Strategic Logics: Complexity, Completeness and Expressivity

Strategic Logics: Complexity, Completeness and Expressivity

A thesis submitted in accordance with the requirements of the University of Liverpool for the degree of Doctor of Philosophy by

Dirk Walther

Department of Computer Science
University of Liverpool
April 2007

To Celia

First Supervisor:	Frank Wolter, University of Liverpool, UK
Second Supervisor:	Michael Wooldridge, University of Liverpool, UK
Advisor:	Wiebe van der Hoek, University of Liverpool, UK
Internal Examiner:	Clare Dixon, University of Liverpool, UK
External Examiner:	Andreas Herzig, Université Paul Sabatier, France

Acknowledgements

The work on this thesis was supported by the EPSRC under the research grant *Virtual Organisations for e-Science* (GR/S62727/01).

I am very grateful for many valuable discussions and the stimulating work with Wiebe van der Hoek, Carsten Lutz, Frank Wolter and Michael Wooldridge.

I also gratefully acknowledge the discussions with and comments by Sophie Pinchinat, Thomas Ågotnes, Dietmar Berwanger, Valentin Goranko, Wojciech Jamroga, Pierre-Yves Schobbens at the Dagstuhl Seminar *Specification, Verification and Test of Open Systems* (06411), October 10–13, 2006.

Contents

CHAPTER 1

Introduction

Computing power is rapidly increasing and the cost of computing capability has steadily decreased since the introduction of integrated circuits in the mid-20th century. This trend was observed by Moore [**180, 181**] in 1965. This has made it possible to introduce processing power into places where it would previously have been uneconomic or impossible. Since the 1980s, many electronic devices have been fitted with a computer in the form of embedded systems, e.g. in radios, watches, TVs, washing machines, cars, etc. In developed countries, it has become normal to own a PC and to communicate over the internet and via mobile phones. Once there are increasingly many computer systems around, the trend is to network them into large distributed systems, an omnipresent example being the internet. As a result, the complexity of tasks that we can automate and delegate to computers has grown steadily. We give more and more control to computers. The realm of our social world is extended to include autonomous computer systems. These systems need the capability to act independently in a way that represents our interests, while interacting with other systems or humans. It is non-trivial to design and build complex systems that exhibit aspects of rationality or human intelligence. Hence there is a need to understand such systems.

"Intelligent" computer systems are studied in Artificial Intelligence (AI) [**215**] whose beginning as a research field was marked by the Dartmouth Conference in 1956 [**171**]. Since the 1980s, Multi-Agent Systems (MASs) have been studied as a sub-discipline of Distributed AI, which focusses on systems in which "intelligent" agents interact with each other. Surveys on MASs can be found in [**232, 231, 271**].

There is no precise definition in AI of what an 'agent' is [**215**], but, generally, an *agent* (Latin *agere*, to act on behalf) is seen as a system that acts within a certain range of autonomy on behalf of a user and with a certain goal. For instance, some programs, robots, but also humans can be seen as agents. A *multi-agent system* is a collection of such agents in which they can interact with each other to meet their objective. The interactions between agents may be cooperative or adversarial. To successfully interact, they will require the ability to cooperate, coördinate, and negotiate with other agents. We can identify four characteristics of MASs [**232**]: first, each agent has incomplete information or capabilities for solving the problem; secondly, there is no global system control; thirdly, data is decentralized; and fourthly, computation is asynchronous. MASs have dynamic environments: from a single agent's viewpoint, the environment is affected by other agents in unpredictable ways.

We now present some advantages of MASs and motivate the interest in MAS research. MASs are appropriate to handle the interactions in domains where people or organizations have different (possibly conflicting) goals and proprietary information. Different organizations will need their own systems to reflect their capabilities and priorities. In particular, MASs are suited to model scheduling tasks, resource management and process control systems. For example, a hospital scheduling system can be modelled with a MAS where people in the hospital together with their interests are represented by agents [**65**]. Further examples are air-traffic control [**164**], electricity transportation management and particle accelerator control [**129**], autonomous space craft control [**220**], climate control [**113**] and electronic commerce. In MASs, control of tasks can be assigned to different agents which provides a method for parallel computation. The system benefits from a speed up and an increase of robustness and reliability by having redundant agents. MASs are inherently modular and thus scalable and flexible. These features are also studied in the research area of Grid Computing in the context of distributed resource sharing systems to make such systems more flexible and adaptable; see, e.g., the series of international workshops on Agent Grid Computing (AGC) [**1**]. Software development can benefit from the modularity inherent to MASs which provides a way of simplifying a problem by assigning subtasks to several agents that interact with each other and different kinds of computational entities. Modular software is easier to develop and maintain than monolithic systems. Thus MASs can be seen as a promising paradigm in software development [**232**], where interaction is recognised as an important characteristic of complex software. Since humans can also be understood as agents, we can simulate MASs to investigate various social processes. In this way, MASs can help to clarify and explain problems in social and life sciences, including intelligence itself [**64**]. The central problems of sociology and MASs are closely related: in sociology one tries to find explanations for the existence of social order among free but interdependent individuals, in MASs one tries to optimize efficiency of the whole system whilst respecting agent's autonomy.

Research in MASs deals with several areas such as interaction between agents, planning and learning, agent-based software design, engineering of practical applications using agent technology, etc. The study of interaction includes agent communication, which facilitates automated cooperation, coordination and negotiation, through techniques such as argumentation, game theory, computational economics and belief-desire-intention models [**271**]. The field of MASs is in particular affected by a quest for an appropriate theoretical foundation. Theory can supply formal methods that provide a semantics to the architectures, languages and tools. In this work, we take a formal approach towards such a foundation. We focus on presentation and reasoning about interaction and knowledge of agents.

There are several models of agents decision making with different degrees of deliberation. The simplest form is the *reactive agent* with no deliberation at all: such agents simply act on observations of the environment [**271, 170**]. Another model is the *rational agent* as used in economics which is characterised by self-interest, i.e., preference is given to those actions that maximise its own utility. The beliefs, desires, and intentions (BDI) model [**40, 209**] incorporates mental attitudes of agents that loosely correspond to human characteristics. These attitudes and the context of an agent specify preferences over actions available to the agent. Other attitudes like knowledge, obligation, commitment, strategy, etc. can be taken into account as well. We can formalise such mental models of agents using modal logic, where attitudes are represented by modal operators [**271**], and we obtain, e.g., a modal logic of knowledge [**79**], a deontic logic [**174**] or a modal logic of strategic ability [**13**]. In this thesis, we use modal logic as a tool to describe dynamic MASs and reason about agent's interactions over time and their knowledge. Modal logics are particularly useful for applications since they often provide a good balance between expressivity and computational complexity. Here, by a 'good balance' we mean that modal logics provide sufficient expressivity for many applications while model checking tasks are solvable with algorithms of relatively low complexity. To this end, we introduce several modal logics such as temporal, epistemic, strategic logic and combinations thereof, and characterise their expressivity and investigate the complexity of reasoning tasks. We focus on relational semantics for modal logic which provides an intuitive way for representing the modal operators. Various logical formalisms were suggested in the multi-agent literature to capture different aspects of MASs; see, e.g., [**182, 100, 13, 195, 247, 125**]. In MASs, we are interested in describing combined abilities of agents belonging to a group of agents. But it is not straightforward to extend the modalities for knowledge and time to a multi-agent setting. Depending on whether we consider knowledge or time, we obtain several different combined modalities. For instance, in epistemic logic, there are at least three ways of combining knowledge of single agents: general-, common- and distributed knowledge [**79**]. A multi-agent account for temporal logic is Alternating-time Temporal Logic (ATL) which was provided by Alur, Henzinger and Kupferman [**13**] in 1997. The novelty of ATL is that it provides modalities for single agents and groups of agents for describing combined strategic abilities. A model for ATL describes all possible outcomes of the actions of interacting agents. However, ATL does not provide any notion of rationality. ATL also does not model the process of cooperation and the notion of a strategy is a purely semantic construct, i.e., there is no account of agents having knowledge and sharing it, of agents in a coalition communicating, negotiating, compromising and reaching a conclusion. To take a step forward to overcome ATL's drawbacks, van der Hoek and Wooldridge [**272**] suggested in 2002 an epistemic variant of ATL. But the interaction between temporal and epistemic abilities of agents is still controversial [**254, 115**] and receives much attention [**219, 125, 117, 118, 120, 119, 110**].

Suppose a scenario where agents fulfill certain roles in a dynamic MAS. Such roles might involve complying with norms such as obligations, permissions, responsibilities or powers. Due to the dynamic character of the MAS, agents might need to interact by transferring normative attributes from an agent to another. Such interactions are called *delegation*. Formal models of delegation and control were studied in, e.g., [**189, 149, 191**]. In this work, we consider the scenario where agents delegate control over propositions to other agents. The distinction between controllable and uncontrollable propositions stems from areas like discrete event systems and control theory, where, e.g., Boutilier [**39**] studied control in the context of deontic logic. Control and controllable propositions were also studied in [**52, 66, 249, 248**].

We now give an overview of the thesis. The main purpose of Chapter 2 is to introduce basic concepts and notation and to review relevant literature. The first section presents a brief survey on modal logic. Then, in sections 2.2, 2.3 and 2.4, we introduce epistemic, temporal and strategic modal logics and state known results that characterise their expressivity and computational complexity. In particular, we consider variants of ATL as extensions of branching-time logics. With such ATL-like logics we can describe dynamic multi-agent interactions. In Section 2.5, we discuss extensions of ATL with epistemic notions. Additionally, we suggest a framework for memory-bounded strategic reasoning. In particular, we introduce an epistemic variant of ATL that accounts for agents with limited memory resources as this case was neglected in the literature to date.

In Chapter 3, we investigate the computational complexity of ATL and its epistemic extension ATEL. We show in detail how the complexity of the satisfiability problem for both logics can be settled at EXPTIME-complete. The part of the chapter about ATL is based on the paper "ATL Satisfiability is Indeed EXPTIME-complete" by Walther, Lutz, Wolter and Wooldridge in the Journal of Logic and Computation, 2006 [**265**], and the part about ATEL is based on the paper "ATEL with Common and Distributed Knowledge is ExpTime-Complete" by Walther which was presented at the 4th Workshop on Methods for Modalities, Humbolt University, Berlin, December 1–2, 2005 [**264**].

In Chapter 4, we aim to extend the expressiveness of ATL without increasing its computational complexity. We introduce explicit names for strategies in the object language and extend modal operators with the possibility to bind agents to strategy names. In this way, we can fix the decisions of agents that possibly belong to several coalitions. By identifying the behaviour of agents, we can reason about the effects of agents changing coalitions. Dynamic coalitions provide more flexibility to adapt abilities to a changing environment. We investigate the expressivity of the resulting logic ATLES and compare it to ATL and ATL*. Moreover, we formulate two model checking problems for ATLES and investigate their complexity as well as the complexity of the satisfiability problem for ATLES. Additionally, we present a complete axiomatisation. This chapter is based on the paper "Alternating-time Temporal Logic with Explicit Strategies" by Walther, van der Hoek and Wooldridge which is going to presented at the 11th Conference on Theoretical Aspects of Rationality and Knowledge (TARK), Brussels, Belgium, June 25–27, 2007 [**266**].

The Chapter 5 is devoted to delegation and control. Control in a MAS is typically decentralised, i.e., decision making is distributed over the agents in the system. The design of decentralised control and efficient coordination mechanisms among agents is a challenging issue. In this chapter, we investigate a logical framework that allows us to reason about agents that have control over propositions and are capable of transferring that control to other agents in the system. This work presents the initial stages towards a more rigorous theoretical framework. We introduce the logic DCL-PC and present a complete axiomatisation. Moreover, we investigate the complexities of the satisfiability and the model checking problems. This chapter is based on work that was undertaken in collaboration with van der Hoek and Wooldridge.

CHAPTER 2

Epistemic and Temporal Logic

2.1. Modal Logic

The purpose of this section is to give a short historical survey of modal logic and to introduce basic propositional modal logic together with concepts such as axiomatisation, satisfiability checking and model checking. The reader familiar with modal logic may skip this section and continue with Section 2.2.

We start with a brief historical outline of modal logic and describe its basic concepts before we present the basic propositional modal logic in more detail. For a more detailed overview of the area, see [**34**].

Modal logic is an area of logic research that originates in philosophy, but is also considered in mathematics, linguistics, computer science, AI and game theory. Modal logics can be seen as formalisations of modalities, which are concepts like possibility and necessity. Many notions have a "modal" character by specifying possibility or necessity within a certain range. Such notions can be found in the original philosophical study of metaphysical modality. In languages, we find auxiliary verbs that indicate a modality of a verb: in English, necessity is expressed by verbs like *must*, *need to* and *have to*, and possibility by *can* and *could*. Several modal logics have been developed that handle modalities of time, space, obligation, conditionality, knowledge, computation, action, etc. Modal logics extend the descriptive range of "standard" logic. From a more technical perspective, modal logics are equivalent to fragments of standard first- or higher-order predicate logic that exhibit an interesting balance between expressive power and computational complexity.

The formal study of modalities started in the early 20th century philosophy with the work by C.I. Lewis [**159**] who in 1918 introduced the first modal operators in an attempt to solve the "paradox" of material implication. Previously, the study of modalities was informal, dating back to the ancient Greeks. In modern notation, Lewis prefixed a logical formula φ with the symbols '$\Box$' or '$\Diamond$' to denote φ's modality: $\Box\varphi$ means 'φ is necessary' and $\Diamond\varphi$ for 'φ is possible'. In 1933, Gödel [**93**] gave a formal semantics to the modal operators to denote mathematical provability, i.e., $\Box\varphi$ means 'φ is provable', and $\Diamond\varphi$ that 'φ is consistent'. Gödel's work turned out to be very influential for the development of many other logical formalisms, and, in particular, for the study of formal provability predicates in arithmetic and set theory. Later, many more modal operators where introduced. In 1951, von Wright [**262**] distinguished between four kinds of modalities: alethic, epistemic, deontic and existential. Alethic modalities (Greek *alêtheia*, truth) express modes of truth (necessity, possibility), epistemic modalities (Greek *epistêmê*, knowledge) the modes of knowledge for single agents or for groups ('known to be true', 'known to be false'), deontic modalities (Greek *deontôs*, as it ought) the modes of morality and norms (obligatory, permitted), and existential modalities the modes of existence (universal, existing). Over the years, various modal logics in different subjects have been developed including temporal, dynamic logic, epistemic and deontic logic, action and strategic logic (or logics of agency), etc.

Several semantics for modal languages have been suggested, among them algebraic, topological and relational semantics [**34**]. We give a brief overview of these semantics, but for more detailed discussion, we refer to [**95**]. In 1952, Jónsson and Tarski [**132, 133**] introduced Boolean algebras with operators and their representation over relational structures, but without mention of modal logic. Later in 1966, Lemmon [**154**] introduced an algebraic semantics for modal logic. In the *algebraic semantics*, we view formulas as terms and evaluate them in suitable algebras, namely Boolean algebras with operators. In the *topological semantics*, modal logic formulas are interpreted in topological spaces; cf. [**8**]. Initially, in the 1920s, the application of topological spaces given by Kuratowski [**146**] suggested the interpretation of closure and interior operators as modal operations. Indeed, in the late 1930s and early 1940s, Tarski and McKinsey developed an algebraic approach to topology [**233, 172**]. This approach evolved into an active research area with many interesting results within modal logic [**78, 224**], and applications in metric spaces, dynamic systems, spatio-temporal reasoning, etc. In the *relational semantics*, modal formulas are evaluated in graph-like structures. The relational semantics has its origins in the fundamental work of Jónsson and Tarski [**132, 133**], Kripke [**142, 143**], Kanger [**135, 136**] and Hintikka [**111**] followed by the work of Lemmon and Scott [**155**]. Relational semantics, also often referred to as *Kripke semantics*, is studied in model theory, where one is interested in the interplay between a logical language and the relational structures (graphs) for that language. For some applications, however, relational semantics was perceived to be too strong, and weaker versions were suggested, among them *neighbourhood semantics*. Around 1970, Neighbourhood semantics were introduced by Montague [**178, 179**] and Scott [**221**] and then further explored by Segerberg [**223**].

Today, modal logic is primarily involved with relational structures where Modal logic is seen as a tool for talking about graphs. A graph consists of vertices, representing *possible worlds*, and edges between vertices, corresponding to relations between the worlds. In the 1960s, Kripke formalised relational semantics by interpreting modal logic over directed coloured graphs, so-called *Kripke structures* [**143**]. However, the idea of possible worlds for the interpretation of modalities can be traced back to the German philosopher Gottfried Leibniz [**153**] in the 17th century. The idea is to interpret the formula $\Box\varphi$ ('φ is necessary') as a claim that φ is true at all possible worlds, and $\Diamond\varphi$ ('φ is possible') as a claim that φ is true at some possible world. This interpretation give rise to a correspondence between between modal and classical logic: '$\Box$' and '$\Diamond$' are linked with the universal quantifier '$\forall$' and the existential quantifier '$\exists$', respectively. Indeed, as Gabbay [**90**] observed, many modal languages translate into fragments of first-order logics using only finitely many variables. These finite variable fragments have appealing computational properties, e.g., the model checking problem can be solved in PTIME [**258**] while it is PSPACE-hard for first-order logic [**50**]. Moreover, in contrast to first-order logic, many modal logics are decidable [**259, 99**]. Many modal logics, as logics on graphs (also called *frames*), also specify a decidable fragment of the monadic second-order logic MSO. Another interesting connection is between graphs and modal algebras, which is studied in *duality theory*: the notions of bounded morphisms, generated subframes and disjoint unions based on relations correspond to the algebraic notions of subalgebras, homomorphic images and direct products [**132, 33, 34**]. Expressivity of modal languages can be characterised using, e.g., the notion of *bisimulation* identifying those models that cannot be distinguished.

Modal logics are applied in many areas with theoretical and practical interest. In mathematics, we find provability logic [**19, 9**] and a strong connection between modal logic and set theory [**226, 240, 29, 28**]. Philosophy considers modal logic of belief change (belief revision [**10, 91**] and belief update [**137**]), the logic of action [**31**] and deontic logic [**263, 86**]. Linguistics uses modal logic to study the semantics of natural language and to analyse its syntactic structure [**241**]. Modal logics in game theory [**21, 190**] are used to describe and reason about games, e.g., game logic [**197**], Coalition Logic [**196**] and Alternating-time temporal logic [**15**]. In AI, modal logics for MAS provide a theoretical framework for intelligent distributed systems [**232, 231, 271**]. Description Logics [**24**], a notational variant of modal logics, are used for knowledge representation and reasoning in AI and, moreover, provide the basis for ontologies in knowledge management systems, medical- and bio-informatics [**228, 229, 87**] as well as in the semantic web [**32, 18**]. In computer science, temporal logics are employed for automated verification of hardware and software [**54**], epistemic, temporal and conditional operators are used in knowledge-based programming [**94**] and modal logics are involved in the analysis of query languages for XML documents [**47, 48**]. Among the many research forums involving modal logic, at least two are devoted specifically to modal logics: *Advances in Modal Logic* (AiML) [**2**] and *Methods for Modalities* (M4M) [**3**].

In this work, we focus on the relational semantics for modal logic. The basic propositional modal language for describing coloured relational structures is inductively defined using countably infinitely many propositional variables $\mathbf{\Pi} = \{p_0, p_1, \dots\}$, the Boolean connectives of conjunction ('$\wedge$') and negation ('$\neg$') and a unary modal operator box ('$\Box$'). The additional Boolean connectives of disjunction ('$\vee$'), implication ('$\rightarrow$') and the unary modal operator diamond ('$\Diamond$') can be defined in terms of '$\wedge$', '$\neg$' and '$\Box$'. A *Kripke structure* $\mathfrak{M} = \langle W, R, \pi \rangle$ is such a coloured graph, where W is a non-empty set of worlds, $R \subseteq W \times W$ is a binary relation defined on W, and $\pi : \mathbf{\Pi} \rightarrow \mathbf{2}^W$ is a (colouring) function assigning propositions to worlds at which they are true. A world w' is considered *possible* wrt. a world w if it is accessible via relation R from w. Given a formula φ, the satisfaction relation '$\models$', where $\mathfrak{M}, w \models \varphi$ means 'φ is true at world w in $\mathfrak{M}$', is inductively defined as:

- $\mathfrak{M}, w \models p_i$ iff $w \in \pi(p_i)$;
- $\mathfrak{M}, w \models \neg\varphi$ iff $\mathfrak{M}, w \not\models \varphi$;
- $\mathfrak{M}, w \models \varphi \vee \psi$ iff $\mathfrak{M}, w \models \varphi$ or $\mathfrak{M}, w \models \psi$;
- $\mathfrak{M}, w \models \Box\varphi$ iff $\mathfrak{M}, v \models \varphi$ for all $v \in W$ such that $(w, v) \in R$.

A formula φ is *satisfiable* if $\mathfrak{M}, w \models \varphi$ for some Kripke structure $\mathfrak{M}$ and some world w in $\mathfrak{M}$, and φ is *valid* if $\mathfrak{M}, w \models \varphi$ for all $\mathfrak{M}$ and all w in $\mathfrak{M}$. Given the class of Kripke structures, it is interesting to look at the modal formulas determined by this class, i.e., the formulas that are valid in these structures. The set of modal formulas that are valid in every structure of a given class is called its *theory*. The theory of the class of *all* Kripke structures is denoted by K. Given a theory of a class of structures, it is sometimes possible to give a syntactical characterisation of it, which makes it possible to reason about those structures on a purely syntactic level. This is a central theme in proof theory [**235**]. Syntactic characterisations can be given, e.g., in the form of Hilbert-style axiom systems that provide axioms and inference rules for deriving other formulas. Table 2.1 presents such a Hilbert-style axiom system for the formulas in K. Axioms reflect essential properties of the modalities and their interaction with one another. In Table 2.1, the modality '$\Box$' is characterised by the *distribution axiom*, also called (K), and the inference rule *necessitation*. A formula φ is called *provable* or

(TAUT)	Propositional tautologies
(Distribution)	$\Box(\varphi \rightarrow \psi) \rightarrow (\Box\varphi \rightarrow \Box\psi)$
(Modus Ponens)	$\dfrac{\varphi, \varphi \rightarrow \psi}{\psi}$
(Necessitation)	$\dfrac{\varphi}{\Box\varphi}$

TABLE 2.1. The axiom system for K.

derivable in the modal system K, written $\vdash_{\mathsf{K}} \varphi$, if there is a finite sequence of formulas that ends in φ such that each formula in this sequence is an instance of an axiom or follows from previous formulas by application of an inference rule. We call such a sequence of formulas a *proof* or a *derivation of* φ. The axiom system is *sound* if every provable formula is also valid, and it is *complete* if every valid formula is provable. The axiom system in Table 2.1 is sound and complete [**33**], which shows that the syntactic and semantic characterisations indeed specify the same logic. Soundness can easily be shown by an induction on the structure of the derivation. *Completeness theory* is concerned with showing completeness of axiom systems, which is usually much harder than showing soundness [**33**]. Several methods have been developed to obtain completeness. Kripke [**142, 143**] originally used tableaux systems; see Fitting [**84**] for an introduction to modal tableauxs. Another way of proving completeness is via normal forms; see Fine [83]. An important tool in completeness theory is the canonical model construction with which completeness results for many modal logics where proven. This proof technique appeared first in 1960s in the work of Makinson [169] and Cresswell [61], and later became standard due to the work of Sahlqvist [216] and Lemmon-Scott [155]. However, there are many incomplete modal logics for which the canonical construction fails. In the late 1970s, Blok [35, 36, 37] showed that incomplete modal logics are not exceptional but rather the norm, and examples of finitely axiomatisable incomplete logic were given by Fine [82] and Thomason [234].

reflexive	$\forall x(xRx)$	$\Box\varphi \to \varphi$
symmetric	$\forall x\forall y(xRy \to yRx)$	$\varphi \to \Box\neg\Box\neg\varphi$
serial	$\forall x\exists y(xRy)$	$\Box\varphi \to \neg\Box\neg\varphi$
transitive	$\forall x\forall y\forall z(xRy \wedge yRz \to xRz)$	$\Box\varphi \to \Box\Box\varphi$
euclidean	$\forall x\forall y\forall z(xRy \wedge xRz \to yRz)$	$\neg\Box\neg\varphi \to \Box\neg\Box\neg\varphi$
weakly dense	$\forall x\forall y(xRy \to \exists z(xRz \wedge zRy))$	$\Box\Box\varphi \to \Box\varphi$
weakly directed	$\forall x\forall y\forall z(xRy \wedge yRz \to \exists v(yRv \wedge zRv))$	$\neg\Box\neg\Box\varphi \to \Box\neg\Box\neg\varphi$

TABLE 2.2. First-order properties expressed in modal logic.

Properties of the relation in a Kripke structure can be expressed in modal logic by a set of modal formulas that are each valid on the structure. Such a set of modal formulas with a common syntactic form can be specified by a scheme. Table 2.2 shows some properties with their definition in first-order logic together with a modal logic formula (a scheme) that specifies the property. However, the expressive power of modal logic and first-order logic do not coincide. Here are two modal formulas that do not correspond to any first-order definable property (see [33]):

$$\text{Löb formula} : \Box(\Box\varphi \to \varphi) \to \Box\varphi$$

$$\text{McKinsey formula} : \Box\neg\Box\neg\varphi \to \neg\Box\neg\Box\varphi$$

For instance, the relations of the Kripke structures on which the Löb formula is valid are transitive and the converses of the relations are well-founded (i.e., no infinite path starts at any world, which excludes, in particular, any loops). The McKinsey formula defines a class of structures with uncountably many worlds. However, the properties specified by the Löb and the McKinsey formula are second-order logic conditions. Conversely, there are first-order properties that cannot be expressed in modal logic, e.g.:

$$\text{asymmetry} : \forall x\forall y(xRy \to \neg(yRx))$$

$$\text{antisymmetry} : \forall x\forall y(xRy \wedge yRx \to x = y)$$

The expressive power of a modal language is usually measured in terms of the distinctions it can draw between two structures. A modal language can distinguish two structures if there is a formula of this language that is true at a world of one structure and false at a world of the other structure. The notion of *bisimulation* helps to characterise the expressive power of a modal language. Informally, a bisimulation is a relation between two Kripke structures, where related worlds are labelled with the same propositional variables, and, for any successor in one of the structures, it is possible to find a matching successor in the other structure that are related again via the bisimulation. The notion of bisimulation is a quite general: it comprises operations on Kripke structures such as disjoint union, generated submodels and bounded morphisms. Moreover, bisimulations can be seen as variation of Ehrenfeucht-Fraïssé games [**112**]. Expressivity of modal logic can be characterised via bisimulations due to the fact that modal satisfiability is *invariant* under bisimulations, i.e., two bisimilar worlds satisfy the same modal formulas, and, conversely, in finitely branching structures, two worlds satisfying the same modal formulas are bisimilar. Moreover, relating to first-order logic, van Benthem [**237**] showed that modal logic is the maximal such language in the sense that each first-order formula invariant under bisimulation is equivalent to a modal logic formula. This result demonstrates how bisimulations reflect the local character of the modal satisfaction relation. Bisimulations were independently developed in modal logic by van Benthem [**236, 237**] and in computer science by Park [**193**] and Milner [**177**]; see [**109**] for further details.

The balance between expressivity and computational complexity of modal logics is interesting for many applications that involve automated reasoning tasks. For instance, applications involving reasoning tasks with temporal logics are hard- and software verification and automated program verification [**54, 55**], another example is the analysis of distributed computer programs [**79**] which uses epistemic logics and combinations of epistemic and temporal logics. Generally, the aim is to find sufficiently expressive logics with low complexity. Automated reasoning is mainly concerned with problems like checking satisfiability and model checking. The *satisfiability problem* for a modal logic asks whether a given formula can be satisfied at some world in some Kripke structure. A logic is called *decidable* if its satisfiability problem can be solved effectively. The *model checking* task is: Given a world w in some structure and a formula φ, is φ satisfied at w? The computational complexity states how much time and memory a computer would need to solve these problems. Decidability of a logic can be established by reducing the problem to known results for other logics, e.g., translating modal formulas into the two-variable fragment of first-order logic which is known to be decidable [**34**]. However, for establishing complexity results, other techniques are usually required that are more constructive. For the basic modal logic, we can show that it has the *finite tree model property* [**259**], i.e., every formula can be satisfied in a finite tree structure. More precisely, they can be satisfied in tree structures whose depth and branching factor is bounded as a function of the length of the input formula. Given this property, it turns out that the satisfiability problem for the basic modal logic K is in PSPACE [**147**]. Note that first-order logic, on the other hand, can enforce infinite models and satisfiability is already undecidable for fragments of first-order logic with three variables [**34**]. Many modal logics are "robustly" decidable, i.e., they remain decidable even after extending their expressive power, e.g. adding path quantification and fixed points to the logic. This is partially due to the tree model property which permits the use of automata-theoretic techniques for showing decidability [**259, 99**]. Model checking a basic modal logic formula φ can be done with an algorithm that labels the worlds of the structure in a bottom-up fashion with subformulas of φ that are true there. In this way, model checking the basic modal logic can be solved in polynomial time in the size of the input formula.

2.2. Epistemic Logic

An important application of modal logic is modelling knowledge and doxastic attitudes like belief – epistemic logic. Intuitively, the goal in epistemic logic is to describe the actual knowledge of agents. But, modeling actual knowledge of human agents appears to be difficult. For instance, we would not always agree with the implication that a person knowing both φ and $\varphi \to \psi$ also knows ψ. Despite these problems, epistemic logic is still useful for modeling knowledge when we make some assumptions about the agents: agents are perfect and rational reasoners. But it is worth keeping in mind that no realistic human agent has such capabilities.

In 1932, C.I. Lewis and C.H. Langford introduced the five systems S1,...,S5 to give an axiomatic account of the alethic modality necessity [**160**]. In particular, the systems S4 and S5 gained much attention for modelling knowledge and belief. Although, Kripke and his colleagues introduced a relational semantics for modal logic in the 1950s, it was Hintikka [**111**] in 1962 who interpreted epistemic logic for the first time in terms of possible worlds. A contemporary account of epistemic logic can be found in [**175, 79**]. In this work, we focus on the epistemic logic S5.

In the remainder of this work, we use the Backus-Naur-Form (BNF) [**138, 85**] as a meta-language to specify the syntax of a logic.

2.2.1. The Logic S5. We now define the syntax and semantics of S5.

DEFINITION 2.1. (S5 SYNTAX). Let Π be a countably infinite set of atomic propositions and $\Sigma = \{1, \dots, n\}$ be a set of n agents. The set of $S5_\Sigma$-formulas φ is given by the following BNF specification:

$$\varphi \quad ::= \quad p \quad | \quad \neg\varphi \quad | \quad \varphi \vee \varphi \quad | \quad K_a\varphi$$

where p ranges over atomic propositions in Π and a over agents in Σ.

Logical truth ($\top$), falsehood ($\bot$) and the Boolean connectives ($\wedge$, $\to$ and $\leftrightarrow$) are defined as usual. The modality K_a is an *epistemic operator* that represents 'agent a knows'. In epistemic logic, it is common to write K_a instead of $\Box_a$.

In the literature, the logic S5 for n agents is usually referred to as $S5_n$ [**79**]. For reasons of consistency of notation, we deviate from this practise and denote this logic $S5_\Sigma$.

S5 is a modal logic which we interpret using a relational semantics. As semantic structures we use a special class of Kripke structures. The relations of these Kripke structures have properties that reflect how we describe knowledge. In the case of S5, the relations are equivalence relations, which we call *epistemic accessibility relations*. The agent's state of knowledge or information is modelled by the current equivalence class of the epistemic accessibility relation for that agent. Considering an agent a, we denote the epistemic accessibility relation for a with $\sim_a$. The *information state* of a at a world w is the set of worlds in the equivalence class $[w]_{\sim_a}$ of w wrt. $\sim_a$. All worlds that are $\sim_a$-accessible from w are considered *possible* by a, or, alternatively, are epistemically *indistinguishable* for a. Agent a "knows φ" if φ is true at all worlds a considers possible, or if φ is true at all worlds that a cannot distinguish according to her knowledge. That is, the smaller the set $[w]_{\sim_a}$, the less worlds agent a considers possible at the world w, or the less worlds are indistinguishable for a at w, and hence the more a knows.

DEFINITION 2.2. (EPISTEMIC STRUCTURE). Given a finite set Σ of agents, an *epistemic structure* for Σ is a tuple $\mathfrak{M} = \langle \Pi, \Sigma, W, \{\sim_a\}_{a\in\Sigma}, \pi\rangle$ where

- $\Pi \subset \mathbf{\Pi}$ is a non-empty set of *atomic propositions*;
- W is a set of *possible worlds*;
- $\sim_a$ is an *epistemic accessibility relation* on W, one for each agent a in Σ, which is required to be an equivalence relation;
- $\pi : \Pi \to 2^W$ is a *valuation function* which assigns to every atomic proposition in Π a set of worlds at which it is true.

For any relation R on the set of worlds (or states), a world w is called *R-accessible* from world v and w is an *R-successor* of v if $(v, w) \in R$. We use this notation throughout this work.

Formally, S5-formulas are evaluated on epistemic structures as follows.

DEFINITION 2.3. (S5 SEMANTICS). Given an epistemic structure $\mathfrak{M} = \langle \Pi, \Sigma, W, \{\sim_a\}_{a\in\Sigma}, \pi\rangle$, the satisfaction relation '$\models$' is inductively defined as follows: For all worlds $w \in W$, all $a \in \Sigma$, it holds that:

- $\mathfrak{M}, w \models p$ iff $w \in \pi(p)$, for all atomic propositions $p \in \Pi$;
- $\mathfrak{M}, w \models \neg\varphi$ iff $\mathfrak{M}, w \not\models \varphi$;
- $\mathfrak{M}, w \models \varphi \vee \psi$ iff $\mathfrak{M}, w \models \varphi$ or $\mathfrak{M}, w \models \psi$;
- $\mathfrak{M}, w \models K_a\varphi$ iff $\mathfrak{M}, v \models \varphi$ for all $\sim_a$-successors v of w.

If for some world w in some epistemic structure $\mathfrak{M}$ we have $\mathfrak{M}, w \models \varphi$, then the S5-formula is *true* at w, and $\mathfrak{M}$ is called a *model* of φ. An S5-formula is *satisfiable* if it has a model, and it is valid if it is true at all worlds in all epistemic structures.

Popular examples for illustrating the relationship of knowledge, communication and action in a distributed system with S5 are variants of the *Cheating Husbands Puzzle* or the *Muddy Children Puzzle*. For a general treatment and detailed analysis of such puzzles, see the work by Moses, Dolev and Halpern [186]. Here, however, we use the knowledge game *Hexa* for that purpose.

EXAMPLE 2.4. (HEXA). We illustrate the use of S5 and epistemic structures by modelling states of the knowledge game Hexa as described by van Ditmarsch [251]. In Hexa, three players a, b and c each hold one of three cards ace, king or queen. Figure 2.1 displays three epistemic structures, where the left one describes the initial situation. Each world is marked with three letters, where, e.g., QKA means that player a holds the queen, b the king and c the ace. The epistemic accessibility relations are denoted by lines, that are labelled by the corresponding agent, between the worlds (omitting the reflexive edges). In the beginning, every player only knows her own card

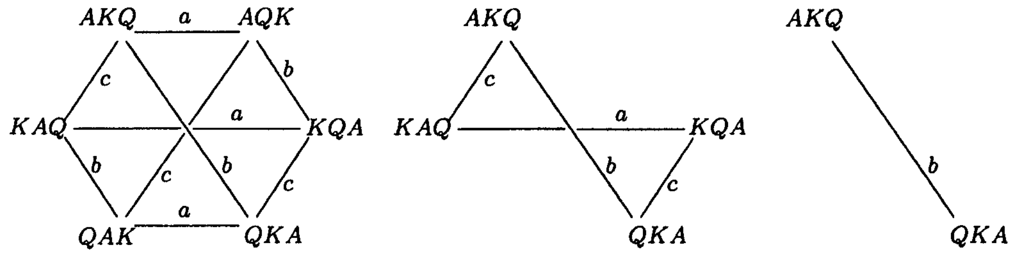


FIGURE 2.1. Three epistemic structures for the knowledge game Hexa.

but cannot observe the cards of the other players. That is, everyone has incomplete information about the actual state of the game. For instance, at world QKA, player a's information state is the set containing the worlds QKA and QAK, i.e., a considers both worlds possible, or, alternatively, a cannot distinguish between them. In other words, a does not know the other players' cards, which we can describe with the S5-formula $\bigwedge_{i=b,c} \neg K_a ace_i \wedge \neg K_a king_i$ using propositional variables like ace_b to represent the fact that player b holds the queen. We can also describe knowledge of a *higher order* in the sense that a player knows about the knowledge of other players. We can state that, at QKA, player a knows that b knows her card with $K_a(K_b ace_b \vee K_b king_b)$. Below we will come back to this example and show what can be said about the players combined knowledge after some players reveal information about their cards to the others. ⊣

So far, we have defined $S5_\Sigma$ purely semantically. We now go on to give a syntactic characterisation of this logic.

2.2.2. Axiom System. The properties of knowledge and belief can be modelled with the following axioms:

T	$K_a\varphi \rightarrow \varphi$	(Veridicality)
D	$K_a\varphi \rightarrow \neg K_a \neg\varphi$	(Consistency)
4	$K_a\varphi \rightarrow K_a K_a\varphi$	(Positive Introspection)
5	$\neg K_a\neg\varphi \rightarrow K_a\neg K_a\neg\varphi$	(Negative Introspection)

Notice that these "axioms" are actually axiom schemes, with φ ranging over S5-formulas and a over agents. When a set Σ of agents is understood, φ ranges over formulas of $S5_\Sigma$ and a over the agents in Σ. The veridicality axiom (T) (sometimes also called factivity axiom) states that knowledge must be the case or equivalently that there is no false knowledge: if agent a knows φ, then φ must hold. When modeling belief, (T) is usually dropped in order to enable false beliefs. For belief, typically (D) is used instead of (T). The consistency axiom (D) states that an agent must be consistent with her knowledge: if a knows φ, then a does not know not φ. The axiom (4) and (5) describe positive and negative introspection of agents: (4) states that if a knows φ, then a knows that she knows φ, and (5) that if a does not know φ, then a knows that she does not know φ.

The axioms (T), (4) and (5) characterise the properties of the epistemic operator K_a. An axiom system for $S5_\Sigma$ is presented in Table 2.3, where a ranges over the agents in Σ. Essentially, it extends the axiom system in Table 2.1 for the basic modal logic K.

(A1)	Propositional tautologies		
(A2)	$K_a(\varphi \to \psi) \to (K_a\varphi \to K_a\psi)$		
(A3)	$K_a\varphi \to \varphi$		
(A4)	$K_a\varphi \to K_aK_a\varphi$		
(A5)	$\neg K_a\neg\varphi \to K_a\neg K_a\neg\varphi$		
(R1)	$\dfrac{\varphi, \varphi \to \psi}{\psi}$	(R2)	$\dfrac{\varphi}{K_a\varphi}$

TABLE 2.3. An axiom system S5.

The semantic and syntactic characterisation are connected by the following theorem.

THEOREM 2.5. *The axiom system for $S5_\Sigma$ is sound and complete.*

This result is can be found in [**176, 101**]. Soundness can easily be shown by an induction on the structure of the derivation. Completeness can proven by contraposition which involves the construction of a canonical model to falsify the negated input formula; see, e.g., Chapter 4 of [**33**].

The axioms give us more insight about the modal logic under consideration since they represent properties of the accessibility relations; see Section 2.1:

T — reflexive;
4 — transitive;
D — serial (right unbounded);
5 — euclidean.

Since the system S5 contains the axioms (T), (4) and (5), we have that all formulas generated by that system are valid on exactly those epistemic models which epistemic accessibility relations are reflexive, transitive and euclidean, i.e. equivalence relations.

Notice that this corresponds to our semantic specification of S5. For modeling belief, one usually uses the axioms (D), (4) and (5) which yields the system KD45. Formulas KD45 are valid on exactly those models with a serial, transitive and euclidian relation.

2.2.3. Combined Knowledge. Combining the knowledge of several agents yields two further interesting concepts: common knowledge and distributed knowledge. Common knowledge was first introduced in 1969, in Lewis's [**162**] philosophical analysis of conventional social practices. It plays an essential role for understanding of natural language in dialogues [**53**] and reaching agreements and coordinating actions [**101, 102**]. Common knowledge is also an important concept in game theory for investigating rationality [**20**]. Distributed knowledge is an important concept for reasoning about knowledge in distributed systems [**101, 81, 102**]. For a survey on multi-agent epistemic logic, see [**79, 175**].

We now add to $S5_\Sigma$ three modalities for combined knowledge: E_A, C_A and D_A, where $A \subseteq \Sigma$. These modalities are *epistemic operators* that respectively stand for 'every agent in A knows', 'it is common knowledge among the agents in A', 'it is distributed knowledge among the agents in A'. Given an epistemic structure $\mathfrak{M} = \langle \Pi, \Sigma, W, \{\sim_a\}_{a\in\Sigma}, \pi \rangle$, we introduce abbreviations for epistemic accessibility relations that correspond to those three kinds of combined knowledge:

$\sim_A^E = \bigcup_{a\in A} \sim_a$ .. the union of the accessibility relations of the agents in A;

$\sim_A^C = (\sim_A^E)^+$ .. the transitive closure of the relation $\sim_A^E$;

$\sim_A^D = \bigcap_{a\in A} \sim_a$ .. the intersection of the accessibility relations of the agents in A.

The new epistemic operators are interpreted in $\mathfrak{M}$ as follows. We extend the definition of $\models$ in Definition 2.3: For all worlds $w \in W$, all $a \in \Sigma$, it holds that

- $\mathfrak{M}, w \models E_A\varphi$ iff $\mathfrak{M}, v \models \varphi$ for all worlds v with $w \sim_A^E v$;
- $\mathfrak{M}, w \models C_A\varphi$ iff $\mathfrak{M}, v \models \varphi$ for all worlds v with $w \sim_A^C v$;
- $\mathfrak{M}, w \models D_A\varphi$ iff $\mathfrak{M}, v \models \varphi$ for all worlds v with $w \sim_A^D v$.

We obtain a hierarchy of combined knowledge of a coalition A:

$$C_A\varphi \Rightarrow \cdots \Rightarrow E_A^3\varphi \Rightarrow E_A^2\varphi \Rightarrow E_A\varphi \Rightarrow D_A\varphi \Rightarrow \varphi$$

where $E_A^m\varphi$ (with $m > 0$) abbreviates $\underbrace{E_A \cdots E_A}_{m\text{-times}}\varphi$. In this hierarchy, common knowledge is the strongest form of combined knowledge since it corresponds to what is publicly known. Distributed knowledge is the weakest form by the fact that it is distributed among A and no member of A necessarily has it. However, the hierarchy collapses in some cases, namely when there is only one agent or, in a multi-agent setting, when all agents have the same knowledge (e.g., in a system where several processes share the same memory).

EXAMPLE 2.6. (HEXA CONTINUED). We now continue the discussion about the card game Hexa in Example 2.4. Suppose that, at this point, player c reveals to the other players that her card is not the king. Clearly, after that public announcement, the players' knowledge changes. The current situation is described by the epistemic structure in the middle of Figure 2.1. At any world, it is now common knowledge among all players that c does not hold the king: $C_{a,b,c}\neg king_c$. Assuming we are in world QKA, player c's statement resulted in the shrinking of a's information state to the singleton set containing QKA, i.e., at this world, a only considers QKA possible and a can distinguish every other world from it. In other words, a learned the other players' cards while b and c did not gain more insight. The formula $K_a(king_b \wedge ace_c)$ is now true at QKA while we still have $\bigwedge_{i=a,c} \neg K_b ace_i \wedge \neg K_b queen_i$ and $\bigwedge_{i=a,b} \neg K_c king_i \wedge \neg K_c queen_i$. If, at this stage in the game, player a announces that she also does not have the king, the situation changes again, which is described by the rightmost structure of Figure 2.1. Again, this announcement is now common knowledge and, thus, $C_{a,b,c}\neg king_a$ holds at all worlds. However, at QKA, player b still did not gain any more insight about the other players' cards. On the other hand, both a and c learned something and we have that QKA satisfies $E_{\{a,c\}}(queen_a \wedge ace_c)$, i.e., everyone among a and c knows each others cards. Moreover, it is now even common knowledge at QKA that b has the king: $QKA \models C_{\{a,b,c\}} king_b$. ⊣

The properties of operators for combined knowledge can be described by the axioms in Table 2.4 (cf. [**101, 103**]), where $\Sigma = \{a_1, \ldots, a_n\}$ and the subscript Σ in E_Σ, C_Σ and D_Σ is dropped. The axiom (A6) defines the operator E in terms of what each

(A6)	$E\varphi \leftrightarrow (K_{a_1}\varphi \wedge \cdots \wedge K_{a_n}\varphi)$
(A7)	$C\varphi \rightarrow E(\varphi \wedge C\varphi)$
(R3)	$\dfrac{\varphi \rightarrow E(\psi \wedge \varphi)}{\varphi \rightarrow C\psi}$
(A8)	$K_a\varphi \rightarrow D\varphi$
(A9)	$D(\varphi \rightarrow \psi) \rightarrow (D\varphi \rightarrow D\psi)$
(A10)	$D\varphi \rightarrow \varphi$
(A11)	$D\varphi \rightarrow DD\varphi$
(A12)	$\neg D\neg\varphi \rightarrow D\neg D\neg\varphi$

TABLE 2.4. Axioms and rules for common- and distributed knowledge.

member in Σ knows. Common knowledge is described with (A7) and (R3), which is often referred to as induction rule. The axiom (A7) characterises common knowledge as a solution to the greatest fixpoint equation $C\varphi \leftrightarrow E(\varphi \wedge C\varphi)$: the direction from right to left is trivial as $E(\varphi \wedge C\varphi)$ implies $C\varphi$, and the left to right direction is given by (A7). Distributed knowledge is characterised with axioms (A8) to (A12). The axiom (A8) describes the interaction between individual knowledge and distributed knowledge: whatever an agent a knows is distributed knowledge. The remaining axioms state that distributed knowledge behaves essentially like knowledge of a single agent.

In order to be able to reason about coalitional combined knowledge, the epistemic operators E, C and D can additionally be parameterized with a set of agents $A \subseteq \Sigma$, i.e., E_A, C_A and D_A. There are at least two new emerging properties:

- $C_A\varphi \rightarrow C_B\varphi$, for all $A, B \subseteq \Sigma$ with $B \subseteq A$;
- $D_A\varphi \rightarrow D_B\varphi$, for all $A, B \subseteq \Sigma$ with $A \subseteq B$.

We are currently not aware of any literature that explicitly treats the coalitional versions C_A and D_A of the operators C and D. However, the complexity results below in Section 2.2.4 is claimed to hold in the more general case [**103**].

We write S5C (S5CD) for the logic S5 enriched with modalities for common knowledge (common- and distributed knowledge). The axiom system for S5C$_\Sigma$ consists of the system S5$_\Sigma$ and the axioms (A6) and (A7) plus the rule (R3) from the Table 2.4. The system for S5CD$_\Sigma$ consists of axioms and rules from the tables 2.3 and 2.4. The following results are well known [**103, 243**].

THEOREM 2.7. *The axiom systems S5C$_\Sigma$ and S5CD$_\Sigma$ are sound and complete.*

In 1985, Halpern and Moses [**101**] gave an axiomatisation of S5D$_\Sigma$. A formal proof of soundness and completeness was presented later in 1992 by Fagin, Halpern and Vardi [**80**]. A complete axiomatisation of S5C$_\Sigma$ with common knowledge was given in [**176, 152, 101**]. A complete axiom system for S5CD$_\Sigma$ with common and distributed knowledge was presented in [**80, 103, 243**].

2.2.4. Complexity. The complexity of the satisfiability problem for variants of S5 depends on the number of agents and the presence of epistemic operators for combined knowledge. The complexity of S5$_\Sigma$ with or without common knowledge and for one or many agents is given in Table 2.5, cf. [**103**]. Adding distributed knowledge to the language does not affect the complexity, i.e., S5D$_\Sigma$ and S5CD$_\Sigma$ are as complex as S5$_\Sigma$ and S5C$_\Sigma$, respectively.

	S5$_\Sigma$, S5D$_\Sigma$	S5C$_\Sigma$, S5CD$_\Sigma$
$\lvert\Sigma\rvert = 1$	NP-complete	PSPACE-complete
$\lvert\Sigma\rvert > 1$	PSPACE-complete	EXPTIME-complete

TABLE 2.5. Complexity of S5-variants.

2.2.5. Expressivity. We now introduce *epistemic bisimulation* as a notion of equivalence between epistemic structures. This is a variant of the notion of bisimulation for the basic modal logic; see, e.g., [**33**].

DEFINITION 2.8. EPISTEMIC BISIMULATION Given two epistemic structures $\mathfrak{M} = \langle \Pi, \Sigma, W, \{\sim_a\}_{a\in\Sigma}, \pi\rangle$ and $\mathfrak{M}' = \langle \Pi, \Sigma, W', \{\sim'_a\}_{a\in\Sigma}, \pi'\rangle$ for a finite set Σ of agents, and an agent $a \in \Sigma$, a binary relation $H \subseteq W \times W'$ is an *a-epistemic bisimulation between*

$\mathfrak{M}$ *and* $\mathfrak{M}'$ if, for all worlds q_1 and q_2 with $(q_1, q_2) \in H$, the following three conditions hold:

(i) $\pi(q_1) = \pi'(q_2)$;
(ii) for every q_1' with $q_1 \sim_a q_1'$, there is a q_2' with $q_2 \sim_a' q_2'$ such that $(q_1', q_2') \in H$;
(iii) for every q_2' with $q_2 \sim_a' q_2'$, there is a q_1' with $q_1 \sim_a q_1'$ such that $(q_1', q_2') \in H$.

If there is an a-epistemic bisimulation H with $(x, y) \in H$, then the worlds x and y are called *a-epistemic bisimilar*, written $x \leftrightarroweq_a^{\varepsilon} y$.

Two worlds x and y are *A-epistemic bisimilar*, written $x \leftrightarroweq_A^{\varepsilon} y$, if there is a binary relation $H \subseteq W \times W'$ with $(x, y) \in H$ satisfying the following three properties:

(iv) H is an a-epistemic bisimulation between $\mathfrak{M}$ and $\mathfrak{M}'$, for all agents $a \in A$;
(v) for all coalitions $B \subseteq A$ and all worlds $q_1, q_1' \in W$ and $q_2 \in W'$ with $(q_1, q_2) \in H$, $q_1 \sim_B^D q_1'$ implies that there is a world $q_2' \in W'$ with $q_2 \sim_B'^D q_2'$ such that $(q_1', q_2') \in H$;
(vi) for all coalitions $B \subseteq A$ and all worlds $q_2, q_2' \in W'$ and $q_1 \in W$ with $(q_1, q_2) \in H$, $q_2 \sim_B'^D q_2'$ implies that there is a world $q_1' \in W$ with $q_1 \sim_B^D q_1'$ such that $(q_1', q_2') \in H$.

Intuitively, $x \leftrightarroweq_a^{\varepsilon} y$ means that, at worlds x and y, the agent a thinks the same possible, and accordingly has the same knowledge. The generalisation $x \leftrightarroweq_A^{\varepsilon} y$ to a coalition A means that at worlds x and y, the agents in A have the same common- and distributed knowledge.

EXAMPLE 2.9. (COMBINED KNOWLEDGE AND BISIMILARITY). Figure 2.2 illustrates a relation H between worlds of two epistemic structures for the agents a and b. The dashed and dotted lines correspond to a's and b's epistemic accessibility relation $\sim_a$ and $\sim_b$, respectively (omitting the reflexive edges). It is readily checked that H is an a- and b-epistemic bisimulation. Notice, however, that the worlds connected by H are not $\{a, b\}$-epistemic bisimilar. The equivalence class of $\sim_{\{a,b\}}^D$ for distributed knowledge in the right structure contains two worlds that satisfy different propositions, namely blackdot and whitedot, respectively. The problem is that the equivalence classes of $\sim_{\{a,b\}}^D$ in the left structure each lack a world to match those propositions. Hence, either condition (v) or (vi) of Definition 2.8 is violated.

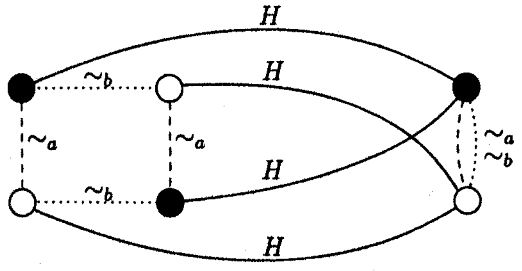


FIGURE 2.2. Bisimilar worlds in two epistemic structures.

Intuitively, at each pair of worlds connected via H, the agent a (agent b) thinks the same possible, and accordingly has the same knowledge. But, when considering the combined knowledge of both agents, they do not have the same distributed knowledge at both worlds. This difference in the combined knowledge of a and b is reflected by the fact that different S5-formulas are satisfied at the worlds connected via H: all worlds of the left structure satisfy either $D_{\{a,b\}}$blackdot or $D_{\{a,b\}}$whitedot, while the worlds of the right structure satisfy neither of these formulas. $\dashv$

The following theorem shows a logical characterisation of epistemic bisimulation. For any set A of agents, we denote with A-S5CD the fragment of S5CD with formulas using only the epistemic operators K_a E_B, C_B and D_B where a ranges over the agents in A, and B ranges over the subsets of A. We denote A-S5D accordingly.

THEOREM 2.10. *Let $\mathfrak{M} = \langle \Pi, \Sigma, W, \{\sim_a\}_{a\in\Sigma}, \pi\rangle$ and $\mathfrak{M}' = \langle \Pi, \Sigma, W', \{\sim'_a\}_{a\in\Sigma}, \pi'\rangle$ be two finite epistemic structures for a finite set Σ of agents, $x \in W$ and $y \in W'$ two worlds, and $A \subseteq \Sigma$ a coalition. The following are equivalent:*

(a) $x \leftrightarrows^{\epsilon}_{A} y$;
(b) *x and y satisfy the same formulas of A-S5D;*
(c) *x and y satisfy the same formulas of A-S5CD.*

PROOF. The proof is along the lines of a proof for a similar theorem, Theorem 2.15 in Section 2.3.5, for CTL [44]. Suppose $\mathfrak{M}$, $\mathfrak{M}'$, x, y and A are as in the theorem. The direction from (c) to (b) is obvious since A-S5D is a fragment of A-S5CD. In the following, the directions from (b) to (a) and from (a) to (c) will be shown, in this order.

"(b) $\Rightarrow$ (a)": Suppose x and y satisfy the same formulas of A-S5D. Define a binary relation $H \subseteq W \times W'$ as follows:

$$H = \{(q, q') \mid \mathfrak{M}, q \models \varphi \text{ iff } \mathfrak{M}', q' \models \varphi, \text{ for all } A\text{-S5D-formulas } \varphi\}.$$

In the following, H is shown to be an A-epistemic bisimulation between $\mathfrak{M}$ and $\mathfrak{M}'$, i.e., that H satisfies the conditions (iv), (v) and (vi) in Definition 2.8 of A-epistemic bisimulation. First, consider condition (iv). Let $a \in A$ be an agent. We need to check that H is an a-epistemic bisimulation, i.e., that the conditions (i) to (iii) in Definition 2.8 are satisfied. To this end, let $(q, q') \in H$.

- ad (i). By definition of H, it follows $\pi(q) = \pi'(q')$.
- ad (ii). Let $\mathfrak{M}'' = \langle \Pi, \Sigma, W'', \{\sim''_a\}_{a\in\Sigma}, \pi''\rangle$ be the disjoint union of $\mathfrak{M}$ and $\mathfrak{M}'$ where $W'' = W \uplus W'$, $\sim''_a = \sim_a \uplus \sim'_a$ for all $a \in \Sigma$, and $\pi''(p) = \pi(p) \uplus \pi'(p)$ for all $p \in \Pi$. Let $\sim$ be a binary relation on W'' such that $s \sim s'$ iff s and s' satisfy the same A-S5D-formulas in $\mathfrak{M}''$. Note that $\sim$ is an equivalence relation. Denote with $W''|_\sim$ the set of equivalence classes induced by $\sim$, and with $[s]$ the equivalence class $\{s' \in W'' \mid s \sim s'\} \subseteq W''|_\sim$ of world s. Clearly, for all sets $[s], [s'] \in W''|_\sim$ with $[s] \neq [s']$, there is an A-S5D-formula φ such that $\mathfrak{M}'', s \models \varphi$ and $\mathfrak{M}'', s' \not\models \varphi$. Arbitrarily choose such a formula φ and set

$\varphi_{[s],[s']} = \varphi$. For all $[s] \in W''|_{\sim}$, let

$$\varphi_{[s]} = \bigwedge_{[s'] \in W''|_{\sim}, [s] \neq [s']} \varphi_{[s],[s']}.$$

Note that W'' and thus $\varphi_{[s]}$ are finite by the finiteness of $\mathfrak{M}$ and $\mathfrak{M}'$. Let $t \in W$ be a world such that $q \sim_a t$ and $\psi = \neg K_a \neg \varphi_{[t]}$. Clearly, ψ is an A-S5D-formula, and we have that $\mathfrak{M}, q \models \psi$. By definition of H, it follows from $(q, q') \in H$ that $\mathfrak{M}', q' \models \psi$. That is, there is a world $t' \in W'$ with $q' \sim'_a t'$ such that $\mathfrak{M}', t' \models \varphi_{[t]}$. For showing (ii), it remains to show that $(t, t') \in H$. By definition of $\varphi_{[t]}$, it follows that $[t] = [t']$, i.e., t and t' satisfy the same A-S5D-formulas. Hence, $(t, t') \in H$ by definition of H.

- ad (iii). This can be shown similarly to condition (ii).

Second, we consider condition (v) only; (vi) is similar. Let $B \subseteq A$ be a coalition, and $q_1, q_2 \in W$ and $q'_1 \in W'$ worlds with $(q_1, q'_1) \in H$ and $q_1 \sim_B^D q_2$. Notice that $\mathfrak{M}, q_2 \models \varphi_{[q_2]}$, where the formula $\varphi_{[q_2]}$ is as in (ii) above. Let $\psi = \neg D_B \neg \varphi_{[q_2]}$. Clearly, ψ is an A-S5D-formula, and we have that $\mathfrak{M}, q_1 \models \psi$. From $(q_1, q'_1) \in H$, it follows by definition of H that $\mathfrak{M}', q'_1 \models \psi$. That is, there is a world $q'_2 \in W'$ with $q'_1 \sim'^D_B q'_2$ such that $\mathfrak{M}', q'_2 \models \varphi_{[q_2]}$. It follows by definition of $\varphi_{[q_2]}$ that $[q_2] = [q'_2]$. Hence, $(q_2, q'_2) \in H$ by definition of H.

"(a) $\Rightarrow$ (c)": Suppose $x \leftrightarroweq^{\epsilon}_A y$, i.e., there is an A-epistemic bisimulation H between $\mathfrak{M}$ and $\mathfrak{M}'$ with $(x, y) \in H$. In the following, we show that

$$\mathfrak{M}, q \models \varphi \quad \text{iff} \quad \mathfrak{M}', q' \models \varphi,$$

for all worlds $q \in W$ and $q' \in W'$, and all A-S5CD-formulas φ. The proof is by induction on the structure of φ. Let $q \in W$ and $q' \in W'$ with $(q, q') \in H$. The only interesting cases are the induction steps for common- and distributed knowledge; the other cases are left to the reader. Suppose the induction hypothesis holds for S5CD-formula φ'. Then:

- $\varphi = C_B \varphi'$, for some $B \subseteq A$. We show the contrapositive of the direction from left to right; the other direction is similar and left to the reader. Suppose $\mathfrak{M}', q' \not\models C_B \varphi'$, i.e., $\mathfrak{M}', t' \models \neg \varphi'$ for some world t' with $q' \sim'^C_B t'$. By definition of $\sim^C_B$, there is a sequence $t'_0 \dots t'_m \in W'^*$, $m \geq 0$, of worlds and a sequence $b_1 \dots b_m \in B^*$ of agents such that $q' = t'_0 \sim'_{b_1} t'_1 \sim'_{b_2} \cdots \sim'_{b_m} t'_m = t'$. We show that there is a sequence $t_0 \dots t_m \in W^*$ of worlds such that $t_0 = q$ and $(t_i, t'_i) \in H$ and $t_{i-1} \sim_{b_i} t_i$, for all $i \leq m$. We show this by induction on m. For $m = 0$, take $t_0 = q$. Then $(t_0, t'_0) \in H$ since $(q, q') \in H$ and $t'_0 = q'$. Consider $m \to m + 1$. By definition of H, it follows from $(t_m, t'_m) \in H$ and $t'_m \sim'_{b_{m+1}} t'_{m+1}$ by (iv) and (iii) in Definition 2.8 of A-epistemic bisimulation that there is a world $s_{m+1} \in W$ such that $t_m \sim_{b_{m+1}} s_{m+1}$. Set $t_{m+1} = s_{m+1}$, which finishes the nested induction. We conclude that $q \sim^C_B t_m$ and, by the induction hypothesis, $\mathfrak{M}, t_m \models \neg \varphi'$. Hence, $\mathfrak{M}, q \not\models C_B \varphi'$

- $\varphi = D_B\varphi'$, for some $B \subseteq A$. We show the contrapositive of the direction from left to right; the other direction is similar. Suppose $\mathfrak{M}', q' \not\models D_B\varphi'$, i.e., $\mathfrak{M}', t' \models \neg\varphi'$ for some world t' with $q' \sim'^{D}_{B} t'$. By definition of H, it follows by (vi) in Definition 2.8 of A-epistemic bisimulation that there is a world $t \in W$ with $q \sim^{D}_{B} t$ such that $(t, t') \in H$. The induction hypothesis yields $\mathfrak{M}, t \models \neg\varphi'$. Hence, $\mathfrak{M}, q \not\models D_B\varphi'$.

□

2.3. Temporal Logic

Another application of modal logic is to reason about time. This was first suggested by Prior in 1957 [**203, 204, 205**] who aimed to formalise with his *tense logic* temporal statements of natural language. Tense logic provides two basic box modalities G and H, where G stands for 'always in the future' and H for 'always in the past'. In the 1970s, Burstall [**46**] suggested temporal logic for reasoning about programs. This was followed by the work of Pratt [**202**], who introduced Propositional Dynamic Logic (PDL), and Pnueli [**200**], who used temporal logic to reason about concurrent programs. Dynamic logic received much attention in computer science; see, e.g., Harel [**106**]. These early works had much influence on a variety of subfields of computer science including databases, specification and verification, temporal knowledge representation, etc. In 1968, Kamp [**134**] extended Prior's basic tense logic with two new binary temporal operators U ('until') and S ('since'). Kamp proved that since and until cannot be defined in terms of the modalities H and G and, moreover, that the expressive power of the since/until logic equals those of first-order logic over Dedekind complete strict total orders (such as $\langle\mathbb{R}, <\rangle$). Later in 1980, Gabbay, Pnueli, Shelah and Stavi [**89**] pointed out that safety properties can be expressed with the until operator. Safety properties are important properties when reasoning about the behaviour of programs. For a survey on temporal logics, see, e.g., Emerson [**69**] and Clarke, Grumberg and Peled [**54**].

Temporal logic can be classified according to aspects like propositional versus first-order, branching versus linear time and discrete versus continuous time. Various temporal logics have been introduced, mainly Propositional Linear Temporal Logic (PLTL), First-order Linear Temporal Logic (FOLTL), Computational Tree Logic (CTL), its extension CTL* and the modal μ-Calculus. Models for time are usually irreflexive and transitive. Often, time is modelled as discrete and linear which makes the natural numbers a preferred model. But also the rational and real numbers have been considered which respectively form dense and continuous linear-time models. Models for branching-time are usually taken to be tree-like structures, in which, at each time point, the future may appear to be branching while the past is linear. The following are the commonly used modalities: the linear temporal operators G ('always'), F ('sometime'), X ('nexttime'), $\mathcal{U}$ ('until'), B or S ('since'), $\overset{\infty}{\mathsf{F}}$ ('infinitely often'), $\overset{\infty}{\mathsf{G}}$ ('almost everywhere') and, in the branching time versions, the path quantifiers A ('for all futures') and E ('for some future'). The operators $\overset{\infty}{\mathsf{F}}$ and $\overset{\infty}{\mathsf{G}}$ express fairness properties [89].

2.3.1. Computational Tree Logic. In 1981, Clarke and Emerson [56] introduced the logic for discrete branching time: Computational Tree Logic (CTL). Later in the 1980s, Emerson and Halpern [73] introduced its extensions CTL$^+$ and CTL*. The propositional μ-Calculus was introduced by Scott and De Bakker [222] as a formalism for specifying and reasoning about concurrent programs. The μ-Calculus attracted much attention and was further developed; see, e.g., Kozen [140], Emerson and Clarke [71] and Stirling [230]. The modal μ-Calculus is the basic tense logic extended with least and greatest fixpoint operators. These operators are expressive enough to encode many temporal logics into the μ-Calculus such as LTL, CTL and CTL*.

We now define the syntax of these four logics. Notice that we choose to define CTL$^+$ and CTL* using a distinction between state- and path formulas, but these languages can also be defined without that distinction.

DEFINITION 2.11. (CTL, CTL$^+$, CTL* SYNTAX). Let Π be a countably infinite set of propositional variables. The logics CTL, CTL$^+$ and CTL* are defined using the following BNF specifications, where p ranges over propositional variables in Π. The set of CTL-formulas φ is defined as:

$$\varphi \ ::= \ p \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{E}\bigcirc\varphi \mid \mathsf{E}(\varphi\,\mathcal{U}\,\varphi) \mid \mathsf{A}(\varphi\,\mathcal{U}\,\varphi)$$

The set of CTL$^+$-formulas is the set of CTL$^+$-state formulas ψ which are simultaneously defined together with CTL$^+$-path formulas ϑ as:

$$\begin{array}{lcl} \psi & ::= & p \mid \psi \vee \psi \mid \neg\psi \mid \mathsf{E}\vartheta \\ \vartheta & ::= & \neg\vartheta \mid \vartheta \vee \vartheta \mid \bigcirc\psi \mid \psi\,\mathcal{U}\,\psi \end{array}$$

The set of CTL*-formulas is the set of CTL*-state formulas θ which are simultaneously defined together with CTL*-path formulas χ as:

$$\begin{array}{lcl} \theta & ::= & p \mid \theta \vee \theta \mid \neg\theta \mid \mathsf{E}\chi \\ \chi & ::= & \theta \mid \chi \vee \chi \mid \neg\chi \mid \bigcirc\chi \mid \chi\,\mathcal{U}\,\chi \end{array}$$

The set of μ-Calculus-formulas π is defined as:

$$\pi \ ::= \ p \mid \neg p \mid \pi \vee \pi \mid \pi \wedge \pi \mid \Box\pi \mid \Diamond\pi \mid \mu p\,\pi \mid \nu p\,\pi$$

where in $\mu p\,\pi$ and $\nu p\,\pi$ the propositional variable p occurs only positively (i.e., $\neg p$ does not occur in π).

The abbreviations $\top$, $\bot$, $\wedge$, $\rightarrow$ and $\leftrightarrow$ are defined as usual. The temporal operator $\bigcirc$ stands for 'next-time' and was originally denoted by X. Moreover, we can define the following abbreviations for some temporal operators:

- $\mathsf{E}\Diamond\varphi = \mathsf{E}(\top\,\mathcal{U}\,\varphi)$;
- $\mathsf{A}\Box\varphi = \neg\mathsf{E}\Diamond\neg\varphi$;

- $\mathsf{A}\Diamond\varphi = \mathsf{A}(\top\,\mathcal{U}\,\varphi)$;
- $\mathsf{E}\Box\varphi = \neg\mathsf{A}\Diamond\neg\varphi$;
- $\mathsf{A}\bigcirc\varphi = \neg\mathsf{E}\bigcirc\neg\varphi$.

In CTL$^+$ and CTL*, we can additionally define the abbreviation $\mathsf{A}(\varphi\,\mathcal{U}\,\psi) = \neg\mathsf{E}\neg(\varphi\,\mathcal{U}\,\psi)$. Notice that CTL* allows for Boolean combinations and nesting of temporal operators inside a path quantifier. CTL$^+$ is a fragment of CTL* that disallows nesting of temporal operators inside a path quantifier. CTL is a fragment of CTL$^+$ that additionally disallows Boolean combinations of temporal operators inside a path quantifier.

Formulas of the CTL variants are interpreted over Kripke structures $\mathfrak{M} = \langle W, R, \pi\rangle$ consisting of a set of states W, a binary relation $R \subseteq W \times W$, and a valuation π mapping every atomic proposition p to a subset $\pi(p)$ of W. W.l.o.g., we assume that the graph of $\mathfrak{M}$ is a tree, since any structure can be unwound into a tree. Moreover, we assume that for every state, there is an R-successor. Given a state $w \in W$, a *w-fullpath* is an infinite sequence $u_0u_1\cdots \in W^\omega$ of states such that $u_0 = w$ and $(u_i, u_{i+1}) \in R$ for all positions $i \geq 0$.

We only present the semantics for CTL. The semantics for CTL$^+$ and CTL* are similar (cf. [**69**]) and fairly straightforward, given the distinction between path- and state formulas in the syntax. A full definition of the μ-Calculus can be found in [**70**].

DEFINITION 2.12. (CTL SEMANTICS). Given a Kripke structure $\mathfrak{M} = \langle \Pi, W, R, \pi\rangle$, the satisfaction relation '$\models$' is inductively defined as follows: For all worlds $w \in W$ and CTL-formulas φ and ψ, it holds that:

- $\mathfrak{M}, w \models p$ iff $w \in \pi(p)$, for all atomic propositions $p \in \Pi$;
- $\mathfrak{M}, w \models \neg\varphi$ iff $\mathfrak{M}, w \not\models \varphi$;
- $\mathfrak{M}, w \models \varphi \vee \psi$ iff $\mathfrak{M}, w \models \varphi$ or $\mathfrak{M}, w \models \psi$;
- $\mathfrak{M}, w \models \mathsf{E}\bigcirc\varphi$ iff there exists an R-successor v of w such that $\mathfrak{M}, v \models \varphi$;
- $\mathfrak{M}, w \models \mathsf{E}(\varphi\,\mathcal{U}\,\psi)$ iff there exists a w-fullpath $u_0u_1\cdots$ and a position $i \geq 0$ such that $\mathfrak{M}, u_i \models \psi$ and $\mathfrak{M}, u_j \models \varphi$ for all positions $j < i$.
- $\mathfrak{M}, w \models \mathsf{A}(\varphi\,\mathcal{U}\,\psi)$ iff for all w-fullpaths $u_0u_1\cdots$, there is a position $i \geq 0$ such that $\mathfrak{M}, u_i \models \varphi$ and $\mathfrak{M}, u_j \models \psi$ for all positions $j < i$.

If for some world w in some Kripke structure $\mathfrak{M}$ we have $\mathfrak{M}, w \models \varphi$, then the CTL-formula is *true* at w, and $\mathfrak{M}$ is called a *model* of φ. A CTL-formula is *satisfiable* if it has a model, and it is valid if it is true at all states in all Kripke structures.

The following valid CTL-formula shows the well-known fact that the operator combination $\mathsf{A}\mathcal{U}$ can be expressed in terms of $\mathsf{A}\Diamond$ and $\mathsf{E}\mathcal{U}$:

$$\mathsf{A}(\varphi\,\mathcal{U}\,\psi) \leftrightarrow \mathsf{A}\Diamond\psi \wedge \neg\mathsf{E}((\neg\psi)\,\mathcal{U}(\neg\varphi \wedge \neg\psi)).$$

Conversely, however, the combination $\mathsf{E}\mathcal{U}$ cannot be expressed in CTL in terms of $\mathsf{E}\Diamond$ and $\mathsf{A}\mathcal{U}$, as Laroussinie [**150**] noted. It can be shown that CTL is strictly more expressive than its fragment only allowing for $\mathsf{E}\Diamond$, $\mathsf{A}\mathcal{U}$ and $\mathsf{E}\bigcirc$ while CTL is no more expressive than the fragment where only $\mathsf{A}\Diamond$, $\mathsf{E}\mathcal{U}$ and $\mathsf{E}\bigcirc$ are allowed [150]. For instance, the formula $\mathsf{E}(\varphi\,\mathcal{U}\,\psi)$ cannot be expressed in the former fragment.

In CTL, we can specify many important properties of concurrent processes. A *safety property* represents the warranty that in all reachable states, i.e., in all states on all paths, a certain property is satisfied. For instance, the safety property

$$\mathsf{A}\Box\neg(\mathsf{inCS}_1 \wedge \mathsf{inCS}_2)$$

where inCS_i means that the process is in critical section i ($i = 1, 2$), states that the process can never be in both critical sections. A *liveness property* is the assurance that a good state fulfilling a certain property must eventually be reached. For instance, the liveness property

$$\mathsf{A}\Diamond\mathsf{delivered}$$

states that at some point a project is delivered. Moreover, CTL can express the property where a response must eventually be given whenever a request is made:

$$\mathsf{A}\Box(\mathsf{request} \rightarrow \mathsf{A}\Diamond\mathsf{response}).$$

The next CTL-formula expresses the property that a system goes infinitely often through a state that fulfills a certain property, say, a checkpoint:

$$\mathsf{A}\Box\mathsf{A}\Diamond\mathsf{checkpoint}$$

This final property states that a state fulfilling a certain property is always reachable.

EXAMPLE 2.13. (COMMUNICATION PROTOCOL). We illustrate how to state process properties with CTL by means of simple communication protocol. The left hand side of Figure 2.3 depicts a self-explanatory state transition diagram of this protocol with four states. The right-hand side, presents an unravelling of this state transition diagram into an infinite tree. Both, the transition graph and its unravelling, can be interpreted as a

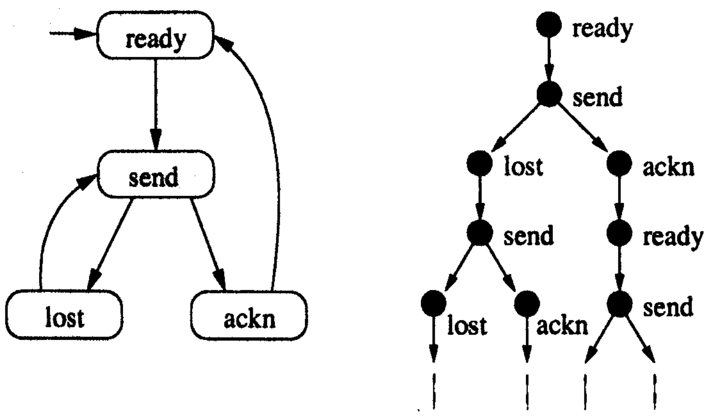


FIGURE 2.3. A state transition diagram and its unravelling into a tree.

Kripke structure with propositional variables ready, send, lost and ackn each labelling the states at which they hold. In this structure, we can describe the safety property that no state will be encountered where a message is lost and acknowledged at the same time with the CTL-formula $\mathsf{A}\Box(\neg\mathsf{lost} \lor \neg\mathsf{ackn})$ being true at all states. We have that the liveness property "the system enters infinitely often the send state" is satisfied, which can be described by $\mathsf{A}\Box\mathsf{A}\Diamond\mathsf{send}$ being satisfied everywhere. However, the liveness property that "the system enters infinitely often the ackn state" does not hold, i.e., $\mathsf{A}\Box\mathsf{A}\Diamond\mathsf{ackn}$ does not hold anywhere. Also it is not the case that "after the system was ready, it will eventually enter the ready state again": the formula $\mathsf{A}\Box(\mathsf{ready} \to \mathsf{A}\Diamond\mathsf{ready})$ does also not hold at any state. But we have that "the system can always reach the ready state from any other state" $\mathsf{A}\Box\mathsf{E}\Diamond\mathsf{ready}$ is true at all states in the system. ⊣

2.3.2. Axioms for CTL. Axiomatisations give us an insight into the expressivity of the considered temporal logic. In 1982, Emerson and Halpern [**72**] presented a complete axiomatisation of CTL; see Table 2.6 for a compact version ($\mathsf{A}\Box\varphi$ is used as an abbreviation for $\neg\mathsf{E}(\top\,\mathcal{U}\,\neg\varphi)$). A complete axiomatisation for CTL* turned out to be more difficult and was not obtained until 2001 by Reynolds [**212**]. Gabbay, Pnueli, Shelah and Stavi [**89**] provide several complete axiomatisations for reasoning about linear time, and Emerson and Halpern [**73**] for reasoning about branching time. A complete axiomatisation of the μ-Calculus was shown by Walukiewicz [**267**].

(A1)	Propositional tautologies				
(A2)	$\mathsf{E}\bigcirc\top \land \mathsf{A}\bigcirc\top$				
(A3)	$\mathsf{E}\bigcirc(\varphi \lor \psi) \leftrightarrow (\mathsf{E}\bigcirc\varphi \lor \mathsf{E}\bigcirc\psi)$				
(A4)	$\mathsf{A}\bigcirc\varphi \leftrightarrow \neg\mathsf{E}\bigcirc\neg\varphi$				
(A5)	$\mathsf{E}(\varphi\,\mathcal{U}\,\psi) \leftrightarrow \psi \lor (\varphi \land \mathsf{E}\bigcirc\mathsf{E}(\varphi\,\mathcal{U}\,\psi))$				
(A6)	$\mathsf{A}(\varphi\,\mathcal{U}\,\psi) \leftrightarrow \psi \lor (\varphi \land \mathsf{A}\bigcirc\mathsf{A}(\varphi\,\mathcal{U}\,\psi))$				
(A7)	$\mathsf{A}\Box(\vartheta \to (\neg\psi \land \mathsf{E}\bigcirc\vartheta)) \to (\vartheta \to \neg\mathsf{A}(\varphi\,\mathcal{U}\,\psi))$				
(A8)	$\mathsf{A}\Box(\vartheta \to (\neg\psi \land \mathsf{A}\bigcirc(\vartheta \lor \neg\mathsf{E}(\varphi\,\mathcal{U}\,\psi))) \to (\vartheta \to \neg\mathsf{E}(\varphi\,\mathcal{U}\,\psi))$				
(R1)	$\dfrac{\varphi,\ \varphi \to \psi}{\psi}$	(R2)	$\dfrac{\varphi}{\mathsf{A}\Box\varphi}$	(R3)	$\dfrac{\varphi \to \psi}{\mathsf{E}\bigcirc\varphi \to \mathsf{E}\bigcirc\psi}$

TABLE 2.6. An axiom system for CTL.

2.3.3. Model Checking. Model checking branching-time logics can be used for verifying high-level properties of finite-state reactive systems. For instance, a concurrent program can be represented as a finite state graph, which in turn can be viewed as a finite Kripke structure (*model*). Verifying the correctness of the program wrt. the desired property can be reduced to checking whether the formula describing the property holds in the model representing the program. Therefore the name *model checking* is used for such kind of verification methods. A survey on model checking temporal logics including CTL and CTL* can be found, e.g., in [**55, 54**]. Model checking for many CTL-variants can be done efficiently. There are several model checking implementations, e.g., SMV for CTL by McMillan [173].

Automata-theoretic techniques have been developed to reduce satisfiability and model checking problems to known automata-theoretic problems (in particular, the non-emptiness problem). Intuitively, each formula is associated with a finite automaton over infinite words [257] (in the linear-time case) or infinite trees [260] (in the branching-time case) that accept exactly all models satisfying the formula. Optimal decision procedures can be obtained by reducing satisfiability to the non-emptiness problem of automata. For branching-time model checking, Kupferman, Vardi and Wolper [145] suggested alternating tree automata as a uniform framework for satisfiability and model checking.

For an overview of model checking algorithms for the μ-Calculus, see, e.g., Emerson [70, 54].

2.3.4. Complexity. Table 2.7 summarises the complexity of both the model checking problem and the satisfiability problem for the four logics CTL, CTL$^+$, CTL* and μ-Calculus. The complexity of the model checking problem is measured in the size of

	Model Checking Problem	Satisfiability Problem
CTL	PTIME-complete [57]	EXPTIME-complete [72]
CTL$^+$	Δ_2^p-complete [151]	2-EXPTIME-complete [130]
CTL*	PSPACE-complete [77]	2-EXPTIME-complete [256, 76]
μ-Calculus	NP $\cap$ co-NP [269]	EXPTIME-complete [74]

TABLE 2.7. Complexities for branching-time logics.

the structure (model) and the length of the formula describing the specification, and the satisfiability problem is relative to the length of the input formula. The model checking problem for CTL can be solved efficiently, while for the other logics CTL$^+$, CTL* and μ-Calculus, it is more complex. Note that CTL$^+$'s model checking complexity Δ_2^p refers to the polynomial-time hierarchy; consult, e.g., [192], for a reference on complexity classes. CTL*-model checking can be polynomially reduced to LTL-model checking [77] which is PSPACE-complete [260]. Model checking LTL and also CTL* can be solved in time linear in the size of the structure and exponential time in the size of the formula. However, in practice, the length of the formula describing the specification is usually rather short, and the exponential growth of the complexity has little impact. If we consider the specification to be fixed and thus only the model as input of the problem, model checking complexity is called *model complexity*. CTL* has the same model complexity as CTL, which is linear in the size of the model.

For formulas of the μ-Calculus that do not contain alternations of least and greatest fixed points, model checking is in linear time as for CTL [58]. For μ-Calculus formulas with unbounded alternation, the precise model checking complexity is an open problem as yet. However, formulas with more than two alternations are rarely required in practice.

For comparison, the complexities of first-order logic are PSPACE-complete for model checking (see Chandra and Merlin [**50**]) and undecidable for satisfiability.

2.3.5. Expressivity. The expressivity of CTL was perceived to be rather weak [**73**, **144**]. For instance, it does not allow any Boolean combination nor nesting of temporal operators after a path quantifier. In particular, CTL lacks the ability to express fairness properties like 'infinitely often' and 'almost everywhere'. This has led to the introduction of more expressive CTL*-fragments with polynomial-time model checking complexity such as CTL^2 by Kupferman and Grumberg [**144**] allowing for two either nested or Boolean-connected temporal operators after a path quantifier, and the extensions ECTL and ECTL^+ of CTL with fairness properties by Emerson and Halpern [**73**].

The expressive power of the logics CTL^+ and CTL is equal [**72**], but translating CTL^+ into CTL yields an exponential blowup in formula length [**268**, **4**]. The μ-Calculus is more expressive than CTL*, and the alternation-free μ-Calculus is more expressive than CTL [**69**, **62**]. CTL can easily be translated into the μ-Calculus: $\mathsf{A}(\varphi\,\mathcal{U}\,\psi)$ translates into $\mu p(\psi \vee \Box(\varphi \wedge p))$ and $\mathsf{E}(\varphi\,\mathcal{U}\,\psi)$ into $\mu p(\psi \vee \Diamond(\varphi \wedge p))$. Translating CTL* into μ-Calculus is more difficult. The currently best known such translation involves a double exponential blowup in formula size; see Dam [**62**]. The expressivity of the modal μ-Calculus can be characterised as follows: μ-Calculus is as expressive as monadic second-order logic over trees [**207**, **75**]. The expressive power of the μ-Calculus is equivalent to the expressive power of alternating tree automata, as described by Niwiński [**188**]. Wilke [**269**] solves both the satisfiability and the model checking problem for the μ-Calculus by reductions on corresponding problems on alternating tree automata. Moreover, Janin and Walukiewicz [**128**] showed that the μ-Calculus is the bisimulation invariant fragment of monadic second-order logic, i.e., every monadic second-order logic formula that does not distinguish between bisimilar structures is equivalent to a formula of the μ-Calculus.

We now introduce *temporal bisimulation* as a notion of equivalence between Kripke structures.

DEFINITION 2.14. BISIMULATION Given two Kripke structures, $\mathfrak{M} = \langle W, R, \pi\rangle$ and $\mathfrak{M}' = \langle W', R', \pi'\rangle$, a binary relation $H \subseteq W \times W'$ is a *temporal bisimulation between* $\mathfrak{M}$ *and* $\mathfrak{M}'$ if for all states q_1 and q_2 with $(q_1, q_2) \in H$ the following three conditions hold:

(i) $\pi(q_1) = \pi'(q_2)$;

(ii) for every R-successor q_1' of q_1, there is an R'-successor q_2' of q_2 such that $(q_1', q_2') \in H$;

(iii) for every R'-successor q_2' of q_2, there is an R-successor q_1' of q_1 such that $(q_1', q_2') \in H$.

If H is a bisimulation and $(x,y) \in H$, then x and y are called *temporally bisimilar*, written $x \leftrightarroweq y$.

In 1988, Browne, Clarke and Grümberg [44] presented the following theorem, which gives a logical characterisation of temporal bisimulation. This theorem shows that two states can be distinguished by a CTL*-formula if, and only if, they can be distinguished by a CTL-formula if, and only if, they are bisimilar.

THEOREM 2.15. *Let $\mathfrak{M} = \langle W, R, \pi \rangle$ and $\mathfrak{M}' = \langle W', R', \pi' \rangle$ be two finite Kripke structures, and let $x \in W$ and $y \in W'$ be two states. The following are equivalent:*

(a) $x \leftrightarroweq y$;
(b) *x and y satisfy the same CTL-formulas;*
(c) *x and y satisfy the same CTL*-formulas.*

2.3.6. RTCTL. In this section, we investigate the quantitative extension of CTL: *Real-Time Computational Tree Logic* (RTCTL). It provides, in addition to the CTL-operators for qualitative temporal assertions, quantitative operators for expressing real-time constraints. RTCTL was introduced by Emerson, Mok, Sistla and Srinivasan [68] for reasoning about real-time environments. In particular, RTCTL makes it possible to formally specify correctness properties of temporal systems in which timing is essential. For instance, we can express bounded response requirement: The RTCTL-formula

$$\mathsf{A}\Box(\mathsf{request} \rightarrow \mathsf{A}\Diamond^{\leq k}\mathsf{response})$$

where k is any natural number, specifies that, whenever a request is made, a response must be given within k time steps, irrespective of how the system evolves after the request.

We propose a new method for polynomially reducing satisfiability in RTCTL (whose numerical parameters are coded in binary) to satisfiability in the same logic with numbers coded in unary. The essence of the reduction is to introduce new propositional variables that serve as the bits of a binary counter measuring distances. We reprove the original result of Emerson *et al.* [68] that RTCTL is in EXPTIME. A similar (but simpler) reduction can be used to show that the corresponding extension of linear-time logic LTL is in PSPACE. The reduction technique was introduced by Lutz, Walther and Wolter [167, 168] to show PSPACE-completeness of the since/until logic over the real line extended with metric operators 'sometime in at most n time units', n coded in binary, even without the finite variability assumption (which states that no propositional variable changes its truth-value infinitely many times in any finite interval).

For the sake of completeness, we first introduce the syntax and semantics of RTCTL.

DEFINITION 2.16. (RTCTL SYNTAX). Let Π be a countably infinite set of propositional variables. *RTCTL* formulas are built according to the following BNF specification, where p ranges over propositional variables in Π and k ranges over natural numbers that are coded in binary:

$$\begin{array}{rccccccccc} \varphi & ::= & p & \mid & \neg\varphi & \mid & \varphi \vee \varphi & \mid & \mathsf{E}\bigcirc\varphi & \mid \\ & & \mathsf{E}(\varphi\mathcal{U}\varphi) & \mid & \mathsf{A}(\varphi\mathcal{U}\varphi) & \mid & \mathsf{E}(\varphi\mathcal{U}^{\leq k}\varphi) & \mid & \mathsf{A}(\varphi\mathcal{U}^{\leq k}\varphi) & \end{array}$$

The abbreviations $\rightarrow$ and $\leftrightarrow$ are defined as usual. Moreover, we abbreviate $\mathsf{A}\bigcirc\varphi = \neg\mathsf{E}\bigcirc\neg\varphi$ and $\mathsf{A}\Box\varphi = \neg\mathsf{E}(\top\mathcal{U}\neg\varphi)$. Notice that a CTL-formula is an RTCTL-formula that does not use the metric version of the until operator.

RTCTL-formulas are evaluated over Kripke structures as CTL.

DEFINITION 2.17. (RTCTL SEMANTICS). We only give the semantics for the metric operators; the other operators are interpreted as in CTL. Given a Kripke structure $\mathfrak{M} = \langle \Pi, W, R, \pi \rangle$, the satisfaction relation '$\models$' is inductively defined as follows: for all worlds $w \in W$, all natural numbers $k \leq 0$, and all RTCTL-formulas φ and ψ, it holds that:

- $\mathfrak{M}, w \models \mathsf{E}(\varphi\mathcal{U}^{\leq k}\psi)$ iff there exists a w-fullpath $u_0u_1\cdots$ and a position $i \leq k$ such that $\mathfrak{M}, u_i \models \psi$ and $\mathfrak{M}, u_j \models \varphi$ for all positions $j < i$;
- $\mathfrak{M}, w \models \mathsf{A}(\varphi\mathcal{U}^{\leq k}\psi)$ iff for all w-fullpaths $u_0u_1\cdots$, there is a position $i \leq k$ such that $\mathfrak{M}, u_i \models \psi$ and $\mathfrak{M}, u_j \models \varphi$ for all positions $j < i$.

Our aim is to prove the following result:

THEOREM 2.18. *The satisfiability problem of RTCTL is* EXPTIME*-complete.*

The lower bound is an immediate consequence of the fact that CTL is a fragment of RTCTL, and the former is EXPTIME-hard. We prove a matching upper bound by a polynomial reduction to satisfiability of CTL, which is known to be in EXPTIME; see Section 2.3.4.

The main idea of the reduction is to replace subformulas $\mathsf{E}(\psi\mathcal{U}^{\leq k}\varphi)$ and $\mathsf{A}(\psi\mathcal{U}^{\leq k}\varphi)$ with a binary counter that is implemented using propositional variables to represent the bits. Notice that, since RTCTL models are not linear, we cannot simply increment the value of a distance-measuring counter when going to a predecessor state. Instead, the value at this predecessor state is determined by incrementing the least or greatest counter value of its successor nodes, depending on whether we are simulating a formula $\mathsf{E}(\psi\mathcal{U}^{\leq k}\varphi)$ or $\mathsf{A}(\psi\mathcal{U}^{\leq k}\varphi)$. For identifying the least and greatest counter value among the successors, we use a marking scheme based on additional propositional variables. Before we describe this marking in detail, let us fix some formalities.

Let φ be an RTCTL-formula whose satisfiability is to be decided. As an upper bound for the number of counter bits needed, let $n_c = \lceil \log_2(k+1) \rceil$ where k is the largest natural number occurring as a parameter to an until operator in φ. For simplicity, we assume w.l.o.g. that φ contains at least one subformula of the form $\mathsf{E}(\psi\mathcal{U}^{\leq k}\varphi')$ and at least one subformula of the form $\mathsf{A}(\psi\mathcal{U}^{\leq k}\varphi')$. Now, let $\chi_0, \ldots, \chi_{\ell'}$ be an enumeration of all of φ's subformulas of the form $\mathsf{E}(\psi\mathcal{U}^{\leq k}\varphi')$, and let $\chi_{\ell'+1}, \ldots, \chi_\ell$ be an enumeration of all subformulas of φ of the form $\mathsf{A}(\psi\mathcal{U}^{\leq k}\varphi')$. If $\chi_i = Q(\psi\mathcal{U}^{\leq k}\varphi')$, $Q \in \{\mathsf{E}, \mathsf{A}\}$, for some $i \leq \ell$, we use ψ_i to denote ψ and φ_i to denote φ'. For the reduction, we use the following propositional variables:

- the bits of the i-th counter, $i \leq \ell$, are represented using propositional variables $c^i_{n_c-1}, \ldots, c^i_0$;
- to mark the bits of the i-th counter, $i \leq \ell$, we use propositional variables $m^i_{n_c-1}, \ldots, m^i_0$.

Intuitively, the marking scheme for finding the greatest counter value among the successors can be understood as follows: start marking bits of the counters in successor nodes by proceeding from the highest ($n_c - 1$-st) to the lowest (0-th) bit, using the following two rules for marking the i-th bit, $i < n_c$, of a successor s' of s:

(1) if, in s', all bits higher than i are marked and all successors of s whose $i+1$-st bit is marked agree on the value of the i-th bit, then mark the i-th bit of s';
(2) if, in s', all bits higher than i are marked and the successors of s whose $i+1$-st bit is marked do not agree on the value of the i-th bit, then mark the i-th bit of s' iff it is one.

The result of this marking is that only those successors of s have *all* marking bits set whose counter value is highest among all the successors of s. A corresponding marking scheme for finding the lowest value is obtained by changing the last part of the second rule to "iff it is zero". Recall that $\chi_0, \ldots, \chi_{\ell'}$ are existentially path-quantified while $\chi_{\ell'+1}, \ldots, \chi_\ell$ are universally quantified. The marking of the i-th counter, $i \leq \ell$, can be implemented using the following formula, where $(i \leq \ell')$ abbreviates $\top$ if $i \leq \ell'$, and $\bot$ otherwise:

$$\begin{aligned}\vartheta^i_1 \;:=\; \bigwedge_{t=0..n_c-1} \Big(& ((\mathsf{A}\bigcirc c^i_t \vee \mathsf{A}\bigcirc\neg c^i_t) \rightarrow \mathsf{A}\bigcirc(m^i_t \leftrightarrow \bigwedge_{t<j<n_c} m^i_j)) \wedge \\ & ((\mathsf{E}\bigcirc c^i_t \wedge \mathsf{E}\bigcirc\neg c^i_t \wedge (i \leq \ell')) \rightarrow \mathsf{A}\bigcirc(m^i_t \leftrightarrow (\neg c^i_t \wedge \bigwedge_{t<j<n_c} m^i_j))) \wedge \\ & ((\mathsf{E}\bigcirc c^i_t \wedge \mathsf{E}\bigcirc\neg c^i_t \wedge (i > \ell')) \rightarrow \mathsf{A}\bigcirc(m^i_t \leftrightarrow (c^i_t \wedge \bigwedge_{t<j<n_c} m^i_j))) \Big)\end{aligned}$$

To implement the counters, we introduce auxiliary formulas. For $1 \leq i \leq \ell$, let

- $(C_i = m)$ be a formula saying that, at the current point, the value of the i-th counter is m, for $0 \leq m < 2^{n_c}$;
- $(C_i \leq m)$ is a formula saying that, at the current point, the value of the i-th counter does not exceed m, for $0 \leq m < 2^{n_c}$.

There are exponentially many such formulas, but we will use only polynomially many of them in the reduction. We now inductively define a translation $(\cdot)^*$ of subformulas of φ to CTL-formulas.

$$\begin{array}{rcl}
(p)^* & := & p \\
(\neg\psi)^* & := & \neg\psi^* \\
(\psi_1 \wedge \psi_2)^* & := & \psi_1^* \wedge \psi_2^* \\
(\mathsf{E}\bigcirc\psi)^* & := & \mathsf{E}\bigcirc\psi^* \\
(\mathsf{E}(\psi_1 \mathcal{U} \psi_2))^* & := & \mathsf{E}(\psi_1^* \mathcal{U} \psi_2^*) \\
(\mathsf{A}(\psi_1 \mathcal{U} \psi_2))^* & := & \mathsf{A}(\psi_1^* \mathcal{U} \psi_2^*) \\
(\mathsf{E}(\psi_1 \mathcal{U}^{\leq k} \psi_2))^* & := & (C_i \leq k) \text{ if } \chi_i = \mathsf{E}(\psi_1 \mathcal{U}^{\leq k} \psi_2) \\
(\mathsf{A}(\psi_1 \mathcal{U}^{\leq k} \psi_2))^* & := & (C_i \leq k) \text{ if } \chi_i = \mathsf{A}(\psi_1 \mathcal{U}^{\leq k} \psi_2)
\end{array}$$

It remains to properly update the counters, which is done by the following formulas, for $i \leq \ell$, set

$$\begin{array}{rcl}
\vartheta_2^i & := & (C_i = 0) \leftrightarrow \varphi_i^* \\
\lambda & := & \neg\psi_i^* \vee \\
& & \big((i \leq \ell') \wedge \mathsf{A}\bigcirc(C_i = 2^{n_c} - 1)\big) \vee \\
& & \big((i > \ell') \wedge \mathsf{E}\bigcirc(C_i = 2^{n_c} - 1)\big) \\
\vartheta_3^i & := & \big((\neg\varphi_i^* \wedge \lambda) \rightarrow (C_i = 2^{n_c} - 1)\big) \wedge \\
& & \Big((\neg\varphi_i^* \wedge \neg\lambda) \rightarrow \bigvee\limits_{t=0..n_c-1} \Big(c_t^i \wedge \mathsf{E}\bigcirc(m_t^i \wedge \neg c_t^i) \wedge \\
& & \qquad \bigwedge\limits_{\ell=0..t-1} (\neg c_\ell^i \wedge \mathsf{E}\bigcirc(m_\ell^i \wedge c_\ell^i)) \wedge \\
& & \qquad \bigwedge\limits_{t<\ell<n_c} (c_\ell^i \leftrightarrow \mathsf{E}\bigcirc(m_\ell^i \wedge c_\ell^i)) \Big)\Big)
\end{array}$$

Intuitively, ϑ_2^i initializes the counter, and ϑ_3^i ensures that the counter is incremented correctly when travelling to a predecessor state. The value $2^{n_c} - 1$ of the i-th counter is used to express that, on respectively all paths (for χ_i being existentially path-quantified) or some path (universal path quantification), the formula φ_i is too far to be of any relevance. Moreover, the value $2^{n_c} - 1$ is also used to indicate that ψ_i does not hold on some point on the way to the next φ_i occurrence.

The following lemma finishes the reduction.

LEMMA 2.19. *φ is satisfiable iff $\varphi^* \wedge \bigwedge\limits_{i \leq \ell} \mathsf{A}\Box(\vartheta_1^i \wedge \vartheta_2^i \wedge \vartheta_3^i)$ is satisfiable.*

The lemma can be proven similarly to the proof of the reduction of the metric since/until logic over the real line to its non-metric counterpart; see Lutz, Walther and Wolter [**167, 168**].

2.4. Strategic Logic

The basic modal logic seems not suitable for describing how several agents are involved in system transitions. This is because a Kripke structure merely models all possible state transitions of a system, and it does not account for how possible simultaneous actions of agents can determine different transitions. In 2001, Pauly [**194**] introduced Coalition Logic, an extension of basic modal logic for reasoning about the strategic abilities of coalitions in multi-agent systems. The semantic structures of Coalition Logic [**195, 196**] correspond to extensive games which makes it possible to describe how single agents or *coalitions* (sets of agents) influence system transitions. The Coalition Logic expression $[C]\varphi$ states that the agents in a coalition C have a joint strategy for ensuring φ in the next state. A drawback of Coalition Logic is its weak expressive power over time: formulas can only use the next-time operator to describe future states since other temporal operators such as until ('$\mathcal{U}$') are missing. In 1997, Alur, Henzinger and Kupferman [**13**] introduced Alternating-time Temporal Logic (ATL). ATL is a logic of *strategic ability*, intended to support reasoning about the abilities of agents and coalitions of agents in *open systems*, i.e., game-like multi-agent systems. ATL can be seen as an extension of Coalition logic with the temporal operators box ('$\Box$') and until ('$\mathcal{U}$') from CTL. Furthermore, from a language point of view, ATL generalises CTL. While in CTL, one is essentially restricted to stating that some property is either inevitable or possible, in ATL, one can also express *adversarial* properties, such as "agents 1 and 2 can ensure that, no matter what the other agents do, the system will not enter an invalid state" (written: $\langle\langle 1,2\rangle\rangle\Box valid$). More precisely, ATL offers a more refined path quantification: while CTL provides path quantifiers 'for all computation paths' ('A') and 'for some path' ('E'), ATL provides path quantifiers, which are parameterized with a coalition of agents, allowing for quantification over subsets of the available paths. From the point of view of semantics, ATL is based on finite structures (called *Alternating Transition Systems* – ATSs) that emphasise the game-like nature of distributed computing, thus reflecting current opinion on the semantics of multi-process systems. More precisely, an ATL-formula of the form $\langle\langle C\rangle\rangle\Phi$ corresponds to a two-player game between a coalition C and its environment (a coalition consisting of all agents outside of C) played on a finite state space. Such a game produces an infinite sequence of states, where, in each round, the next state is determined by both a choice of the coalition and a choice of its environment at the current state. A coalitional choice is composed of the choices of the agents that are members of that coalition. And finally, from the verification point of view, the model checking problem for ATL is no more complex than for its counterpart CTL.

2.4.1. ATL. We now define the syntax of Alternating-time Temporal Logics ATL and ATL* together with the syntax of the Alternating-time μ-Calculus (AMC); cf. [**13, 14, 15**].

DEFINITION 2.20. (ATL SYNTAX). Let Π be a countably infinite set of atomic propositions and Σ a countable infinite set of agents. A *coalition* is a finite set $C \subset \Sigma$ of agents. The logics ATL, ATL* and AMC are defined using the following BNF specifications, where p ranges over atomic propositions in Π. The set of ATL-formulas φ is defined as:

$$\varphi \quad ::= \quad p \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle\!\langle C \rangle\!\rangle \bigcirc \varphi \mid \langle\!\langle C \rangle\!\rangle \Box \varphi \mid \langle\!\langle C \rangle\!\rangle (\varphi \,\mathcal{U}\, \varphi)$$

The set of ATL*-formulas is the set of ATL* state formulas ψ which are simultaneously defined together with ATL* path formulas ϑ as:

$$\begin{array}{lcl} \psi & ::= & p \mid \neg\psi \mid \psi \vee \psi \mid \langle\!\langle C \rangle\!\rangle \vartheta \\ \vartheta & ::= & \psi \mid \neg\vartheta \mid \vartheta \vee \vartheta \mid \bigcirc\vartheta \mid \vartheta \,\mathcal{U}\, \vartheta \end{array}$$

The set of AMC-formulas ξ is defined as:

$$\xi \quad ::= \quad p \mid \neg\xi \mid \xi \vee \xi \mid \langle\!\langle C \rangle\!\rangle \bigcirc \xi \mid \mu p.\xi$$

The modality $\langle\!\langle\ \rangle\!\rangle$ is called a *path quantifier*, $\bigcirc$ ('next'), $\Box$ ('always') and $\mathcal{U}$ ('until') are *temporal operators*. The symbols $\top$ and $\bot$, and the operators $\wedge$, $\rightarrow$ and $\leftrightarrow$ are defined as usual. The operator $\langle\!\langle C \rangle\!\rangle \Diamond \varphi$ is defined as $\langle\!\langle C \rangle\!\rangle \top \,\mathcal{U}\, \varphi$. Observe that, in ATL, the operator $\langle\!\langle C \rangle\!\rangle \Box$ cannot be expressed in terms of $\langle\!\langle C \rangle\!\rangle\, \mathcal{U}$ since formulas of the form $\langle\!\langle C \rangle\!\rangle \neg(\top \,\mathcal{U} \neg\varphi)$, which are equivalent to $\langle\!\langle C \rangle\!\rangle \Box \varphi$, are not acceptable according to the ATL syntax. Clearly, ATL is a proper fragment of ATL*.

The following example formula illustrates why ATL is suitable for modelling MAS; see [15]. Suppose we would like to specify the property that a process or transaction denoted by, say a, does not encounter a deadlock, which prevents a from both reading and writing the memory. Using ATL, we can express this property as

$$\langle\!\langle \emptyset \rangle\!\rangle \Box (\langle\!\langle a \rangle\!\rangle \Diamond \mathsf{read} \wedge \langle\!\langle a \rangle\!\rangle \Diamond \mathsf{write}).$$

This formula states that process a can always access the memory, no matter what the other processes do that may be competing with a for resources.

Several versions of the semantics of ATL and the related Coalition Logic have been presented in the literature: concurrent game structures, alternating transition systems and coalition effectivity models. All of these have been shown to be equivalent by Goranko and Jamroga [96]. Initially, Alur *et al.* [13] defined ATL over turn-based game structures, where each transition is determined by some single agent. Later in [14], they introduce alternating transition systems, where the choices of all agents determine the system transitions. Yet another variant of the ATL semantics was introduced in [15] with concurrent game structures, where each choice is represented by a label, to improve understandability of the logic and clarity of the presentation. In this work, however, we choose to work with *alternating transition systems* as introduced in [14].

DEFINITION 2.21. (ATS). An *alternating transition system (ATS)* for a set of agents Σ is a tuple $\mathcal{S} = \langle \Pi, \Sigma, Q, \pi, \delta \rangle$ with $n \geq 1$ where

- $\Pi \subset \mathbf{\Pi}$ is a finite, non-empty set of *atomic propositions*;
- $\Sigma = \{a_1, \ldots, a_n\} \subset \mathbf{\Sigma}$ is a (finite) set of n *agents*;
- Q is a finite, non-empty set of *states*;
- $\pi : Q \to 2^{\Pi}$ is a *valuation function* which assigns to every state a set of propositions which are true there; and
- $\delta : Q \times \Sigma \to 2^{2^Q}$ is a *transition function* which maps a state $q \in Q$ and an agent $a \in \Sigma$ to a set $\delta(q,a)$ of *choices* available to a at q such that the following condition is satisfied: for all states $q \in Q$ and all sets $Q_{a_1}, \ldots, Q_{a_n}$ of *choices* $Q_{a_i} \in \delta(q, a_i)$, $1 \leq i \leq n$, the intersection $Q_{a_1} \cap \cdots \cap Q_{a_n}$ is a singleton set.

Observe that the system is completely determined when all the agents have made their choice, since then there is a unique successor state. Relaxing the condition on the transition function δ about singleton sets by allowing for arbitrary sets introduces non-determinism. Note that this non-determinism corresponds to having an additional unknown agent that is not contained in Σ. In open systems, such an additional agent can be seen as representing the environment. However, in the following, we will use the ATSs as defined above, i.e., we keep the condition about singleton sets. In Chapter 3, we define a variant of ATL's semantics that respects non-determinism by accounting for at least one additional agent in the structures that does not occur in any input formula.

Intuitively, $\delta(q,a)$ describes the *a-choices* available in q: when in state q, agent a chooses a set from $\delta(q,a)$ to ensure that the "next state" will be among those in the chosen set. It is natural to generalise this notion to *C-choices* for coalitions C: let $\mathcal{S} = \langle \Pi, \Sigma, Q, \pi, \delta \rangle$ be an ATS. For each state $q \in Q$ and each coalition $C \subseteq \Sigma$, set

$$\delta(q,C) := \begin{cases} \{Q_C \subseteq Q \mid Q_C = \bigcap_{a \in C} Q_a \text{ where for each } a \in C,\ Q_a \in \delta(q,a)\} & \text{if } C \neq \emptyset \\ \{\bigcup \delta(q,\Sigma)\} & \text{if } C = \emptyset \end{cases}.$$

When in state q, the coalition C may jointly choose a set from $\delta(q,C)$ to ensure that the next state is from this set. Since $\delta(q,C)$ is non-empty for all q and C, and $\delta(q,\Sigma)$ is a set of singleton sets, the states appearing in singleton sets in $\delta(q,\Sigma)$ are the *successors* of q, i.e., whatever choices the individual agents make, the next state of the system will be from $\bigcup \delta(q,\Sigma)$. This explains the definition of $\delta(q,\emptyset)$: the empty set of agents cannot influence the behaviour of the system, so the only choice that the empty coalition has is the set of all successors.

An infinite sequence $\lambda = q_0 q_1 q_2 \cdots \in Q^{\omega}$ of states is a *computation* if, for all positions $i \geq 0$, there is a choice $\{q_{i+1}\} \in \delta(q_i, \Sigma)$ (i.e., q_{i+1} is a successor of q_i). As a notational convention for any finite or infinite sequence $\lambda = \lambda_0 \lambda_1 \cdots$ and any $i \geq 0$, denote with $\lambda[i]$ the i-th component λ_i in λ and with $\lambda[0,i]$ the initial sequence $\lambda_0 \cdots \lambda_i$ of λ.

A *strategy* for an agent $a \in \Sigma$ is a mapping $\sigma_a : Q^+ \to 2^Q$ such that for all $\lambda \in Q^*$ and all $q \in Q$, $\sigma_a(\lambda \cdot q) \in \delta(q,a)$. Note that a strategy σ_a maps each finite sequence $\lambda \cdot q$

of states to a choice in $\delta(q, a)$ available to agent a at the state q. A *strategy* for agents in a coalition $C \subseteq \Sigma$ is a set of strategies $\sigma_C = \{\sigma_a \mid a \in C\}$, one for each agent in C.

The set $\mathsf{out}(q, \sigma_C)$ of *outcomes* of a strategy σ_C starting at a state $q \in Q$ is the set of all computations $\lambda = q_0 q_1 q_2 \cdots \in Q^\omega$ such that $q_0 = q$ and $q_{i+1} \in \bigcap_{\sigma_a \in \sigma_C} \sigma_a(\lambda[0, i])$ for all $i \geq 0$.

DEFINITION 2.22. (ATL SEMANTICS). Given an ATS $\mathcal{S} = \langle \Pi, \Sigma, Q, \pi, \delta \rangle$, the satisfaction relation $\models$ is inductively defined as follows: For all states q of $\mathcal{S}$, coalitions of agents $C \subseteq \Sigma$, agents $a \in \Sigma$, and ATL-formulas φ, φ_1, and φ_2, it holds that

- $\mathcal{S}, q \models p$ iff $p \in \pi(q)$ for all propositions $p \in \Pi$;
- $\mathcal{S}, q \models \neg\varphi$ iff $\mathcal{S}, q \not\models \varphi$;
- $\mathcal{S}, q \models \varphi_1 \vee \varphi_2$ iff $\mathcal{S}, q \models \varphi_1$ or $\mathcal{S}, q \models \varphi_2$;
- $\mathcal{S}, q \models \langle\!\langle C \rangle\!\rangle \bigcirc \varphi$ iff there is a strategy σ_C such that for all computations $\lambda \in \mathsf{out}(q, \sigma_C)$, it holds that $\mathcal{S}, \lambda[1] \models \varphi$;
- $\mathcal{S}, q \models \langle\!\langle C \rangle\!\rangle \Box \varphi$ iff there is a strategy σ_C such that for all computations $\lambda \in \mathsf{out}(q, \sigma_C)$, it holds that $\mathcal{S}, \lambda[i] \models \varphi$ for all positions $i \geq 0$;
- $\mathcal{S}, q \models \langle\!\langle C \rangle\!\rangle \varphi_1 \mathcal{U} \varphi_2$ iff there is a strategy σ_C such that for all computations $\lambda \in \mathsf{out}(q, \sigma_C)$, there is a position $i \geq 0$ such that $\mathcal{S}, \lambda[i] \models \varphi_2$ and $\mathcal{S}, \lambda[j] \models \varphi_1$ for all positions j with $0 \leq j < i$.

If for some state q of some ATS $\mathcal{S}$ it holds that $\mathcal{S}, q \models \varphi$, then the ATL-formula φ is *true* at q, and $\mathcal{S}$ is called a *model* of φ. An ATL-formula is *satisfiable* if it has a model, and it is *valid* if it is true at all states in any ATS.

Notice that there is an intimate relationship between CTL and ATL. Let the finite set Σ be the set of all agents in an ATS $\mathcal{S}$. On $\mathcal{S}$, we can then interpret CTL's existential path quantifier E in ATL as the expression $\langle\!\langle \Sigma \rangle\!\rangle$, while we can interpret CTL's universal path quantifier A in ATL as the expression $\langle\!\langle \emptyset \rangle\!\rangle$. Clearly, this translation only works if the set of agents Σ is finite and known in advance.

To better understand ATL and its limitations, consider the following example.

EXAMPLE 2.23. (DESCRIPTION OF COALITIONAL STRATEGIC ABILITY). This example illustrates how ATL and ATL* can be used to describe agents' abilities and composed abilities in a MAS with two agents. Figure 2.4 shows an ATS for two agents a and b. The arrows denote possible system transitions. The four lower states have each only one outgoing arrow, i.e., at these states, the next state of the system is already determined. The top-most state, however, has four outgoing arrows which means that the agents' choices can determine four possible system transitions. The round boxes labelled with either a or b denote the choices available to the agents a and b at the top-most state. As illustrated in Figure 2.4, the states each satisfy some of the three propositions $\mathsf{blackdot}$, $\mathsf{whitedot}$ and start. Suppose the top-most state is the current one. We have that the ATL-formula $\langle\!\langle a \rangle\!\rangle \bigcirc \mathsf{blackdot}$ is satisfied which states that agent a can enforce the next state to have the property $\mathsf{blackdot}$. The formula $\langle\!\langle a \rangle\!\rangle \Box \mathsf{blackdot}$ asserts

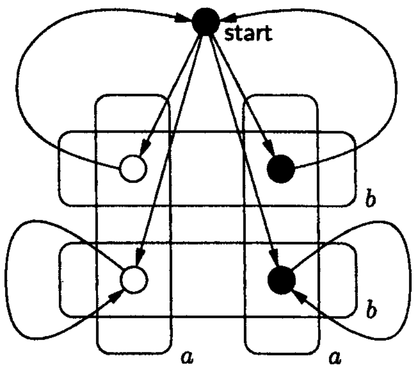


FIGURE 2.4. An ATS for two agents a and b.

controllability of the overall system by agent a wrt. the proposition blackdot. That is, it states that a can ensure that the property blackdot always holds in the system, no matter how the other components, here only agent b, behave. Also we can describe agent b's capability to ensure that the state satisfying start is reached (a) infinitely many times, or (b) finitely many times: both ATL*-formulas $\langle\langle b\rangle\rangle\Box\Diamond\mathsf{start}$ for the *liveness property* (a), and $\langle\langle b\rangle\rangle\Diamond\Box\neg\mathsf{start}$ for the property (b) are true at the current state. Using ATL*, we can describe a composition of the agents' abilities: the formula $\langle\langle a,b\rangle\rangle\Box(\mathsf{blackdot}\wedge\Diamond\mathsf{start})$ being true at the current state means that the coalition of a and b has the capability to ensure the system to always satisfy blackdot and to infinitely often reach the start state. The formula $\bigwedge_{i=a,b}\neg\langle\langle i\rangle\rangle\Diamond\Box\mathsf{whitedot}$ holding at the top state means that neither agent, a or b, is capable of enforcing the system to eventually enter and then to never leave the set of whitedot states, irrespective of what the other agent does. In other words, each agent can ensure that a blackdot state is always reachable: $\bigwedge_{i=a,b}\langle\langle i\rangle\rangle\Box\Diamond\mathsf{blackdot}$ is true at the top state. However, both agents together are capable of ensuring the system to enter infinitely many times a whitedot state and infinitely many times a blackdot state: the top state satisfies the ATL*-formula $\langle\langle a,b\rangle\rangle(\Box\Diamond\mathsf{whitedot}\wedge\Box\Diamond\mathsf{blackdot})$. Notice that this property cannot be expressed in ATL since it requires a conjunction of temporal properties inside a path quantifier which is not according to ATL syntax. To overcome this limitation of ATL's expressiveness, we introduce in Chapter 4 an extension of ATL with explicit strategies (ATLES) which allows us to fix a strategy for an agent and thus to fix the set of paths determined by a path quantifier. Then, by using the same strategies in another path quantifier, we can describe several temporal properties of these paths. $\dashv$

One interesting aspect of ATL is that its model checking problem subsumes a number of other computational problems, such as *program synthesis*, *module checking* and *controllability* [15, p.676]. However, the problem of *social procedure design* or *mechanism design* in ATL can be understood as a (constructive) *satisfiability checking* problem: given a specification of a social mechanism (such as a voting procedure [194]), expressed as an ATL-formula φ, the question is whether or not there exists a procedure that satisfies the specification φ; and if so, exhibit it. Since such a procedure corresponds to a model of φ, the design of a social procedure can be viewed as a proof of the satisfiability of φ. The design is constructive in the sense that an actual model is generated. In the next example, we discuss this in more detail.

EXAMPLE 2.24. (SOCIAL PROCEDURES). One area of interest is the use of ATL in the specification, verification, and synthesis of *social procedures* such as voting protocols [**194**]. Consider the following example (adapted from [**194**]).

> *Two agents, A and B, must choose between two outcomes, p and q. We want a mechanism that will allow them to choose, which will satisfy the following requirements. First, whatever happens, we definitely want an outcome to result – that is, we want either p or q to be selected. Second, we really do want the agents to be able to collectively choose an outcome. However, we do not want them to be able to bring about both outcomes simultaneously. Similarly, we do not want either agent to dominate: we want them both to have equal power.*

We can elegantly capture these requirements using ATL, as follows.

$$\langle\langle\emptyset\rangle\rangle\bigcirc(p \vee q) \tag{2.1}$$
$$(\langle\langle A,B\rangle\rangle\bigcirc p) \wedge (\langle\langle A,B\rangle\rangle\bigcirc q) \tag{2.2}$$
$$\neg\langle\langle A,B\rangle\rangle\bigcirc(p \wedge q) \tag{2.3}$$
$$(\neg\langle\langle A\rangle\rangle\bigcirc p) \wedge (\neg\langle\langle B\rangle\rangle\bigcirc p) \tag{2.4}$$
$$(\neg\langle\langle A\rangle\rangle\bigcirc q) \wedge (\neg\langle\langle B\rangle\rangle\bigcirc q) \tag{2.5}$$

The first requirement states that an outcome *must* result: this will happen inevitably, whatever the agents do. Requirement (2.2) states that the two agents can choose between the two outcomes: they have a collective strategy such that, if they follow this strategy, outcome x will occur, where x is either p or q. Requirement (2.3), however, says that the agents cannot choose *both* outcomes. Requirements (2.4) and (2.5) state that neither agent can bring about an outcome alone.

Now it is easy to see that there exists a voting protocol that satisfies these requirements. Consider the following mechanism, (from [**198**]), intended to permit the agents to select between the outcomes in accordance with these requirements.

> *The two agents vote on the outcomes, i.e., they each choose either p or q. If there is a consensus, then the consensus outcome is selected; if there is no consensus, (i.e., if the two agents vote differently), then an outcome p or q is selected non-deterministically.*

Notice that, given this simple mechanism, the agents really can collectively choose the outcome, by cooperating. If they do not cooperate, however, then an outcome is chosen for them.

It is similarly easy to see how ATL can be used, in this way, to specify much more complex voting protocols and other related social procedures. It is important to note that this example is chosen for pedagogic reasons, and realistic protocols will be rather more elaborate. One issue for which our complexity analysis becomes important is that often we have a considerably larger number of agents who participate in a protocol than just two. The example above can naturally be extended to any number n of agents. In particular, we would get a formula expressing the requirements for n agents.

Once we have a protocol, we can use an ATL model checker (such as MOCHA [17]), to automatically verify that our implementation of the requirements is correct. However, the problem of *synthesising* a protocol from such a specification corresponds to (constructively) checking the *satisfiability* of the specification—whence our interest in the satisfiability problem. $\dashv$

2.4.2. Axioms. In 2006, Goranko and van Drimmelen [98] presented a sound and complete axiom system for ATL_Σ; see Figure 2.8. ATL_Σ is the fragment of ATL that only allows for formulas whose agents are from the set Σ. This axiomatisation extends Pauly's [194] axiomatisation of Coalition Logic with axioms and rules for fixed point formulas characterising the temporal operators. Completeness is proven by constructing an infinite but bounded branching tree model, for each consistent formula. Since such tree models correspond to unfolded finite models, this also shows the finite model property for ATL_Σ.

For an axiom system of ATL (without fixing the set of agents), we refer to Section 4.1.2 in Chapter 4 where we present in Table 4.1 a complete axiom system for an extension of ATL with explicit strategies. This axiom system can also be used to derive valid ATL-formulas.

2.4.3. Model Checking. Alur, Henzinger and Kupferman [15] described the symbolic model checking algorithm in Figure 2.5. The function ATL–*eval*$(\cdots)$ computes, for a given formula ψ and an ATS $S = \langle \Pi, \Sigma, Q, \pi, \delta \rangle$, the *extension* $[\![\psi]\!]$ of ψ in S, where $[\![\psi]\!]$ is the set of states in S satisfying ψ. ATL$-$*eval*$(\cdots)$ uses the *pre-image* operator *Pre*, which maps a coalition A and a set Q' of states to the set $Pre(A, Q')$ containing the states at which A can enforce the next state of the system to lie in Q'. Formally, for all $A \subseteq \Sigma$ and all $Q' \subseteq Q$:

$$\begin{aligned} Pre(A, Q') \quad := \quad & \{q \in Q \mid \text{there is a choice } Q_C \in \delta(q, A) \\ & \qquad\qquad \text{such that } Q_C \subseteq Q'\}. \end{aligned}$$

The model checking algorithm is called *symbolic*, because the state space is symbolically represented by formulas, which is more succinct than an explicit representation by lists or tables. An implementation of model checking for ATL is, e.g., MOCHA by

(TAUT)	Propositional tautologies
($\perp$)	$\neg\langle\langle A\rangle\rangle\bigcirc\perp$
($\top$)	$\langle\langle A\rangle\rangle\bigcirc\top$
(Σ)	$\neg\langle\langle\emptyset\rangle\rangle\bigcirc\neg\varphi \to \langle\langle\Sigma\rangle\rangle\bigcirc\varphi$
(S)	$(\langle\langle A\rangle\rangle\bigcirc\varphi \wedge \langle\langle B\rangle\rangle\bigcirc\psi) \to \langle\langle A\cup B\rangle\rangle\bigcirc(\varphi\wedge\psi)$ where $A\cap B=\emptyset$
($\mathrm{FP}_\Box$)	$\langle\langle A\rangle\rangle\Box\varphi \leftrightarrow \varphi\wedge\langle\langle A\rangle\rangle\bigcirc\langle\langle A\rangle\rangle\Box\varphi$
($\mathrm{GFP}_\Box$)	$\langle\langle\emptyset\rangle\rangle\Box(\theta\to(\varphi\wedge\langle\langle A\rangle\rangle\bigcirc\theta)) \to \langle\langle\emptyset\rangle\rangle\Box(\theta\to\langle\langle A\rangle\rangle\Box\varphi)$
($\mathrm{FP}_\mathcal{U}$)	$\langle\langle A\rangle\rangle\psi\,\mathcal{U}\,\varphi \leftrightarrow \varphi\vee(\psi\wedge\langle\langle A\rangle\rangle\bigcirc\langle\langle A\rangle\rangle\psi\,\mathcal{U}\,\varphi)$
($\mathrm{LFP}_\mathcal{U}$)	$\langle\langle\emptyset\rangle\rangle\Box((\varphi\vee(\psi\wedge\langle\langle A\rangle\rangle\bigcirc\theta))\to\theta) \to \langle\langle\emptyset\rangle\rangle\Box(\langle\langle A\rangle\rangle\psi\,\mathcal{U}\,\varphi\to\theta)$
(Modus Ponens)	$\dfrac{\varphi,\ \varphi\to\psi}{\psi}$ ($\langle\langle\emptyset\rangle\rangle\Box$-Necessitation) $\dfrac{\varphi}{\langle\langle\emptyset\rangle\rangle\Box\varphi}$
($\langle\langle A\rangle\rangle\bigcirc$-Monotonicity)	$\dfrac{\varphi\to\psi}{\langle\langle A\rangle\rangle\bigcirc\varphi\to\langle\langle A\rangle\rangle\bigcirc\psi}$

TABLE 2.8. An axiom system for ATL_Σ.

Alur *et al.* [**17**]. A modification of the ATL model checking algorithm [**15**] was proposed by Jamroga [**116**] to obtain an algorithm for multi-agent planning.

```
function ATL-$eval(\psi, \mathcal{S} = \langle\Pi,\Sigma,Q,\pi,\delta\rangle)$ returns the extension $[\![\psi]\!]$ of $\psi$ in $\mathcal{S}$
    case $\psi = p$:                                   return $\pi(p)$
    case $\psi = \neg\vartheta$:                       return $Q \setminus$ ATL-$eval(\vartheta,\mathcal{S})$
    case $\psi = \vartheta_1 \vee \vartheta_2$:        return ATL-$eval(\vartheta_1,\mathcal{S}) \cup$ ATL-$eval(\vartheta_2,\mathcal{S})$
    case $\psi = \langle\langle A\rangle\rangle\bigcirc\vartheta$:   return $Pre(A,$ ATL-$eval(\vartheta,\mathcal{S}))$
    case $\psi = \langle\langle A\rangle\rangle\Box\vartheta$:       $\Delta_1 := Q$; $\Delta_2 =$ ATL-$eval(\vartheta,\mathcal{S})$
                                                       while $\Delta_1 \not\subseteq \Delta_2$ do $\Delta_1 := \Delta_2$
                                                           $\Delta_2 := Pre(A,\Delta_1) \cap$ ATL-$eval(\vartheta,\mathcal{S})$ od
                                                       return $\Delta_1$
    case $\psi = \langle\langle A\rangle\rangle\vartheta_1\,\mathcal{U}\,\vartheta_2$:  $\Delta_1 := \emptyset$; $\Delta_2 =$ ATL-$eval(\vartheta_2,\mathcal{S})$
                                                       while $\Delta_2 \not\subseteq \Delta_1$ do $\Delta_1 := \Delta_1 \cup \Delta_2$
                                                           $\Delta_2 := Pre(A,\Delta_1) \cap$ ATL-$eval(\vartheta_1,\mathcal{S})$ od
                                                       return $\Delta_1$
end-function
```

FIGURE 2.5. ATL symbolic model checking.

ATL* generalises ATL by allowing for Boolean combinations and nestings of temporal operators after path quantifiers. The resulting additional expressivity of ATL* might be useful for certain applications. ATL* combines the advantages of LTL with the ability to reason about agents' interaction: Specifications written in LTL are thought to be more intuitive and more appropriate for reasoning about concurrent systems [**148**]. However, the drawback of ATL* is its 2-EXPTIME-complete model checking problem as opposed to PTIME for ATL. To be able to benefit from both, the intuitive and expressive syntax of ATL* and ATL's inexpensive model checking, Harding, Ryan and

Schobbens [105] suggested an approximation of ATL*-formulas by formulas of ATL. The approximation uses a set of rewrite rules for forming the ATL formulas. More precisely, a given ATL*-formula φ_{ATL^*} is approximated by two formulas of ATL, a strong bound φ_{strong} and a weak bound φ_{weak}, such that:

$$\varphi_{\text{strong}} \rightarrow \varphi_{\text{ATL}^*} \rightarrow \varphi_{\text{weak}}.$$

Then, using the model-checker MOCHA [17], we can determine whether φ_{ATL^*} is satisfied at a world in an ATS: If φ_{strong} is true, then φ_{ATL^*} holds there as well; and, if φ_{weak} is false, so is φ_{ATL^*}.

2.4.4. Complexity. Table 2.9 shows the known complexities of the model checking and satisfiability problems for ATL, ATL* and AMC. The model checking problems

	Model Checking Problem	Satisfiability Problem
ATL	PTIME-complete [13]	EXPTIME-complete [265, 252]
ATL*	2-EXPTIME-complete [13]	2-EXPTIME-hard [213] 3-EXPTIME [63, 218]
AMC	EXPTIME-complete [13]	EXPTIME-complete [218, 141]

TABLE 2.9. Complexities for ATL variants.

for ATL and the AMC-fragment without alternation of fixpoint operators are PTIME-complete. More precisely, they can be solved in time $\mathcal{O}(m \cdot \ell)$ for an ATS with m transitions and an ATL-formula of length ℓ [15]. Model checking AMC can be solved in time $\mathcal{O}((m \cdot \ell)^{d+1})$, where $d \geq 1$ is the alternation depth of the formula [15].

Jamroga and Dix [121] refined the ATL model checking complexity for the case where the number of agents is considered as part of the input for the model checking problem. The authors point out that the number of transitions in an ATL-model are usually exponential in the number of agents that occur in the input formula. Following this observation, the complexity of the model checking problem is settled at Σ_2^P-complete for concurrent game structures (introduced in [15]), and at NP-complete for ATSs. Although concurrent game structures were considered to be more elegant or intuitive [96], this complexity result [121] indicates an advantage of using ATSs instead of concurrent game structures.

The complexity of the satisfiability problem for AMC is EXPTIME-complete: the upper bound is a recent result employing alternating parity automata over infinite trees [218], and the lower bound stems from the EXPTIME-hard μ-Calculus [141] which is a fragment of AMC. ATL* satisfiability is 2-EXPTIME-hard and can be solved in 3-EXPTIME: the lower bound stems from the 2-EXPTIME-hardness of Boolean games with LTL objectives [213], and the upper bound follows from the fact that ATL*-formulas can be translated into formulas of AMC with doubly exponential increase of the formula size [63] and that AMC satisfiability can be decided in EXPTIME [218].

Investigating the complexity of the satisfiability problem for ATL is one main objective of this work and it will be described in detail later in Chapter 3.

2.4.5. Expressivity. A strategy for an agent in ATL or ATL* might depend on the full history, i.e., an unbounded sequence of states up to the current state. However, as it was already remarked in [**15**], for ATL and ATL* over finite transition systems, strategies depending on finite histories suffice. This is due to the correspondence of the ATL and ATL* semantics over finite structures to ω-regular games. In such games, the existence of a winning strategy implies the existence of a finite-state winning strategy [**45, 208**]. For ATL-formulas of the form $\langle\langle A\rangle\rangle\Diamond\varphi$, which correspond to finite reachability games, it suffices to consider strategies without history, i.e., strategies that depend on the current state only.

To establish a liveness property, say, of the form $\langle\langle A\rangle\rangle\Box\Diamond\varphi$ on all computations specified by coalition A, the computations that delay satisfying φ forever need to be ruled out. For this, ATL is not expressive enough. However, fairness constraints added to the semantics of ATL can rule out certain computations. The resulting logic is called Fair-ATL [**15**]. Since ATL* can express such fairness conditions, there is no need for a Fair-ATL*.

AMC is more expressive than ATL* and the alternation-free fragment of AMC is more expressive than ATL [**15**]. Notice that this is analogous to the relationships between μ-Calculus and CTL*, and alternation-free μ-Calculus and CTL as described in Section 2.3.5. Note that for one-player ATS (where $|\Sigma| = 1$) the (alternation-free) AMC is the same as (alternation-free) μ-Calculus, ATL* is the same as CTL*, and ATL is the same as CTL. A canonical translation of ATL* into AMC is described in [**63**]. This translation involves an inevitable doubly exponential blow-up of the formula size.

Below in Chapter 4, we consider the extension ATLES of ATL with explicit names for strategies in the object language. ATLES is clearly more expressive than ATL and it can describe some properties that cannot be expressed in ATL* and AMC.

As bisimulation is a useful tool for characterising expressivity of a modal logic structurally, we now generalise temporal bisimulation (cf. Definition 2.14) to the alternating case and introduce alternating bisimulation [**16**].

DEFINITION 2.25. ALTERNATING BISIMULATION Given two ATSs $S = \langle \Pi, \Sigma, Q, \pi, \delta\rangle$ and $S' = \langle \Pi, \Sigma, Q', \pi', \delta'\rangle$ for a finite set of agents Σ, and a coalition $A \subseteq \Sigma$, a binary relation $H \subseteq Q \times Q'$ is an *A-bisimulation between S and S'* if for all states q_1 and q_2 with $(q_1, q_2) \in H$ the following three conditions hold:

(i) $\pi(q_1) = \pi'(q_2)$;
(ii) for every choice $Q_A \in \delta(q_1, A)$, there is a choice $Q'_A \in \delta'(q_2, A)$ such that for every state $q'_2 \in Q'_A$, there is a state $q'_1 \in Q_A$ such that $(q'_1, q'_2) \in H$;
(iii) for every choice $Q'_A \in \delta'(q_2, A)$, there is a choice $Q_A \in \delta(q_1, A)$ such that for every state $q'_1 \in Q_A$, there is a state $q'_2 \in Q'_A$ such that $(q'_1, q'_2) \in H$.

If H is an A-bisimulation and $(x,y) \in H$, then x and y are called *A-bisimilar*, written $x \leftrightarrows_A y$.

Intuitively, $x \leftrightarrows_A y$ means that at states x and y, the agents in coalition A can achieve the same, i.e., ensure the same outcome.

We now develop a logical characterisation of alternating bisimulation and correct a problem with the characterisation given by Alur, Henzinger, Kupferman and Vardi [16]. A similar bisimulation characterisation was given for CTL by Browne, Clarke and Grümberg [44]. We use the following notation: For a coalition A, denote with A-ATL (A-ATL*) the fragment of ATL (ATL*) where the path quantifiers of all formulas are parameterised with A. Denote with A-ATL$^{\bigcirc}$ the fragment of A-ATL where the only temporal operator occurring in any formula is $\bigcirc$. The following theorem gives a logical characterisation of A-bisimulation.

THEOREM 2.26. *Let $S = \langle \Pi, \Sigma, Q, \pi, \delta \rangle$ and $S' = \langle \Pi, \Sigma, Q', \pi', \delta' \rangle$ be two finite ATSs, $x \in Q$ and $y \in Q'$ two states, and $A \subseteq \Sigma$ a set of agents. The following are equivalent:*

(a) *$x \leftrightarrows_A y$;*
(b) *x and y satisfy the same A-ATL$^{\bigcirc}$-formulas;*
(c) *x and y satisfy the same A-ATL*-formulas.*

Before proving this theorem, we point out the problem with a similar logical characterisation of alternating bisimulation in [16].

REMARK 2.27. Theorem 9 of [16] states that, given a set of agents $A \subseteq \Sigma$, two states x and y are A-bisimilar if, and only if, x and y satisfy the same ATL*-formulas in positive normal form where only A or $\Sigma \setminus A$ occur in any path quantifier. Consider the following counterexample for the left-to-right direction of Theorem 9 in [16]. Let x and y be two states of two ATSs for the set $\Sigma = \{a,b\}$ of agents. Figure 2.6 shows the choices for agents a and b at a state x (left) and y (right). At x, the only choices for a and for b coincide and contain only one state, and, at y, there are three a-choices and two b-choices available that intersect as shown. A black dot and a white dot indicates that the corresponding states satisfy different atomic propositions. Verify that x and y are a-bisimilar provided that x and y satisfy the same atomic propositions (cf. Definition 2.25). According to Theorem 9 in [16], both states, x and y, should satisfy the positive next formula $\langle\langle \Sigma \setminus \{a\} \rangle\rangle \bigcirc \mathit{blackdot}$. It is easy to see that this is not the case since y does not, while x does. Notice that nevertheless both, x and y, satisfy the next formulas $\langle\langle a \rangle\rangle \bigcirc \mathit{blackdot}$ and $\neg\langle\langle a \rangle\rangle \bigcirc \neg \mathit{blackdot}$. Table 2.10 shows implications of ATL-next formulas at two A-bisimilar states x and y in two ATSs for a set Σ of agents. Depending on the coalition A, we distinguish three cases: A is a proper, non-empty subset of Σ, $A = \emptyset$ or $A = \Sigma$. For instance, we can read of Table 2.10 that, given $x \leftrightarrows_A y$ with $\emptyset \subset A \subset \Sigma$, we have $x \models \langle\langle A \rangle\rangle \bigcirc p$ implies $y \models \neg\langle\langle \Sigma \setminus A \rangle\rangle \bigcirc \neg p$, but not vice versa. $\dashv$

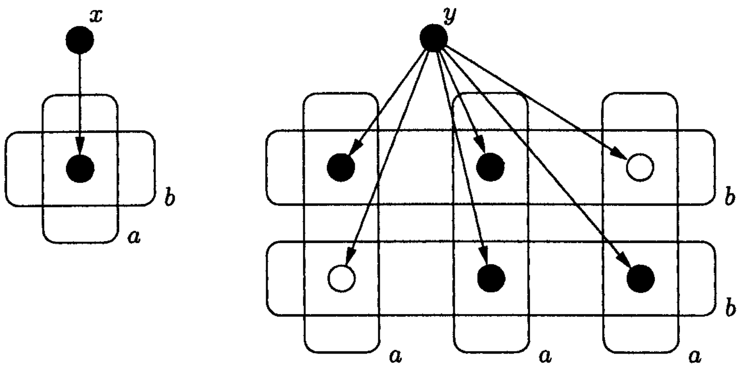


FIGURE 2.6. The choices for agents a and b at states in two ATSs.

$\emptyset \subset A \subset \Sigma$:	$x \models \neg\langle\langle \Sigma \setminus A \rangle\rangle \bigcirc \neg p$	$\not\Leftarrow\ \not\Rightarrow$	$x \models \langle\langle \Sigma \setminus A \rangle\rangle \bigcirc p$
	$\Uparrow\ \not\Downarrow$		$\not\Uparrow\ \Downarrow$
	$x \models \langle\langle A \rangle\rangle \bigcirc p$	$\not\Leftarrow\ \not\Rightarrow$	$x \models \neg\langle\langle A \rangle\rangle \bigcirc \neg p$
	$\Updownarrow$ (by $x \rightleftarrows_A y$)		$\Updownarrow$ (by $x \rightleftarrows_A y$)
	$y \models \langle\langle A \rangle\rangle \bigcirc p$	$\not\Leftarrow\ \not\Rightarrow$	$y \models \neg\langle\langle A \rangle\rangle \bigcirc \neg p$
	$\not\Uparrow\ \Downarrow$		$\Uparrow\ \not\Downarrow$
	$y \models \neg\langle\langle \Sigma \setminus A \rangle\rangle \bigcirc \neg p$	$\not\Leftarrow\ \not\Rightarrow$	$y \models \langle\langle \Sigma \setminus A \rangle\rangle \bigcirc p$
$A = \emptyset$:	$x \models \langle\langle \emptyset \rangle\rangle \bigcirc p$	$\not\Leftarrow\ \Rightarrow$	$x \models \neg\langle\langle \emptyset \rangle\rangle \bigcirc \neg p$
	$\Updownarrow$ (by $x \rightleftarrows_\emptyset y$)		$\Updownarrow$ (by $x \rightleftarrows_\emptyset y$)
	$y \models \langle\langle \emptyset \rangle\rangle \bigcirc p$	$\not\Leftarrow\ \Rightarrow$	$y \models \neg\langle\langle \emptyset \rangle\rangle \bigcirc \neg p$
$A = \Sigma$:	$x \models \langle\langle \Sigma \rangle\rangle \bigcirc p$	$\Leftarrow\ \not\Rightarrow$	$x \models \neg\langle\langle \Sigma \rangle\rangle \bigcirc \neg p$
	$\Updownarrow$ (by $x \rightleftarrows_\Sigma y$)		$\Updownarrow$ (by $x \rightleftarrows_\Sigma y$)
	$y \models \langle\langle \Sigma \rangle\rangle \bigcirc p$	$\Leftarrow\ \not\Rightarrow$	$y \models \neg\langle\langle \Sigma \rangle\rangle \bigcirc \neg p$

TABLE 2.10. Implications of ATL$^{\bigcirc}$-formulas at two A-bisimilar states x and y.

We now give the proof of Theorem 2.26.

PROOF. The proof is along the lines of the proof for CTL bisimulation characterisation by Browne, Clarke and Grümberg [44]. Suppose S, S', x, y and A are as in the theorem. The direction from (c) to (b) is obvious since A-ATL$^{\bigcirc}$ is a fragment of

A-ATL*. In the following, the directions from (b) to (a) and from (a) to (c) will be shown, in this order.

"(b) $\Rightarrow$ (a)": Suppose x and y satisfy the same A-ATL$^{\bigcirc}$-formulas. Define a binary relation $H \subseteq Q \times Q'$ as follows:

$$H = \{(q, q') \mid \text{for all } A\text{-ATL}^{\bigcirc}\text{-formulas } \varphi,\ S, q \models \varphi \text{ iff } S', q' \models \varphi\}.$$

In the following, H is shown to be an A-bisimulation between S and S'. Notice that $(x, y) \in H$. Then it follows that $x \leftrightarrows_A y$.

Let $(q, q') \in H$. It needs to be shown that the conditions (i) to (iii) in Definition 2.25 of A-bisimulation are satisfied.

- ad (i). By definition of H, $S, q \models p$ iff $S', q' \models p$ for all atomic propositions $p \in \Pi$. Thus $\pi(q) = \pi'(q')$.
- ad (ii). Let $S'' = \langle \Pi, \Sigma, Q'', \pi'', \delta'' \rangle$ be the disjoint union of S and S' where $Q'' = Q \uplus Q'$, $\pi'' = \pi \uplus \pi'$ and $\delta'' = \delta \uplus \delta'$. Let $\sim \subseteq Q'' \times Q''$ be a binary relation such that $s \sim s'$ iff s and s' satisfy the same A-ATL$^{\bigcirc}$-formulas in S''. Note that $\sim$ is an equivalence relation. Denote with $Q''|_\sim$ the set of equivalence classes induced by $\sim$. Clearly, for all $[s], [s'] \in Q''|_\sim$, there is an A-ATL$^{\bigcirc}$-formula φ such that $S'', s \models \varphi$ and $S'', s' \not\models \varphi$. Arbitrarily choose such a formula φ and set $\varphi_{[s],[s']} = \varphi$. For all $[s] \in Q''|_\sim$, let
$$\varphi_{[s]} = \bigwedge_{[s'] \in Q''|_\sim, [s] \neq [s']} \varphi_{[s],[s']}.$$
Note that Q'' and thus $\varphi_{[s]}$ are finite by the finiteness of S and S'. Let $Q_A \in \delta(q, A)$ be a choice and $\psi = \langle\!\langle A \rangle\!\rangle \bigcirc \bigvee_{t \in Q_A} \varphi_{[t]}$. Since the set Q_A is finite, so is ψ. That is, ψ an A-ATL$^{\bigcirc}$-formula. It is evident that $S, q \models \psi$. By definition of H, it follows from $(q, q') \in H$ that $S', q' \models \psi$. That is, there is a choice $Q'_A \in \delta'(q', A)$ such that for every state $t' \in Q'_A$, $S', t' \models \bigvee_{t \in Q_A} \varphi_{[t]}$. For showing (ii), it remains to show that for every state $t' \in Q'_A$, there is a state $t \in Q_A$ such that $(t, t') \in H$. Let $t' \in Q'_A$. Then $S', t' \models \varphi_{[t]}$ for some $t \in Q_A$. By definition of $\varphi_{[t]}$, it follows $[t] = [t']$, i.e., t and t' satisfy the same A-ATL$^{\bigcirc}$-formulas. Hence, $(t, t') \in H$ by definition of H.
- ad (iii). This can be shown similarly to Condition (ii).

"(a) $\Rightarrow$ (c)": Suppose $x \leftrightarrows_A y$, i.e., there is an A-bisimulation H such that $(x, y) \in H$. In the following, it is shown that

$$S, q \models \varphi \quad \text{iff} \quad S', q' \models \varphi$$

for all states $q \in Q$ and $q' \in Q'$, and all A-ATL*-state formulas φ, and

$$S, \rho \models \psi \quad \text{iff} \quad S', \rho' \models \psi$$

for all infinite sequences $\rho \in Q^\omega$ and $\rho' \in Q'^\omega$ with $(\rho[i], \rho'[i]) \in H$ for every $i \geq 0$, and all A-ATL*-path formulas ψ. Then from $(x, y) \in H$ it follows that x and y satisfy the same ATL*-formulas.

The proof is by simultaneous induction on the structures of φ and ψ. Let $q \in Q$ and $q' \in Q'$ with $(q, q') \in H$, and $\rho \in Q^\omega$ and $\rho' \in Q'^\omega$ with $(\rho[i], \rho'[i]) \in H$ for every $i \geq 0$. For the induction base, let $\varphi = p$ where $p \in \Pi$ is an atomic proposition. By Condition (i) in Definition 2.25 of A-bisimulation, it holds that $p \in \pi(q)$ iff $p \in \pi'(q')$. Thus $\mathcal{S}, q \models p$ iff $\mathcal{S}', q' \models p$. Consider the induction step. Suppose the induction hypothesis holds for A-ATL*-state formulas φ', φ_1 and φ_2, and for A-ATL*-path formulas ψ', ψ_1 and ψ_2. It is shown for A-ATL*-formula φ and A-ATL*-path formula ψ as follows:

- $\varphi = \neg\varphi'$. Then we have that $\mathcal{S}, q \models \neg\varphi'$ iff $\mathcal{S}, q \not\models \varphi'$ iff $\mathcal{S}', q' \not\models \varphi'$ (by the induction hypothesis) iff $\mathcal{S}', q' \models \neg\varphi'$.
- $\varphi = \varphi_1 \vee \varphi_2$. Then we have that $\mathcal{S}, q \models \varphi_1 \vee \varphi_2$ iff $\mathcal{S}, q \models \varphi_1$ or $\mathcal{S}, q \models \varphi_2$ iff $\mathcal{S}', q' \models \varphi_1$ or $\mathcal{S}', q' \models \varphi_2$ (by the induction hypothesis) iff $\mathcal{S}', q' \models \varphi_1 \vee \varphi_2$.
- $\varphi = \langle\langle A \rangle\rangle\psi'$. In the following, only the the direction from left to right is considered; the other direction is similar. Suppose $\mathcal{S}, q \models \langle\langle A \rangle\rangle\psi'$. Then there is a strategy σ_A in $\mathcal{S}$ such that for all computations $\lambda \in \mathsf{out}_{\mathcal{S}}(q, \sigma_A)$, it holds that $\mathcal{S}, \lambda \models \psi'$. In the following, a strategy σ'_A in $\mathcal{S}'$ is constructed such that for all computations $\lambda' \in \mathsf{out}_{\mathcal{S}'}(q', \sigma'_A)$, there is a computation $\lambda \in \mathsf{out}_{\mathcal{S}}(q, \sigma_A)$ such that $(\lambda[i], \lambda'[i]) \in H$ for all $i \geq 0$. Then, together with the induction hypothesis, it follows that $\mathcal{S}', \lambda' \models \psi'$ for all $\lambda' \in \mathsf{out}_{\mathcal{S}'}(q', \sigma'_A)$. Thus $\mathcal{S}', q' \models \langle\langle A \rangle\rangle\psi'$.

  Define a strategy $\sigma'_A : Q'^+ \to 2^{Q'}$ by induction on the length $n + 1$ of sequences $\lambda' = \lambda'_0 \dots \lambda'_n \in Q'^+$ such that the following property $(*)$ is satisfied: For all $\lambda' = \lambda'_0 \dots \lambda'_n \in Q'^+$ with $n \geq 0$, $\lambda'_0 = q'$ and $\lambda'_{i+1} \in \sigma'_A(\lambda'[0, i])$ for all $i < n$, there is a computation $\lambda \in \mathsf{out}_{\mathcal{S}}(q, \sigma_A)$ such that $(\lambda[i], \lambda'[i]) \in H$ for all $i \leq n$.

  For $n = 0$: Since H is an A-bisimulation, it follows by the definition of A-bisimulation that there is a choice $Q_A \in \delta'(q', A)$ with the property: for all states $t' \in Q_A$, there is a state $t \in \sigma_A(q)$ such that $(t, t') \in H$. Arbitrarily choose such a $Q_A \in \delta'(q', A)$ and set $\sigma'_A(q') = Q_A$. For all other states $s \in Q'$ with $s \neq q'$, arbitrarily choose a choice $Q'_A \in \delta'(s, A)$ and set $\sigma'_A(s) = Q'_A$.

  For $n \to n + 1$: Let $\lambda' = \lambda'_0 \dots \lambda'_n \in Q'^+$ and suppose $\sigma'_A(\lambda')$ is already defined. For each state $s \in Q'$, distinguish two cases:

  - $\lambda'_0 = q'$, $\lambda'_{i+1} \in \sigma'_A(\lambda'[0, i])$ for all $i < n$, and $s \in \sigma'_A(\lambda')$. By $(*)$, there is a computation $\lambda \in \mathsf{out}_{\mathcal{S}}(q, \sigma_A)$ such that $(\lambda[i], \lambda'[i]) \in H$ for all $i \leq n$. Arbitrarily fix such a $\lambda \in \mathsf{out}_{\mathcal{S}}(q, \sigma_A)$ such that $(\lambda[n + 1], s) \in H$. Since H is an A-bisimulation, it follows by the definition of A-bisimulation that there is a choice $Q_A \in \delta'(s, A)$ with the property: for all states $t' \in Q_A$, there is a state $t \in \sigma_A(\lambda[0, n+1])$ such that $(t, t') \in H$. Arbitrarily choose such a $Q_A \in \delta'(s, A)$ and set $\sigma'_A(\lambda' \cdot s) = Q_A$.

- Otherwise, arbitrarily choose a choice $Q'_A \in \delta'(s, A)$ and set $\sigma'_A(\lambda' \cdot s) = Q'_A$.

Note that in the above construction of σ'_A, such choices Q_A and Q'_A always exist by definition of ATS. Thus σ'_A is well-defined.

- $\psi = \varphi'$. Then we have that $\mathcal{S}, \rho \models \varphi'$ iff $\mathcal{S}, \rho[0] \models \varphi'$ iff $\mathcal{S}', \rho'[0] \models \varphi'$ (by the induction hypothesis) iff $\mathcal{S}', \rho' \models \varphi'$.
- $\psi = \neg\psi'$. Then we have that $\mathcal{S}, \rho \models \neg\psi'$ iff $\mathcal{S}, \rho \not\models \psi'$ iff $\mathcal{S}', \rho' \not\models \psi'$ (by the induction hypothesis) iff $\mathcal{S}', \rho' \models \neg\psi'$.
- $\psi = \psi_1 \vee \psi_2$. Then we have that $\mathcal{S}, \rho \models \psi_1 \vee \psi_2$ iff $\mathcal{S}, \rho \models \psi_1$ or $\mathcal{S}, \rho \models \psi_2$ iff $\mathcal{S}', \rho' \models \psi_1$ or $\mathcal{S}', \rho' \models \psi_2$ (by the induction hypothesis) iff $\mathcal{S}', \rho' \models \psi_1 \vee \psi_2$.
- $\psi = \bigcirc\psi'$. Then we have that $\mathcal{S}, \rho \models \bigcirc\psi'$ iff $\mathcal{S}, \rho[1, \infty] \models \psi'$ iff $\mathcal{S}', \rho'[1, \infty] \models \psi'$ (by the induction hypothesis) iff $\mathcal{S}', \rho' \models \bigcirc\psi'$.
- $\psi = \psi_1 \mathcal{U} \psi_2$. Then we have that $\mathcal{S}, \rho \models \psi_1 \mathcal{U} \psi_2$ iff there is an $i \geq 0$ such that $\mathcal{S}, \rho[i, \infty] \models \psi_2$ and $\mathcal{S}, \rho[j, \infty] \models \psi_1$ for all j with $0 \leq j < i$ iff there is an $i \geq 0$ such that $\mathcal{S}', \rho'[i, \infty] \models \psi_2$ and $\mathcal{S}', \rho'[j, \infty] \models \psi_1$ for all j with $0 \leq j < i$ (by the induction hypothesis) iff $\mathcal{S}', \rho' \models \psi_1 \mathcal{U} \psi_2$.

□

2.5. Combination of Epistemic and Strategic Logic

ATL can describe coalitional strategic abilities in MASs, but it does not describe how agents interact such as the process of cooperation. One missing aspect is the knowledge of agents, which might be necessary for agents' decision making. In 2002, Wooldridge and van der Hoek [**245, 272, 246**] combined ATL with S5 and introduced the extension of ATL with knowledge operators: Alternating-time Temporal Epistemic Logic (ATEL). This extension widens the scope of reasoning over MASs with epistemic notions and enables reasoning about agents acting under incomplete information. Reasoning about and verification of strategic abilities of agents and their knowledge is useful in several application areas such as security [**104**], planning [**245**], games [**125**] and communication protocols [**166**].

An account for ATL with incomplete information but without knowledge operators was already given by Alur *et al.* [**15**].

2.5.1. ATEL. We define the syntax of the Alternating-time Temporal Epistemic Logic as combination of ATL (Definition 2.20) and S5CD (cf. Section 2.2.3).

DEFINITION 2.28. (ATEL SYNTAX). Let $\mathbf{\Pi}$ be a countable infinite set of atomic propositions and $\mathbf{\Sigma}$ a countable infinite set of agents. The set of ATEL-formulas φ is defined by the following BNF specification:

$$\begin{array}{rcl} \varphi & ::= & p \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle\langle A\rangle\rangle\bigcirc\varphi \mid \langle\langle A\rangle\rangle\Box\varphi \mid \langle\langle A\rangle\rangle(\varphi\,\mathcal{U}\,\varphi) \mid \\ & & K_a\varphi \mid E_A\varphi \mid C_A\varphi \mid D_A\varphi \end{array}$$

where p ranges over atomic propositions in $\mathbf{\Pi}$, a over agents in $\mathbf{\Sigma}$ and $A \subset \mathbf{\Sigma}$ over coalitions.

For an illustration of ATEL, consider the following formula, which states that if agent a knows a secret, then she can keep it to herself, i.e., a can make sure that her secret will never be known by another agent b (cf. [**264**]):

$$K_a secret \wedge \neg K_b secret \rightarrow \langle\!\langle a \rangle\!\rangle \Box \neg K_b secret.$$

As a semantic structure for ATEL, we extend ATSs to alternating epistemic transition systems; cf. [**272**].

Definition 2.29. (AETS). An *alternating epistemic transition system (AETS)* for a set of agents $\Sigma = \{1, \ldots, n\}$ is a tuple $\mathcal{S} = \langle \Pi, \Sigma, Q, \{\sim_a\}_{a \in \Sigma}, \pi, \delta \rangle$ that fulfils the following two conditions:

- $\langle \Pi, \Sigma, Q, \pi, \delta \rangle$ forms an ATS for Σ (cf. Definition 2.21), and
- $\langle \Pi, \Sigma, Q, \{\sim_a\}_{a \in \Sigma}, \pi \rangle$ forms an epistemic structure for Σ (cf. Definition 2.2).

We use the notions of a *successor*, a *computation*, a *strategy* for an agent or a coalition and the set of *outcomes* of a strategy as defined for ATL in Section 2.4.1. Moreover, we make use of the abbreviations $\sim_A^E$, $\sim_A^C$ and $\sim_A^D$ for coalitional epistemic accessibility relations as defined for S5CD in Section 2.2.3.

The semantics of ATEL combines the semantics for ATL and S5CD; cf. [**272**].

Definition 2.30. (ATEL Semantics). Given an AETS $\mathcal{S} = \langle \Pi, \Sigma, Q, \{\sim_a\}_{a \in \Sigma}, \pi, \delta \rangle$, the satisfaction relation $\models$ is inductively defined as follows: For all states q of $\mathcal{S}$ and ATEL-formulas ψ, we have that $\mathcal{S}, q \models \psi$ is defined as

- for ATL if ψ is an atomic proposition or of the form $\neg\varphi$, $\varphi_1 \vee \varphi_2$, $\langle\!\langle A \rangle\!\rangle \bigcirc \varphi$, $\langle\!\langle A \rangle\!\rangle \Box \varphi$ or $\langle\!\langle A \rangle\!\rangle \varphi_1 \mathcal{U} \varphi_2$, then $\mathcal{S}, q \models \psi$ is defined as (cf. Definition 2.22);
- for S5CD if ψ is of the form $K_a\varphi$, $E_A\varphi$, $C_A\varphi$ or $D_A\varphi$ (cf. Definition 2.3 and Section 2.2.3).

If for some state q of some AETS $\mathcal{S}$ it holds that $\mathcal{S}, q \models \varphi$, then the ATEL-formula φ is *true* at q, and $\mathcal{S}$ is called a *model* of φ. An ATEL-formula is *satisfiable* if it has a model.

Example 2.31. (Interplay of Temporal and Epistemic Notions). This example illustrates the complexity of the properties involving temporal and epistemic notions that can be expressed with ATEL. Figure 2.7 shows a part of an AETS for two agents a and b: The dashed lines denote the agents' epistemic accessibility relations labelled with $\sim_a$ and $\sim_b$; the solid arrows denote the system transitions; and the round boxes denote the choices available to a and b at the upper left state of Figure 2.7. Suppose this upper left state is the current state. Using ATEL, we can express epistemic properties, temporal properties and combinations of both. For instance, we have the epistemic property that agent a knows proposition p, but agent b does not know p, i.e.,

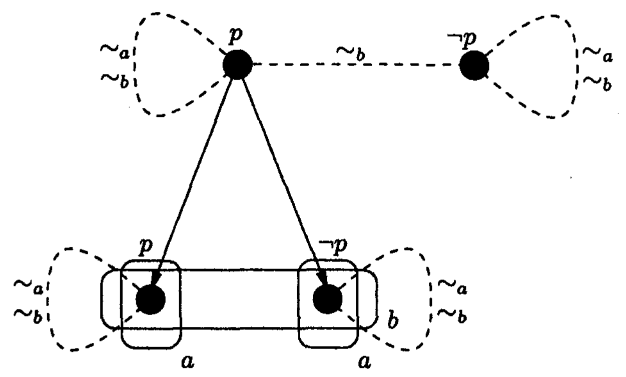


FIGURE 2.7. A part of an epistemic alternating structure.

the formula $K_a p \wedge \neg K_b p$ holds at the current state. Also the formulas $\langle\langle a\rangle\rangle \bigcirc p \vee \langle\langle a\rangle\rangle \bigcirc \neg p$ and $\neg\langle\langle b\rangle\rangle \bigcirc p \wedge \neg\langle\langle b\rangle\rangle \bigcirc \neg p$ are satisfied describing the temporal properties that a can enforce p or $\neg p$ at the next state of the system, whereas agent b cannot. A property combining epistemic and temporal notions is, e.g., that a can ensure that b knows p at the next state. Indeed, we find that $\langle\langle a\rangle\rangle \bigcirc K_b p$ is satisfied. Moreover, the fact of a being capable of bringing about b to know p at the next state implies that a must know p already is reflected by the formula $\langle\langle a\rangle\rangle \bigcirc K_b p \rightarrow K_a p$ being true at the current state. $\dashv$

2.5.2. Axioms for ATEL. Van der Hoek and Wooldridge [**272**, **246**] provided some sound axioms for ATEL but no complete axiomatisation. Goranko, Jamroga and van Drimmelen [**97**] axiomatised ATEL without operators for distributed knowledge and conjectured an axiomatisation of ATEL with distributed knowledge operators by extending the axiomatic system for ATL [**98**].

The axiom system of ATEL with distributed knowledge suggested in [**97**] is similar to the union of the axioms and rules for ATL in Table 2.8, for S5 in Table 2.3 and for the S5-operators for common- and distributed knowledge in Table 2.4 (parameterized with sets of agents). Additionally, we need two axiom schemes:

- $C_A\varphi \rightarrow C_B\varphi$, for all $A, B \subseteq \Sigma$ with $B \subseteq A$;
- $D_A\varphi \rightarrow D_B\varphi$, for all $A, B \subseteq \Sigma$ with $A \subseteq B$.

Notice that the resulting axiom system deviates from the one conjectured for ATEL in [**97**] and, in this work, is not proven to be complete wrt. ATEL. Anyway, the system suggests that epistemic notions in ATEL do not interfere with temporal operators.

2.5.3. Model Checking. ATEL model checking is tractable as its ATL counterpart: the model checking problem is PTIME-complete [**246**]. In 2004, a reduction of the model checking of a fragment of ATEL to ATL model checking was suggested by van Otterloo, van der Hoek and Wooldridge [**255**]. This reduction works by translating some ATEL-formulas and transition structures into formulas and structures of ATL in a satisfiability preserving way. After that, a similar but more general technique was suggested by Goranko and Jamroga [96], which embeds full ATEL with common and distributed knowledge into ATL by "simulating" the epistemic layer of ATEL by adding new agents.

2.5.4. Complexity. The complexity of ATEL's satisfiability problem is settled at ExpTime-complete [264]. That is, ATEL is no more complex than ATL and the presence of the knowledge operators in the language does not increase the complexity. In Chapter 3, we extend the decision procedure for ATL to decide ATEL.

2.5.5. Expressivity. We now demonstrate how the notions of epistemic bisimulation (cf. Definition 2.8) and alternating bisimulation (cf. Definition 2.25) can be combined to develop a notion of equivalence between alternating epistemic transition systems.

Definition 2.32. Epistemic Alternating Bisimulation. Let $S = \langle \Pi, \Sigma, Q, \{\sim_a\}_{a\in\Sigma}, \pi, \delta\rangle$ and $S = \langle \Pi, \Sigma, Q', \{\sim'_a\}_{a\in\Sigma}, \pi', \delta'\rangle$ be two AETSs for a finite set Σ of agents, and $A \in \Sigma$ a coalition. Two states $x \in Q$ and $y \in Q'$ are *A-epistemic and alternating bisimilar*, written $x \leftrightarrows^{\varepsilon,\tau}_A y$, if there are two binary relations $H, H^\varepsilon \subseteq Q \times Q'$ satisfying the following three properties:

(i) H^ε is an A-epistemic bisimulation with $(x, y) \in H^\varepsilon$ (cf. Definition 2.8);
(ii) H is an A-alternating bisimulation with $(x, y) \in H$ (cf. Definition 2.25);
(iii) for all agents $a \in A$, and all states $q_1, q_2 \in Q$ and $q'_1, q'_2 \in Q'$ with $(q_1, q'_1) \in H \cap H^\varepsilon$, $(q_2, q'_2) \in H^\varepsilon$, $q_1 \sim_a q_2$ and $q'_1 \sim'_a q'_2$, there is a state $q''_2 \in Q'$ such that
   (a) $\pi'(q'_2) = \pi'(q''_2)$;
   (b) $q'_1 \sim'_a q''_2$;
   (c) $(q_2, q''_2) \in H$.

This notion of bisimulation expresses that, given $x \leftrightarrows^{\varepsilon,\tau}_A y$, at states x and y the agents in A have the same knowledge and the same ability and, moreover, no single agent thinks it is possible that the agents in A could have different abilities.

It would be interesting to develop this notion further to characterize the expressivity of ATEL and it variants, which are introduced in the following sections.

2.5.6. Problems with ATEL. A strategic logic for reasoning about agents under incomplete information appears to be non-trivial. After the introduction of ATEL, several problems with the integration of the semantics of temporal and epistemic operators were pointed out [246, 131, 219, 254, 115, 6, 125, 110]. The criticism is mainly directed to the fact that the semantics of ATEL, as defined in [272], does not incorporate incomplete information consistently: epistemic operators assume incomplete information while temporal operators do not. Another issue is that ATEL does not account for the notion of agent's acting rationally. Many alternative solutions were suggested and still no satisfactory solution seems to be found. As emphasised in [6], the interaction should be analysed in the context of existing research about the interaction between knowledge and action, e.g., Moore [182], Morgenstern [183, 184], Halpern and

Fagin [**100**] and also Lespérance [**156, 157**]. In this section, we will review the basic problems of ATEL and present some of the proposed solutions.

ATEL describes strategic ability of single agents or groups of agents, and it models the knowledge of a single agent or the combined knowledge of the agents in a group. We observe that decision making and knowledge interfere: at two states, which an agent cannot distinguish to the best of her knowledge, she should make the same decision. The knowledge of an agent refers to her information state, i.e., which states she considers possible or which states are epistemically indistinguishable; cf. Section 2.2. Given this observation, we identify a number of problems with ATEL. The first problem has to do with the use of inappropriate strategies as illustrated by the following example.

EXAMPLE 2.33. (A TWO-PLAYER CARD GAME). This example illustrates one limitation of ATEL in modelling the interaction of an agent's knowledge and her strategic ability. Figure 2.8 depicts an AETS for three agents (two players a and b and a card dealer) that describes a simple card game. The arrows denote possible system transitions. The dashed lines denote epistemic accessibility relations for the agents a and b (omitting reflexive lines) labelled with $\sim_a$ and $\sim_b$, respectively. The card game involves only three cards: an ace, a king and a queen. The ace beats the king, the king the queen and the queen the ace. The first of three moves of the game is made at the initial state s_0 by a neutral player who randomly deals each player one card and puts the third card on the table. Neither player knows the others' card nor the card on the table. The second move is made by player a who can either keep her card or trade it with the card on the table. The final move is made by player b who has the same options. Which card each player is holding at a state is indicated by propositions of the form XY, where X and Y range over 'A' for ace, 'K' for king and 'Q' for queen and X indicates a's card and Y the card held by b. At the end, each player needs to show her card. The winner is the player holding the stronger card. Which player won is indicated by propositions win_a and win_b. Ignoring the epistemic accessibility relations, we can state

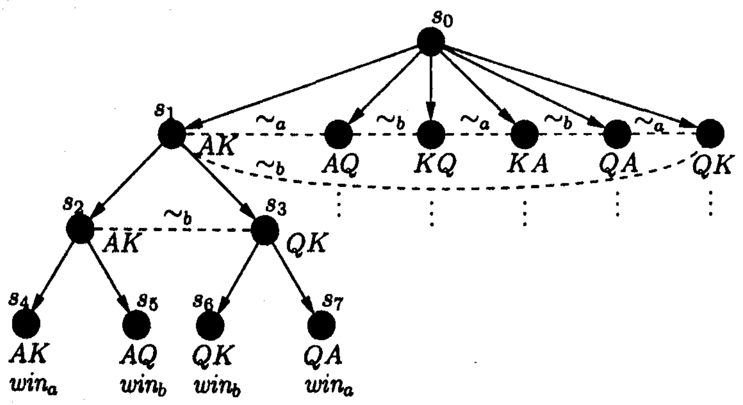


FIGURE 2.8. A simple card game described by an AETS.

that at the initial state s_0 agent b has a winning strategy, i.e., $s_0 \models \langle\langle b \rangle\rangle \Diamond \mathsf{win}_b$. However, the situation is different if we take the epistemic relations into account. Traverse the game tree and consider the states s_2 and s_3. At these states, b has to take two different choices in order to win at the next state, namely 'trade' at s_2 and 'keep' at s_3. But b's decision of whether to trade or to keep her card depends on the card held by player a which b cannot observe. That is, taking b's knowledge into account, player b cannot distinguish the states s_2 and s_3 and, consequently, she cannot act differently at these states. Hence, b does not have a winning strategy at s_0 — a contradiction. $\dashv$

The problem illustrated in Example 2.33 can be solved by changing ATEL's semantics to restrict the space of strategies to *uniform strategies* only. Uniform strategies in the context of strategic logic of agents under incomplete information were suggested and applied, e.g., in [**219, 115, 125, 254, 110**]. The concept of uniform strategies can also be found in other areas such as game theory, see van Benthem [**238, 239**] and von Neumann and Morgenstern [**261**].

Uniform strategies are also applied in game theory; see van Benthem [**238, 239**]. Intuitively, a uniform strategy for an agent disallows the agent to take different choices at for this agent epistemically indistinguishable states. That is, uniform strategies require the identification of choices. To this end, we extend the alternating epistemic transition systems with names for choices. Here, we choose to identify choices with a natural number. For applications, however, it might be more intuitive to incorporate proper names for choices in the definition of the semantic structure; see, e.g., the concurrent game structures for ATL [**15**].

DEFINITION 2.34. (AETS$^\sharp$). An *alternating epistemic transition system with choice enumeration (AETS$^\sharp$)* for a set of agents Σ is a tuple $S = \langle \Pi, \Sigma, Q, \{\sim_a\}_{a\in\Sigma}, \pi, \delta, \sharp\rangle$ that fulfils the following three conditions:

- $\langle \Pi, \Sigma, Q, \pi, \delta\rangle$ forms an ATS for Σ (cf. Definition 2.21), and
- $\langle \Pi, \Sigma, Q, \{\sim_a\}_{a\in\Sigma}, \pi\rangle$ forms an epistemic structure for Σ (cf. Definition 2.2).
- $\sharp : Q \times \Sigma \times \mathbf{2}^Q \to \mathbb{N}$ is a partial function $\sharp$ mapping each state q, each agent a, and each choice $Q_i \subseteq Q$, $1 \le i \le \ell$, in the set of choices $\delta(q, a) = \{Q_1, \ldots, Q_\ell\}$ to a natural number $\sharp(q, a, Q_i) = i$.

A strategy σ_a for agent a is called *uniform* if for all sequences $q_1 \ldots q_n \in Q^*$ and $q'_1 \ldots q'_n \in Q^*$ of the same length $n > 0$ with $q_i \sim_a q'_i$ for all $1 \le i \le n$, it holds that

$$\sharp(q_n, a, \sigma_a(q_1 \ldots q_n)) = \sharp(q'_n, a, \sigma_a(q'_1 \ldots q'_n)).$$

REMARK 2.35. Given the notion of uniform strategies, an interesting question arises about the expressivity of ATEL and ATL interpreted over AETS$^\sharp$. Namely, can ATEL or ATL distinguish the presence of uniform strategies? Equivalently, do ATEL and ATL with and without uniform strategies have the same valid formulas? We leave this question open. $\dashv$

Another related problem with the interaction of knowledge and strategic ability in ATEL is concerned with the knowledge of the identity of a strategy. Intuitively, when an agent has the capability to bring about φ, this should mean that the agent knows the identity of a strategy available to her that can enforce φ. In AI, Konolige [**139**] and Levesque [**158**] argued that knowing something requires having an identifier for it. In this context, the two knowledge modalities *de re* (Latin, of the thing) and *de dicto* (Latin, of the word) can be distinguished: an agent has knowledge *de re* of a strategy if she knows the identity of the strategy, and she has knowledge *de dicto* of a strategy if she knows that this strategy exists but she does not necessarily know its identity. This distinction between knowledge *de re* and *de dicto* was already analysed in the context of philosophy of language by Quine [**206**] in 1956, and it plays an important role in Moore's [**182**] account of the interaction between knowledge and action in which he considers the combination of first-order dynamic logic with the modal logic S4 for belief. Later, Morgenstern [**184, 185**] extended and generalised Moore's work. Moore's account of ability has been criticised for that the knowledge required for actions is indexical (also called *de se*) rather than *de re* [**156**]. We do not consider indexical knowledge here; for further information, see, e.g., Lewis [**161**] and Lespérance [**157**]. To give another reference to work about the interaction between knowledge and action where between *de re* and *de dicto* is distinguished, see Wooldridge [**270**] who focusses on the belief-desire-intention (BDI) model of agents and develops a logic for rational agents.

In ATEL, we can express knowledge *de dicto*, e.g., with formulas of the form $K_a\langle\langle a\rangle\rangle\Phi$, i.e., agent a knows that she has the strategy to achieve the temporal expression Φ, but the strategy can be a different one at each state a cannot distinguish from the current one. However, knowledge *de re* cannot be captured in ATEL. We illustrate knowledge *de dicto* by continuing Example 2.33.

EXAMPLE 2.36. (A TWO-PLAYER CARD GAME CONTINUED). As illustrated in Figure 2.8, the states s_2 and s_3 are indistinguishable for player b. At each of these states, we have that player b can act in a way in order to win the game at the next state, i.e., $\langle\langle b\rangle\rangle\Diamond\mathsf{win}_b$ that is satisfied at s_2 and s_3. The winning strategies for b differ at both states: at s_2 she has to trade her card and at s_3 she has to keep it. Since b cannot distinguish s_2 and s_3, she can only know of the fact that she can win without knowing how. This corresponds to b having knowledge *de dicto* of her ability to win, which we can express in ATEL with the formula $K_b\langle\langle b\rangle\rangle\Diamond\mathsf{win}_b$. $\dashv$

Since the existing operators in ATEL lack the ability to express knowledge *de re*, additional operators combining strategic ability and knowledge are needed. Jamroga and Ågotnes [**120, 119**] introduced an ATEL-variant with incomplete information and imperfect recall (it uses historyless uniform strategies) that provides operators expressing knowledge *de re* and *de dicto*. This variant uses a non-standard semantics, i.e., formulas are interpreted over sets of states instead of single states. The authors suggest new epistemic operators for "constructive" knowledge that capture the notion of knowledge *de re*, while the standard epistemic operators refer to knowledge de dicto. The new epistemic operators for "constructive" knowledge are $\mathbb{K}_a$, $\mathbb{C}_A$, $\mathbb{E}_A$ and $\mathbb{D}_A$ formalising respectively an agent a's "constructive" knowledge, "constructive" common, everybody's and distributed knowledge of agents in a coalition A. Below we will revise the definition of ATEL and formally introduce the epistemic operators for "constructive" knowledge.

Intuitively, an agents' knowledge depends on what was previously known. In ATEL, an agent's history and knowledge do not interfere with each other, i.e., the information of an agent does not depend on previously traversed states. Generally, two criteria can be distinguished: perfect or imperfect information, and perfect and imperfect recall. To denote which combination is meant, Schobbens [**219**] introduced a useful naming scheme: In Schobbens' [**219**] account for refining the interaction of knowledge and action in ATEL, he suggested four alternative ATL-variants with incomplete information: ATL_{IR}, ATL_{iR}, ATL_{Ir} and ATL_{ir}, where the subscript 'I' and 'i' encode whether respectively perfect or imperfect information is assumed, and 'R' and 'r' encode whether agents have perfect or imperfect recall, respectively. Jonker [**131**] suggests an ATEL-extension "Feasible ATEL" by introducing additional operators for finding a suitable uniform strategy. Yet another account to improve ATEL was given by Jamroga and van der Hoek [**125**] who proposed two ATEL-variants: Alternating-time Temporal Observational Logic (ATOL) for agents with bounded recall of the past, and Alternating-time Temporal Epistemic Logic with Recall (ATEL-R*) for reasoning about both perfect and imperfect recall. Notice that the variants for imperfect information and imperfect recall, ATL_{ir}, "Feasible ATEL" and ATOL, have an NP-complete model checking problem [**219, 125, 122**], whereas the model checking problem for the variants for imperfect information and perfect recall, ATL_{iR} and ATEL-R* are believed to be undecidable [**125, 120**].

Recently in 2006, Herzig and Troquard [**110**] investigate an interesting alternative approach for reasoning about strategic ability and knowledge within the logic of "Seeing To It That" (STIT), a logic about time and choices of agents proposed in the 1990s in the domain of philosophy of action; see work by Belnap, Perloff [**30**] and Xu [**31**]. In [**110**], the authors show that the STIT-framework can accommodate uniform strategies. Broersen, Herzig, Troquard [**43**] present an an extension of ATL, introducing ideas from STIT-theory, they present a translation from Pauly's Coalition Logic to Chellas' STIT logic in [**42**], and extend this work further in [**41**] to a translation of ATL into STIT.

In the following, we revise the definition of ATEL from Section 2.5.1 and integrate several suggestions to overcome some of the issues with ATEL. The revised version of ATEL is for incomplete information and perfect recall and, following Schobbens' [**219**] naming scheme, we denote it by ATEL_{iR}. To ensure consistency between an agent's knowledge and her strategic abilities, ATEL_{iR} only allows for uniform strategies. To make knowledge depend on the past, ATEL_{iR}-formulas are interpreted over histories rather than states. This follows ATEL-R* by Jamroga and van der Hoek [**125**] whose semantics is based on that of CTL* with past time [150], where a linear past is assumed. For reasoning about knowledge de re and de dicto, ATEL$_{iR}$ provides additional epistemic operators $\mathbb{K}$, $\mathbb{E}$, $\mathbb{C}$, $\mathbb{D}$ for "constructive" knowledge as suggested by Jamroga and Ågotnes [120, 119].

DEFINITION 2.37. (ATEL SYNTAX REVISITED). Let Π be a countable infinite set of atomic propositions and Σ a countable infinite set of agents. The set of ATEL-formulas φ is defined by the following BNF specification:

$$\begin{array}{lllllllllllll} \varphi & ::= & p & \mid & \neg\varphi & \mid & \varphi \vee \varphi & \mid & \langle\langle A\rangle\rangle \bigcirc \varphi & \mid & \langle\langle A\rangle\rangle \Box \varphi & \mid & \langle\langle A\rangle\rangle (\varphi \mathcal{U} \varphi) \mid \\ & & K_a\varphi & \mid & E_A\varphi & \mid & C_A\varphi & \mid & D_A\varphi & \mid & & & \\ & & \mathbb{K}_a\varphi & \mid & \mathbb{E}_A\varphi & \mid & \mathbb{C}_A\varphi & \mid & \mathbb{D}_A\varphi & & & & \end{array}$$

where p ranges over atomic propositions in Π, a over agents in Σ and $A \subset \Sigma$ over coalitions.

In order to be able to identify uniform strategies, formulas of the revised version of ATEL will be interpreted over AETS$^\sharp$s. Moreover, we want to overcome the problem that, in ATEL, knowledge is not dependent on what was previously known, i.e., the knowledge of an agent does not depend on previously traversed information states. To this end, we define the notion of an information history for agents. Consider an AETS$^\sharp$ $S = \langle \Pi, \Sigma, Q, \{\sim_a\}_{a\in\Sigma}, \pi, \delta, \sharp\rangle$. Recall that, given an epistemic accessibility relation $\sim_a$, the equivalence class $[q]_{\sim_a}$ of a state $q \in Q$ is the information state of agent a at q. A finite, non-empty sequence of states $h = q_1q_2\dots q_n \in Q^+$ is a *history* if, for all positions $1 \leq i \leq n$, there is a choice $\{q_{i+1}\} \in \delta(q_i, \Sigma)$ (i.e., q_{i+1} is a successor of q_i). The final state q_n of the history h is denoted with $\hat{h}$. Intuitively, histories are finite computations up to the current state. The *information history* for agent a of a history (or finite computation) $\lambda = q_1q_2\dots q_n \in Q^+$ with $n \geq 1$ is the sequence $h_a(\lambda) = Q_1Q_2\dots Q_n \in 2^{Q^+}$ of sets of states such that $Q_i = [q_i]_{\sim_a}$, for each i with $1 \leq i \leq n$. In the literature (e.g. [250]), usually two kinds of information histories are distinguished: synchronous and asynchronous. Intuitively, an information history of an agent is *synchronous* if the agent can observe a global clock, i.e., she can detect system transitions although her information state does not change. An information history for an agent is *asynchronous* if the agent cannot observe a global clock, i.e., all adjacent information states in the sequence must be different from each other. Here we will focus on synchronous information states.

We now define a notion of equivalence between information histories. Two histories λ and λ' are equivalent according to a's knowledge, written $\lambda \approx_a \lambda$, if they give rise to the same information histories for a. Formally,

$$\lambda \approx_a \lambda' \text{ iff } h_a(\lambda) = h_a(\lambda').$$

This means that, given $\lambda \approx_a \lambda'$, (in the synchronous setting) the histories are of the same length, i.e., $|\lambda| = |\lambda'|$, and that the information histories $h_a(\lambda)$, $h_a(\lambda')$ for a agree on every information state. Notice that $\approx_a$ is an equivalence relation and, therefore, it is suitable for interpreting it as epistemic accessibility relation, cf. Section 2.2. To account for the combined knowledge of several agents, we define three epistemic accessibility relations in terms of $\approx_a$ corresponding to "everyone knows", common and distributed knowledge similarly to combined knowledge in S5 in Section 2.2.3: For all coalitions C, we set

$$\begin{aligned} \approx_A^E &:= \bigcup_{a\in A} \approx_a; \\ \approx_A^C &:= (\approx_A^E)^+; \\ \approx_A^D &:= \bigcap_{a\in A} \approx_a . \end{aligned}$$

We redefine the notion of a strategy to account for agents under incomplete information. A *strategy* for an agent $a \in \Sigma$ is a mapping $\sigma_a : (2^Q)^+ \rightarrow 2^Q$ such that for all histories $\lambda \in Q^+$, the choice $\sigma_a(h_a(\lambda))$ is in $\delta(\hat{h}, a)$. A *strategy* for agents in a coalition $C \subseteq \Sigma$ is a set of strategies $\sigma_C = \{\sigma_a \mid a \in C\}$, one for each agent in C.

We now redefine the outcomes of a strategy in an AETS$^\sharp$ wrt. a set of histories. Let $H \subseteq Q^+$ be a set of histories. The set $\mathsf{out}(H, \sigma_C)$ of *outcomes* of a strategy σ_C starting with a set of possible histories H is a set of pairs (λ, i), where $\lambda \in Q^*$ is a computation and $i \geq 0$ a position, such that $\lambda[0, i] = h$, for some history $h \in H$, and $\lambda[j+1] \in \bigcap_{\sigma_a \in \sigma_C} \sigma_a(h_a(\lambda[0, j]))$, for all positions $j \geq i$.

DEFINITION 2.38. (ATEL$_{iR}$ SEMANTICS). Given an AETS$^\sharp$ $\mathcal{S} = \langle \Pi, \Sigma, Q, \{\sim_a\}_{a\in\Sigma}, \pi, \delta, \sharp\rangle$, the satisfaction relation $\models$ is inductively defined as follows where H ranges over sets of histories in $\mathcal{S}$, $C \subseteq \Sigma$ over coalitions of agents, a over agents in Σ, $\mathcal{K}$ ranges over K, E, C, D and $\mathfrak{K}$ over $\mathbb{K}, \mathbb{E}, \mathbb{C}, \mathbb{D}$, and φ, φ_1 and φ_2 range over ATEL-formulas:

- $\mathcal{S}, H \models p$ iff $p \in \pi(\hat{h})$, for all histories $h \in H$ and all propositions $p \in \Pi$;
- $\mathcal{S}, H \models \neg\varphi$ iff $\mathcal{S}, H \not\models \varphi$;
- $\mathcal{S}, H \models \varphi_1 \vee \varphi_2$ iff $\mathcal{S}, H \models \varphi_1$ or $\mathcal{S}, H \models \varphi_2$;
- $\mathcal{S}, H \models \langle\!\langle C \rangle\!\rangle \bigcirc \varphi$ iff there is an uniform strategy σ_C such that for all outcomes $(\lambda, i) \in \mathsf{out}(H, \sigma_C)$, it holds that $\mathcal{S}, \{\lambda[i+1]\} \models \varphi$;
- $\mathcal{S}, H \models \langle\!\langle C \rangle\!\rangle \Box \varphi$ iff there is an uniform strategy σ_C such that for all outcomes $(\lambda, i) \in \mathsf{out}(H, \sigma_C)$, it holds that $\mathcal{S}, \{\lambda[j]\} \models \varphi$ for all positions $j \geq i$;
- $\mathcal{S}, H \models \langle\!\langle C \rangle\!\rangle \varphi_1 \mathcal{U} \varphi_2$ iff there is an uniform strategy σ_C such that for all outcomes $(\lambda, i) \in \mathsf{out}(H, \sigma_C)$, there is a position $j \geq i$ such that $\mathcal{S}, \{\lambda[j]\} \models \varphi_2$ and $\mathcal{S}, \{\lambda[k]\} \models \varphi_1$ for all positions k with $i \leq k < j$;
- $\mathcal{S}, H \models \mathcal{K}_A \varphi$ iff $\mathcal{S}, \{h\} \models \varphi$ for all histories h, h' with $h \approx_A^{\mathcal{K}} h'$ and $h' \in H$;
- $\mathcal{S}, H \models \mathfrak{K}_A \varphi$ iff $\mathcal{S}, H' \models \varphi$ where $H' = \bigcup_{h\in H} [h]_{\approx_A^{\mathfrak{K}}}$.

If for some set H of histories of some AETS$^\sharp$ $\mathcal{S}$ it holds that $\mathcal{S}, H \models \varphi$, then the ATEL-formula φ is *true* or *satisfied* at q, and $\mathcal{S}$ is called a *model* of φ.

To better understand how ATEL$_{iR}$ works and to see that it overcomes many of the ATEL-problems, consider the following example.

EXAMPLE 2.39. (KNOWLEDGE ABOUT ABILITY IN ATEL$_{iR}$). This example illustrates how we can express properties of an agent's knowledge *de re* and *de dicto* depending on her information history using ATEL$_{iR}$. Figure 2.9 depicts a section of an AETS$^\sharp$ for an agent a. The arrows denote the system transitions. The dashed ellipses contain the states that are epistemically indistinguishable for a. The labelled boxes correspond to the choices of a: the labels '$a\sharp1$' and '$a\sharp2$' respectively mark the first and the second choice available to a at the previous state. Suppose the system is currently

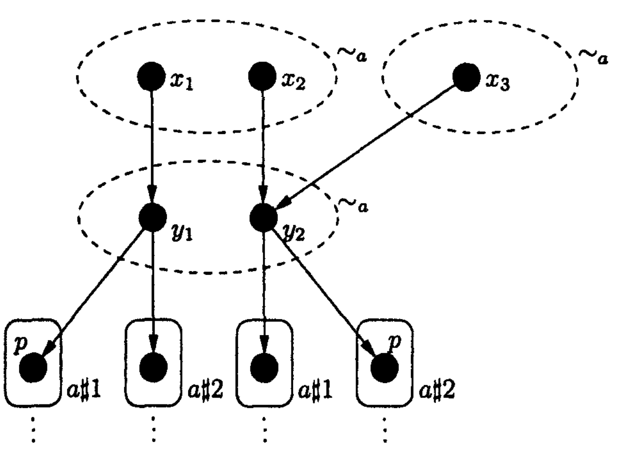


FIGURE 2.9. AETS$^\sharp$ for one agent with imperfect information.

in the state y_1 and agent a can recall one previous state, i.e., the current history is $h_1 = x_1y_1$. Notice that a cannot distinguish between the histories h_1 and $h_2 = x_2y_2$ since both give rise to the same information history for a, i.e., $h_1 \approx_a h_2$. But a can distinguish $h_3 = x_3y_2$, i.e., $h_3 \not\approx_a h_i$, for $i \in \{1,2\}$. We can express the fact that agent a has a strategy at h_1 to bring about the next state of the system to satisfy p with $\{h_1\} \models \langle\langle a\rangle\rangle \bigcirc p$. The same holds for h_2. Notice, however, that the strategies at h_1 and h_2 differ: for bringing about p at the next state, a needs to choose the choice $a\sharp1$ at h_1, and the choice $a\sharp2$ at h_2. So, what can be said about a's knowledge at h_1? Agent a knows that she has a strategy to ensure p at the next state, but she does not know the right strategy in the current situation, which can be h_1 or h_2. That is, a has knowledge *de dicto* at h_1 of being able to bring about p at the next state, but not knowledge *de re*. We can formalise the *de dicto* part with $\{h_1\} \models K_a\langle\langle a\rangle\rangle \bigcirc p$ and the *de re* part with $\{h_1\} \not\models \mathbb{K}_a\langle\langle a\rangle\rangle \bigcirc p$. Notice that the situation is different at h_3: here we have $\{h_3\} \models \mathbb{K}_a\langle\langle a\rangle\rangle \bigcirc p$. ⊣

Model checking under incomplete information and perfect information can lead to undecidability. An undecidability result of model checking epistemic modal logic with common knowledge under incomplete information and perfect recall was provided by van der Meyden [250]. In this work model checking S5$_n$ with common knowledge under incomplete information and perfect recall is PSPACE-complete for the synchronous case (i.e., with a global clock visible to all agents) and undecidable for the asynchronous case. The undecidability result is based on a reduction of the Halting problem of Turing machines. Azhar, Peterson and Reif [**22, 23**] show that multi-player games of incomplete information can be undecidable, unless some restrictions apply to the distribution of information. The authors model multi-player games of incomplete information by extending the alternating Turing machine of Chandra, Kozen and Stockmeyer [**49**] and the alternating machines by Reif [**211**]. The model checking problem for ATL with incomplete information is claimed to be undecidable by Alur *et al.* [**15**]. For a special case in the setting of Fair-ATL with incomplete information, Alur *et al.* refer to some undecidability results on asynchronous multi-player games with incomplete information [**199, 201**]. Although a formal proof of this statement has not been published, the ATL model checking problem under incomplete information and perfect recall seems widely believed to be undecidable [**219, 125, 165, 120**]. Model checking ATL with incomplete information was done, e.g., by Lomuscio and Raimondi [**165**], Jamroga and Dix [**123, 124**] and Schobbens [**219**].

In the remainder of this section, we cite some literature that appears to be relevant for capturing the concept of rationality with modal logic. ATL-like logics provide reasoning about strategic abilities of agents, but they do not account for rationality. Intuitively, rational agents should act in a way as it serves best their own interests. Clearly, modelling rational behaviour is interesting to many research areas such as economics, philosophy, cognitive science, computer science and AI. Van der Hoek and Wooldridge [**247**] present an article surveying over theories for rational agency including the Cohen and Levesque's [**59**] intention logic, Rao and Georgeff's BDI logics [**209, 210**], and the KARO framework of Linder, van der Hoek and Meyer [**253**]. A modal characterisation of the game-theoretical concept of Nash equilibrium was provided by Harrenstein, van der Hoek, Meyer and Witteveen [**108**]. Van Otterloo and Jonker [**254**] introduce an ATEL-variant Epistemic Temporal Strategic Logic (ETSL) which captures properties of agents acting rationally using a notion of *strategy domination* to express that a goal can be achieved better in one way than another. Later, Jamroga [**117, 118**] revised the semantics for ETSL and showed that a rational player knows that she will succeed if, and only if, she knows how to play to succeed, which is not true for rational coalitions of players. The link between model logics of games and logics of rational agency, where agents are characterised in terms of attitudes such as belief, desire and intention, was investigated by Jamroga, van der Hoek, Wooldridge [**126, 127**].

2.6. Strategic Logic for Agents with Limited Memory

In this section, we introduce a new ATL-variant that accounts for reasoning about strategic abilities of agents with limited amount of memory in the setting of incomplete information and imperfect recall.

Currently, one of the goals of the agent literature is to find an appropriate framework for describing multi-agent systems under incomplete information. Many ATL-variants have been suggested (cf. the previous section) to overcome the inherent problems with the original version of ATEL [272]. One issue is the amount of resources available to an agent. It appears to be a reasonable assumption to restrict the amount of memory available to an agent as any real computer system is equipped with a certain finite amount of memory. The agent literature suggests to limit the memory of agents and proposes to base agents' decision making on *imperfect-* rather than *perfect recall*; cf., e.g., the logic ATOL by Jamroga and van der Hoek [125]. Remember that perfect recall means that an agent memorizes the entire sequence of the previously traversed states (the history), no matter how many states this might be. Strategies in the perfect recall setting are based on finite but unbounded histories. To memorize such strategies the agent needs an unlimited amount of memory at her disposal. Imperfect recall, however, only allows for strategies that are based on histories of a certain length. For instance, the ATL-variants ATL_{Ir}, ATL_{ir} introduced by Schobben's [219] and ATEL-R* introduced by Jamroga and van der Hoek [125] account for imperfect recall. The first two logics use historyless strategies, i.e., the length of the history is reduced to one such that choices merely dep end on the current state.

Actually, adopting imperfect recall, i.e. basing strategies on bounded histories, is just half the way to limit the memory consumption of an agent. This approach does not account for the fact that storing a strategy even in the imperfect recall setting can require a lot of memory. As a strategy specifies a choice for every state in a model, the size of a strategy grows linearly with the number of states in the considered model. Since models can have many states, the required memory for storing an entire strategy may easily exceed the memory available to an agent.

In the following, we introduce a new family of ATL-variants called *Alternating-time Temporal Logic with Bounded Memory* (ATLBM) for reasoning about strategic abilities of agents under incomplete information, with imperfect recall and limited amount of memory. We adopt Schobben's [219] naming scheme and write ATLBM^m_{ir}, where the letter '*i*' in the subscript indicates incomplete information, the letter '*r*' limited recall and the natural number '*m*' (with $m \geq 0$) in the superscript stands for the number of previously traversed states an agent can recall. The language of ATLBM extends the language of ATL in that the ATL path quantifiers are additionally parameterized with a natural number that indicates the amount of memory available to the agents. An ATLBM-formula of the form $\langle\langle a \rangle\rangle^n \Phi$ means that agent a can bring about the temporal expression Φ, where a has a memory of size n. That is, agent a has a strategy of size n to enforce Φ, or, equivalently, a can achieve Φ without having to account for more than n different situations (or n of her information states).

For an overview on the logical omniscience problem and finite memory, see, e.g., Ågotnes [5]. Notice the difference to the study of resource-bounded agency, where the resource-bounded reasoning of agents is modelled. Agents are assumed to be reasoners with a given space and time bound. The question that arises is how much memory an agent needs to derive a formula, which is of theoretical and practical interest. The former is concerned about the deductive strength of a particular logic, and the latter about the memory requirements of agents to achieve certain goals. For more information, we refer to the work by Duc [**67**], Alechina and Logan [**11**] and Whitsey [**12**], and Jago [**114**]. For a general overview on modelling bounded rationality, see, e.g., Rubinstein [**214**].

2.6.1. ATLBM. In the following, we define the syntax of ATLBM^m_{ir}. An extension with knowledge operators should be straightforward.

DEFINITION 2.40. (ATLBM^m_{ir} SYNTAX). Let $\mathbf{\Pi}$ be a countable infinite set of atomic propositions and $\mathbf{\Sigma}$ a countable infinite set of agents. A *coalition* is a finite set $C \subset \mathbf{\Sigma}$ of agents. The language of ATLBM^m_{ir} is defined using the following BNF specification, where p ranges over atomic propositions in $\mathbf{\Pi}$, the coalition C ranges over finite subsets of $\mathbf{\Sigma}$ and n ranges over natural numbers. The set of ATLBM^m_{ir}-formulas φ is defined as:

$$\varphi \ ::= \ p \ \mid \ \neg\varphi \ \mid \ \varphi \vee \varphi \ \mid \ \langle\!\langle C\rangle\!\rangle^n \bigcirc\varphi \ \mid \ \langle\!\langle C\rangle\!\rangle^n \Box\varphi \ \mid \ \langle\!\langle C\rangle\!\rangle^n (\varphi\, \mathcal{U}\, \varphi)$$

The difference to the language of ATL is that each path quantifier $\langle\!\langle C\rangle\!\rangle^n$ in ATLBM^m_{ir} is additionally parameterised with a natural number n, which determines the amount of memory available to the coalition C. For instance, this enables us to express with the formula

$$\langle\!\langle a\rangle\!\rangle^{n+1}\Diamond\varphi \ \wedge \ \neg\langle\!\langle a\rangle\!\rangle^{n}\Diamond\varphi$$

the fact that agent a with memory of size $n+1$ can bring about eventually φ while she cannot achieve that only with memory of size n at her disposal.

The semantic structures of ATLBM^m_{ir} are $\mathrm{AETS}^\sharp$ as introduced in Definition 2.34. Notice that, as for ATL, we could also use concurrent game structures to interpret ATLBM^m_{ir}-formulas which would yield an equivalent semantics; see Section 2.4.1 for a similar discussion about ATL semantics. We now refine the notions of a strategy and the outcomes of a strategy to account for agents with incomplete information and imperfect recall. Consider an $\mathrm{AETS}^\sharp$ $\mathcal{S} = \langle \Pi, \Sigma, Q, \{\sim_a\}_{a\in\Sigma}, \pi, \delta, \sharp\rangle$. Given two natural numbers m and n, an *m-step strategy with n decisions* for an agent $a \in \Sigma$ is a partial function $\sigma_a : (\mathbf{2}^Q)^* \to \mathbf{2}^Q$ such that for at most n histories $\lambda_1, \ldots, \lambda_{n'} \in Q^+$, $n' \leq n$, with pairwise distinct information histories for a, i.e., $h_a(\lambda_i) \neq h_a(\lambda_j)$ for all $i, j \leq n'$ with $i \neq j$, the set $\sigma_a(h_a(\lambda_i))$ of states is a choice in $\delta(\hat{\lambda}_i, a)$. Notice that, in the setting of incomplete information, an agent a can only observe information states, i.e., states modulo the epistemic accessibility relation $\sim_a$ for a. This carries over to sequences of states: an agent observes information histories instead of histories. Two distinct histories λ_1 and λ_2 (with $\lambda_1 \neq \lambda_2$) cannot be distinguished by a if they give rise to the same information history for a, i.e., $h_a(\lambda_1) = h_a(\lambda_2)$. Consequently, agent a is not supposed to be able to make different decisions at λ_1 and λ_2. This intuition is reflected by the requirement of 'pairwise distinct information histories' in the definition of an m-step strategy with n decisions. Also notice that an 'm-step strategy' is a strategy that takes histories up to length m into account. Since the logic ATLBM^m_{ir} is parameterised with m, every agent uses m-step strategies and, thus, we assume that every agent has the same capability to recall at most $m-1$ previous states. Intuitively, an 'n-decision strategy' is a strategy that determines a choice at not more than n different information states/information histories. Therefore, a strategy with n decisions can be stored by an agent with memory of size n.

A m-step strategy σ_a with n decisions for agent a is called *uniform* if for all histories $\lambda = q_1 \dots q_n$ and $\lambda' = q'_1 \dots q'_n$ of the same length $n > 0$ that give rise to the same information histories, i.e., $h_a(\lambda) = h_a(\lambda')$, it holds that

$$\sharp(\hat{\lambda}, a, \sigma_a(h_a(\lambda))) = \sharp(\hat{\lambda}', a, \sigma_a(h_a(\lambda'))).$$

A *uniform m-step strategy with n decisions* for a coalition C is a set of uniform m-step strategies with n decisions $\sigma_C = \{\sigma_a \mid a \in C\}$, one for each agent in C.

For defining the outcomes of a strategy, we introduce an auxiliary notion for chopping off a history of length at most m of an infinite computation. Given a computation $\lambda = q_0 q_1 q_2 \cdots \in Q^\omega$ and a position $i \geq 0$ in λ, the history of length $m \geq 0$ determined by λ and i is the sequence $h_{\leq m}(\lambda, i)$ of at most m states defined as follows:

$$h_{\leq m}(\lambda, i) := \begin{cases} q_0 q_1 \dots q_i & \text{if } i < m; \\ q_{i-m+1} \dots q_{i-1} q_i & \text{if } i \geq m. \end{cases}$$

The set $\mathsf{out}(h, \sigma_C)$ of *outcomes* of a strategy σ_C starting at a history h is the set of all pairs (λ, i), where $\lambda \in Q^\omega$ is a computation and $i \geq 0$ a position in λ, such that for all positions $j \geq i$ and all strategies $\sigma_a \in \sigma_C$:

- $\lambda[0, i] = h$, and
- $\lambda[j+1] \in \sigma_a(h_a(h_{\leq m}(\lambda, j)))$ if σ_a is defined for $h_a(h_{\leq m}(\lambda, j))$.

The semantics for ATLBM$^m_{ir}$ is similar to the revised semantics of ATEL in Definition 2.38. One difference is that ATLBM$^m_{ir}$ does not have operators for knowledge and, thus, formulas are interpreted over single histories instead of sets of histories. Another difference is that ATLBM$^m_{ir}$ models imperfect recall of agents, i.e., the histories are limited up to length m.

DEFINITION 2.41. (ATLBM$^m_{ir}$ SEMANTICS). Given an AETS$^\sharp$ $\mathcal{S} = \langle \Pi, \Sigma, Q, \{\sim_a\}_{a \in \Sigma}, \pi, \delta, \sharp \rangle$, the satisfaction relation $\models$ is inductively defined as follows, where h ranges over histories in $\mathcal{S}$, $C \subseteq \Sigma$ ranges over coalitions of agents, a over agents in Σ, and φ, φ_1 and φ_2 range over ATLBM$^m_{ir}$-formulas:

- $\mathcal{S}, h \models p$ iff $p \in \pi(\hat{h})$, for all propositions $p \in \Pi$;
- $\mathcal{S}, h \models \neg\varphi$ iff $\mathcal{S}, h \not\models \varphi$;
- $\mathcal{S}, h \models \varphi_1 \vee \varphi_2$ iff $\mathcal{S}, h \models \varphi_1$ or $\mathcal{S}, h \models \varphi_2$;
- $\mathcal{S}, h \models \langle\!\langle C \rangle\!\rangle^n \bigcirc \varphi$ iff there is an uniform m-step strategy σ_C with n decisions such that for all outcomes $(\lambda, i) \in \mathsf{out}(h, \sigma_C)$, it holds that $\mathcal{S}, h_{\leq m}(\lambda, i+1) \models \varphi$;
- $\mathcal{S}, h \models \langle\!\langle C \rangle\!\rangle^n \Box \varphi$ iff there is an uniform m-step strategy σ_C with n decisions such that for all outcomes $(\lambda, i) \in \mathsf{out}(h, \sigma_C)$, it holds that $\mathcal{S}, h_{\leq m}(\lambda, j) \models \varphi$ for all positions $j \geq i$;

- $S, h \models \langle\!\langle C\rangle\!\rangle^n \varphi_1 \,\mathcal{U}\, \varphi_2$ iff there is an uniform m-step strategy σ_C with n decisions such that for all outcomes $(\lambda, i) \in \mathrm{out}(h, \sigma_C)$, there is a position $j \geq 0$ such that $S, h_{\leq m}(\lambda, j) \models \varphi_2$ and $S, h_{\leq m}(\lambda, k) \models \varphi_1$ for all positions k with $i \leq k < j$.

If for some set H of histories of some AETS$^\sharp$ S it holds that $S, q \models \varphi$, then the ATEL-formula φ is *true* or *satisfied* at q, and S is called a *model* of φ.

To understand ATLBM$^m_{ir}$ better, consider the following example.

EXAMPLE 2.42. (EXPRESSING PROPERTIES WITH ATLBM$^m_{ir}$). This example illustrates how ATLBM$^m_{ir}$ can be used to express properties of a MAS in which agents' memory is limited. Figure 2.10 shows a section of an AETS$^\sharp$ for agent a. The arrows denote the system transitions. The labelled boxes denote a's choices: the label '$a\sharp1$' marks the first choice of a at the previous state and '$a\sharp2$' the second choice. The dashed ellipse comprises the states which are epistemically indistinguishable for agent a. Note that a can distinguish the other states. Suppose a's decisions are solely based on her current information state, i.e., a cannot recall any previously traversed states. Consequently, to express statements about agent a, we use the logic ATLBM$^1_{ir}$. The

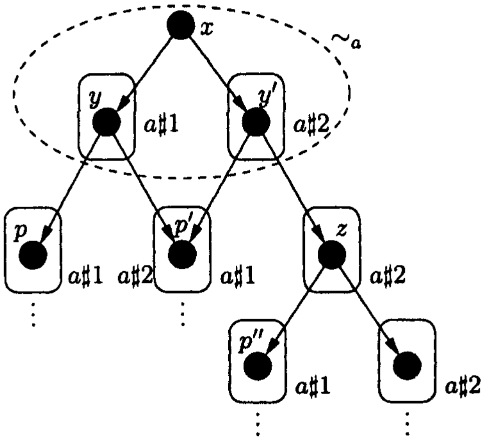


FIGURE 2.10. An AETS$^\sharp$ for one agent with bounded memory.

system properties a can enforce depend on the memory available to a. First, consider the scenario where a has a memory of size one, i.e., a strategy for a specifies a decision at exactly one of a's information states. Suppose a decides at state x for the choice $a\sharp1$ which determines the next state of the system to be state y. Since a does not recall any history and the states x and y are epistemically indistinguishable for a, she again decides for the choice $a\sharp1$. Thus, the system enters a state at which p is satisfied. That is, at x, agent a can bring about p at some point in the future. Using ATLBM$^1_{ir}$, this property can be expressed as $x \models \langle\!\langle a\rangle\!\rangle^1 \Diamond p$. Contrary, agent a cannot eventually bring about p'' starting at state x. To see this, suppose a decides for the choice $a\sharp2$ at x which determines the next two system transitions and leads the system into state z. Notice that a can distinguish z from her previous information state. But this means that agent

a does not know which choice to take at z since her 1-decision strategy does not specify any choice. Therefore, a has to pick a choice at random which does not necessarily lead the system into the p''-state. This can be expressed with $x \not\models \langle\langle a\rangle\rangle^1 \Diamond p''$. However, if we increase the memory of agent a, she becomes capable of eventually bringing about p''. Consider the scenario where a has a memory of size two, i.e., a strategy for a specifies a decision at two of her information states. Now, a is able to choose the choice $a\sharp 1$ at state z and thus enforce p''. The property that, at x, agent a can enforce p'' at some point in the future can be formalised as $x \models \langle\langle a\rangle\rangle^2 \Diamond p''$. Adding more memory does not necessarily increase the agents' capability. For instance, suppose agent a has any finite amount of memory at her disposal. Given that the states x, y and y' are epistemically indistinguishable for a, it is readily checked that she will not be able to bring about p' starting from x. This can be formalised as $x \not\models \langle\langle a\rangle\rangle^n \Diamond p'$ for any $n \geq 0$. $\dashv$

2.6.2. Axioms. While ATL accounts for agents with unlimited memory, the agents in ATLBM$^m_{ir}$ have limited memory. The axioms in Table 2.11 characterise some interactions of the ATLBM$^m_{ir}$ path quantifiers which determine the amount of memory available to the agents. Note that, for the parameters n and m in Table 2.11, we have $n \geq 0$ and $m \geq 1$. Intuitively, the axioms $(\langle\langle A\rangle\rangle^0\bigcirc)$, $(\langle\langle A\rangle\rangle^0\Box)$ and $(\langle\langle A\rangle\rangle^0\mathcal{U})$ express the fact that

$(\langle\langle A\rangle\rangle^0\bigcirc)$	$\langle\langle A\rangle\rangle^0\bigcirc\varphi \leftrightarrow \langle\langle\emptyset\rangle\rangle^n\bigcirc\varphi$
$(\langle\langle A\rangle\rangle^0\Box)$	$\langle\langle A\rangle\rangle^0\Box\varphi \leftrightarrow \langle\langle\emptyset\rangle\rangle^n\Box\varphi$
$(\langle\langle A\rangle\rangle^0\mathcal{U})$	$\langle\langle A\rangle\rangle^0\varphi\mathcal{U}\psi \leftrightarrow \langle\langle\emptyset\rangle\rangle^n\varphi\mathcal{U}\psi$
$(\langle\langle A\rangle\rangle^1\bigcirc)$	$\langle\langle A\rangle\rangle^1\bigcirc\varphi \leftrightarrow \langle\langle A\rangle\rangle^{n+1}\bigcirc\varphi$
$(\langle\langle A\rangle\rangle^{+1}\Box)$	$\langle\langle A\rangle\rangle^n\Box\varphi \rightarrow \langle\langle A\rangle\rangle^{n+1}\Box\varphi$
$(\langle\langle A\rangle\rangle^{+1}\mathcal{U})$	$\langle\langle A\rangle\rangle^n\varphi\mathcal{U}\psi \rightarrow \langle\langle A\rangle\rangle^{n+1}\varphi\mathcal{U}\psi$

TABLE 2.11. Some axioms of ATLBM$^m_{ir}$.

coalitions of agents without memory can bring about as much as the empty coalition with arbitrary amount of memory. This is because, coalitions of agents without memory cannot store any strategy, i.e., they can only achieve what is inevitable in the system. Observe that an ATLBM$^m_{ir}$-formula of the form $\langle\langle C\rangle\rangle^0\Phi$ is equivalent to the formula $\langle\langle\emptyset\rangle\rangle\Phi$ in ATL (where strategies depend on histories of length at most m). The Axiom $(\langle\langle A\rangle\rangle^1\bigcirc)$ states a coalition of agents with memory of size one can enforce as much at the next system state as the same coalition where agents have more memory at their disposal. In other words, the amount of memory required for the agents in a coalition to enforce a property at the next system state is exactly one. The axioms $(\langle\langle A\rangle\rangle^{+1}\Box)$ and $(\langle\langle A\rangle\rangle^{+1}\mathcal{U})$ state that, with a larger amount of memory for each agent, a coalition can achieve at least as much, or more.

An axiom system for ATLBM$^m_{ir}$ could consist of the axioms in Table 2.11 together with the axiom system for ATL$_\Sigma$ in Table 2.8 where each ATL-axiom is adapted to

ATLBM$_{ir}^{m}$ by extending each path quantifier with an additional parameter denoting the amount of memory available to the agents. This adaptation is straightforward for the ATL-axioms and rules in Table 2.12, where $n \geq 0$.

$(\bot^n)$	$\neg\langle\langle A\rangle\rangle^n \bigcirc \bot$
$(\top^n)$	$\langle\langle A\rangle\rangle^n \bigcirc \top$
(Σ^n)	$\neg\langle\langle \emptyset\rangle\rangle^{n+1} \bigcirc \neg\varphi \rightarrow \langle\langle \Sigma\rangle\rangle^{n+1} \bigcirc \varphi$
(S^n)	$(\langle\langle A\rangle\rangle^n \bigcirc \varphi \wedge \langle\langle B\rangle\rangle^n \bigcirc \psi) \rightarrow \langle\langle A \cup B\rangle\rangle^n \bigcirc (\varphi \wedge \psi)$ where $A \cap B = \emptyset$
(Modus Ponens)	$\dfrac{\varphi, \varphi \rightarrow \psi}{\psi}$
($\langle\langle \emptyset\rangle\rangle^n \Box$-Necessitation)	$\dfrac{\varphi}{\langle\langle \emptyset\rangle\rangle^n \Box \varphi}$
($\langle\langle A\rangle\rangle^n \bigcirc$-Monotonicity)	$\dfrac{\varphi \rightarrow \psi}{\langle\langle A\rangle\rangle^n \bigcirc \varphi \rightarrow \langle\langle A\rangle\rangle^n \bigcirc \psi}$

TABLE 2.12. Some ATL-axioms and rules adapted to ATLBM$_{ir}^{m}$.

$(FP_{\Box}^{n})$	$\langle\langle A\rangle\rangle^n \Box\varphi \rightarrow \varphi \wedge \langle\langle A\rangle\rangle^1 \bigcirc \langle\langle A\rangle\rangle^n \Box\varphi$
$(FP_{\Box}^{n+1})$	$\langle\langle A\rangle\rangle^{n+1} \Box\varphi \leftarrow \varphi \wedge \langle\langle A\rangle\rangle^1 \bigcirc \langle\langle A\rangle\rangle^n \Box\varphi$
$(FP_{\mathcal{U}}^{n} \leftarrow$ - a)	$\langle\langle a\rangle\rangle^{n+1} \psi \mathcal{U} \varphi \rightarrow \varphi \vee (\psi \wedge (\langle\langle a\rangle\rangle^0 \bigcirc \langle\langle a\rangle\rangle^{n+1} (\psi \mathcal{U} \varphi) \vee \langle\langle a\rangle\rangle^1 \bigcirc \langle\langle a\rangle\rangle^n (\psi \mathcal{U} \varphi)))$
$(FP_{\mathcal{U}}^{n} \rightarrow)$	$\langle\langle A\rangle\rangle^{n+1} \psi \mathcal{U} \varphi \leftarrow \varphi \vee (\psi \wedge (\langle\langle A\rangle\rangle^0 \bigcirc \langle\langle A\rangle\rangle^{n+1} (\psi \mathcal{U} \varphi) \vee \langle\langle A\rangle\rangle^1 \bigcirc \langle\langle A\rangle\rangle^n (\psi \mathcal{U} \varphi))$

TABLE 2.13. Some ATL-axioms adapted to ATLBM$_{ir}^{m}$ where $m = 1$.

Notice that more care is needed when adapting the ATL-axioms (FP$_{\Box}$) and (FP$_{\mathcal{U}}$) since the length of the history the agents are assumed to recall is crucial. Table 2.13 contains some axioms for ATLBM$_{ir}^{m}$ where $m = 1$, i.e., for the case, where agents' decision are solely based on the current state. For each of the ATL-axioms (FP$_{\Box}$) and (FP$_{\mathcal{U}}$), we present two axioms: $(FP_{\Box}^{n})$ and $(FP_{\Box}^{n+1})$, and $(FP_{\mathcal{U}}^{n} \leftarrow$ - a) and $(FP_{\mathcal{U}}^{n} \rightarrow)$, respectively. Notice the the different parameters of the path quantifiers that indicate the memory available to the agents. Also note that the axiom $(FP_{\mathcal{U}}^{n} \leftarrow$ - a) only captures single agent coalitions.

We leave the complete axiomatisation of ATLBM$_{ir}^{m}$ for future work.

2.6.3. Expressivity. At first sight, the expressivity of ATLBM$_{ir}^{m}$ and ATL seem to be incomparable: ATLBM$_{ir}^{m}$ accounts for agents with incomplete information, imperfect recall and bounded memory, whereas ATL (as defined in Section 2.4.1) does neither of these. In order to enable a comparison, consider the logic ATLBM with perfect information and perfect recall, which we denote by ATLBM$_{IR}$. Notice that ATL and ATLBM$_{IR}$ are both interpreted over ATSs, and that, due to perfect recall, the superscript 'm' is not needed anymore. Consider the following equivalence which shows that ATL-formulas of the form $\langle\langle A\rangle\rangle\Phi$ would be equivalent to infinitely long formulas of ATLBM$_{IR}$. (Of course, infinite formulas are not allowed in ATLBM$_{IR}$.)

$$\langle\langle A\rangle\rangle\Phi \leftrightarrow \bigvee_{n\geq 0} \langle\langle A\rangle\rangle^n\Phi$$

To avoid infinitely long formulas, we can enrich the language of ATLBM with additional path quantifiers of the form $\langle\langle\,\cdot\,\rangle\rangle^*$. The semantics of the additional path quantifiers is defined as follows: for all AETS$^\sharp$ S, all states x of S, all ATLBM-formulas of the form $\langle\langle C\rangle\rangle^n\Phi$,

$$S, x \models \langle\langle C\rangle\rangle^*\Phi \text{ iff } S, x \models \langle\langle C\rangle\rangle^n\Phi \text{ for some natural number } n \geq 0.$$

Intuitively, $\langle\langle C\rangle\rangle^*\Phi$ states that coalition C can enforce the temporal property Φ with some finite amount of memory for strategies, no matter what the other agents do. ATL can be embedded into the enriched version of ATLBM$_{IR}$ using the following translation $(\cdot)^t$:

$$\begin{aligned}
(p)^t &:= p;\\
(\neg\varphi)^t &:= \neg\varphi;\\
(\varphi\vee\psi)^t &:= \varphi\vee\psi;\\
(\langle\langle C\rangle\rangle\bigcirc\varphi)^t &:= \langle\langle C\rangle\rangle^*\bigcirc\varphi;\\
(\langle\langle C\rangle\rangle\Box\varphi)^t &:= \langle\langle C\rangle\rangle^*\Box\varphi;\\
(\langle\langle C\rangle\rangle(\varphi\mathcal{U}\psi))^t &:= \langle\langle C\rangle\rangle^*(\varphi\mathcal{U}\psi).
\end{aligned}$$

2.6.4. Conclusion. We now conclude this section and give some suggestions for future work. We introduced a family ATLBM$^m_{ir}$ of ATL-variants for reasoning about strategic ability in MASs of resource-bounded agents. In particular, the setting of agents with limited memory appears to be closer to real-world applications. ATLBM$^m_{ir}$ considers agents under incomplete information, with imperfect recall and limited amount of memory for storing strategies. There are three more variants ATLBM$^m_{Ir}$, ATLBM$_{iR}$ and ATLBM$_{IR}$ that, respectively, account for complete information and imperfect recall, incomplete information and perfect recall, or complete information and perfect recall. It would be interesting to investigate the ATLBM-variants further, in particular, to compare their expressivity to each other and to ATL, to characterise their expressivity syntactically via axiom systems, or structurally via bisimulations, and to explore the computational complexities of their model checking and satisfiability problems. Moreover, it seems interesting to investigate epistemic extensions of ATLBM$^m_{ir}$, say ATELBM$^m_{ir}$, which would additionally facilitate reasoning about the agents' knowledge in a resource-bounded setting.

CHAPTER 3

Complexity of Reasoning in ATL/ATEL

ATL by Alur, Henzinger and Kupferman [**15**] is being increasingly widely applied in formal system specification and verification of distributed systems and game-like multi-agent systems. The extension ATEL by van der Hoek and Wooldridge [**272, 246**] of ATL with knowledge modalities widens the scope of reasoning over multi-agent systems with epistemic notions. In this chapter, we investigate the computational complexities of the satisfiability problem for ATL and ATEL with epistemic operators for common and distributed knowledge. For the case where the set of agents is fixed in advance, ATL satisfiability was settled at EXPTIME-complete by van Drimmelen [**252**], cf. Section 2.4.4 in Chapter 2. If the set of agents is not fixed in advance, this result yields a 2-EXPTIME upper bound. In this chapter, we focus on the latter case and, in Section 3.1.1, we define three natural variations of the satisfiability problem. Although none of these variations fixes the set of agents in advance, we are able to prove containment of ATL satisfiability in EXPTIME for all of them by means of a type elimination construction. We present the ATL decision procedure in Section 3.1.2. In Section 3.2.1, we expand the result and show containment of the satisfiability problem for ATEL with epistemic operators for common and distributed knowledge in EXPTIME. Thus ATEL is no more complex than ATL.

3.1. Complexity of ATL

Although the complexity of the model checking problem for ATL was classified in the very first publications on ATL [**13**], the complexity of the satisfiability problem was not considered by the developers of the logic. The fact that ATL is a generalisation of CTL (almost always) immediately gives an EXPTIME lower bound, but the question of whether or not ATL satisfiability was in EXPTIME was left open. By "almost always" we mean that, as we shall see later, some variants of the ATL satisfiability problem do not inherit EXPTIME-hardness from CTL in an immediate way.

For the case where the set of agents considered is defined in advance, the complexity of the satisfiability problem for ATL was settled in 2003 with an automata-based EXPTIME decision procedure by van Drimmelen [**252**]. The approach in [**252**] was to show that ATL satisfiability can be reduced to the non-emptiness problem for alternating Büchi tree automata. For the overall decision procedure to have exponential running time, the branching degree of the constructed trees has to be polynomial in the size of the input formula. In the proof in [**252**], the constructed trees have branching degree k^n, where n is the number of agents and k is polynomial in the size of the input formula. Thus, a polynomial branching degree, and hence an overall EXPTIME upper bound, is only obtained if the number of agents allowed to appear in input formulas is fixed beforehand, rather than being regarded as part of the input. Thus, the obtained EXPTIME result by van Drimmelen can be stated as follows.

THEOREM 3.1. *Suppose Σ is a fixed, finite set of agents. Then satisfiability of ATL-formulas based on Σ in an ATS over Σ is* EXPTIME-*complete.*

If input formulas may contain arbitrarily many agents, then n, the number of agents, *is clearly dependent on the input formula.* In this case, the branching degree of the constructed trees becomes exponential, and the decision procedure only yields a 2-EXPTIME upper bound. Thus, if we do not fix the set of agents in advance, the complexity of satisfiability in ATL was still open (until recently) – between EXPTIME and 2-EXPTIME. Note that the automata-based approach in [252] cannot be generalised by choosing a better tree construction since in ATL it is possible to devise a formula that enforces a branching degree exponential in the number of agents. To illustrate that this branching degree cannot easily be reduced, we exhibit a sequence of ATL-formulas $(\varphi_i)_{i\in\mathbb{N}}$ such that, for any ATS S, state q, and $i \geq 0$, $S, q \models \varphi_i$ implies that q has at least 2^i successors in S:

$$\varphi_i := \bigwedge_{1\leq j\leq i} (\langle\langle a_j\rangle\rangle \bigcirc p_j \wedge \langle\langle a_j\rangle\rangle \bigcirc \neg p_j)$$

As every agent a_i may choose the propositional letter p_i to be true or false at a successor state, jointly the agents $a_1, \ldots, a_k$ may choose any possible valuation of $p_1, \ldots, p_k$ for a successor state. As there are 2^i such valuations, there must be as many successors.

Considering ATL with an unbounded supply of agents, we find there are several different ways of framing the satisfiability problem with respect to the agents that can appear in both the formula and the structure that satisfies the formula. In particular, when not having fixed an agent set in beforehand, there are different possibilities for the number of agents that occur in an ATS over which a formula is to be interpreted. With this observation in mind, consider the following three formulations of the ATL satisfiability problem, where the set of agents is not fixed in advance:

(a) Given a finite set Σ of agents and a formula φ over Σ, is φ satisfiable in an ATS over Σ?
(b) Given a formula φ, is there a finite set Σ of agents (containing the agents referred to in φ) such that φ is satisfiable in an ATS over Σ?
(c) Given a formula φ, is φ satisfiable in an ATS over exactly the agents which occur in φ?

Note that the construction in [252] does not give an EXPTIME upper bound for any of these three variations. The main contribution of this work is thus to prove that all three variations are in fact EXPTIME-complete.

3.1.1. Varieties of Satisfiability for ATL.

The key syntactic difference between ATL and its predecessor CTL is that formulas of ATL explicitly refer to agents. We must be mindful of the way in which such agents are interpreted in the formulation of the ATL satisfiability problem. To better understand why, consider the following ATL-formula (adapted from [**194**, p.47]).

$$\neg\langle\langle a\rangle\rangle\bigcirc p \wedge \neg\langle\langle a\rangle\rangle\bigcirc q \wedge \langle\langle a\rangle\rangle\bigcirc(p \vee q)$$

This formula expresses the fact that, in the next state, agent a cannot make p true, and cannot make q true; but it can make either p or q true. Now the question is whether this formula is satisfiable. The answer depends on the range of ATSs we are prepared to consider. If we admit *arbitrary* ATSs as witness to its satisfiability, then the answer is yes: one can easily construct an ATS containing two or more agents that satisfies it. However, suppose we only consider ATSs that contain at most *one* agent. By virtue of the fact that the formula is well-formed, the agent in the structure must be a. But then the choice sets for this agent must be singletons, and it is then easy to see that the formula could not be satisfied in such a model. So: the agents in the structures we are prepared to consider *are* important in determining the satisfiability or otherwise of a formula, and even unknown agents that are not referred to in a formula can play a part in determining whether or not the formula is satisfied in a structure. Notice that the presence of unknown agents introduces an element of non-determinism, as the agents that occur in a formula cannot completely determine the behaviour of the system anymore.

With these concerns in mind, consider the following three variations of satisfiability for ATL.

(a) *Satisfiability over given sets of agents*:
Given a finite set Σ of agents and a formula φ over Σ, is φ satisfiable in an ATS over Σ?

(b) *Satisfiability over arbitrary sets of agents*:
Given a formula φ, is there a finite set Σ of agents (containing the agents referred to in φ) such that φ is satisfiable in an ATS over Σ?

(c) *Satisfiability over formula-defined sets of agents*:
Given a formula φ, is φ satisfiable in an ATS over exactly the agents which occur in φ?

Notice that in none of these variations of the problem is the set of agents fixed externally, in the problem definition, as in the case of Theorem 3.1. Moreover, the construction of van Drimmelen [**252**] does not give an EXPTIME upper bound for any of these variations: the automata theoretic algorithm presented in [**252**] yields a 2-EXPTIME upper bound for all three cases. In variants (a) and (c), the sets of agents are given *as part of the input*, while in Variant (b), we are asked whether an ATS over an *arbitrary* set of agents exists such that this structure satisfies the formula. Our main result is as follows.

THEOREM 3.2. *The variants (a), (b) and (c) of the satisfiability problem for ATL are* EXPTIME*-complete.*

Section 3.1.2 is largely devoted to the proof of the ExpTime upper bound. The lower bound will be discussed briefly at the end of Section 3.1.2. We begin by showing that it suffices to prove the upper bound for Variant (c), as the other two cases may be reduced to this. Notice that with Σ_φ we denote the set of agents that occur in the formula φ.

Lemma 3.3. *The variants (a) and (c) of the satisfiability problem for ATL are polynomially reducible to each other, while Variant (b) is polynomially reducible to (a). In fact, we even have the stronger property that, for each formula φ and each set of agents $\Sigma \supset \Sigma_\varphi$, φ is satisfiable in an ATS for Σ iff it is satisfiable in an ATS for $\Sigma_\varphi \cup \{a\}$, for one fresh agent a.*

Proof. Note that (c) is a special case of (a), where Σ coincides with the set of agents Σ_φ which occur in φ. Conversely, given a finite set Σ and φ from (a), conjunctively add to φ any valid formula ψ containing exactly the agents from Σ which do not occur in φ. Then φ is satisfiable in an ATS for Σ iff $\varphi \wedge \psi$ is satisfiable in an ATS for the agents which occur in $\varphi \wedge \psi$. It thus remains to prove the second part, i.e., that for each formula φ and each set of agents $\Sigma \supset \Sigma_\varphi$, φ is satisfiable in an ATS for Σ iff it is satisfiable in an ATS for $\Sigma_\varphi \cup \{a\}$, for one fresh agent a.

"$\Rightarrow$": Suppose $\mathcal{S}$ is an ATS for $\Sigma \supset \Sigma_\varphi$ such that $\mathcal{S}$ satisfies φ. We convert $\mathcal{S}$ into an ATS $\mathcal{S}'$ for $\Sigma_\varphi \uplus \{a\}$ that also satisfies φ. Define the transition function δ' of $\mathcal{S}'$ in terms of δ in $\mathcal{S}$ as follows: for all $q \in Q$,

- $\delta'(q, a') := \delta(q, a')$ for each $a' \in \Sigma_\varphi$, and
- $\delta'(q, a) := \delta(q, \Sigma \setminus \Sigma_\varphi)$.

It is easy to show by structural induction that, for all $q \in Q$ and all formulas ψ using only agents from the set Σ_φ, we have $\mathcal{S}, q \models \psi$ iff $\mathcal{S}', q \models \psi$. Thus, $\mathcal{S}'$ is a model of φ as required.

"$\Leftarrow$": Suppose $\mathcal{S}$ is an ATS for $\Sigma_\varphi \uplus \{a\}$ such that $\mathcal{S}$ satisfies φ. Let $a' \in \Sigma \setminus \Sigma_\varphi$. We convert $\mathcal{S}$ into an ATS $\mathcal{S}'$ for Σ such that $\mathcal{S}'$ still satisfies φ. Define δ' of $\mathcal{S}'$ in terms of δ in $\mathcal{S}$ as follows: for all $q \in Q$,

- $\delta'(q, a'') = \delta(q, a'')$ for each $a'' \in \Sigma_\varphi$,
- $\delta'(q, a') = \delta(q, a)$, and
- $\delta'(q, a'') = \{Q\}$ for each $a'' \in \Sigma \setminus (\Sigma_\varphi \cup \{a'\})$.

Again, it is easy to show by structural induction that, for all $q \in Q$ and all formulas ψ using only agents from the set Σ_φ, we have $\mathcal{S}, q \models \psi$ iff $\mathcal{S}', q \models \psi$. □

3.1.2. ATL Decision Procedure. This section is devoted to the proof of Theorem 3.2. The main result is containment in ExpTime of Problem (c) from the previous section, i.e., satisfiability of ATL-formulas φ in ATSs over exactly the agents occurring in φ. By Lemma 3.3, this yields ExpTime upper bounds also for problems (a) and (b). The ExpTime lower bounds for (a) and (c) are immediate by reduction of CTL as sketched after the definition of ATL in Section 2.4.1 of Chapter 2. To establish an ExpTime lower bound for Problem (b), we will reduce the global consequence problem in the modal logic K.

We start with the ExpTime upper bound for Problem (c). Our approach is to use a type elimination construction similar to the one commonly used for CTL by Emerson and Halpern [**72, 69**]. One advantage of this approach is that it is constructive: if the input formula φ is satisfiable, then the proof actually constructs a model of φ. Moreover, this model is finite and of bounded size: the number of states is at most exponential in the length of φ. Thus, our algorithm can be used, e.g., for the synthesis of social procedures as sketched in Example 2.24 in Section 2.4.1 of Chapter 2. We should note that recently a similar construction has been independently developed by Goranko and van Drimmelen [**98**]. However, Goranko and van Drimmelen use their construction to prove completeness of an ATL axiomatisation rather than for obtaining upper complexity bounds.

We start the presentation of our decision procedure with a number of definitions.

Definition 3.4 (Extended Closure). Let φ be an ATL-formula. The *extended closure* $\mathsf{ecl}(\varphi)$ of φ is the smallest set which is closed under the following conditions:

- $\varphi \in \mathsf{ecl}(\varphi)$;
- $\mathsf{ecl}(\varphi)$ is closed under subformulas;
- $\mathsf{ecl}(\varphi)$ is closed under single negation;
- if $\langle\langle A\rangle\rangle\Box\psi \in \mathsf{ecl}(\varphi)$, then $\langle\langle A\rangle\rangle\bigcirc\langle\langle A\rangle\rangle\Box\psi \in \mathsf{ecl}(\varphi)$;
- if $\langle\langle A\rangle\rangle\psi\,\mathcal{U}\,\vartheta \in \mathsf{ecl}(\varphi)$, then $\langle\langle A\rangle\rangle\bigcirc\langle\langle A\rangle\rangle\psi\,\mathcal{U}\,\vartheta \in \mathsf{ecl}(\varphi)$.

It is easy to verify that for a given ATL-formula φ, the cardinality of the extended closure $\mathsf{ecl}(\varphi)$ is linear in the length of φ.

Definition 3.5 (ATL Type). Let φ be an ATL-formula. The set $\Psi \subseteq \mathsf{ecl}(\varphi)$ is a *type for* φ if the following conditions are satisfied:

(T1) $\psi_1 \vee \psi_2 \in \Psi$ iff $\psi_1 \in \Psi$ or $\psi_2 \in \Psi$, for all $\psi_1 \vee \psi_2 \in \mathsf{ecl}(\varphi)$;

(T2) $\psi \in \Psi$ iff $\neg\psi \notin \Psi$, for all $\neg\psi \in \mathsf{ecl}(\varphi)$;

(T3) $\langle\langle A\rangle\rangle\Box\psi \in \Psi$ iff $\{\psi, \langle\langle A\rangle\rangle\bigcirc\langle\langle A\rangle\rangle\Box\psi\} \subseteq \Psi$, for all $\langle\langle A\rangle\rangle\Box\psi \in \mathsf{ecl}(\varphi)$;

(T4) $\langle\langle A\rangle\rangle\psi\,\mathcal{U}\,\vartheta \in \Psi$ iff $\vartheta \in \Psi$ or $\{\psi, \langle\langle A\rangle\rangle\bigcirc\langle\langle A\rangle\rangle\psi\,\mathcal{U}\,\vartheta\} \subseteq \Psi$, for all $\langle\langle A\rangle\rangle\psi\,\mathcal{U}\,\vartheta \in \mathsf{ecl}(\varphi)$.

The set of all ATL types for φ is designated by Γ_φ.

Before continuing, we introduce some convenient notions. We assume that $|\Sigma_\varphi| = n$ implies $\Sigma_\varphi = \{1, \ldots, n\}$, i.e., the agents are numbered and their name coincides with their number. We call ATL-formulas of the form $\langle\langle A\rangle\rangle\psi_1\,\mathcal{U}\,\psi_2$ or $\neg\langle\langle A\rangle\rangle\Box\psi$ *eventualities*. A *next-formula* is a formula of the form $\langle\langle A\rangle\rangle\bigcirc\psi$ or $\neg\langle\langle A\rangle\rangle\bigcirc\psi$. For each formula φ, assume that all next-formulas in $\mathsf{ecl}(\varphi)$ are linearly ordered and use $\sharp_\psi$ to denote the number of the next-formula $\psi \in \mathsf{ecl}(\varphi)$ (the numbering starts with 0 and the formula φ will be clear from the context). The ordering is such that no negative next-formula occurs before a positive one. Since there are as many positive next-formulas in $\mathsf{ecl}(\varphi)$ as negative ones, we obtain an enumeration $\psi_0, \ldots, \psi_{k-1}$ of k next-formulas, with positive next-formulas $\psi_0, \ldots, \psi_{k/2-1}$ and negative next-formulas $\psi_{k/2}, \ldots, \psi_{k-1}$.

To understand the following, central definition, let us sketch some details of the ATS $\mathcal{S}$ that our algorithm attempts to build as a model for the input formula φ. If $n = |\Sigma_\varphi|$ and k is the number of next-formulas in $\mathsf{ecl}(\varphi)$, then (regardless of some technical details) the states of $\mathcal{S}$ consist of sequences of n-tuples whose components take values from $\{0, \ldots, k-1\}$. The set of all such n-tuples is denoted with $[k/n]$, and the states of $\mathcal{S}$ will thus be from $[k/n]^*$. If $q \in [k/n]^*$ is a state of $\mathcal{S}$, then $\{q \cdot \vec{t} \mid \vec{t} \in [k/n]\}$ will be the set of its potential successors, i.e., the choices in $\delta(q, a)$ will be subsets of this set. When constructing $\mathcal{S}$, each state $q \in [k/n]^*$ will have to satisfy a number of positive next-formulas and a number of negative next-formulas. Clearly, having to satisfy a positive next-formula $\langle\langle A\rangle\rangle\bigcirc\psi$ at q means that there has to be an A-choice $C \in \delta(q, A)$ such that all states in C satisfy ψ. Similarly, having to satisfy a negative next-formula $\neg\langle\langle A\rangle\rangle\bigcirc\psi$ at q means that all A-choices $C \in \delta(q, A)$ have to contain a state satisfying $\neg\psi$. This can be achieved by defining the transition function and assigning formulas to successors as follows:

(i) For each agent a, we set

$$\delta(q,a) \ := \ \{\{q \cdot \vec{t} \in [k/n] \mid \vec{t} = (t_0, \ldots, t_{n-1}) \text{ and } t_a = p\} \mid p < k/2\}$$

Recall that we assume agents to be natural numbers. Intuitively, every agent "owns" a position in $\vec{t}$, and via this position he can make an a-choice in state q by "voting" for a positive next-formula that he wants to be satisfied.

Due to this definition, for coalitions of agents A we can characterise A-choices as follows: A subset $S_A \subseteq [k/n]$ is called an *A-voting set* if there exists a mapping $\tau : A \rightarrow \{0, \ldots, k/2-1\}$ such that

$$S_A \ := \ \{\vec{t} = (t_0, \ldots, t_{n-1}) \mid t_a = \tau(a) \text{ for all } a \in A\}.$$

Then, the elements of $\delta(q, A)$ are exactly the sets $\{q \cdot \vec{t} \mid \vec{t} \in S_A\}$ with S_A being an A-voting set.

(ii) To satisfy a positive next-formula $\langle\langle A\rangle\rangle\bigcirc\psi$ at q, we use the voting set S_A in which all agents $a \in A$ vote for this formula, i.e.,

$$S_A = \{\vec{t} = (t_0, \ldots, t_{n-1}) \mid t_a = \sharp_{\langle\langle A\rangle\rangle\bigcirc\psi} \text{ for all } a \in A\}.$$

The ATS $\mathcal{S}$ is constructed such that all states in the corresponding A-choice $\{q \cdot \vec{t} \mid \vec{t} \in S_A\} \in \delta(q, A)$ make ψ true.

(iii) To satisfy a negative next-formula $\neg\langle\langle A\rangle\rangle\bigcirc\psi$ at q, we have to pick an element from every A-choice $C \in \delta(q, A)$, and then make ψ false at the picked elements.

Note that, in being a member of an A-choice, a picked element will automatically also be a member of an A'-choice for all $A' \subseteq A$. This is fine as $\neg\langle\langle A\rangle\rangle\bigcirc\psi$ implies $\neg\langle\langle A'\rangle\rangle\bigcirc\psi$.

However, if $B \not\subseteq A$, then $\neg\langle\langle A\rangle\rangle\bigcirc\psi$ does not imply $\neg\langle\langle B\rangle\rangle\bigcirc\psi$. Thus we should be careful that the picked elements from A-choices are not elements of B-choices for any such B. This is implemented by demanding that the element $\vec{t} = (t_0, \dots, t_{n-1})$ picked for an A-choice satisfies $t_a \geq k/2$ for each $a \notin A$.

The description of how exactly we pick elements is given after the next definition.

(iv) A special role is played by negative next-formulas $\neg\langle\langle\Sigma_\varphi\rangle\rangle\bigcirc\psi$. As we are working with formula-defined sets of agents, Σ_φ is the set of all agents in S. For this reason, such negative next-formulas behave differently from formulas referring to smaller sets of agents. For example, $\neg\langle\langle A\rangle\rangle\bigcirc\psi$ and $\neg\langle\langle A\rangle\rangle\bigcirc\psi'$ imply $\neg\langle\langle A\rangle\rangle\bigcirc(\psi \vee \psi')$ if, and only if, $A = \Sigma_\varphi$. However, dealing with formulas $\neg\langle\langle\Sigma_\varphi\rangle\rangle\bigcirc\psi$ is simple: they merely state that no successor of q satisfies ψ.

The whole picture of the ATS construction is somewhat more complicated due to the presence of box formulas and until formulas, which we will address later.

We now introduce a "refutation function" whose purpose is to pick, for states q that have to satisfy a negative next-formula $\neg\langle\langle A\rangle\rangle\bigcirc\psi$, successors that refute ψ as explained under (iii) above.

Definition 3.6 (Refutation Function). Let φ be an ATL-formula, $n = |\Sigma_\varphi|$ and k the number of next-formulas in $\mathsf{ecl}(\varphi)$. We define a partial function

$$f : [k/n] \times 2^{\Sigma_\varphi} \to \{k/2, \dots, k-1\}$$

mapping vectors and coalitions of agents to (numbers of) negative next-formulas: for each set $A \subset \Sigma_\varphi$ of agents , fix an agent $a_A \in \Sigma_\varphi \backslash A$. Then set, for all $\vec{t} = (t_0, \dots, t_{n-1}) \in [k/n]$ and $A \subseteq \Sigma_\varphi$,

$$f(\vec{t}, A) := \begin{cases} \sharp_{\neg\langle\langle A\rangle\rangle\bigcirc\psi} & \text{if } t_{a_A} = \sharp_{\neg\langle\langle A\rangle\rangle\bigcirc\psi} \text{ and for all } a \in \Sigma_\varphi,\ t_a < k/2 \text{ iff } a \in A \\ \text{undefined} & \text{if there is no such } \neg\langle\langle A\rangle\rangle\bigcirc\psi. \end{cases}$$

Intuitively, $f(\vec{t}, A) = \sharp_{\neg\langle\langle A\rangle\rangle\bigcirc\psi}$ means that, for every state q satisfying $\neg\langle\langle A\rangle\rangle\bigcirc\psi$, the successor $q \cdot \vec{t}$ has to refute ψ. Note that there may be more than one successor of q refuting ψ: at least one element of each A-choice. Formally, the most important properties of the refutation function are the following:

(1) for each formula $\neg\langle\langle A'\rangle\rangle\bigcirc\psi' \in \mathsf{ecl}(\varphi)$ with $A' \subset \Sigma_\varphi$ and each A'-voting set $S_{A'}$, there is an element $\vec{t}' \in S_{A'}$ such that $f(\vec{t}', A') = \sharp_{\neg\langle\langle A'\rangle\rangle\bigcirc\psi'}$ (Recall that A'-voting sets correspond to A'-choices; cf. explanation (i) above.);

(2) for all $\vec{t}' = (t'_0, \dots, t'_{n-1}) \in [k/n]$, $f(\vec{t}', A') = \sharp_{\neg\langle\langle A'\rangle\rangle\bigcirc\psi'}$ implies $t'_a \geq k/2$ for all $a \in \Sigma_\varphi \setminus A'$ (cf. explanation (iii));

(3) for each $\vec{t} \in [k/n]$, there is at most a single $A \subseteq \Sigma_\varphi$ with $f(\vec{t}, A)$ defined.

It is easily verified that the function f from Definition 3.6 indeed satisfies these properties. A different function satisfying the properties is given by van Drimmelen [**252**].

To determine the satisfiability of an ATL-formula φ, the algorithm developed in this section will check for the existence of a model that is composed from certain trees, so-called *φ-trees.*

DEFINITION 3.7 (φ-tree, ϑ-vector, $\bigcirc$-matching). Let φ be an ATL-formula, $n = |\Sigma_\varphi|$ and k the number of next-formulas in $\mathsf{ecl}(\varphi)$. For each next-formula $\vartheta \in \mathsf{ecl}(\varphi)$ and vector $\vec{t} = (t_0, \ldots, t_{n-1}) \in [k/n]$,

- if $\vartheta = \langle\!\langle A \rangle\!\rangle \bigcirc \psi$ and $t_a = \sharp_\vartheta$ for each $a \in A$, then $\vec{t}$ is called a *ϑ-vector*;
- if $\vartheta = \neg\langle\!\langle A \rangle\!\rangle \bigcirc \psi$ with $A \subset \Sigma_\varphi$, then $\vec{t}$ is called a *ϑ-vector* if $f(\vec{t}, A) = \sharp_\vartheta$;
- if $\vartheta = \neg\langle\!\langle \Sigma_\varphi \rangle\!\rangle \bigcirc \psi \in \mathsf{ecl}(\varphi)$, then $\vec{t}$ is called a *ϑ-vector.*

For each type $\Psi \in \Gamma_\varphi$ and each $\vec{t} = (t_0, \ldots, t_{n-1}) \in [k/n]$, let $S_\Psi(\vec{t}) \subseteq \mathsf{ecl}(\varphi)$ be the smallest set such that

(M1) if $\langle\!\langle A \rangle\!\rangle \bigcirc \psi \in \Psi$ and $\vec{t}$ is a $\langle\!\langle A \rangle\!\rangle \bigcirc \psi$-vector, then $\psi \in S_\Psi(\vec{t})$, and
(M2) if $\neg\langle\!\langle A \rangle\!\rangle \bigcirc \psi \in \Psi$ and $\vec{t}$ is a $\neg\langle\!\langle A \rangle\!\rangle \bigcirc \psi$-vector, then $\neg\psi \in S_\Psi(\vec{t})$.

Given a set M, a *$\langle M, k, n\rangle$-tree* T is a mapping T from a finite prefix-closed subset of $[k/n]^*$ to M. A *$\langle \Gamma_\varphi, k, n\rangle$-tree* is called a *$\varphi$-tree.* A φ-tree T is called *$\bigcirc$-matching* if, for all $\alpha \in \mathsf{dom}(T)$ and all $\vec{t} \in [k/n]$, $\alpha \cdot \vec{t} \in \mathsf{dom}(T)$ implies $S_{T(\alpha)}(\vec{t}) \subseteq T(\alpha \cdot \vec{t})$.

Intuitively, a vector $\vec{t}$ is a $\langle\!\langle A \rangle\!\rangle \bigcirc \psi$-vector if, for all states q satisfying $\langle\!\langle A \rangle\!\rangle \bigcirc \psi$, the successor $q \cdot \vec{t}$ has to satisfy ψ, cf. explanation (ii) above. The $\neg\langle\!\langle A \rangle\!\rangle \bigcirc \psi$-vectors can be understood in an analogous way. This intuition is reflected in (M1) and (M2) and in the definition of $\bigcirc$-matching.

Up to this point, we have set up the basic machinery to define ATSs based on the states $[k/n]^*$, and to treat satisfaction of (positive and negative) next-formulas. To deal with eventualities, we introduce φ-trees that witness their satisfaction, so-called *witness trees.* Their definition is largely analogous to the corresponding construction for CTL [69].

DEFINITION 3.8 (Witness Tree). Let φ be an ATL-formula, Γ a set of types for φ, and $\Psi \in \Gamma$. A φ-tree T is called a *witness-tree rooted at Ψ in Γ for a formula $\langle\!\langle A \rangle\!\rangle \psi \,\mathcal{U}\, \vartheta$* if it satisfies the following properties:

(1) for all $\alpha \in \mathsf{dom}(T)$, $T(\alpha) \in \Gamma$;
(2) T is $\bigcirc$-matching;
(3) $T(\varepsilon) = \Psi$;
(4) for all $\alpha \in \mathsf{dom}(T)$, $\langle\!\langle A \rangle\!\rangle \psi \,\mathcal{U}\, \vartheta \in T(\alpha)$;
(5) for all non-leaf nodes α, $\psi \in T(\alpha)$;
(6) for all leaf nodes α, $\vartheta \in T(\alpha)$;
(7) if $\alpha \in \mathsf{dom}(T)$, $\langle\!\langle A \rangle\!\rangle \bigcirc \langle\!\langle A \rangle\!\rangle \psi \,\mathcal{U}\, \vartheta \in T(\alpha)$, $\vartheta \notin T(\alpha)$ and $\vec{t}$ is a $\langle\!\langle A \rangle\!\rangle \bigcirc \langle\!\langle A \rangle\!\rangle \psi \,\mathcal{U}\, \vartheta$-vector, then $\alpha \cdot \vec{t} \in \mathsf{dom}(T)$.

T is called a *witness-tree rooted at Ψ in Γ* for a formula $\neg\langle\!\langle A \rangle\!\rangle \Box \psi$ if it satisfies the following properties:

(1) for all $\alpha \in \mathsf{dom}(T)$, $T(\alpha) \in \Gamma$;
(2) T is $\bigcirc$-matching;
(3) $T(\varepsilon) = \Psi$;
(4) for all $\alpha \in \mathsf{dom}(T)$, $\neg\langle\langle A\rangle\rangle\Box\psi \in T(\alpha)$;
(5) for all leaf nodes α, $\neg\psi \in T(\alpha)$
(6) if $\alpha \in \mathsf{dom}(T)$, $\neg\langle\langle A\rangle\rangle\bigcirc\langle\langle A\rangle\rangle\Box\psi \in T(\alpha)$, $\neg\psi \notin T(\alpha)$ and $\vec{t}$ is a $\neg\langle\langle A\rangle\rangle\bigcirc\langle\langle A\rangle\rangle\Box\psi$-vector, then $\alpha \cdot \vec{t} \in \mathsf{dom}(T)$.

Our decision procedure is based on the following core notion of *realizability*. Intuitively, a type Ψ is realizable in a set of types Γ if it is possible to (a) satisfy all next-formulas in Ψ using only types from Γ, and (b) construct witness trees for all eventualities in Ψ using only types from Γ.

DEFINITION 3.9 (ATL Realizability). Let φ be an ATL-formula and Γ a set of types for φ. A type $\Psi \in \Gamma$ is *ATL-realizable in* Γ if the following conditions are satisfied:

1. for each $\vec{t} \in [k/n]$, there is a $\Psi' \in \Gamma$ such that $S_\Psi(\vec{t}) \subseteq \Psi'$;
2. for each $\langle\langle A\rangle\rangle\psi\mathcal{U}\vartheta \in \Psi$, there is a $\langle\langle A\rangle\rangle\psi\mathcal{U}\vartheta$-witness tree rooted at Ψ in Γ;
3. for each $\neg\langle\langle A\rangle\rangle\Box\psi \in \Psi$, there is a $\neg\langle\langle A\rangle\rangle\Box\psi$-witness tree rooted at Ψ in Γ.

We are now ready to describe the decision procedure. The idea is to start with all types for the input formula, then repeatedly eliminate types that are not ATL-realizable, and finally check whether there is a type that survived elimination and contains the input formula. Let φ be an ATL-formula whose satisfiability is to be decided. The type elimination algorithm for ATL is presented as function ATL–*sat*(φ) in Figure 3.1. The

```
1.  function ATL–sat($\varphi$) returns 'Yes, ...' or 'No, ...'
2.     $m := 0$
3.     $\Delta_m := \Gamma_\varphi$
4.     do
5.        $m := m + 1$
6.        $\Delta_m := \{\Psi \in \Delta_{m-1} \mid \Psi$ is ATL-realizable in $\Delta_{m-1}\}$
7.     until $\Delta_m = \Delta_{m-1}$
8.     if $\varphi \in \Psi$, for some $\Psi \in \Delta_m$,
9.        then return 'Yes, $\varphi$ is satisfiable in an ATS for $\Sigma_\varphi$.'
10.       else return 'No, $\varphi$ is not satisfiable.'
11. end-function
```

FIGURE 3.1. A type-elimination algorithm for ATL.

algorithm starts with the set of all types Γ_φ for the input formula φ (line 2 and 3). Then it inductively computes a sequence $\Delta_0, \Delta_1, \cdots$ of sets of types for φ as shown in the do–until loop (lines 4–7) while repeatedly eliminating the types that are not ATL-realizable. Since there are only finitely many types to start with, the algorithm eventually leaves the loop with a set Δ_m of types such that $\Delta_{m+1} = \Delta_m$ for some $m \geq 0$ (line 7). Notice that, at this point, Δ_m is a set of types for φ that are all ATL-realizable in Δ_m. The algorithm returns 'Yes, φ is satisfiable in an ATS for Σ_φ' if the input formula φ is contained in some type of Δ_m (line 8 and 9); otherwise it returns 'No, φ is not satisfiable.' (line 8 and 10).

We proceed as follows: first we show that this procedure is effective by proving that the existence of witness trees is decidable in exponential time.

LEMMA 3.10. *Let Γ be a set of types for an ATL-formula φ. Then the existence of witness trees in Γ can be decided in time exponential in the length of φ.*

Second, we prove soundness and completeness of the procedure.

LEMMA 3.11. *Let φ be an ATL-formula. Then the procedure returns 'Yes, the input formula φ is satisfiable in an ATS for Σ_φ' iff this is indeed the case.*

Finally, we establish that it runs in exponential time.

LEMMA 3.12. *The described type elimination procedure runs in exponential time.*

The proofs of these lemmas can be found in the next section.

Let us now consider the lower bound. By Lemma 3.3, it is sufficient to prove EXPTIME-hardness for variant (b) of the ATL satisfiability problem. For this variant, EXPTIME-hardness does *not* trivially follow from EXPTIME-hardness of CTL: the translation from CTL to ATL described after the definition of ATL in Section 2.4.1 cannot be used since, when concerned with satisfiability over arbitrary sets of agents, there is no obvious ATL-equivalent of CTL-formulas of the form $\mathsf{E}\varphi\,\mathcal{U}\,\psi$. Note in particular that we cannot use $\langle\!\langle\Sigma\rangle\!\rangle\varphi\,\mathcal{U}\,\psi$ since the coalition Σ of all agents is not available for a formula. Moreover, the $\langle\!\langle\Sigma\rangle\!\rangle\varphi\,\mathcal{U}\,\psi$ equivalent formula $\neg\langle\!\langle\emptyset\rangle\!\rangle\neg(\varphi\,\mathcal{U}\,\psi)$ is not according to the syntax of ATL.

To show the lower bound for ATL satisfiability over arbitrary sets of agents, we reduce the EXPTIME-hard global consequence problem in the modal logic K. For the syntax and semantics of K, we refer to Section 2.1 in Chapter 2 or to, e.g., [**227**]. To avoid confusion with the operators of ATL, we denote the diamond and box of K with $\blacklozenge$ and $\blacksquare$. Recall that the global consequence problem is to decide, given two K-formulas φ and ψ, whether it is the case that for every Kripke structure $\mathcal{M}$, if φ is true in every state of $\mathcal{M}$ (in symbols, $\mathcal{M} \models \varphi$), then ψ is true in every state of $\mathcal{M}$.

For the reduction, we use the following facts: (i) a Kripke structure $\mathcal{M}$ with an accessibility relation that is serial (right unbounded) is equivalent to an alternating transition system with a single agent [**15, 96**]; (ii) $\langle\!\langle\emptyset\rangle\!\rangle\bigcirc\varphi$ expresses that φ is true at all successors; and (iii) $\langle\!\langle\emptyset\rangle\!\rangle\Box\varphi$ expresses that φ holds in the (reachable part of the) whole model. Now let φ and ψ be K-formulas. Translate φ and ψ into formulas of ATL using the following translation $(\cdot)^\sharp$:

$$\begin{aligned} p^\sharp &:= p; \\ (\varphi \wedge \psi)^\sharp &:= \varphi^\sharp \wedge \psi^\sharp; \end{aligned}$$

$$\begin{aligned}(\neg\varphi)^\sharp &:= \neg\varphi^\sharp;\\ (\blacklozenge\varphi)^\sharp &:= \neg\langle\langle\emptyset\rangle\rangle\bigcirc\neg\varphi^\sharp;\\ (\blacksquare\varphi)^\sharp &:= \langle\langle\emptyset\rangle\rangle\bigcirc\varphi^\sharp.\end{aligned}$$

Now it suffices to show the following:

LEMMA 3.13. *ψ follows globally from φ iff $\langle\langle\emptyset\rangle\rangle\Box\varphi^\sharp \wedge \neg\psi^\sharp$ is unsatisfiable in ATL with arbitrarily many agents.*

PROOF. It is straightforward to prove by structural induction that, for all Kripke structures $\mathcal{M}$ with serial accessibility relations, all states q of $\mathcal{M}$, and all K-formulas ϑ, we have $\mathcal{M}, q \models \vartheta$ iff $\mathcal{M}, q \models \vartheta^\sharp$, where, on the right hand side, $\mathcal{M}$ is viewed as a single-agent ATS. We leave details to the reader and continue with a proof of the lemma.

"$\Leftarrow$". We show the contrapositive. Thus suppose ψ does not globally follow from φ. Then there is a Kripke structure $\mathcal{M}$ such that $\mathcal{M} \models \varphi$ and $\mathcal{M}, q \not\models \psi$ for some state q of $\mathcal{M}$. Then, $\mathcal{M}, q \models \langle\langle\emptyset\rangle\rangle\Box\varphi^\sharp$ and $\mathcal{M}, q \not\models \psi^\sharp$. Hence, $\mathcal{M}, q \models \langle\langle\emptyset\rangle\rangle\Box\varphi^\sharp \wedge \neg\psi^\sharp$, and this formula is satisfiable in a single-agent ATS.

"$\Rightarrow$". We again show the contrapositive. Suppose $\langle\langle\emptyset\rangle\rangle\Box\varphi^\sharp \wedge \neg\psi^\sharp$ is satisfiable and take an ATS $\mathcal{M}$ and a state q of $\mathcal{M}$ such that $\mathcal{M}, q \models \langle\langle\emptyset\rangle\rangle\Box\varphi^\sharp \wedge \neg\psi^\sharp$. By Lemma 3.3 and since $\langle\langle\emptyset\rangle\rangle\Box\varphi^\sharp \wedge \neg\psi^\sharp$ does not refer to any agents, we may assume that $\mathcal{M}$ is a single-agent ATS and thus a Kripke structure. We have $\mathcal{M}, q \models \langle\langle\emptyset\rangle\rangle\Box\varphi^\sharp$ and $\mathcal{M}, q \not\models \psi^\sharp$. By the former, $\varphi^\sharp$ holds at all points reachable from q. Denote by $\mathcal{N}$ the model induced by $\mathcal{M}$ on those points. Then $\mathcal{N} \models \varphi$ but $\mathcal{N}, q \not\models \psi$. Hence, ψ does not follow globally from φ. ☐

3.1.3. Proofs for the ATL Decision Procedure.

LEMMA 3.10. *Let Γ be a set of types for an ATL-formula φ. Then the existence of witness trees in Γ can be decided in time exponential in the length of φ.*

PROOF. Let φ and Γ be as in the lemma, and Ψ_0 a type in Γ. We only show how to check the existence of witness trees for formulas of the form $\langle\langle A\rangle\rangle\psi\mathcal{U}\vartheta$. Witness trees for formulas $\neg\langle\langle A\rangle\rangle\Box\psi$ can be treated analogously. Thus, suppose we want to check the existence of a witness tree for $\langle\langle A\rangle\rangle\psi\mathcal{U}\vartheta \in \Psi_0$ rooted at Ψ_0 in Γ. Start with identifying leaf nodes of possible witness trees by marking all types in Γ which contain $\langle\langle A\rangle\rangle\psi\mathcal{U}\vartheta$ and ϑ. Then identify inner nodes of possible witness trees as follows: For all unmarked types $\Psi \in \Gamma$ such that $\langle\langle A\rangle\rangle\psi\mathcal{U}\vartheta \in \Psi$, mark Ψ if $\psi \in \Psi$ and for all $\langle\langle A\rangle\rangle\bigcirc\langle\langle A\rangle\rangle\psi\mathcal{U}\vartheta$-vectors $\vec{t} \in [k/n]$, there is a type $\Psi' \in \Gamma$ such that:

(i) $S_\Psi(\vec{t}) \subseteq \Psi'$, and
(ii) Ψ' is marked.

Repeatedly apply this procedure until no more types in Γ get marked. Note that this process must terminate since Γ contains only finitely many types.

It is easy to see that a witness tree for $\langle\langle A\rangle\rangle\psi\mathcal{U}\vartheta$ exists iff Ψ_0 was marked. For the left-to-right direction, suppose a witness tree exists. At the beginning of the marking procedure, all leaf nodes of possible witness trees are marked. In every subsequent round, all inner nodes of possible witness trees in increasing distance from the leaf nodes will be marked. Hence, eventually Ψ_0 will be marked. For the right-to-left direction, assume that Ψ_0 is marked and consider the following informal construction of a witness tree. Take the type Ψ_0 as the root. For each leaf and for all $\langle\langle A\rangle\rangle\bigcirc\langle\langle A\rangle\rangle\psi\mathcal{U}\vartheta$-vectors, add a successor type that satisfies (i) and (ii) until a type is reached which was marked at the beginning. Note that this process will not generate any infinite paths since, when choosing a successor type for some type Ψ, we can only use a type that was marked strictly before Ψ was marked. It is readily checked that all properties of a witness tree are fulfilled by the obtained tree.

For the complexity of the algorithm that checks for witness trees consider the following. Let $n = |\varphi|$. Note that the cardinality of the extended closure $\mathsf{ecl}(\varphi)$ is linear in n, i.e., $|\mathsf{ecl}(\varphi)| = c \cdot n$ for some constant $c \geq 1$. Since $\Gamma \subseteq \Gamma_\varphi \subseteq 2^{\mathsf{ecl}(\varphi)}$, it holds that $|\Gamma| \leq 2^{c \cdot n}$. For marking the leaf nodes, maximal $2^{c \cdot n}$ types have to be considered. For the marking of the inner nodes consider the following. Since there are at most $2^{c \cdot n}$ types in Γ and in each marking round at least one type gets marked, there are maximal $2^{c \cdot n}$ marking rounds. In each such round, maximal $2^{c \cdot n}$ yet unmarked types have to be checked. In order to find out whether to mark such a type, not more than n^n vectors have to be considered and for each such vector, tests for conditions (i) and (ii) with maximal $2^{c \cdot n}$ types need to be performed. Altogether this yields an upper bound of $2^{c \cdot n} + 2^{c \cdot n} \cdot 2^{c \cdot n} \cdot n^n \cdot 2^{c \cdot n} = 2^{\mathcal{O}(n^2)}$ steps. Thus the existence of witness trees can be checked in time exponential in the length of φ. □

LEMMA 3.11. *Let φ be an ATL-formula. Then the procedure returns 'Yes, the input formula φ is satisfiable in an ATS for Σ_φ' iff this is indeed the case.*

PROOF. Suppose φ is given, and let $\Sigma = \Sigma_\varphi$ and $n = |\Sigma_\varphi|$.

"$\Rightarrow$" (Soundness) Assume that the elimination procedure was started on input φ and returns "Yes, the input formula φ is satisfiable". Let $\Gamma = \{\Psi_0, \ldots, \Psi_{m-1}\}$ be the computed set of types. Then all types of Γ are ATL-realizable in Γ and there is a type $\Psi \in \Gamma$ with $\varphi \in \Psi$. Our aim is to construct an ATS that is a model of φ.

To this end, enumerate all eventualities in $\mathsf{ecl}(\varphi)$ by $\psi_0, \ldots, \psi_{\ell-1}$. For each i with $i < \ell$ and each j with $j < m$, fix a φ-tree $T_{\langle\psi_i,\Psi_j\rangle}$ as follows:

- If $\psi_i \in \Psi_j$, then fix a ψ_i-witness tree T rooted at Ψ_j in Γ. Supplement all inner nodes of T with missing successors: for each inner node $\alpha \in \mathsf{dom}(T)$ and each $\vec{t} \in [k/n]$, if $\alpha \cdot \vec{t} \notin \mathsf{dom}(T)$, then add it and set $T(\alpha \cdot \vec{t}) = \Psi$ for some $\Psi \in \Gamma$ such that $S_{T(\alpha)}(\vec{t}) \subseteq \Psi$. Note that such a Ψ must exist by Condition 1 of Definition 3.9. Let $T_{\langle\psi_i,\Psi_j\rangle}$ be the result of augmenting T in this way.

- If $\psi_i \notin \Psi_j$, then let $T_{\langle\psi_i,\Psi_j\rangle}$ be the tree comprised of the nodes $\{\varepsilon\} \cup [k/n]$ such that $T_{\langle\psi_i,\Psi_j\rangle}(\varepsilon) = \Psi_j$ and, for each $\vec{t} \in [k/n]$, $T_{\langle\psi_i,\Psi_j\rangle}(\vec{t}) = \Psi$ for some $\Psi \in \Gamma$ with $S_{\Psi_j}(\vec{t}) \subseteq \Psi$.

It is easy to see that all trees $T_{\langle\psi_i,\Psi_j\rangle}$ are $\bigcirc$-matching. To construct a model of φ,

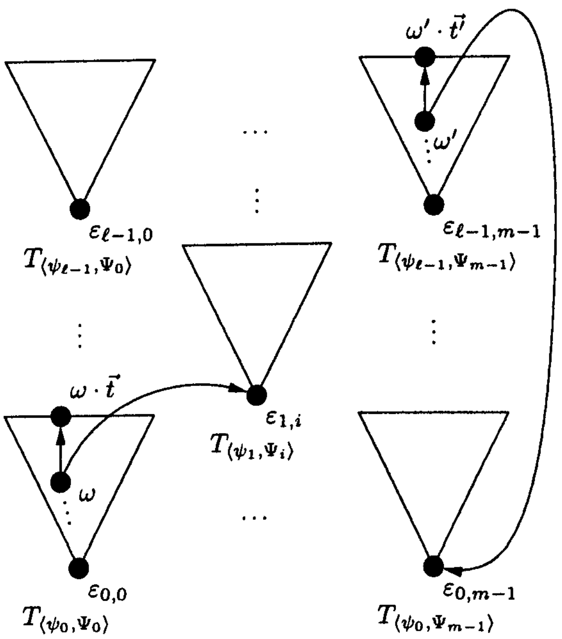

$\varepsilon_{\ell-1,0}$
$T_{\langle\psi_{\ell-1},\Psi_0\rangle}$
$\omega \cdot \vec{t}$
ω
$\varepsilon_{0,0}$
$T_{\langle\psi_0,\Psi_0\rangle}$
$\varepsilon_{1,i}$
$T_{\langle\psi_1,\Psi_i\rangle}$
$\omega' \cdot \vec{t'}$
ω'
$\varepsilon_{\ell-1,m-1}$
$T_{\langle\psi_{\ell-1},\Psi_{m-1}\rangle}$
$\varepsilon_{0,m-1}$
$T_{\langle\psi_0,\Psi_{m-1}\rangle}$

FIGURE 3.2. $\ell \times m$-matrix of witness trees.

intuitively we do the following: we arrange the selected witness trees in an $\ell \times m$-matrix such that the rows range over the eventualities $\psi_0, \ldots, \psi_{\ell-1}$ and the columns over the types $\Psi_0, \ldots, \Psi_{m-1}$, and then replace all leaf nodes by an 'arrow' from the leaf node's predecessor to the root of some other witness tree. For an illustration of the $\ell \times m$-matrix of the witness trees, see Figure 3.2.

We now define the ATS $\mathcal{S} = (\Pi, \Sigma, Q, \pi, \delta)$ that we then prove to be a model of φ. Π and Σ are the sets of those propositions and agents that occur in the input formula φ. For defining the set of states Q, fix symbols $\varepsilon_{i,j}$ with $i < \ell$ and $j < m$. Then set:

$$Q \ := \ \{\varepsilon_{i,j}w \mid w \in \mathsf{dom}(T_{\langle\psi_i,\Psi_j\rangle}) \text{ is inner node}\}.$$

Next, the valuation π is easily defined: for $q = \varepsilon_{i,j}w \in Q$, set

$$\pi(q) \ := \ T_{\langle\psi_i,\Psi_j\rangle}(w) \cap \Pi.$$

To define the transition function δ, we first define a successor function on Q: for each $q = \varepsilon_{i,j}w \in Q$ and each $\vec{t} \in [k/n]$, set

$$s_{\vec{t}}(q) \ := \ \begin{cases} \varepsilon_{s,p} & \text{if } w \cdot \vec{t} \text{ is a leaf node of } T_{\langle\psi_i,\Psi_j\rangle}, \\ & s = i+1 \bmod \ell \text{ and } T_{\langle\psi_i,\Psi_j\rangle}(w \cdot \vec{t}) = \Psi_p; \\ q \cdot t & \text{if } w \cdot \vec{t} \text{ is an inner node of } T_{\langle\psi_i,\Psi_j\rangle}. \end{cases}$$

Now the definition of δ is straightforward: for each $q \in Q$ and $a \in \Sigma$, set

$$\delta(q,a) \;:=\; \{\{s_{\vec{t}}(q) \mid \vec{t} = (t_0,\ldots,t_{n-1}) \in [k/n] \text{ and } t_a = p\} \mid p < k/2\}.$$

To show that $\mathcal{S}$ is indeed a model of φ, we introduce some auxiliary notions:

For each strategy $\sigma_A = \{\sigma_a \mid a \in A\}$ for a set of agents $A \subseteq \Sigma$ and each sequence of states $\lambda \in Q^+$, we write $\sigma_A(\lambda)$ to denote the set of states $\bigcap_{a\in A} \sigma_a(\lambda)$. Observe that, by definition of strategies for single agents, we have $\sigma_A(\lambda \cdot q) \in \delta(q, A)$ for all $\lambda \in Q^+$ and $q \in Q$.

For each positive next-formula $\langle\!\langle A\rangle\!\rangle\bigcirc\psi$, the *$\langle\!\langle A\rangle\!\rangle\bigcirc\psi$-strategy* is the strategy $\sigma_A = \{\sigma_a \mid a \in A\}$ for the set of agents A that is defined by setting, for each $a \in A$,

$$\sigma_a(\lambda \cdot q) \;:=\; \{s_{\vec{t}}(q) \mid \vec{t} = (t_0,\ldots,t_{n-1}) \text{ and } t_a = \sharp_{\langle\!\langle A\rangle\!\rangle\bigcirc\psi}\}.$$

It is readily checked that we have

$$\sigma_A(\lambda \cdot q) = \{s_{\vec{t}}(q) \mid \vec{t} \text{ is a } \langle\!\langle A\rangle\!\rangle\bigcirc\psi\text{-vector}\}. \tag{3.1}$$

For each negative next-formula $\neg\langle\!\langle A\rangle\!\rangle\bigcirc\psi$, a *$\neg\langle\!\langle A\rangle\!\rangle\bigcirc\psi$-computation for a strategy σ_A rooted at a state $q \in Q$* is a computation $\lambda \in \mathsf{out}(q, \sigma_A)$ such that, for all positions $i \geq 0$, $\lambda[i+1] = s_{\vec{t}}(\lambda[i])$ for some $\neg\langle\!\langle A\rangle\!\rangle\bigcirc\psi$-vector $\vec{t} \in [k/n]$.

CLAIM 3.12. *Let $\neg\langle\!\langle A\rangle\!\rangle\bigcirc\psi$ be a next-formula, σ_A a strategy and $q \in Q$. Then there exists a $\neg\langle\!\langle A\rangle\!\rangle\bigcirc\psi$-computation for σ_A rooted at q.*

PROOF OF CLAIM Let $\neg\langle\!\langle A\rangle\!\rangle\bigcirc\psi$, σ_A and q be as in the claim. Inductively define a $\neg\langle\!\langle A\rangle\!\rangle\bigcirc\psi$-computation $\lambda \in Q^\omega$ for σ_A rooted at q as follows:

- $\lambda[0] := q$, and
- for each $i \geq 0$, $\lambda[i+1] := s_{\vec{t}}(\lambda[i])$ for some $\neg\langle\!\langle A\rangle\!\rangle\bigcirc\psi$-vector $\vec{t} \in [k/n]$ such that $s_{\vec{t}}(\lambda[i]) \in \sigma_A(\lambda[i])$.

In order to show that λ is well-defined, it remains to show that for each $i \geq 0$, there is a state $s_{\vec{t}}(\lambda[i]) \in \sigma_A(\lambda[i])$ such that $\vec{t}$ is a $\neg\langle\!\langle A\rangle\!\rangle\bigcirc\psi$-vector. Distinguish two cases:

- $A = \Sigma$. By definition, every vector $\vec{t} \in [k/n]$ is a $\neg\langle\!\langle \Sigma\rangle\!\rangle\bigcirc\psi$-vector. Since $\sigma_A(\lambda[i]) \in \delta(\lambda[i], A)$, $\sigma_A(\lambda[i])$ is non-empty. Thus, any vector from $\sigma_A(\lambda[i])$ is suitable.
- $A \neq \Sigma$. By definition of δ and since $\sigma_A(\lambda[i]) \in \delta(\lambda[i], A)$, $\sigma_A(\lambda[i]) = \{s_{\vec{t}}(\lambda[i]) \mid \vec{t} \in S_A\}$ for some A-voting set S_A. By condition 1 of the function f used in the definition of $\neg\langle\!\langle A\rangle\!\rangle\bigcirc\psi$-vectors, S contains a $\neg\langle\!\langle A\rangle\!\rangle\bigcirc\psi$-vector. Thus, there is a state $s_{\vec{t}}(\lambda[i]) \in \sigma_A(\lambda[i])$ such that $\vec{t}$ is a $\neg\langle\!\langle A\rangle\!\rangle\bigcirc\psi$-vector.

◀

To denote the intended type of a state $q = \varepsilon_{i,j}w \in Q$, we set $t(q) := T_{\langle\psi_i,\Psi_j\rangle}(w)$. Using the construction of $\mathcal{S}$ and property 2 of witness trees, it is straightforward to prove the following claim, which, intuitively, states that our ATS is $\bigcirc$-matching:

CLAIM 3.13. *For all $q \in Q$ and $\vec{t} \in [k/n]$, $S_{t(q)}(\vec{t}) \subseteq t(s_{\vec{t}}(q))$.*

The next claim establishes the property of $\mathcal{S}$ that is crucial for showing that it is a model of φ.

CLAIM 3.14. *For any state $q \in Q$ and any formula $\psi \in ecl(\varphi)$, $\psi \in t(q)$ iff $\mathcal{S}, q \models \psi$.*

PROOF OF CLAIM Let q and ψ be as in the claim. The proof is by induction on the structure of ψ. Since the base case and the Boolean cases are straightforward, we concentrate on path quantifiers:

- $\psi = \langle\langle A\rangle\rangle\bigcirc\psi'$. "$\Rightarrow$": Suppose $\langle\langle A\rangle\rangle\bigcirc\psi' \in t(q)$. Let σ_A be the $\langle\langle A\rangle\rangle\bigcirc\psi'$-strategy and $\lambda \in \mathrm{out}(q, \sigma_A)$ a computation. By equation (3.1), $\lambda[1] = s_{\vec{t}}(\lambda[0])$ for some $\langle\langle A\rangle\rangle\bigcirc\psi'$-vector $\vec{t}$. From $\langle\langle A\rangle\rangle\bigcirc\psi' \in t(\lambda[0])$ and Claim 3.13, it follows that $\psi' \in t(\lambda[1])$. The induction hypothesis yields $\mathcal{S}, \lambda[1] \models \psi'$. Hence, $\mathcal{S}, q \models \langle\langle A\rangle\rangle\bigcirc\psi'$ by the semantics.

  "$\Leftarrow$": Suppose $\langle\langle A\rangle\rangle\bigcirc\psi' \notin t(q)$. Then $\neg\langle\langle A\rangle\rangle\bigcirc\psi' \in t(q)$ by (T2). Let σ_A be any strategy for the agents in the coalition A and λ a $\neg\langle\langle A\rangle\rangle\bigcirc\psi'$-computation for σ_A rooted at q. Note that by Claim 3.12 there is such a λ. Since $\lambda[1] = s_{\vec{t}}(\lambda[0])$ for some $\neg\langle\langle A\rangle\rangle\bigcirc\psi'$-vector $\vec{t}$, $\neg\psi' \in t(\lambda[1])$ by Claim 3.13. Condition (T2) yields $\psi' \notin t(\lambda[1])$. Thus $\mathcal{S}, \lambda[1] \not\models \psi'$ by the induction hypothesis. Hence, $\mathcal{S}, q \not\models \langle\langle A\rangle\rangle\bigcirc\psi'$ by the semantics.
- $\psi = \langle\langle A\rangle\rangle\Box\psi'$. "$\Rightarrow$": Suppose $\langle\langle A\rangle\rangle\Box\psi' \in t(q)$. Let σ_A be the $\langle\langle A\rangle\rangle\bigcirc\langle\langle A\rangle\rangle\Box\psi'$-strategy and $\lambda \in \mathrm{out}(q, \sigma_A)$ a computation. We show by induction on i that, for $i \geq 0$, the following holds:

  (1) $\langle\langle A\rangle\rangle\Box\psi' \in t(\lambda[i])$;
  (2) $\langle\langle A\rangle\rangle\bigcirc\langle\langle A\rangle\rangle\Box\psi' \in t(\lambda[i])$
  (3) $\psi' \in t(\lambda[i])$.

  For the base case, (1) is immediate since $\lambda[0] = q$. Thus, condition (T3) yields $\langle\langle A\rangle\rangle\bigcirc\langle\langle A\rangle\rangle\Box\psi' \in t(\lambda[0])$ and $\psi' \in t(\lambda[0])$ which gives us (2) and (3). For the induction step, the induction hypothesis gives us $\langle\langle A\rangle\rangle\bigcirc\langle\langle A\rangle\rangle\Box\psi' \in t(\lambda[i-1])$. By (3.1), $\lambda[i] = s_{\vec{t}}(\lambda[i-1])$ for some $\langle\langle A\rangle\rangle\bigcirc\langle\langle A\rangle\rangle\Box\psi'$-vector $\vec{t}$. By Claim 3.13, it follows that $\langle\langle A\rangle\rangle\Box\psi' \in t(\lambda[i])$ and thus we have (1). Now we may again use (T3) to infer $\langle\langle A\rangle\rangle\bigcirc\langle\langle A\rangle\rangle\Box\psi' \in t(\lambda[i])$ and $\psi' \in t(\lambda[i])$ and obtain (2) and (3). This finishes the induction on i. Finally, (3) and the induction hypothesis yield $\mathcal{S}, \lambda[i] \models \psi'$ for all $i \geq 0$ and we are done.

  "$\Leftarrow$": Suppose $\langle\langle A\rangle\rangle\Box\psi' \notin t(q)$. Then $\neg\langle\langle A\rangle\rangle\Box\psi' \in t(q)$ by (T2). Let σ_A be any strategy for the agents in A and λ a $\neg\langle\langle A\rangle\rangle\bigcirc\langle\langle A\rangle\rangle\Box\psi'$-computation for σ_A rooted at q. In the following, it is shown that $\neg\psi' \in t(\lambda[i])$ for some $i \geq 0$. Then (T2) yields $\psi' \notin t(\lambda[i])$ and the induction hypothesis $\mathcal{S}, \lambda[i] \not\models \psi'$. Hence, $\mathcal{S}, q \not\models \langle\langle A\rangle\rangle\Box\psi'$ by the semantics as desired.

  Suppose by contradiction that $\neg\psi' \notin t(\lambda[i])$ for all $i \geq 0$. By (T2), $\psi' \in t(\lambda[i])$ for all $i \geq 0$. We show by induction on i that, for $i \geq 0$, the following holds as well:

  (1) $\neg\langle\langle A\rangle\rangle\Box\psi' \in t(\lambda[i])$;

(2) $\neg\langle\!\langle A\rangle\!\rangle\bigcirc\langle\!\langle A\rangle\!\rangle\Box\psi' \in t(\lambda[i])$

For the base case, (1) has already been shown. Since $\psi' \in t(\lambda[0])$, (T3) and (1) imply $\langle\!\langle A\rangle\!\rangle\bigcirc\langle\!\langle A\rangle\!\rangle\Box\psi' \notin t(\lambda[0])$. Thus, (2) follows by (T2). For the induction step, the induction hypothesis gives us $\neg\langle\!\langle A\rangle\!\rangle\bigcirc\langle\!\langle A\rangle\!\rangle\Box\psi' \in t(\lambda[i-1])$. By definition of λ, $\lambda[i] = s_{\vec{t}}(\lambda[i-1])$ for some $\neg\langle\!\langle A\rangle\!\rangle\bigcirc\langle\!\langle A\rangle\!\rangle\Box\psi'$-vector $\vec{t}$. By Claim 3.13 and the matching condition (M2) in Definition 3.7 of $\bigcirc$-matching, we thus have $\neg\langle\!\langle A\rangle\!\rangle\Box\psi' \in t(\lambda[i])$. To establish (2), we may argue as in the base case.

By the matrix construction of $\mathcal{S}$, there is a position $i \geq 0$ such that $\lambda[i]$ is the root of the φ-tree $T_{\langle\vartheta,\Psi\rangle}$ where $\vartheta = \neg\langle\!\langle A\rangle\!\rangle\Box\psi'$ and $\Psi = t(\lambda[i])$. Since $\neg\langle\!\langle A\rangle\!\rangle\Box\psi' \in \Psi$ by (1), $T_{\langle\vartheta,\Psi\rangle}$ is a witness tree for the eventuality $\neg\langle\!\langle A\rangle\!\rangle\Box\psi'$ rooted at Ψ in Γ. The finiteness of $T_{\langle\vartheta,\Psi\rangle}$ implies that there is a position $j \geq i$ such that the type $t(\lambda[j])$ labels one of its leaf nodes $\lambda[j]$. Hence, $\neg\psi' \in t(\lambda[j])$ by definition of the witness tree $T_{\langle\vartheta,\Psi\rangle}$; a contradiction.

- $\psi = \langle\!\langle A\rangle\!\rangle\psi'\,\mathcal{U}\,\vartheta$. This case is similar to the previous one and left to the reader.

◀

Since $\varphi \in \Psi$ for some type $\Psi \in \Gamma$, there is a state $q \in Q$ such that $\psi \in t(q)$. Then it follows from Claim 3.14 that $\mathcal{S}, q \models \varphi$.

"$\Leftarrow$" (Completeness): Suppose φ is satisfiable in an ATS $\mathcal{S} = \langle \Pi, \Sigma, Q, \pi, \delta\rangle$ in a state $q_\varphi \in Q$. For each state $q \in Q$, let $t(q)$ be the type $\{\psi \in \mathsf{ecl}(\varphi) \mid \mathcal{S}, q \models \psi\}$. Denote with $\mathsf{types}(Q)$ the set of all types associated with some state in Q. We first establish the following claim:

CLAIM 3.15. *Let $q \in Q$ and $\vec{t} \in [k/n]$. Then there is a state $q' \in Q$ such that $S_{t(q)}(\vec{t}) \subseteq t(q')$. Moreover, the following holds: if $\langle\!\langle A\rangle\!\rangle\bigcirc\psi \in t(q)$ and σ_A is a strategy such that, for all computations $\lambda \in out(q, \sigma_A)$, we have $\mathcal{S}, \lambda[1] \models \psi$, we can choose q' such that $q' \in \sigma_A(q)$.*

PROOF OF CLAIM Let q and $\vec{t}$ be as in the claim. Also, select a formula $\langle\!\langle A\rangle\!\rangle\bigcirc\psi$ and a strategy σ_A as in the "moreover" part of the claim. Note first that, by Property 3 of the function f used in the definition of vectors for negative next-formulas, $\vec{t}$ is a vector for at most a single formula $\neg\langle\!\langle A'\rangle\!\rangle\bigcirc\psi'$ with $A' \subset \Sigma_\varphi$. Let

- $\langle\!\langle A_1\rangle\!\rangle\bigcirc\psi_1, \ldots, \langle\!\langle A_\ell\rangle\!\rangle\bigcirc\psi_\ell$ be all positive next-formulas from $t(q)$ for which $\vec{t}$ is a vector; this includes the selected formula $\langle\!\langle A\rangle\!\rangle\bigcirc\psi$;
- $\neg\langle\!\langle A'\rangle\!\rangle\bigcirc\psi'$ be the single negative next-formula from $t(q)$ with $A' \subset \Sigma_\varphi$ for which $\vec{t}$ is a vector, if such a formula exists;
- $\neg\langle\!\langle \Sigma_\varphi\rangle\!\rangle\bigcirc\psi''_1, \ldots, \neg\langle\!\langle \Sigma_\varphi\rangle\!\rangle\bigcirc\psi''_m$ be all negative next-formulas from $t(q)$ quantifying over the set of all agents Σ_φ.

Observe that, by definition, $\vec{t}$ is a vector for all negative next-formulas quantifying over Σ_φ, so that $\neg\psi''_1, \ldots, \neg\psi''_m \in S_{t(q)}(\vec{t})$. Next note that, by definition of vectors for positive next-formulas, we have $A_i \cap A_j = \emptyset$ for $1 \leq i < j \leq \ell$.

For $1 \leq i \leq \ell$, let σ_{A_i} be a strategy such that for all computations $\lambda \in \mathsf{out}(q, \sigma_{A_i})$, $\mathcal{S}, \lambda[1] \models \psi_i$. Such a strategy exists since $\langle\!\langle A_i \rangle\!\rangle \bigcirc \psi_i \in t(q)$. For the selected formula $\langle\!\langle A \rangle\!\rangle \bigcirc \psi$ from the "moreover" part of the claim, choose the selected strategy σ_A. Let $B = \bigcup_{1 \leq i \leq \ell} A_i$ and set $\sigma_B = \bigcup_{1 \leq i \leq \ell} \sigma_{A_i}$ which is well-defined since $A_i \cap A_j = \emptyset$ for $1 \leq i < j \leq \ell$. Thus for all $\lambda \in \mathsf{out}(q, \sigma_B)$, we have $\mathcal{S}, \lambda[1] \models \psi_i$ for $1 \leq i \leq \ell$.

Next, select a computation $\lambda \in \mathsf{out}(q, \sigma_B)$. If there is no negative next-formula in $t(q)$, for which $\vec{t}$ is a vector, then choose an arbitrary element $\lambda \in \mathsf{out}(q, \sigma_B)$ ($\mathsf{out}(q, \sigma_B)$ is non-empty since $\delta(q, A)$ is non-empty for all q and A). Otherwise, choose a $\lambda \in \mathsf{out}(q, \sigma_B)$ such that $\mathcal{S}, \lambda[1] \models \neg\psi'$. Such an element exists since, first, $\neg\langle\!\langle A' \rangle\!\rangle \bigcirc \psi'$ is in $t(q)$ and, second, $\vec{t}$ being a vector for this formula implies (by property 2 of the refutation function f used in the definition of such vectors) that $A_i \subseteq A'$ for $1 \leq i \leq \ell$.

Finally, we have $\mathcal{S}, \lambda[1] \models \neg\psi_i''$ for $1 \leq i \leq m$ since $\neg\langle\!\langle \Sigma_\varphi \rangle\!\rangle \bigcirc \psi_i'' \in t(q)$ implies that $\mathcal{S}, \lambda'[1] \models \neg\psi_i''$ for any computation λ' rooted at q.

Summing up, we have shown that $S_{t(q)}(\vec{t}) \subseteq t(\lambda[1]) \in \mathsf{types}(Q)$. Thus, $\lambda[1]$ is the state whose existence is stated in the claim. ◀

In the following, it is shown that all types in $\mathsf{types}(Q)$ are ATL-realizable in $\mathsf{types}(Q)$. Let $q \in Q$ be a state. We have to check that each type $t(q)$ in $\mathsf{types}(Q)$ satisfies conditions 1 to 3 of Definition 3.9.

1. Let $\vec{t} \in [k/n]$. We have to show that $S_{t(q)}(\vec{t}) \subseteq \Psi$ for some $\Psi \in \mathsf{types}(Q)$. Clearly, this is an immediate consequence of Claim 3.15 (the "moreover" part is not needed).
2. Suppose $\langle\!\langle A \rangle\!\rangle \psi \mathcal{U} \vartheta \in t(q)$. It is our aim to construct a $\langle\!\langle A \rangle\!\rangle \psi \mathcal{U} \vartheta$-witness tree rooted at the type $t(q)$ in $\mathsf{types}(Q)$. Since $\mathcal{S}, q \models \langle\!\langle A \rangle\!\rangle \psi \mathcal{U} \vartheta$, there is a strategy σ_A such that for all computations $\lambda \in \mathsf{out}(q, \sigma_A)$, there is a position $i \geq 0$ such that $\mathcal{S}, \lambda[i] \models \vartheta$ and $\mathcal{S}, \lambda[j] \models \psi$ for $j < i$.

   Using the semantics, it is not difficult to prove that σ_A satisfies the following property: if $\lambda \in \mathsf{out}(q, \sigma_A)$ and $i \in \mathbb{N}$ is smallest such that $\mathcal{S}, \lambda[i] \models \vartheta$, then
   (a) $\mathcal{S}, \lambda[j] \models \langle\!\langle A \rangle\!\rangle \psi \mathcal{U} \vartheta$ for $j \leq i$;
   (b) $\mathcal{S}, \lambda[j] \models \langle\!\langle A \rangle\!\rangle \bigcirc \langle\!\langle A \rangle\!\rangle \psi \mathcal{U} \vartheta$ for $j < i$;
   (c) $\mathcal{S}, \lambda[j] \models \psi$ for $j < i$.

   We use σ_A to define a $\langle Q, k, n \rangle$-tree T. This tree can then easily be converted into the required witness-tree. For a member α of $[k/n]^*$, denote by $\alpha[0, i]$ the initial segment of α of length $i + 1$. In particular, α coincides with $\alpha[0, |\alpha| - 1]$, where $|\alpha|$ denotes the length of α. Now, the construction proceeds by induction as follows: In the induction start, set $T(\varepsilon) := q$. In the induction step, let $\alpha \in \mathsf{dom}(T)$ be such such that $T(\alpha)$ is already defined. If $\mathcal{S}, T(\alpha) \models \vartheta$, then α is a leaf and we do not further extend this branch. Otherwise, for each $\langle\!\langle A \rangle\!\rangle \bigcirc \langle\!\langle A \rangle\!\rangle \psi \mathcal{U} \vartheta$-vector $\vec{t}$, set $T(\alpha \cdot \vec{t}) := q'$ for some $q' \in Q$ such that
   (i) $S_{t(T(\alpha))}(\vec{t}) \subseteq t(q')$, and
   (ii) $q' \in \sigma_A(\lambda)$ where $\lambda = T(\alpha[0,0])\, T(\alpha[0,1]) \cdots T(\alpha[0, |\alpha| - 1])$.

The existence of such a q' is a consequence of Claim 3.15: since $\lambda = \lambda' \cdot T(\alpha)$ for some λ', there exists a strategy σ'_A such that $\sigma'_A(T(\alpha)) = \sigma_A(\lambda)$. Take any such strategy. Now apply Claim 3.15 including the "moreover" part, using the next-formula $\langle\langle A\rangle\rangle\bigcirc\langle\langle A\rangle\rangle\psi\,\mathcal{U}\,\vartheta$ and the strategy σ'_A, to get the desired state q'. Note that the prerequisites of the "moreover" part are satisfied:

- $\mathcal{S}, T(\alpha) \models \langle\langle A\rangle\rangle\bigcirc\langle\langle A\rangle\rangle\psi\,\mathcal{U}\,\vartheta$ holds by (b);
- for all computations $\lambda \in \mathsf{out}(T(\alpha), \sigma'_A)$, we have $\mathcal{S}, \lambda[1] \models \langle\langle A\rangle\rangle\psi\,\mathcal{U}\,\vartheta$ since σ'_A is based on σ_A and by (a).

We now show that T is finite. Suppose by contradiction that there is an infinite path $\tau \in [k/n]^\omega$ in T. Let $\lambda \in Q^\omega$ be the infinite sequence defined by setting $\lambda[i] := T(\tau[0,i])$. By (ii), $\lambda \in \mathsf{out}(q, \sigma_A)$. Then there is a position $i \geq 0$ such that $\mathcal{S}, \lambda[i] \models \vartheta$. Thus $\vartheta \in t(\lambda[i]) = t(T(\tau[0,i]))$ and the node $\tau[0, i+1]$ is a leaf; a contradiction.

Since the nodes in T are labelled by states, the composition $t(T(\cdot))$ yields a finite φ-tree. To show that $t(T(\cdot))$ is a $\langle\langle A\rangle\rangle\psi\,\mathcal{U}\,\vartheta$-witness tree rooted at $t(q)$ in $\mathsf{types}(Q)$, it remains to show that T satisfies properties 1 to 7 in the definition of a $\langle\langle A\rangle\rangle\psi\,\mathcal{U}\,\vartheta$-witness tree: properties 1, 3, 6 and 7 are immediate by definition of T; property 2 is a consequence of (i), property (4) a consequence of (a) and property 5 a consequence of (c).

3. This case is similar to the previous one and left to the reader.

From $\mathcal{S}, q_\varphi \models \varphi$ it follows that $\varphi \in t(q_\varphi)$. Then $\mathsf{types}(Q)$ is a set of types that are each ATL-realizable in $\mathsf{types}(Q)$ and $t(q_\varphi)$ is a type in $\mathsf{types}(Q)$ such that $\varphi \in t(q_\varphi)$. Let Δ be the set of types for φ computed by the type elimination algorithm. It is easy to see that $\mathsf{types}(Q) \subseteq \Delta$. Hence, the algorithm returns "Yes, the input formula φ is satisfiable". □

LEMMA 3.12. *The described elimination procedure runs in exponential time.*

PROOF. Suppose φ is given and let $n = |\varphi|$. Recall that the size of the extended closure $\mathsf{ecl}(\varphi)$ is linear in the length of φ, i.e., $|\mathsf{ecl}(\varphi)| = c \cdot n$ for some constant $c \geq 1$. The algorithm computes a sequence $\Delta_0, \ldots, \Delta_m$ of sets of types such that $\Delta_0 \supsetneq \Delta_1 \supsetneq \cdots \supsetneq \Delta_m$. Since $\Delta_0 = \Gamma_\varphi \subseteq 2^{\mathsf{ecl}(\varphi)}$, this sequence is finite with $m \leq 2^{c\cdot n}$. For each i with $0 \leq i < m$, it holds that $|\Delta_i| < |\Delta_0| \leq 2^{c\cdot n}$. Thus, to compute the set Δ_{i+1} at most $2^{c\cdot n}$ types in Δ_i need to be checked whether they are ATL-realizable in Δ_i. In the following, it is shown that for a type $\Psi \in \Delta_i$ at most $2^{\mathcal{O}(n^2)}$ steps are needed to check for Ψ's realizability in Δ_i. Consider the 3 points in Definition 3.9:

1. For at most for n^n vectors (as $k \leq n$), inclusion tests for maximal $2^{c\cdot n}$ types in Δ_i have to be performed. Hence, this takes not more than $n^n \cdot 2^{c\cdot n} = 2^{\mathcal{O}(n^2)}$ steps.

2.,3. By Lemma 3.10, to decide the existence of witness trees takes not more than $2^{\mathcal{O}(n^2)}$ steps. This has to be done for at most n formulas of the form $\langle\langle A\rangle\rangle\psi\,\mathcal{U}\,\vartheta$ or $\neg\langle\langle A\rangle\rangle\Box\psi$ in Ψ. Thus, altogether maximal $2^{\mathcal{O}(n^2)}$ steps are needed.

Note that checking whether there is a type in Δ_m that contains φ takes not more than $2^{O(n)}$ steps. We conclude that our decision procedure runs in time exponential in the size of the input. □

3.2. Complexity of ATEL

In what follows, we consider ATEL, the extension of ATL by epistemic operators for individual knowledge, common and distributed knowledge as defined in Section 2.5.1 in Chapter 2.

In 2004, Goranko, Jamroga and van Drimmelen [97] axiomatised the fragment of ATEL without operators for distributed knowledge. From this axiomatisation, it becomes clear that the extension of ATL by operators for individual and common knowledge is the independent fusion ATL$\otimes$S5^C in the sense of [88] of the logics ATL and S5^C. Although both logics, ATL [**252, 265**] and S5^C [**103**], are ExpTime-complete, it does not follow from general results on fusions of modal logics that the fusion is ExpTime-complete as well. For instance, an increase in computational complexity when taking fusions can be observed with the fusion of S5 with itself: while S5 is in NP [**103**], the fusion is PSpace-hard [**88**]. From general results on complexity transfer for fusions [**27, 25, 26**], it only follows that ATL$\otimes$S5^C is in 2-ExpTime. In the next section, we show that the satisfiability problem for ATEL is ExpTime-complete using a type elimination construction that extends the construction for ATL presented in the previous section with a technique of Halpern and Moses [**103**] to handle the additional knowledge operators. This complexity result for ATEL was presented in [**264**].

As for ATL, when formulating satisfiability problems for ATEL some care is needed as it was analyzed in Section 3.1.1. In particular, the range of semantic structures, over which a formula is to be interpreted, needs to be specified. Three variants of the satisfiability problem for ATL were suggested depending on the possibilities for the number of agents to occur in semantic structures. For ATEL, however, we concentrate only on one of these problems; the other two satisfiability problems can be reduced to this:

> *Satisfiability over formula-defined sets of agents:* Given a formula φ, is φ satisfiable in a structure for exactly the agents which occur in φ?

In the next section, we show that this satisfiability problem for ATEL is ExpTime-complete.

3.2.1. ATEL Decision Procedure. In this section, we show that ATEL satisfiability is ExpTime-complete by extending the ExpTime-completeness result for ATL in Section 3.1.2 to ATEL, which shows that adding epistemic operators for common knowledge and distributed knowledge to ATL does not yield an increase in computational complexity.

Theorem 3.13. *The satisfiability problem for ATEL is* ExpTime-*complete.*

The lower complexity bound carries over from the fragment ATL, which was shown to be EXPTIME-hard at the end of Section 3.1.2. For the upper bound, we now show that ATEL satisfiability is in EXPTIME. Correctness of the decision procedure follows from Lemma 3.19 while Lemma 3.20 proves its exponential running time.

The decision procedure implements an extended version of the type elimination construction for ATL from Section 3.1.2 that additionally accounts for ATEL's epistemic operators K_a, E_A, C_A and D_A for each agent a and coalition A of agents where the epistemic operators are dealt with similarly to [103]. We start with refining the definition of an extended closure with knowledge operators. The closure for an input formula φ contains all formulas that are relevant for deciding φ.

DEFINITION 3.14 (Extended Closure with Epistemic Operators). Let φ be an ATEL-formula and n the number of agents occurring in φ. The *extended closure* $\mathsf{ecl}(\varphi)$ *of* φ is the smallest set which is closed under the following conditions:

- $\varphi \in \mathsf{ecl}(\varphi)$;
- $\mathsf{ecl}(\varphi)$ is closed under subformulas;
- $\mathsf{ecl}(\varphi)$ is closed under single negation;
- if $\langle\langle A\rangle\rangle\Box\psi \in \mathsf{ecl}(\varphi)$, then $\langle\langle A\rangle\rangle\bigcirc\langle\langle A\rangle\rangle\Box\psi \in \mathsf{ecl}(\varphi)$;
- if $\langle\langle A\rangle\rangle\psi\,\mathcal{U}\,\vartheta \in \mathsf{ecl}(\varphi)$, then $\langle\langle A\rangle\rangle\bigcirc\langle\langle A\rangle\rangle\psi\,\mathcal{U}\,\vartheta \in \mathsf{ecl}(\varphi)$;
- if $E_A\psi \in \mathsf{ecl}(\varphi)$ with $A = \{a_1, \ldots, a_\ell\}$, then $\neg(\neg K_{a_1}\psi \vee \cdots \vee \neg K_{a_\ell}\psi) \in \mathsf{ecl}(\varphi)$;
- if $C_A\psi \in \mathsf{ecl}(\varphi)$, then $E_A C_A\psi \in \mathsf{ecl}(\varphi)$;
- if $K_a\psi \in \mathsf{ecl}(\varphi)$ and $D_A\vartheta \in \mathsf{ecl}(\varphi)$ with $a \in A$, then $D_A\psi \in \mathsf{ecl}(\varphi)$;
- if $D_A\psi \in \mathsf{ecl}(\varphi)$ and $D_B\vartheta \in \mathsf{ecl}(\varphi)$ with $A \subseteq B$, then $D_B\psi \in \mathsf{ecl}(\varphi)$.

Note that the cardinality of $\mathsf{ecl}(\varphi)$ is linear in the length of φ.

Now the notion of types is adapted to additionally account for epistemic operators.

DEFINITION 3.15 (ATEL Type). Let φ be an ATEL-formula. The set $\Psi \subseteq \mathsf{ecl}(\varphi)$ is a *type for* φ if the following conditions are satisfied:

(T1) $\psi_1 \vee \psi_2 \in \Psi$ iff $\psi_1 \in \Psi$ or $\psi_2 \in \Psi$, for all $\psi_1 \vee \psi_2 \in \mathsf{ecl}(\varphi)$;

(T2) $\psi \in \Psi$ iff $\neg\psi \notin \Psi$, for all $\neg\psi \in \mathsf{ecl}(\varphi)$;

(T3) $\langle\langle A\rangle\rangle\Box\psi \in \Psi$ iff $\{\psi, \langle\langle A\rangle\rangle\bigcirc\langle\langle A\rangle\rangle\Box\psi\} \subseteq \Psi$, for all $\langle\langle A\rangle\rangle\Box\psi \in \mathsf{ecl}(\varphi)$;

(T4) $\langle\langle A\rangle\rangle\psi\,\mathcal{U}\,\vartheta \in \Psi$ iff $\vartheta \in \Psi$ or $\{\psi, \langle\langle A\rangle\rangle\bigcirc\langle\langle A\rangle\rangle\psi\,\mathcal{U}\,\vartheta\} \subseteq \Psi$, for all $\langle\langle A\rangle\rangle\psi\,\mathcal{U}\,\vartheta \in \mathsf{ecl}(\varphi)$;

(T5) if $K_a\psi \in \Psi$, then $\psi \in \Psi$;

(T6) $E_A\psi \in \Psi$ iff $\neg(\neg K_{a_1}\psi \vee \cdots \vee \neg K_{a_\ell}\psi) \in \Psi$, for all $E_A\psi \in \mathsf{ecl}(\varphi)$ with $A = \{a_1, \ldots, a_\ell\}$;

(T7) $C_A\psi \in \Psi$ iff $\{\psi, E_A C_A\psi\} \subseteq \Psi$, for all $C_A\psi \in \mathsf{ecl}(\varphi)$;

(T8) if $D_A\psi \in \Psi$, then $\psi \in \Psi$;

(T9) if $K_a\psi \in \Psi$, then $D_A\psi \in \Psi$, for all $D_A\vartheta \in \mathsf{ecl}(\varphi)$ with $a \in A$;

(T10) if $D_A\psi \in \Psi$, then $D_B\psi \in \Psi$, for all $D_B\vartheta \in \mathsf{ecl}(\varphi)$ with $A \subseteq B$.

The set of all types for φ is designated by Γ_φ.

Note that the cardinality of Γ_φ is exponential in the length of φ.

Intuitively, an ATEL type describes a state q of a transition system by means of formulas from $\mathsf{ecl}(\varphi)$ which are true at q. The decision procedure will use all types for an input formula to determine its decidability. The conditions (T1) to (T10) state restrictions on the formulas in a type: (T1) to (T4) deal with Boolean and temporal formulas; the conditions (T5) to (T10) were added to handle epistemic formulas. In particular, the relation between formulas with epistemic operators K and D are taken care of by (T9) and (T10).

As for ATL, we assume that all next formulas in the extended closure $\mathsf{ecl}(\varphi)$ are linearly ordered such that no negative next formula occurs before a positive one. We denote with $\sharp_\psi$ the number of the next formula $\psi \in \mathsf{ecl}(\varphi)$ in this ordering; the numbering starts with 0. Since there are as many positive next formulas in $\mathsf{ecl}(\varphi)$ as negative ones, obtain the following enumeration $\psi_1, \ldots \psi_k$ with $\psi_1, \ldots, \psi_{k/2-1}$ positive next formulas and $\psi_{k/2}, \ldots, \psi_k$ negative next formulas.

Intuitively, the presented type elimination algorithm eliminates those types that cannot be used for building an AETS S that is a model for an input formula φ. Every state q of S corresponds to some type that was not eliminated and the construction of S is aiming at satisfying every formula of this type at q. In particular, q will have to satisfy a number of positive and negative next formulas, negated box formulas and until formulas. These formulas are dealt with as for ATL: satisfying a positive next formula corresponds to a coalition voting for it (cf. explanation (ii) in Section 3.1.2); for satisfying negative next formulas, we use the notion of a refutation function (cf. Definition 3.6); and for negated box formulas and until formulas, we use so-called witness trees that witness the satisfaction of these formulas at some states in the model (cf. Definition 3.7 and 3.8). The construction of a model, however, is more involved due to the presence of formulas of the form $\neg C_A\psi$. This is because the satisfaction of such formulas needs to be witnessed by some state in a model. Intuitively, since epistemic and temporal operators do not interact in ATEL, formulas of the form $\neg C_A\psi$ can be treated separately with a notion of witness paths, which are defined as follows.

Definition 3.16 (Witness Path). Let φ be an ATEL-formula, Γ a set of types for φ and $\Psi \in \Gamma$. A sequence $\Psi_0 \cdots \Psi_m \in \Gamma^*$, $m \geq 1$, is a *witness path rooted at* Ψ *in* Γ *for a formula* $\neg C_A\psi$ if the following three conditions are satisfied:

(1) $\Psi_0 = \Psi$;

(2) for all $i < m$, there is an agent $a \in A$ such that for all $K_a\psi' \in \mathsf{ecl}(\varphi)$, $K_a\psi' \in \Psi_i$ iff $K_a\psi' \in \Psi_{i+1}$;

(3) $\neg\psi \in \Psi_m$.

As for ATL, the decision procedure for ATEL employs a type elimination algorithm which in turn relies essentially on the notion of *realizability*. Realizability for ATL (cf. Definition 3.9) is extended with three additional conditions to account for epistemic operators K_a, C_A and D_A. Intuitively, a type Ψ is ATEL-realizable in a set of types Γ if it is possible to (a) satisfy all next-formulas in Ψ and all formulas of the form $\neg K_a\psi$ or $\neg D_A\psi$ in Ψ using only types from Γ, (b) construct witness trees for all negated box formulas and until formulas in Ψ, and (c) construct witness paths for all formulas of the form $\neg C_A\psi$ in Ψ using only types from Γ.

DEFINITION 3.17 (ATEL Realizability). Let φ be an ATEL-formula and Γ a set of types for φ. A type $\Psi \in \Gamma$ is *ATEL-realizable in* Γ if the following conditions are satisfied:

1. for all $\vec{t} \in [k/n]$, there is a $\Psi' \in \Gamma$ such that $S_\Psi(\vec{t}) \subseteq \Psi'$;
2. for all $\langle\langle A\rangle\rangle\psi\mathcal{U}\vartheta \in \Psi$, there is a $\langle\langle A\rangle\rangle\psi\mathcal{U}\vartheta$-witness tree rooted at Ψ in Γ;
3. for all $\neg\langle\langle A\rangle\rangle\Box\psi \in \Psi$, there is a $\neg\langle\langle A\rangle\rangle\Box\psi$-witness tree rooted at Ψ in Γ;
4. for all $\neg K_a\psi \in \Psi$, there is a $\Psi' \in \Gamma$ such that
   (a) $\neg\psi \in \Psi'$, and
   (b) $K_a\psi' \in \Psi$ iff $K_a\psi' \in \Psi'$, for all $K_a\psi' \in \mathsf{ecl}(\varphi)$;
5. for all $\neg C_A\psi \in \Psi$, there is $\neg C_A\psi$-witness path rooted at Ψ in Γ;
6. for all $\neg D_A\psi \in \Psi$, there is a $\Psi' \in \Gamma$ such that
   (a) $\neg\psi \in \Psi'$, and
   (b) $D_A\psi' \in \Psi$ iff $D_A\psi' \in \Psi'$, for all $D_A\psi' \in \mathsf{ecl}(\varphi)$.

Finally, we describe the decision procedure for ATEL using type elimination. The algorithm is presented as function ATEL–$sat(\varphi)$ in Figure 3.3. For the input formula φ,

```
1. function ATEL–sat($\varphi$) returns 'Yes, ...' or 'No, ...'
2.    $m := 0$
3.    $\Delta_m := \Gamma_\varphi$
4.    do
5.       $m := m + 1$
6.       $\Delta_m := \{\Psi \in \Delta_{m-1} \mid \Psi$ is ATEL-realizable in $\Delta_{m-1}\}$
7.    until $\Delta_m = \Delta_{m-1}$
8.    if $\varphi \in \Psi$, for some $\Psi \in \Delta_m$,
9.       then return 'Yes, $\varphi$ is satisfiable in an AETS for $\Sigma_\varphi$.'
10.      else return 'No, $\varphi$ is not satisfiable.'
11. end-function
```

FIGURE 3.3. A type-elimination algorithm for ATEL.

the algorithm starts with the set of all types Γ_φ for φ (line 2 and 3) and repeatedly eliminates the types that are not ATEL-realizable (lines 4–7). Since there are only finitely many types to start with, the algorithm eventually leaves the loop with a set Δ_m of types for some $m \geq 0$ (line 7). Notice that, at this point, Δ_m is a set of types for ψ that are all ATEL-realizable in Δ_m. The algorithm returns 'Yes, φ is satisfiable in an AETS for Σ_φ' if the input formula φ is contained in some type of Δ_m (line 8 and 9); otherwise it returns 'No, φ is not satisfiable.' (line 8 and 10).

As for the ATL decision procedure, we state three lemmas to establish that the described decision procedure for ATEL is correct and can be computed within exponential running time in the length of the input formula.

LEMMA 3.18. *Let Γ be a set of types for an ATEL-formula φ. Then the existence of witness trees and paths in Γ can be decided in time exponential in the length of φ.*

LEMMA 3.19. *Let φ be an ATEL-formula. Then the procedure returns 'Yes, the input formula φ is satisfiable in an AETS for Σ_φ' iff it is indeed the case.*

LEMMA 3.20. *The described type elimination procedure runs in exponential time.*

The proofs are presented in the following section. The lemmas 3.18 and 3.20 can be shown in a fairly similar way to the corresponding lemmas for the ATL decision procedure: Lemma 3.10 and Lemma 3.12, respectively. The proof of Lemma 3.19, however, needs to be extended compared to Lemma 3.11 in order to account for the additional epistemic operators.

We now give a brief outline of the proof of Lemma 3.19. For showing soundness, i.e., the direction from left to right, we assume that the elimination procedure was started on input φ and returns 'Yes, the input formula φ is satisfiable'. Then all types of the computed set, Γ say, are ATEL-realizable in Γ and there is a type $\Psi \in \Gamma$ with $\varphi \in \Psi$. We construct an AETS S of φ where states correspond to types in Γ. This is done by constructing witness trees, for all until formulas and negated box formulas in the closure $\mathsf{ecl}(\varphi)$, and, for each type in Γ, arranging them in a matrix form such that the rows range over the eventualities and the columns over the types. Then we replace all leaf nodes by an edge from the leaf node's predecessor to the root of some other witness tree in the next row of the matrix such that this root is labelled by the same type as the replaced leaf. This construction is similar to the one for CTL; cf. [69]. So far, the proof is similar to the proof of the Lemma 3.11 for ATL. Initially, the epistemic operator D_A is interpreted over an epistemic accessibility relation $\sim^D_A$ in S that is explicitly defined, i.e., the equation $\sim^D_A = \bigcap_{a \in A} \sim_a$, for all coalitions A, does not necessarily hold. However, a proper AETS S' can be constructed in terms of S by introducing epistemic witness trees on each state in S and rearranging, for each agent a, its epistemic accessibility relation $\sim_a$ such that temporal and epistemic transitions are kept separate. For completeness, i.e., the right-to-left direction, we suppose that φ is satisfiable in an AETS S. Take Q to be the set of states in S and denote with $\mathsf{types}(Q)$ the set of all ATEL types associated with some state in Q. Then, it can be shown that all ATEL types in $\mathsf{types}(Q)$ are ATEL-realizable in $\mathsf{types}(Q)$. The formulas where the outermost operator is an ATL path quantifier can be dealt with as in Lemma 3.11; the other formulas with epistemic operators as outermost operators are treated explicitly.

3.2.2. Proofs for ATEL Decision Procedure.

LEMMA 3.23. *Let Γ be a set of types for an ATEL-formula φ. Then the existence of witness trees and paths in Γ can be decided in time exponential in the length of φ.*

PROOF. The proof for until formulas and negated box formulas is as the proof of Lemma 3.10. In the following, we only concentrate on checking for the existence of a witness path for formulas of the form $\neg C_A\psi$, which works in a similar fashion as for witness trees. Suppose we have to check for the existence of a witness path for $\neg C_A\psi \in \Psi_0$ rooted at Ψ_0 in Γ. Start with marking all types in Γ which contain $\neg C_A\psi$ and $\neg\psi$. For all unmarked types $\Psi \in \Gamma$ with $\neg C_A\psi \in \Psi$, mark Ψ if there is a type $\Psi' \in \Gamma$ such that for some agent $a \in A$ it holds that

(i) for each $K_a\psi' \in \mathsf{ecl}(\varphi)$, $K_a\psi' \in \Psi$ iff $K_a\psi' \in \Psi'$, and
(ii) Ψ' is marked.

Repeat this procedure until no more types in Γ get marked. The required witness path exists if Ψ_0 was marked; otherwise not. This can be shown similarly to the argument above.

Again, it is not hard to verify that a witness path for $\neg C_A\psi$ exists iff Ψ_0 was marked. This can be done in a similar way as for witness trees.

In the following, the complexity of the algorithm that checks for witness paths is discussed. Initially, at most $2^{c\cdot n}$ types containing $\neg C_A\psi$ and $\neg\psi$ will be marked. Then in each marking round, at least one type gets marked, i.e., there are not more that $2^{c\cdot n}$ marking rounds. In each such round, maximal $2^{c\cdot n}$ yet unmarked types have to be checked. In order to find out whether to mark such a type, not more than $2^{c\cdot n}$ types have to be considered. Then for maximal n agents, each such type is to be checked whether it satisfies conditions (i) and (ii). Altogether this yields an upper bound of $2^{c\cdot n} + 2^{c\cdot n} \cdot 2^{c\cdot n} \cdot 2^{c\cdot n} \cdot n = 2^{\mathcal{O}(n)}$ steps. Consequently, checking for existence of witness paths is time exponential in the length of φ. □

LEMMA 3.24. *Let φ be an ATEL-formula. Then the procedure returns 'Yes, the input formula φ is satisfiable in an AETS for Σ_φ' iff it is indeed the case.*

PROOF. Suppose φ is given. Let $\Sigma = \Sigma_\varphi$, $n = |\Sigma_\varphi|$ and k be the number of next-formulas in the extended closure $\mathsf{ecl}(\varphi)$.

"$\Rightarrow$" (Soundness) Assume that the elimination procedure was started on input φ and returns "Yes, the input formula φ is satisfiable". Let $\Gamma = \{\Psi_0, \ldots, \Psi_{m-1}\}$ be the computed set of types. Then all types of Γ are ATEL-realizable in Γ and there is a type $\Psi \in \Gamma$ with $\varphi \in \Psi$. Our aim is to construct an AETS that is a model of φ.

To this end, enumerate all eventualities in $\mathsf{ecl}(\varphi)$ by $\psi_0, \ldots, \psi_{\ell-1}$. For each i with $i < \ell$ and each j with $j < m$, fix a φ-tree $T_{\langle\psi_i,\Psi_j\rangle}$ as follows:

- If $\psi_i \in \Psi_j$, then fix a ψ_i-witness tree T rooted at Ψ_j in Γ. Supplement all inner nodes of T with missing successors: for each inner node $\alpha \in \mathsf{dom}(T)$ and each $\vec{t} \in [k/n]$, if $\alpha \cdot \vec{t} \notin \mathsf{dom}(T)$, then add $\alpha \cdot \vec{t}$ to $\mathsf{dom}(T)$ and set $T(\alpha \cdot \vec{t}) = \Psi$ for some $\Psi \in \Gamma$ such that $S_{T(\alpha)}(\vec{t}) \subseteq \Psi$. Note that such a Ψ must exist by Condition 1 in Definition 3.17 of ATEL realizability. Let $T_{\langle\psi_i,\Psi_j\rangle}$ be the result of augmenting T in this way.

- If $\psi_i \notin \Psi_j$, then let $T_{\langle\psi_i,\Psi_j\rangle}$ be the tree comprised of the nodes $\{\varepsilon\} \cup [k/n]$ such that $T_{\langle\psi_i,\Psi_j\rangle}(\varepsilon) = \Psi_j$ and, for each $\vec{t} \in [k/n]$, $T_{\langle\psi_i,\Psi_j\rangle}(\vec{t}) = \Psi$ for some $\Psi \in \Gamma$ with $S_{\Psi_j}(\vec{t}) \subseteq \Psi$.

Verify that all trees $T_{\langle\psi_i,\Psi_j\rangle}$ are $\bigcirc$-matching. To construct a model of φ, intuitively we do the following: we arrange the selected witness trees in an $\ell \times m$-matrix such that the rows range over the eventualities $\psi_0, \dots, \psi_{\ell-1}$ and the columns over the types $\Psi_0, \dots, \Psi_{m-1}$, and then replace all leaf nodes by an edge from the leaf node's predecessor to the root of some other witness tree such that this root is labelled by the same type as the replaced leaf.

Now define an AETS $\mathcal{S} = (\Pi, \Sigma, Q, \sim_1, \dots, \sim_n, \pi, \delta)$ that will be shown to satisfy φ. Π and Σ are the sets of those atomic propositions and agents that occur in the input formula φ. For defining the set of states Q, fix symbols $\varepsilon_{i,j}$ with $i \leq \ell$ and $j \leq m$. Then set:

$$Q \;:=\; \{\varepsilon_{i,j}w \mid w \in \mathsf{dom}(T_{\langle\psi_i,\Psi_j\rangle}) \text{ is an inner node of } T_{\langle\psi_i,\Psi_j\rangle}\}.$$

Next, the valuation π is easily defined: for $q = \varepsilon_{i,j}w \in Q$, set

$$\pi(q) \;:=\; T_{\langle\psi_i,\Psi_j\rangle}(w) \cap \Pi.$$

To define the transition function δ, first define a successor function on Q: for each $q = \varepsilon_{i,j}w \in Q$ and each $\vec{t} \in [k/n]$, set

$$s_{\vec{t}}(q) \;:=\; \begin{cases} \varepsilon_{s,p} & \text{if } w \cdot \vec{t} \text{ is a leaf node of } T_{\langle\psi_i,\Psi_j\rangle}, \\ & s = i+1 \bmod \ell \text{ and } T_{\langle\psi_i,\Psi_j\rangle}(w \cdot \vec{t}) = \Psi_p \\ q \cdot t & \text{if } w \cdot \vec{t} \text{ is an inner node of } T_{\langle\psi_i,\Psi_j\rangle} \end{cases}$$

Now the definition of δ is straightforward: for each $q \in Q$ and $a \in \Sigma$, set

$$\delta(q,a) \;:=\; \{\{s_{\vec{t}}(q) \mid \vec{t} = (t_0, \dots, t_{n-1}) \in [k/n] \text{ and } t_a = p\} \mid p < k/2\}.$$

The epistemic accessibility relations $\sim_a$ are defined as follows. Denote with $t(q)$ the intended type $T_{\langle\psi_i,\Psi_j\rangle}(w)$ of a state $q = \varepsilon_{i,j}w \in Q$. For each $q, q' \in Q$ and $a \in \Sigma$, set

$$q \sim_a q' \text{ iff for all } K_a\psi' \in \mathsf{ecl}(\varphi),\ K_a\psi' \in t(q) \text{ iff } K_a\psi' \in t(q').$$

In the following, each $D_A\vartheta \in \mathsf{ecl}(\varphi)$ is interpreted by the relation $\sim_A^D$, explicitly defined by setting: for all $q, q' \in Q$,

$$q \sim_A^D q' \text{ iff for all } D_A\psi' \in \mathsf{ecl}(\varphi),\ D_A\psi' \in t(q) \text{ iff } D_A\psi' \in t(q'). \tag{3.2}$$

Note that the condition $\sim_A^D = \bigcap_{a\in A} \sim_a$ does not necessarily hold. Thus $\mathcal{S}$ plus all $\sim_A^D$ is in general not a proper AETS. Nevertheless, for the time being $\mathcal{S}$ will be used to interpret ATEL-formulas with distributed knowledge operator D. To show that $\mathcal{S}$ is indeed a model of φ, we introduce some auxiliary notions:

For each strategy $\sigma_A = \{\sigma_a \mid a \in A\}$ for a set of agents $A \subseteq \Sigma$ and each sequence of states $\lambda \in Q^+$, we write $\sigma_A(\lambda)$ to denote the set of states $\bigcap_{a\in A} \sigma_a(\lambda)$. Observe that, by definition of strategies for single agents, we have $\sigma_A(\lambda \cdot q) \in \delta(q, A)$ for all $\lambda \in Q^*$ and $q \in Q$.

For each positive next-formula $\langle\langle A\rangle\rangle\bigcirc\psi$, the *$\langle\langle A\rangle\rangle\bigcirc\psi$-strategy* is the strategy $\sigma_A = \{\sigma_a \mid a \in A\}$ for the set of agents A that is defined by setting, for each $a \in A$, $\lambda \in Q^*$ and $q \in Q$.

$$\sigma_a(\lambda \cdot q) \ := \ \{s_{\vec{t}}(q) \mid \vec{t} = (t_0, \ldots, t_{n-1}) \text{ and } t_a = \sharp_{\langle\langle A\rangle\rangle\bigcirc\psi}\}.$$

It is readily checked that we have

$$\sigma_A(\lambda \cdot q) = \{s_{\vec{t}}(q) \mid \vec{t} \text{ is a } \langle\langle A\rangle\rangle\bigcirc\psi\text{-vector}\}.$$

For each negative next-formula $\neg\langle\langle A\rangle\rangle\bigcirc\psi$, a *$\neg\langle\langle A\rangle\rangle\bigcirc\psi$-computation for a strategy σ_A rooted at a state* $q \in Q$ is a computation $\lambda \in \mathsf{out}(q, \sigma_A)$ such that, for all positions $i \geq 0$, $\lambda[i+1] = s_{\vec{t}}(\lambda[i])$ for some $\neg\langle\langle A\rangle\rangle\bigcirc\psi$-vector $\vec{t} \in [k/n]$.

CLAIM 3.25. *Let $\neg\langle\langle A\rangle\rangle\bigcirc\psi$ be a next-formula, σ_A a strategy and $q \in Q$. Then there exists a $\neg\langle\langle A\rangle\rangle\bigcirc\psi$-computation for σ_A rooted at q.*

This claim is equivalent to Claim 3.12 in the proof of Lemma 3.11.

Using the construction of $\mathcal{S}$ and Property 2 of witness trees, it is straightforward to prove the following claim, which, intuitively, states that $\mathcal{S}$ is $\bigcirc$-matching:

CLAIM 3.26. *For all $q \in Q$ and $\vec{t} \in [k/n]$, $S_{t(q)}(\vec{t}) \subseteq t(s_{\vec{t}}(q))$.*

This claim is as Claim 3.13 in the proof of Lemma 3.11.

The next claim establishes the property of $\mathcal{S}$ that is crucial for showing that $\mathcal{S}$ is a model of φ.

CLAIM 3.27. *For any state $q \in Q$ and any formula $\psi \in \mathsf{ecl}(\varphi)$, $\psi \in t(q)$ iff $\mathcal{S}, q \models \psi$.*

PROOF OF CLAIM Let q and ψ be as in the claim. The proof is by induction on the structure of ψ. The base case and the Boolean cases are straightforward. The cases for $\psi = \langle\langle A\rangle\rangle\bigcirc\psi'$, $\psi = \langle\langle A\rangle\rangle\Box\psi'$ and $\psi = \langle\langle A\rangle\rangle\psi'\,\mathcal{U}\,\vartheta$ were already proved in a similar claim for ATL (Claim 3.14) in the proof of Lemma 3.11. Therefore, we only concentrate on the epistemic operators K_a, C_A and D_A:

- $\psi = K_a\psi'$. "$\Rightarrow$": Suppose $K_a\psi' \in t(q)$. Let q' be a state such that $q \sim_a q'$. By definition of $\sim_a$, it holds that for all $K_a\vartheta \in \mathsf{ecl}(\varphi)$, $K_a\vartheta \in t(q)$ iff $K_a\vartheta \in t(q')$. Thus $K_a\psi' \in t(q')$. Then (T5) yields $\psi' \in t(q')$ and the induction hypothesis $\mathcal{S}, q' \models \psi'$. Hence, $\mathcal{S}, q \models K_a\psi'$ by the semantics.

  "$\Leftarrow$": Suppose $K_a\psi' \notin t(q)$. Then $\neg K_a\psi' \in t(q)$ by (T2). Since the type $t(q)$ of Γ is ATEL-realizable in Γ, from Condition 4 in Definition 3.17 of ATEL realizability and the construction of $\mathcal{S}$, it follows that there is a state $q' \in Q$ such that (a) $\neg\psi' \in t(q')$ and (b) $q \sim_a q'$. Then (T2) yields $\psi' \notin t(q')$ and the induction hypothesis $\mathcal{S}, q' \not\models \psi'$. Hence, $\mathcal{S}, q \not\models K_a\psi'$ by the semantics.

- $\psi = E_A\psi'$. "$\Rightarrow$": Suppose $E_A\psi' \in t(q)$ where $A = \{a_1, \dots, a_\ell\}$. Then $\neg(\neg K_{a_1}\psi' \vee \cdots \vee \neg K_{a_\ell}\psi') \in t(q)$ by (T6). The conditions (T1) and (T2) yield $K_a\psi' \in t(q)$ for all $a \in A$. Let $a \in A$ and $q' \in Q$ such that $q \sim_a q'$. By definition of $\sim_a$, it holds that for all $K_a\vartheta \in \mathsf{ecl}(\varphi)$, $K_a\vartheta \in t(q)$ iff $K_a\vartheta \in t(q')$. Thus $K_a\psi' \in t(q')$. Then (T5) yields $\psi' \in t(q')$ and the induction hypothesis $\mathcal{S}, q' \models \psi'$. From the fact $\sim_A^E = \bigcup_{a\in A} \sim_a$, it follows that $\mathcal{S}, q' \models \psi'$ for all states $q' \in Q$ with $q \sim_A^E q'$. Hence, $\mathcal{S}, q \models E_A\psi'$ by the semantics.

  "$\Leftarrow$": Suppose $E_A\psi' \notin t(q)$ where $A = \{a_1, \dots, a_\ell\}$. Then (T6) and (T2) yield $\neg K_{a_1}\psi' \vee \cdots \vee \neg K_{a_\ell}\psi' \in t(q)$. By (T1), $\neg K_a\psi' \in t(q)$ for some $a \in A$. Since the type $t(q)$ of Γ is ATEL-realizable in Γ, from Condition 4 in Definition 3.17 of ATEL realizability and the construction of $\mathcal{S}$, it follows that there is a state $q' \in Q$ such that (a) $\neg\psi' \in t(q')$ and (b) $q \sim_a q'$. Then (T2) yields $\psi' \notin t(q')$ and the induction hypothesis $\mathcal{S}, q' \not\models \psi'$. Note that $q \sim_A^E q'$ since $\sim_A^E = \bigcup_{a\in A} \sim_a$. Hence, $\mathcal{S}, q \not\models E_A\psi'$ by the semantics.
- $\psi = C_A\psi'$. "$\Rightarrow$": Suppose $C_A\psi' \in t(q)$ where $A = \{a_1, \dots, a_\ell\}$. Let q' be a state such that $q \sim_A^C q'$. Then by definition of $\sim_A^C$, there is a sequence $q_0 \cdots q_j \in Q^*$, $j \geq 0$, of states and a sequence $b_0 \cdots b_{j-1} \in A^*$ of agents such that $q = q_0 \sim_{b_0} q_1 \sim_{b_1} \cdots \sim_{b_{j-1}} q_j = q'$. It is shown by a subinduction that $C_A\psi' \in t(q_i)$ for all $i \leq j$. The induction base is clear since $C_A\psi' \in t(q) = t(q_0)$. For the induction step, suppose $C_A\psi' \in t(q_i)$. Then (T7) yields $E_AC_A\psi' \in t(q_i)$ and (T6) $\neg(\neg K_{a_1}C_A\psi' \vee \cdots \vee \neg K_{a_\ell}C_A\psi') \in t(q_i)$. Thus $K_aC_A\psi' \in t(q_i)$ for all $a \in A$ by (T1) and (T2). By definition of $\sim_{b_i}$, it holds that $K_{b_i}C_A\psi' \in t(q_{i+1})$. Then $C_A\psi' \in t(q_{i+1})$ by (T5). Hence, the above subinduction yields $C_A\psi' \in t(q_j) = t(q')$. Then $\psi' \in t(q')$ by (T7) and $\mathcal{S}, q' \models \psi'$ by the induction hypothesis. Hence, $\mathcal{S}, q \models C_A\psi'$ by the semantics.

  "$\Leftarrow$": Suppose $C_A\psi' \notin t(q)$ where $A = \{a_1, \dots, a_\ell\}$. Then $\neg C_A\psi' \in t(q)$ by (T2). Since the type $t(q)$ of Γ is ATEL-realizable in Γ, from Condition 5 in Definition 3.17 of ATEL realizability and the construction of $\mathcal{S}$, it follows that there is a sequence $q_0 \cdots q_j \in Q^*$ of states such that the sequence $t(q_0) \cdots t(q_j)$ of types is a $\neg C_A\psi'$-witness path rooted at $t(q)$ in Γ. Since $t(q) = t(q_0)$ by Point 1 in Definition 3.16 of witness paths, set $q_0 = q$. For all $i < j$, Point 2 together with the definition of $\sim_{a_i}$ yields that $q_i \sim_{a_i} q_{i+1}$ for some $a_i \in A$. Thus $q \sim_A^C q_j$. Point 3 yields $\neg\psi' \in t(q_j)$. Then $\psi' \notin t(q_j)$ by (T2) and $\mathcal{S}, q_j \not\models \psi'$ by the induction hypothesis. Hence, $\mathcal{S}, q \not\models C_A\psi'$ by the semantics.
- $\psi = D_A\psi'$. "$\Rightarrow$": Suppose $D_A\psi' \in t(q)$. Let q' be a state such that $q \sim_A^D q'$. By definition of $\sim_A^D$, it holds that for all $D_A\vartheta \in \mathsf{ecl}(\varphi)$, $D_A\vartheta \in t(q)$ iff $D_A\vartheta \in t(q')$. Thus $D_A\psi' \in t(q')$. Then (T8) yields $\psi' \in t(q')$ and the induction hypothesis $\mathcal{S}, q' \models \psi'$. Hence, $\mathcal{S}, q \models D_A\psi'$ by the semantics.

  "$\Leftarrow$": Suppose $D_A\psi' \notin t(q)$. Then $\neg D_A\psi' \in t(q)$ by (T2). Since the type $t(q)$ of Γ is ATEL-realizable in Γ, from Condition 6 in Definition 3.17 of ATEL realizability and the construction of $\mathcal{S}$, it follows that there is a state $q' \in Q$ such that (a) $\neg\psi' \in t(q')$ and (b) $q \sim_A^D q'$. Then (T2) yields $\psi' \notin t(q')$ and the induction hypothesis $\mathcal{S}, q' \not\models \psi'$. Hence, $\mathcal{S}, q \not\models D_A\psi'$ by the semantics.

◀

Notice that, in general, the epistemic accessibility relations in $\mathcal{S}$ together with all explicitly defined epistemic accessibility relations $\sim_A^D$ for distributed knowledge do not satisfy property (3.2). More precisely, it holds that, for every $D_A\psi \in \mathsf{ecl}(\varphi)$, $\sim_A^D \subseteq \sim_a$ for all agents $a \in A$ by (T9), but $\bigcap_{a\in A} \sim_a \subseteq \sim_A^D$ does not necessarily hold.

In order to show that φ is satisfiable, it remains to show that there is a proper AETS satisfying property (3.2) in which also φ is satisfied at some state.

First, consider some auxiliary notions. Enumerate the $\mathsf{ecl}(\varphi)$-formulas ψ of the form $\neg K_a\vartheta$ or $\neg D_A\vartheta$ and denote the number of ψ with $\sharp_\psi$ (the numbering starts with 1). Let k^e be the total number of $\mathsf{ecl}(\varphi)$-formulas of this form. Let $\prec$ be some linear order on Q^* such that

(i) $\alpha \prec \alpha'$ if $|\alpha| < |\alpha'|$, and
(ii) $\alpha\alpha' \prec \alpha\alpha''$ if $\alpha' \prec \alpha''$

where $|\alpha|$ denotes the length of sequence $\alpha \in Q^*$.

A sequence $q_0 \cdots q_n \in Q^*$, $n \geq 0$, of states *accomplishes* the ATEL-formula $\neg C_A\psi$ at a state $q \in Q$ if there is a sequence $a_0 \cdots a_{n-1} \in A^*$ of agents such that $q = q_0 \sim_{a_0} q_1 \sim_{a_1} \cdots \sim_{a_{n-1}} q_n$ and $\mathcal{S}, q_n \models \neg\psi$. An ATEL-formula $\neg K_a\psi$ is *relevant* for satisfying the ATEL-formula $\neg C_A\psi'$ at a state $q \in Q$ if

- $\neg K_a\psi, \neg C_A\psi' \in t(q)$ and $\neg\psi' \notin t(q)$,
- $\psi = C_A\psi'$, and
- $q_0 \sim_a q_1$ where $q_0 q_1 \cdots q_n \in Q^*, n \geq 1$, is, among the sequences that accomplish $\neg C_A\psi'$ at q, minimal wrt. $\prec$.

Verify that $\neg C_A\psi' \in t(q)$ iff $\mathcal{S}, q \models \neg C_A\psi'$ iff there is a sequence that accomplishes $\neg C_A\psi'$ at q, where the first equivalence follows from Claim 3.27.

For each state $q \in Q$, define a partial mapping $\tau_q : \{1, \ldots, k^e\}^* \times \{q\} \rightarrow 2^\Sigma \times Q$ where $\{\alpha \mid (\alpha, q) \in \mathsf{dom}(\tau_q)\}$ is a finite prefix-closed subset of $\{1, \ldots, k^e\}^*$. Intuitively, τ_q is an epistemic witness tree with q as its root. Each successor node in τ_q corresponds to a formula of the form $\neg K_a\psi$ or $\neg D_A\psi$ and is mapped to a witnessing state of the previous ATES $\mathcal{S}$. In particular, for each formula $\neg C_A\psi$ at q, there is a path in τ_q that corresponds to a witness path for $\neg C_A\psi$ in Q. Formally, τ_q is defined inductively as follows. For each $(\alpha, q) \in \{1, \ldots, k^e\}^* \times \{q\}$, refer to the first and second component of the tuples (α, q) and $\tau_q((\alpha, q))$ with $(\alpha, q)^1$, $(\alpha, q)^2$, $\tau_q((\alpha, q))^1$ and $\tau_q((\alpha, q))^2$, respectively. For the induction base, set

$$\tau_q((\varepsilon, q)) := (\Sigma, q).$$

For the induction step, let $\alpha \in \{1, \ldots, k^e\}^*$ be such that $\tau_q((\alpha, q))$ is already defined. Let $q' = \tau_q((\alpha, q))^2$ be a state in Q. For each $i \in \{1, \ldots, k^e\}$ with $i = \sharp_\psi$, if $\psi \in t(q')$, then depending on ψ distinguish the following three cases:

(1) $\psi = \neg K_a \psi'$ and $\neg K_a \psi'$ is relevant for satisfying some $\neg C_A \vartheta$ at q'. Among the sequences that accomplish $\neg C_A \vartheta$ at q', let $q_0 q_1 \cdots q_n \in Q^*$, $n \geq 0$, be the minimal one wrt. $\prec$. Set $\tau_q((\alpha \cdot i, q)) := (\{a\}, q_1)$.

(2) $\psi = \neg K_a \psi'$ and $\neg K_a \psi'$ is not relevant for satisfying any $\neg C_A \vartheta$ at q'. Arbitrarily choose a state $q'' \in Q$ such that the type $t(q'')$ fulfills Condition 4 of Definition 3.17 of ATEL realizability. Set $\tau_q((\alpha \cdot i, q)) := (\{a\}, q'')$.

(3) $\psi = \neg D_A \psi'$. Arbitrarily choose a state $q'' \in Q$ such that the type $t(q'')$ fulfills Condition 6 of Definition 3.17 of ATEL realizability. Set $\tau_q((\alpha \cdot i, q)) := (A, q'')$.

For an illustration of an epistemic witness tree τ_q at a state q in Q, see Figure 3.4. Note that Q is the state set of $\mathcal{S}$ and the dotted line and the dashed line respectively correspond to the (possibly different) epistemic accessibility relations $\sim_a$ and $\sim_b$ (omitting reflexive edges).

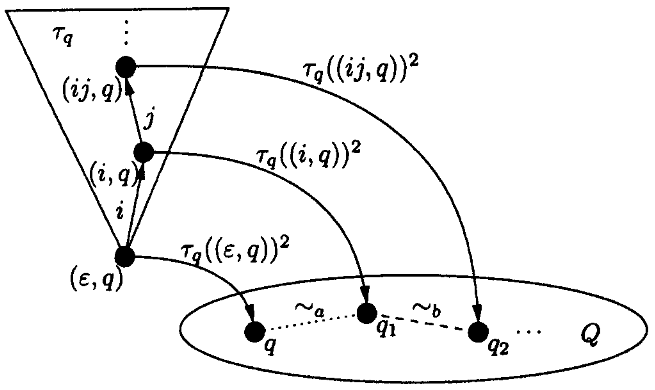


FIGURE 3.4. An epistemic witness tree τ_q at state q.

In order to see that τ_q is well-defined, consider the following. For Case (1), note that, since $\neg K_a \psi'$ is relevant for satisfying some $\neg C_A \vartheta$ at q', there is such a sequence accomplishing $\neg C_A \vartheta$ at q'. In both remaining cases such a state q'' exists since respectively $\neg K_a \vartheta' \in t(q)$ implies $\mathcal{S}, q \models \neg K_a \vartheta'$ and $\neg D_A \vartheta' \in t(q)$ implies $\mathcal{S}, q \models \neg D_A \vartheta'$ by Claim 3.27.

Notice that a tree τ_q, as it is defined above, is not finite in general. However, it can be made finite as follows: For all nodes α and $\alpha \cdot \alpha'$ on the same path in τ_q, if τ_q maps α and $\alpha \cdot \alpha'$ to the same state q' of $\mathcal{S}$, then identify the successors of $\alpha \cdot \alpha'$ with the successors of α. In order not to introduce more notation, assume that, in each τ_q, the relevant successors are reconnected such that τ_q is finite.

For notational convenience, consider the following abbreviations: For all states $q \in Q$, denote the set $\mathsf{dom}(\tau_q) \setminus \{(\varepsilon, q)\}$ of pairs with $\mathsf{dom}(\tau_q)^{-1}$, identify q with (ε, q) and let $\tau_q(q) = (\Sigma, q)$. Moreover, $q'^1 \cdot i$ denotes $(q'^1 \cdot i, q)$ for each $q' \in \mathrm{dom}(\tau_q)$ and each $i \in \{1, \dots, k^e\}$.

In the following, the ATES $\mathcal{S}$ is extended with the nodes of the above defined trees τ_q, one tree for each state q of $\mathcal{S}$. Moreover, in the resulting ATES $\mathcal{S}'$ the formula defined epistemic accessibility relations $\sim_a$ of $\mathcal{S}$ are redefined to include the additional states corresponding to the nodes of the trees τ_q. Formally, define $\mathcal{S}' = \langle \Pi, \Sigma, Q', \sim'_1, \dots, \sim'_n, \pi', \delta' \rangle$ where

- $Q' = Q \cup \{\mathrm{dom}(\tau_q)^{-1} \mid q \in Q\}$;
- for each $a \in \Sigma$, $\sim'_a \subseteq Q' \times Q'$ is smallest equivalence relation such that for any $q, q' \in Q'$, $q \sim'_a q'$ iff there is a $q_\varepsilon \in Q$, $\alpha \in \{1, \dots, k^e\}^*$ and $i \in \{1, \dots, k^e\}$ with either $q = (\alpha, q_\varepsilon)$ and $q' = (\alpha \cdot i, q_\varepsilon)$, or $q = q_\varepsilon$ and $q' = (i, q_\varepsilon)$ such that $\tau_{q_\varepsilon}(q')$ is defined and $a \in \tau_{q_\varepsilon}(q')^1$;
- for each $q \in Q'$,
$$\pi'(q) = \begin{cases} \pi(q) & \text{if } q \in Q, \text{ or} \\ \{p \in \Pi \mid p \in t(\tau_{q'}(q)^2)\} & \text{if } q \in \mathrm{dom}(\tau_{q'})^{-1} \text{ for some } q' \in Q; \end{cases}$$
- for each $q \in Q'$ and $a \in \Sigma$,
$$\delta'(q, a) = \begin{cases} \delta(q, a) & \text{if } q \in Q, \text{ or} \\ \delta(\tau_{q'}(q)^2, a) & \text{if } q \in \mathrm{dom}(\tau_{q'})^{-1} \text{ for some } q' \in Q. \end{cases}$$

For an illustration of the transition relation δ' in the ATES $\mathcal{S}'$, see Figure 3.5: the choices in $\delta'(q, a)$ for an agent a at a node q of an epistemic witness tree $\tau_{q'}$, where $q = (\alpha, q')$, are the choices in $\delta(q'', a)$ for a at the state $q'' \in Q$ with $q'' = \tau_{q'}((\alpha, q'))^2$. The choices are depicted by the boxes with round corners. Note that the set Q' of states in $\mathcal{S}'$ contains the nodes of all witness trees (except their roots) and the states in Q.

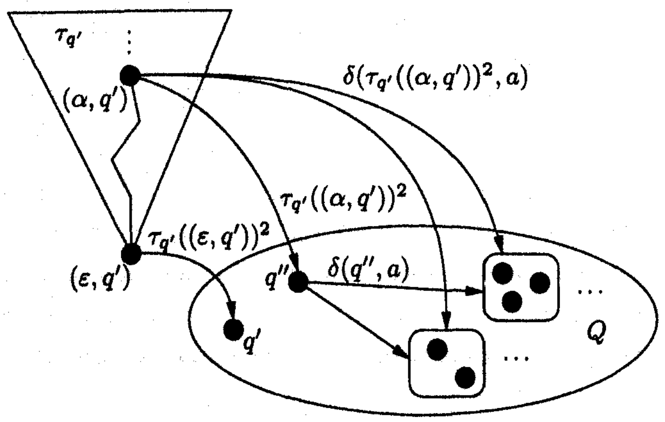

$\tau_{q'}$
(α, q')
$\delta(\tau_{q'}((\alpha, q'))^2, a)$
$\tau_{q'}((\alpha, q'))^2$
$\tau_{q'}((\varepsilon, q'))^2$
(ε, q')
q''
$\delta(q'', a)$
q'
Q

FIGURE 3.5. The transition relation δ' in the ATES $\mathcal{S}'$.

From now on, we interpret the epistemic operator D_A by the epistemic accessibility relation $\sim'^D_A$, which is implicitly defined in terms of $\sim_a$, for all agents $a \in A$, as $\sim'^D_A = \bigcap_{a \in A} \sim'_a$.

Notation: For each state $q \in Q' \backslash Q$, there is a state $q_\varepsilon \in Q$ such that $q \in \mathsf{dom}(\tau_{q_\varepsilon})^{-1}$ by definition of $\mathcal{S}'$. Denote with $t(q)$ the type $t(\tau_{q_\varepsilon}(q)^2)$.

It remains to show that $\mathcal{S}'$ is a model for φ. To this end, consider the following two claims.

CLAIM 3.28. *For all states $q, q' \in Q'$ and agents $a \in \Sigma$, $q \sim'_a q'$ implies $K_a\psi \in t(q)$ iff $K_a\psi \in t(q')$ for all $K_a\psi \in ecl(\varphi)$.*

PROOF OF CLAIM Let q, q' and a be as in the claim. Suppose $q \sim'_a q'$. By definition of $\mathcal{S}'$, $q \in \mathsf{dom}(\tau_{q_\varepsilon})$ for some $q_\varepsilon \in Q$. Then $q' \in \mathsf{dom}(\tau_{q_\varepsilon})$ by definition of $\sim'_a$. The transitivity of $\sim'_a$ together with the definition of $\sim'_a$ and the tree-like structure of τ_{q_ε} yield that there is a sequence $q_0 \cdots q_\ell$ in $\mathsf{dom}(\tau_{q_\varepsilon})^*$ with $q = q_0 \sim'_a q_1 \sim'_a \cdots \sim'_a q_\ell = q'$ such that, for all $j < \ell$, either $q^1_{j+1} = q^1_j \cdot i$ or $q^1_j = q^1_{j+1} \cdot i$ for some $i \in \{1, \ldots, k^e\}$. Let $q_0 \cdots q_\ell$ be the shortest such sequence. In the following, it is shown that $\tau_{q_\varepsilon}(q_j)^2 \sim_a \tau_{q_\varepsilon}(q_{j+1})^2$ for all $j < \ell$. Then $\tau_{q_\varepsilon}(q)^2 \sim_a \tau_{q_\varepsilon}(q')^2$ by transitivity of $\sim_a$. From the definition of $\sim_a$, it follows that $K_a\psi \in t(\tau_{q_\varepsilon}(q)^2) = t(q)$ iff $K_a\psi \in t(\tau_{q_\varepsilon}(q')^2) = t(q')$ for all $K_a\psi \in \mathsf{ecl}(\varphi)$ which shows the claim.

Now it is shown that $\tau_{q_\varepsilon}(q_j)^2 \sim_a \tau_{q_\varepsilon}(q_{j+1})^2$ for all $j < \ell$. Let $j < \ell$. Consider only the former case where $q^1_{j+1} = q^1_j \cdot i$ for some $i \in \{1, \ldots, k^e\}$; the latter one is similar. By definition of $\sim'_a$, it holds that $a \in \tau_{q_\varepsilon}(q_{j+1})^1$. Let $\vartheta \in t(q_j)$ be such that $\sharp_\vartheta = i$. According to the induction step in the definition of τ_{q_ε}, distinguish three cases: In the cases (1) and (2) where $\vartheta = \neg K_a\vartheta' \in t(q_j)$, it follows that $\tau_{q_\varepsilon}(q_j)^2 \sim_a \tau_{q_\varepsilon}(q_{j+1})^2$. In Case (3) where $\vartheta = \neg D_A\vartheta' \in t(q_j)$, $\tau_{q_\varepsilon}(q_{j+1})$ is defined such that $\tau_{q_\varepsilon}(q_{j+1})^1 = A$ and $\tau_{q_\varepsilon}(q_j)^2 \sim^D_A \tau_{q_\varepsilon}(q_{j+1})^2$. Observe that (T9) together with the definition of $\sim^D_A$ implies $\sim^D_A \subseteq \sim_b$ for all $b \in A$. Then it follows from $a \in \tau_{q_\varepsilon}(q_{j+1})^1 = A$ that $\sim^D_A \subseteq \sim_a$. Thus $\tau_{q_\varepsilon}(q_j)^2 \sim_a \tau_{q_\varepsilon}(q_{j+1})^2$. ◀

CLAIM 3.29. *For any state $q \in Q'$ and any formula $\psi \in ecl(\varphi)$, $\psi \in t(q)$ iff $\mathcal{S}', q \models \psi$.*

PROOF OF CLAIM Let q and ψ be as in the claim. The proof is by induction on the structure of ψ.

- $\psi = p$ for $p \in \Pi$ or $\psi = \neg\psi'$ or $\psi = \psi_1 \vee \psi_2$. These cases are similar to those ones in the proof of Claim 3.27.
- $\psi = \langle\langle A\rangle\rangle\bigcirc\psi'$, $\psi = \langle\langle A\rangle\rangle\Box\psi'$, or $\psi = \langle\langle A\rangle\rangle\psi\,\mathcal{U}\,\vartheta$.

  Then: $\psi \in t(q)$ iff $\psi \in t(\tau_{q_\varepsilon}(q)^2)$ for some $q_\varepsilon \in Q$ with $q \in \mathsf{dom}(\tau_{q_\varepsilon})$
  iff $\mathcal{S}, \tau_{q_\varepsilon}(q)^2 \models \psi$ by Claim 3.27
  iff $\mathcal{S}', q \models \psi$ since for all $a \in \Sigma$, $\delta'(q, a) = \delta(\tau_{q_\varepsilon}(q)^2, a)$.

- $\psi = K_a\psi'$. "$\Rightarrow$": Suppose $K_a\psi' \in t(q)$. Let $q' \in Q'$ be a state such that $q \sim'_a q'$. Then $K_a\psi' \in t(q')$ by Claim 3.28 and $\psi' \in t(q')$ by (T5). The induction hypothesis yields $\mathcal{S}', q' \models \psi'$. Hence, $\mathcal{S}', q \models K_a\psi'$ by the semantics.

  "$\Leftarrow$": Suppose $K_a\psi' \notin t(q)$. Then $\neg K_a\psi' \in t(q)$ by (T2). By definition of $\mathcal{S}'$, $q \in \mathsf{dom}(\tau_{q_\epsilon})$ for some $q_\epsilon \in Q$. It follows from the induction step in the definition of τ_{q_ϵ} that $\tau_{q_\epsilon}(q')$ is defined for $q' = q^1 \cdot \sharp_\vartheta$ where $\vartheta = \neg K_a\psi'$. Moreover, $a \in \tau_{q_\epsilon}(q')^1$ and $\neg\psi' \in t(\tau_{q_\epsilon}(q')^2) = t(q')$. Then $q \sim'_a q'$ by definition of $\sim'_a$. The induction hypothesis yields $\mathcal{S}', q' \models \neg\psi'$. Hence, $\mathcal{S}', q \not\models K_a\psi'$ by the semantics.
- $\psi = E_A\psi'$. "$\Rightarrow$": Suppose $E_A\psi' \in t(q)$ where $A = \{a_1, \ldots, a_\ell\}$. Then $\neg(\neg K_{a_1}\psi' \vee \cdots \vee \neg K_{a_\ell}\psi') \in t(q)$ by (T6). The conditions (T1) and (T2) yield $K_a\psi' \in t(q)$ for all $a \in A$. Let $a \in A$ and $q' \in Q'$ such that $q \sim'_a q'$. Then $K_a\psi' \in t(q')$ by Claim 3.28 and $\psi' \in t(q')$ by (T5). The induction hypothesis yields $\mathcal{S}, q' \models \psi'$. From the fact $\sim^{E'}_A = \bigcup_{a\in A} \sim'_a$, it follows that $\mathcal{S}, q' \models \psi'$ for all states $q' \in Q'$ with $q \sim^{E'}_A q'$. Hence, $\mathcal{S}, q \models E_A\psi'$ by the semantics.

  "$\Leftarrow$": Suppose $E_A\psi' \notin t(q)$ where $A = \{a_1, \ldots, a_\ell\}$. By definition of $\mathcal{S}'$, $q \in \mathsf{dom}(\tau_{q_\epsilon})$ for some $q_\epsilon \in Q$. Then (T6) and (T2) yield $\neg K_{a_1}\psi' \vee \cdots \vee \neg K_{a_\ell}\psi' \in t(q)$. By (T1), $\neg K_a\psi' \in t(q)$ for some $a \in A$. It follows from the induction step in the definition of τ_{q_ϵ} that $\tau_{q_\epsilon}(q')$ is defined for $q' = q^1 \cdot \sharp_\vartheta$ where $\vartheta = \neg K_a\psi'$. Moreover, $a \in \tau_{q_\epsilon}(q')^1$ and $\neg\psi' \in t(\tau_{q_\epsilon}(q')^2) = t(q')$. Then $q \sim'_a q'$ by definition of $\sim'_a$, and $q \sim^{E'}_A q'$ since $\sim'_a \subseteq \sim^{E'}_A$ by definition of $\sim^{E'}_A$. The induction hypothesis yields $\mathcal{S}', q' \models \neg\psi'$. Hence, $\mathcal{S}', q \not\models E_A\psi'$ by the semantics.
- $\psi = C_A\psi'$. "$\Rightarrow$": Suppose $C_A\psi' \in t(q)$ where $A = \{a_1, \ldots, a_\ell\}$. Let q' be a state such that $q \sim^{C'}_A q'$. Then by definition of $\sim^{C'}_A$, there is a sequence $q_0 \cdots q_j \in Q'^*$, $j \geq 0$, of states and a sequence $b_0 \cdots b_{j-1} \in A^*$ of agents such that $q = q_0 \sim'_{b_0} q_1 \sim'_{b_1} \cdots \sim'_{b_{j-1}} q_j = q'$. It is shown by a subinduction that $C_A\psi' \in t(q_i)$ for all $i \leq j$. The induction base is clear since $C_A\psi' \in t(q) = t(q_0)$. For the induction step, suppose $C_A\psi' \in t(q_i)$. Then (T7) yields $E_AC_A\psi' \in t(q_i)$ and (T6) $\neg(\neg K_{a_1}C_A\psi' \vee \cdots \vee \neg K_{a_\ell}C_A\psi') \in t(q_i)$. Thus $K_aC_A\psi' \in t(q_i)$ for all $a \in A$ by (T1) and (T2). Since $q_i \sim'_{b_i} q_{i+1}$, it follows by Claim 3.28 that $K_{b_i}C_A\psi' \in t(q_{i+1})$. Then $C_A\psi' \in t(q_{i+1})$ by (T5). Hence, the above subinduction yields $C_A\psi' \in t(q_j) = t(q')$. Then $\psi' \in t(q')$ by (T7) and $\mathcal{S}, q' \models \psi'$ by the induction hypothesis. Hence, $\mathcal{S}, q \models C_A\psi'$ by the semantics.

  "$\Leftarrow$": Suppose $C_A\psi' \notin t(q)$ where $A = \{a_1, \ldots, a_\ell\}$. Then $\neg C_A\psi' \in t(q)$ by (T2). By definition of $\mathcal{S}'$, $q \in \mathsf{dom}(\tau_{q_\epsilon})$ for some $q_\epsilon \in Q$. Since $t(q) = t(\tau_{q_\epsilon}(q)^2)$, $\neg C_A\psi' \in t(\tau_{q_\epsilon}(q)^2)$. Among the sequences that accomplish $\neg C_A\psi$ at $\tau_{q_\epsilon}(q)^2$, let $q_0 \cdots q_n \in Q^*$, $n \geq 0$, be the minimal one wrt. $\prec$. Note that such a sequence exists since $\mathcal{S}, \tau_{q_\epsilon}(q)^2 \models \neg C_A\psi'$ by Claim 3.27. Thus there is a sequence $a_0 \cdots a_{n-1} \in A^*$ of agents such that $\tau_{q_\epsilon}(q)^2 = q_0 \sim_{a_0} q_1 \sim_{a_1} \cdots \sim_{a_{n-1}} q_n$.

By Condition (i) in the definition of $\prec$, there is no shorter such accomplishing sequence, i.e., for all $j < n$, $S, q_j \not\models \neg\psi'$ and thus $\neg\psi' \notin t(q_j)$ by Claim 3.27. Inductively define a sequence $q'_0 \cdots q'_n \in \mathsf{dom}(\tau_{q_\varepsilon})^*$: set $q'_0 := q$ and for all $i < n$, set $q'_{i+1} := q'^1_i \cdot \sharp_\vartheta$ with $\vartheta = \neg K_{a_n} C_A \psi'$.

In the following, it is shown that τ_{q_ε} maps each state of the sequence $q'_0 \cdots q'_n$ in $\mathsf{dom}(\tau_{q_\varepsilon})^*$ to the corresponding states of the sequence $q_0 \cdots q_n$ in Q^*. Formally, it is shown by a subinduction that $\tau_{q_\varepsilon}(q'_j) = (\{a_{j-1}\}, q_j)$ and $q'_{j-1} \sim_{a_{j-1}} q'_j$ for all $j \le n$. In the induction base, it holds that $\tau_{q_\varepsilon}(q'_0) = (\{a\}, q_0)$ for some $a \in A$. For the induction step, suppose $\tau_{q_\varepsilon}(q'_j) = (\{a_{j-1}\}, q_j)$. It follows by $q_0 \cdots q_n$ being a sequence that accomplishes $\neg C_A \psi$ that $C_A \psi' \in t(q_j)$. Then (T7) yields $\neg E_A C_A \psi' \in t(q_j)$ and (T6) $\neg(\neg K_{a_1} C_A \psi' \vee \cdots \vee \neg K_{a_\ell} C_A \psi') \in t(q_j)$. Thus $K_a C_A \psi' \in t(q_j)$ for all $a \in A$ by (T1) and (T2). In particular, it holds for agent a_j that $\neg K_{a_j} C_A \psi' \in t(q_j)$. Together with the facts $q_j \sim_{a_j} q_{j+1}$ and $\neg\psi' \notin t(q_j)$, it follows that $\neg K_{a_j} C_A \psi'$ is relevant for satisfying $\neg C_A \psi'$ at q_j. By Condition (ii) in the definition of $\prec$, $q_j q_{j+1} \cdots q_n$ is, among the sequences that accomplish $\neg C_A \psi$ at q_j, the minimal one wrt. $\prec$. Then, by Case (1) in the induction step of the definition of τ_{q_ε}, $\tau_{q_\varepsilon}(q'^1_j \cdot i) = (\{a_j\}, q_{j+1})$ where $i = \sharp_\vartheta$ and $\vartheta = \neg K_{a_j} C_A \psi'$. Since $q'_{j+1} = q'^1_j \cdot i$ by definition of $q'_0 \ldots q'_n$, it holds that $\tau_{q_\varepsilon}(q'_{j+1}) = (\{a_j\}, q_{j+1})$. By definition of $\sim'_{a_j}$, it holds that $q'_j \sim'_{a_j} q'_{j+1}$ which finishes the induction step. Since $\sim'^C_A = (\bigcup_{a \in A} \sim'_a)^*$, it holds that $q = q'_0 \sim'^C_A q'_n$. From $S, q_n \models \neg\psi'$, it follows by Claim 3.27 that $\neg\psi' \in t(q_n)$. Then $\neg\psi' \in t(q'_n)$ since $t(q'_n) = t(q_n)$. The induction hypothesis yields $S', q'_n \models \neg\psi'$. Hence, $S', q \not\models C_A \psi'$ by the semantics.

- $\psi = D_A \psi'$. "$\Rightarrow$": Suppose $D_A \psi' \in t(q)$. By definition of S', $q \in \mathsf{dom}(\tau_{q_\varepsilon})$ for some $q_\varepsilon \in Q$. Let $q' \in Q'$ be a state such that $q \sim'^D_A q'$. Since $q \sim'_a q'$ for each $a \in A$ by definition of $\sim'^D_A$, it follows that $q' \in \mathsf{dom}(\tau_{q_\varepsilon})$ by definition of $\sim'_a$. The transitivity of $\sim'^D_A$ together with the definition of $\sim'_a$, $a \in A$, and the tree-like structure of τ_{q_ε} yield that there is a sequence $q_0 \cdots q_\ell$ in $\mathsf{dom}(\tau_{q_\varepsilon})^*$ with $q = q_0 \sim'^D_A q_1 \sim'^D_A \ldots \sim'^D_A q_\ell = q'$ such that, for all $j < \ell$, either $q^1_{j+1} = q^1_j \cdot i$ or $q^1_j = q^1_{j+1} \cdot i$ for some $i \in \{1, \ldots, k^e\}$. Let $q_0 \cdots q_\ell$ be the shortest such sequence. In the following, it is shown that $\tau_{q_\varepsilon}(q_j)^2 \sim^D_A \tau_{q_\varepsilon}(q_{j+1})^2$ for all $j < \ell$. Then $\tau_{q_\varepsilon}(q)^2 \sim^D_A \tau_{q_\varepsilon}(q')^2$ by transitivity of $\sim^D_A$. From $D_A \psi' \in t(q)$, it follows by definition of $\sim^D_A$ that $D_A \psi' \in t(q')$. Then (T8) yields $\psi' \in t(q')$ and the induction hypothesis $S', q' \models \psi'$. Hence, $S', q \models D_A \psi'$ by the semantics.

Now it is shown that $\tau_{q_\varepsilon}(q_j)^2 \sim^D_A \tau_{q_\varepsilon}(q_{j+1})^2$ for all $j < \ell$. Let $j < \ell$. Consider only the former case where $q^1_{j+1} = q^1_j \cdot i$ for some $i \in \{1, \ldots, k^e\}$; the latter one is similar. By definition of $\sim'^D_A$, it holds that $q_j \sim'_a q_{j+1}$ for each $a \in A$. Then $a \in \tau_{q_\varepsilon}(q_{j+1})^1$ by definition of $\sim'_a$. Thus $A \subseteq \tau_{q_\varepsilon}(q_{j+1})^1$. Let $\vartheta \in t(q_j)$ be such that $\sharp_\vartheta = i$. According to the induction step in the definition of τ_{q_ε}, distinguish three cases: In cases (1) and (2) where $\vartheta = \neg K_a \vartheta' \in t(q_j)$, it easily follows that $\tau_{q_\varepsilon}(q_j)^2 \sim_a \tau_{q_\varepsilon}(q_{j+1})^2$. Note that $A = \{a\}$ since $\tau_{q_\varepsilon}(q_{j+1})^1 = \{a\}$.

Thus $\tau_{q_\varepsilon}(q_j)^2 \sim_A^D \tau_{q_\varepsilon}(q_{j+1})^2$. In Case (3) where $\vartheta = \neg D_B \vartheta' \in t(q_j)$, $\tau_{q_\varepsilon}(q_{j+1})$ is defined such that $\tau_{q_\varepsilon}(q_{j+1})^1 = B$ and $\tau_{q_\varepsilon}(q_j)^2 \sim_B^D \tau_{q_\varepsilon}(q_{j+1})^2$. Observe that (T10) together with the definition of $\sim_B^D$ implies $\sim_B^D \subseteq \sim_{B'}^D$ for all $B' \subseteq B$. Then it follows from $A \subseteq \tau_{q_\varepsilon}(q_{j+1})^1 = B$ that $\sim_A^D \subseteq \sim_B^D$. Thus $\tau_{q_\varepsilon}(q_j)^2 \sim_A^D \tau_{q_\varepsilon}(q_{j+1})^2$.

"$\Leftarrow$": Suppose $D_A\psi' \notin t(q)$. Then $\neg D_A \psi' \in t(q)$ by (T2). By definition of S', $q \in \mathsf{dom}(\tau_{q_\varepsilon})$ for some $q_\varepsilon \in Q$. It follows from the induction step in the definition of τ_{q_ε} that $\tau_{q_\varepsilon}(q')$ is defined where $q' = q^1 \cdot \sharp_\vartheta$ and $\vartheta = \neg D_A \psi'$. Moreover, $\tau_{q_\varepsilon}(q')^1 = A$ and $\neg\psi' \in t(\tau_{q_\varepsilon}(q')^2) = t(q')$. Then $q \sim_a' q'$ for each $a \in A$ by definition of $\sim_a'$. Thus $q \sim_A^{D'} q'$. The induction hypothesis yields $S', q' \models \neg\psi'$. Hence, $S', q \not\models D_A\psi'$ by the semantics.

◀

Since $\varphi \in \Psi$ for some type $\Psi \in \Gamma$, there is a state $q \in Q$ such that $\psi \in t(q)$. Then it follows from Claim 3.29 that $S', q \models \varphi$.

"$\Leftarrow$" (Completeness): Suppose φ is satisfiable in an AETS $\mathcal{S} = \langle \Pi, \Sigma, Q, \sim_1, \ldots, \sim_n, \pi, \delta \rangle$ in a state $q_\varphi \in Q$. For each state $q \in Q$, let $t(q)$ be the type $\{\psi \in \mathsf{ecl}(\varphi) \mid \mathcal{S}, q \models \psi\}$. Denote with $\mathsf{types}(Q)$ the set of all types associated with some state in Q.

In the following, it is shown that all types in $\mathsf{types}(Q)$ are ATEL-realizable in $\mathsf{types}(Q)$. Let $q \in Q$ be a state. It is to check that each type $t(q)$ in $\mathsf{types}(Q)$ satisfies conditions 1 to 6 of Definition 3.17 of ATEL realizability. Conditions 1 to 3 are can be shown as in Lemma 3.11. We now show the remaining conditions 4 to 6:

4. Suppose $\neg K_a \psi \in t(q)$. Then $\mathcal{S}, q \models \neg K_a\psi$, i.e., there is a state $q' \in Q$ with $q \sim_a q'$ such that $\mathcal{S}, q' \models \neg\psi$. Thus $t(q')$ is a type in $\mathsf{types}(Q)$ such that $\neg\psi \in t(q')$ which satisfies Condition 4(a). For 4(b), it is to show that $K_a\psi' \in t(q)$ iff $K_a\psi' \in t(q')$ for each $K_a\psi' \in \mathsf{ecl}(\varphi)$. For the direction from left to right, suppose $K_a\psi' \in t(q)$. Then $\mathcal{S}, q \models K_a\psi'$, i.e., $\mathcal{S}, q'' \models \psi'$ for all states $q'' \in Q$ with $q \sim_a q''$. The symmetry of $\sim_a$ implies $q' \sim_a q$, and by transitivity of $\sim_a$, it holds that $q' \sim_a q''$ for all states $q'' \in Q$ with $q \sim_a q''$. Thus $\mathcal{S}, q'' \models \psi'$ for all states $q'' \in Q$ with $q' \sim_a q''$. Then the semantics yields $\mathcal{S}, q' \models K_a\psi'$. Hence $K_a\psi' \in t(q')$. The right-to-left direction is similar.
5. Suppose $\neg C_A\psi \in t(q)$. Then $\mathcal{S}, q \models \neg C_A\psi$, i.e., there is a state $q' \in Q$ with $q \sim_A^C q'$ such that $\mathcal{S}, q' \models \neg\psi$. Since $\sim_A^C = (\bigcup_{a \in A} \sim_a)^*$, there is a sequence $q_0 \cdots q_\ell \in Q^*$, $\ell \geq 0$, of states and a sequence $a_0 \cdots a_{\ell-1} \in A^*$ of agents such that $q = q_0 \sim_{a_0} q_1 \sim_{a_1} \cdots \sim_{a_{\ell-1}} q_\ell = q'$. In the following, it is shown that the sequence $t(q_0) \cdots t(q_\ell)$ of types fulfills the conditions of a $\neg C_A\psi$-witness path rooted at $t(q)$ in $\mathsf{types}(Q)$. Condition 1 is fulfilled by $q = q_0$ and Condition 3 since $\neg\psi \in t(q') = t(q_\ell)$. For Condition 2, it suffices to show that $K_{a_i}\psi' \in t(q_i)$ iff $K_{a_i}\psi' \in t(q_{i+1})$ for all $i < \ell$ and each $K_{a_i}\psi' \in \mathsf{ecl}(\varphi)$. For the direction from left to right, let $i < \ell$ and suppose $K_{a_i}\psi' \in t(q_i)$. Then $\mathcal{S}, q_i \models K_{a_i}\psi'$, i.e., $\mathcal{S}, q'' \models \psi'$ for all states $q'' \in Q$ with $q_i \sim_{a_i} q''$. The symmetry of $\sim_{a_i}$ implies $q_{i+1} \sim_{a_i} q_i$, and by transitivity of $\sim_{a_i}$, it holds that $q_{i+1} \sim_{a_i} q''$ for all

states $q'' \in Q$ with $q_i \sim_{a_i} q''$. Thus $\mathcal{S}, q_{i+1} \models \psi'$ for all states $q'' \in Q$ with $q' \sim_a q''$. Then the semantics yields $\mathcal{S}, q_{i+1} \models K_{a_i}\psi'$. Hence $K_{a_i}\psi' \in t(q_{i+1})$. The right-to-left direction is similar.

6. This case is similar to 4.

From $\mathcal{S}, q_\varphi \models \varphi$ it follows that $\varphi \in t(q_\varphi)$. Then types(Q) is a set of types that are each ATEL-realizable in types(Q) and $t(q_\varphi)$ is a type in types(Q) such that $\varphi \in t(q_\varphi)$. Let Δ be the set of types for φ computed by the type elimination algorithm. It is easy to see that types(Q) $\subseteq \Delta$. Hence, the algorithm returns "Yes, the input formula φ is satisfiable". □

LEMMA 3.25. *The described type elimination procedure runs in exponential time.*

PROOF. This proof is similar to the proof of Lemma 3.12. Suppose φ is given and let $n = |\varphi|$. Recall that the size of the extended closure ecl(φ) is linear in the length of φ, i.e., $|\text{ecl}(\varphi)| = c \cdot n$ for some constant $c \geq 1$. The algorithm computes a sequence $\Delta_0, \ldots, \Delta_m$ of sets of types such that $\Delta_0 \supsetneq \Delta_1 \supsetneq \cdots \supsetneq \Delta_m$. Since $\Delta_0 = \Gamma_\varphi \subseteq 2^{\text{ecl}(\varphi)}$, this sequence is finite with $m \leq 2^{c \cdot n}$. For each i with $0 < i \leq m$, it holds that $|\Delta_i| < |\Delta_0| \leq 2^{c \cdot n}$. Thus, to compute the set Δ_{i+1} at most $2^{c \cdot n}$ types in Δ_i need to be checked whether they are ATEL-realizable in Δ_i. In the following, it is shown that for a type $\Psi \in \Delta_i$ at most $2^{\mathcal{O}(n^2)}$ steps are needed to check for Ψ's ATEL realizability in Δ_i. Consider the six points in Definition 3.17:

1. For at most n^n vectors (as $k \leq n$), inclusion tests for maximal $2^{c \cdot n}$ types in Δ_i have to be performed. Hence, this takes not more than $n^n \cdot 2^{c \cdot n} = 2^{\mathcal{O}(n^2)}$ steps.

2.,3. By Lemma 3.18, to decide the existence of witness trees takes not more than $2^{\mathcal{O}(n^2)}$ steps. This has to be done for at most n formulas of the form $\langle\langle A\rangle\rangle\psi\,\mathcal{U}\,\vartheta$ or $\neg\langle\langle A\rangle\rangle\Box\psi$ in Ψ. Thus, altogether maximal $2^{\mathcal{O}(n^2)}$ steps are needed.

4., 6. For maximal n formulas of the form $\neg K_a\psi$ ($\neg D_A\psi$) in Ψ, conditions 4(a) and 4(b) (6(a) and 6(b)) have to be checked for at most $2^{c \cdot n}$ types in Δ_i. Checking for these conditions takes only polynomially many steps wrt. n. Consequently, at most $2^{\mathcal{O}(n)}$ steps are needed.

5. To check for the existence of witness paths takes not more than $2^{\mathcal{O}(n)}$ many steps by Lemma 3.18. This has to be done for at most n formulas of the form $\neg C_A\psi$ in Ψ. Hence, entirely at most $2^{\mathcal{O}(n)}$ steps are needed.

Note that checking whether there is a type in Δ_m that contains φ takes not more than $2^{\mathcal{O}(n)}$ steps. We conclude that our decision procedure runs in time exponential in the size of the input. □

3.3. Conclusion

In this chapter, we have revisited the satisfiability problem for ATL, which was settled at EXPTIME-complete in a result of van Drimmelen [**252**] for cases where the set of agents was fixed externally in advance. We pointed out that if the set of agents is not fixed externally, then van Drimmelen's construction yielded only a 2-EXPTIME upper bound. This motivated the statement of three variations of the ATL satisfiability problem, where the set of agents was not fixed externally in advance. We have shown that each of these variations is EXPTIME-complete.

Moreover, we combined the decision procedure for ATL with the technique of [**103**] to show that the epistemic extension ATEL of ATL with common and distributed knowledge is EXPTIME-complete as well. This result shows that adding epistemic operators for common and distributed knowledge to ATL does not yield an increase in computational complexity.

ATEL, as it is defined in Section 2.5.1, does not allow for any interaction between knowledge and time. For instance, it is possible for an agent a that at two for a epistemically indistinguishable states a makes different choices. For future work, it would be interesting to investigate the complexity of variants of ATEL that capture various desired and reasonable interactions between knowledge and time; cf. the discussion in Section 2.5.6. Another significant issue to address is the precise complexity of ATL*'s satisfiability problem which lies between 2-EXPTIME and 3-EXPTIME; cf. Section 2.4.4 in the previous chapter.

CHAPTER 4

ATL with Explicit Strategies

4.1. Preliminaries

In this chapter, we introduce ATLES, a variant of ATL with explicit names for strategies in the object language. ATLES makes it possible to refer to the same strategy in different occurrences of path quantifiers, and as a consequence it becomes possible to express some properties in ATLES that cannot even be expressed in ATL*. We present a complete axiomatic system for ATLES. Moreover, we show that the satisfiability problem for ATLES is no more complex than for ATL: it is EXPTIME-complete. We identify two variants of the model checking problem for ATLES and investigate their computational complexity.

ATL [15] is a logic in which one can represent and reason about the strategic abilities of agents in game-like, multi-agent systems. The key-construct in ATL is $\langle\langle A\rangle\rangle\Phi$, expressing that coalition A has a strategy so as to ensure that the temporal property Φ holds. As it was pointed out in the literature [**242**, **38**], the semantics of ATL is "richer" than its language: strategies are present in the semantics but in the language one can only quantify over strategies without explicitly referring to them. As a consequence, ATL does not facilitate explicit reasoning about strategies. In this chapter, we aim to narrow the gap between semantics and language by introducing *Alternating-time Logic with Explicit Strategies* (ATLES). This logic is quite general and extends many existing logics for reasoning about strategic capabilities of agents such as "Commitment ATL" (CATL) by van der Hoek, Jamroga and Wooldridge [**242**], Coalition Logic by Pauly [**194**], "Action Logic" (AL) by Borgo [**38**], ATL$_\Sigma$ with fixing the set Σ of agents as considered by Goranko and van Drimmelen [**98**], and ATL without fixing the set of agents as defined by Walther, Lutz, Wolter and Wooldridge [**265**].

The key difference in the syntax with CATL [**242**] is that CATL's commitment operators $C_a(\varrho, \varphi)$, where ϱ is a commitment for agent a, are dropped and, instead, each path quantifier $\langle\langle A\rangle\rangle_\rho$ is additionally parameterized with a *commitment function* denoted by ρ. A commitment function, ρ, is a partial function mapping agents to strategy terms. That is, each agent b for which ρ is defined (i.e., $b \in \mathsf{dom}(\rho)$) commits to the strategy $\rho(b)$. Then $\langle\langle A\rangle\rangle_\rho\varphi$ means that 'while the agents in $\mathsf{dom}(\rho)$ act according to their commitments, the coalition A can cooperate to ensure φ as an outcome'. Originally, the CATL commitment operators $C_a(\varrho, \varphi)$ were interpreted using an update semantics [**242**] which had the effect that, once an agent committed to a strategy, she could not change or even undo that commitment. For ATLES, however, we use a different approach for the semantics with which we overcome that restriction: instead of changing (updating) the model, we define a notion of strategies for coalitions that accounts for the individual commitment of agents to strategies. Moreover, in the strategies we allow for agents with perfect information, i.e., making a choice depends on the full history of previously traversed states. In [**242**], CATL was defined for a fixed set of agents and a fixed set of strategy terms. ATLES is defined without fixing the number of agents and strategy terms in advance as it was done for ATL in Section 2.4.1 of Chapter 2.

Pauly's Coalition Logic can be conceived as the next fragment of ATL; see [**98**]. Recently, Borgo [**38**] presented Action Logic which, interestingly, can be seen as the next fragment of ATLES. The basic construct in AL is $\vec{v}\varphi$, where $\vec{v}$ is a vector with for every agent i a place that can be filled with either a constant action term a_i, (meaning that agent i has committed to a_i) a quantifier $\exists x_i$ (agent i has a choice to make) or $\forall y_i$ (for all actions of i). The interpretation of $\vec{v}\varphi$ is that under the "assignment" $\vec{v}$, the formula φ will hold in the next state. But this corresponds to the ATLES expression $\langle\langle A\rangle\rangle_\rho\varphi$, where A is the set of agents with an $\exists x_i$ in $\vec{v}$, and the function ρ collects all the pairs (i, a_i) with a_i in $\vec{v}$.

Recently, Ågotnes, Goranko and Jamroga [**7**] introduced the ATL-variants IATL and MIATL. These variants extend ATL with irrevocable strategies, i.e., an agent choosing a strategy means that she commits to that strategy without ever changing it. This is achieved with an update semantics, where, whenever an agent chooses a strategy, the model is truncated in a way to make it impossible for the agent to choose another strategy later. In ATLES, we can "simulate" irrevocable strategies without altering the model by assigning agents to the same strategy terms in the path quantifiers of subformulas.

This chapter is organized as follows. After introducing ATLES in this section, in Section 4.2, we investigate its expressivity, in Section 4.3, we formulate two variants of the model checking problem for ATLES and establish their computational complexities, while, in Section 4.4 we settle the complexity of ATLES's satisfiability problem and, finally, in Section 4.5 we show completeness with respect to ATLES semantics.

4.1.1. ATLES.

We now introduce ATLES, which provides explicit names for strategies.

DEFINITION 4.1. (ATLES SYNTAX). Let $\mathbf{\Pi}$ be a countable infinite set of *atomic propositions*, $\mathbf{\Sigma}$ a countable infinite set of *agents* and $\mathbf{\Upsilon}$ a set of *strategy terms* with $\mathbf{\Upsilon} = \bigcup_{a\in\mathbf{\Sigma}} \mathbf{\Upsilon}_a$, where $\mathbf{\Upsilon}_a$ is a countable infinite set of *strategy terms for agent a*. A *coalition* is a finite set $A \subset \mathbf{\Sigma}$ of agents. A *commitment function* is a partial function $\rho : \mathbf{\Sigma} \to \mathbf{\Upsilon}$ mapping finitely many agents $a \in \mathbf{\Sigma}$ to a strategy term $\rho(a) \in \mathbf{\Upsilon}_a$ for a. The set of ATLES-formulas is generated by the following grammar, where $p \in \mathbf{\Pi}$, A ranges over coalitions, ρ over commitment functions and φ over ATLES-formulas:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle\langle A\rangle\rangle_\rho \bigcirc\varphi \mid \langle\langle A\rangle\rangle_\rho \Box\varphi \mid \langle\langle A\rangle\rangle_\rho \varphi\,\mathcal{U}\,\varphi.$$

Logical truth ($\top$), the Boolean connectives ($\wedge$, $\rightarrow$ and $\leftrightarrow$) and $\Diamond$ are defined as usual. Observe that in ATLES the operators $\langle\langle A\rangle\rangle_\rho\Box$ are explicitly defined in the syntax, since $\langle\langle A\rangle\rangle_\rho\Box$ cannot be expressed in terms of $\langle\langle A\rangle\rangle_\rho$ and $\mathcal{U}$ within ATLES. ATL is the fragment of ATLES only allowing for commitment functions that are undefined for all agents.

As semantic structures, an extension of alternating transition systems were suggested in **[242]** that explicitly account for actions and action pre-conditions, so-called action-based alternating transition systems. In this work, however, we confine ourselves to a variant of the alternating transition systems introduced in **[15]** extended with strategy terms and a denotation function mapping strategy terms to strategies. These transition systems can easily be seen to be equivalent to the action-based structures of **[242]**.

DEFINITION 4.2. (ATSN). Let $\Sigma = \{1,\ldots,n\} \subset \mathbf{\Sigma}$, with $n \geq 1$, be a finite set of agents and, for each agent $a \in \Sigma$, $\Upsilon_a \subset \mathbf{\Upsilon}_a$ a finite set of a-strategy terms. An *alternating transition system with strategy names (ATSN) for* Σ is a tuple $\mathcal{S} = \langle \Pi, \Sigma, Q, \{\Upsilon_a\}_{a\in\Sigma}, \pi, \delta, ||\cdot||\rangle$ where

- $\Pi \subseteq \mathbf{\Pi}$ is a finite, non-empty set of *atomic propositions*,
- Q is a finite, non-empty set of *states*,
- $\pi : Q \rightarrow 2^{\Pi}$ is a *valuation function* which assigns to every state a set of atomic propositions that are true there,
- $\delta : Q \times \Sigma \rightarrow 2^{2^Q}$ is a *transition function* which maps a state $q \in Q$ and an agent $a \in \Sigma$ to a non-empty set of *choices* $\delta(q,a)$ available to a at q such that the following condition is satisfied: for every state $q \in Q$ and every set $Q_1,\ldots,Q_n$, where n is the number of agents, of choices $Q_i \in \delta(q,i)$, $1 \leq i \leq n$, the intersection $Q_1 \cap \cdots \cap Q_n$ is a singleton set, and
- $||\cdot|| : \Upsilon \rightarrow (Q^+ \rightarrow 2^Q)$ is a *denotation function*, where $\Upsilon = \bigcup_{a\in\Sigma} \Upsilon_a$, which, for each agent $a \in \Sigma$, maps an a-strategy term to an a-strategy (to be defined below).

Notice that an ATSN contains finitely many strategy terms, although there can be infinitely many different strategies. This is sufficient since an ATLES-formula is evaluated over ATSNs that contain at least the strategy terms occurring in the formula rather than a strategy term for each possible strategy.

Intuitively, $\delta(q,a)$ describes the *a-choices* available in q: when in state q, agent a chooses a set from $\delta(q,a)$ to ensure that the "next state" will be among those in the chosen set. This notion of a-choices is generalised to *A-choices* for coalitions A of agents as follows: Given an ATSN $\mathcal{S} = \langle \Pi, \Sigma, Q, \{\Upsilon_a\}_{a\in\Sigma}, \pi, \delta, ||\cdot||\rangle$, for each state $q \in Q$ and each coalition $A \subseteq \Sigma$, set

$$\delta(q,A) := \begin{cases} \{Q_A \subseteq Q \mid Q_A = \bigcap_{a\in A} Q_a \text{ where } Q_a \in \delta(q,a)\} & \text{if } A \neq \emptyset \\ \{\bigcup \delta(q,\Sigma)\} & \text{if } A = \emptyset \end{cases}.$$

When in state q, the agents in coalition A collectively choose a set from $\delta(q, A)$ to ensure that the "next state" is from this set. Note that $\delta(q, A)$ is non-empty for each state q and coalition A, and $\delta(q, \Sigma)$ is a set of singletons. The states in the singleton sets of $\delta(q, \Sigma)$ are the *successors* of q, i.e., the system is completely determined when all the agents have made their choice. Since the empty coalition cannot influence the behaviour of the system, $\delta(q, \emptyset)$ is set to $\bigcup \delta(q, \Sigma)$, the set of all possible successors of q.

An infinite sequence $\lambda = q_0 q_1 q_2 \cdots \in Q^\omega$ of states is a *computation* if, for all positions $i \geq 0$, there is a choice $\{q_{i+1}\} \in \delta(q_i, \Sigma)$. Denote with $\lambda[i]$ the i-th component q_i in λ, and with $\lambda[0, i]$ the initial sequence $q_0 \cdots q_i$ of λ.

A *strategy* for an agent $a \in \Sigma$ is a function $\sigma_a : Q^+ \to \mathbf{2}^Q$ that maps all finite sequences $\lambda \cdot q \in Q^+$ of states to a choice $\sigma_a(\lambda \cdot q) \in \delta(q, a)$ available to agent a at q. Note that $\lambda \cdot q$ denotes the concatenation of the finite sequence λ with the state q. A *strategy for a coalition* A is a set of strategies $\sigma_A = \{\sigma_a \mid a \in A\}$, one for each agent in A. Given a commitment function ρ, we augment the notion of strategies for A to that of a *ρ-strategy for A*. This is a set of strategies containing for each committed agent in $\mathsf{dom}(\rho)$, the strategy she committed to and, for each free agent in $A \setminus \mathsf{dom}(\rho)$, an arbitrary strategy for this agent. Formally, a *ρ-strategy for A* is a strategy $\sigma_{A \cup \mathsf{dom}(\rho)}$ for the agents in $A \cup \mathsf{dom}(\rho)$ such that for all agents $a \in \mathsf{dom}(\rho)$, the strategy σ_a for a in $\sigma_{A \cup \mathsf{dom}(\rho)}$ is such that $\sigma_a = ||\rho(a)||$.

The set $\mathsf{out}(q, \sigma_A)$ of *outcomes* of a strategy σ_A for the agents in A starting at a state q is the set of all computations $\lambda = q_0 q_1 q_2 \cdots \in Q^\omega$ such that $q_0 = q$ and $q_{i+1} \in \bigcap_{\sigma_a \in \sigma_A} \sigma_a(\lambda[0, i])$ for all $i \geq 0$.

Now we can be more precise about the meaning of $\langle\langle A \rangle\rangle_\rho \varphi$:

> $\langle\langle A \rangle\rangle_\rho \varphi$ means that, given the commitments of the agents $b \in \mathsf{dom}(\rho)$ to use strategy $\rho(b)$, the agents $a \in A \setminus \mathsf{dom}(\rho)$ have a strategy such that, no matter what the agents $c \in \Sigma \setminus (\mathsf{dom}(\rho) \cup A)$ will do, φ will result.

DEFINITION 4.3. (ATLES SEMANTICS). Given an ATSN $\mathcal{S} = \langle \Pi, \Sigma, Q, \{\Upsilon_a\}_{a \in \Sigma}, \pi, \delta, || \cdot || \rangle$, the satisfaction relation $\models$ is inductively defined as follows:

- $\mathcal{S}, q \models p$ iff $p \in \pi(q)$, for all atomic propositions $p \in \Pi$;
- $\mathcal{S}, q \models \neg\psi$ iff $\mathcal{S}, q \not\models \psi$;
- $\mathcal{S}, q \models \psi \vee \varphi$ iff $\mathcal{S}, q \models \psi$ or $\mathcal{S}, q \models \varphi$;
- $\mathcal{S}, q \models \langle\langle A \rangle\rangle_\rho \bigcirc \varphi$ iff there is a ρ-strategy $\sigma_{A \cup \mathsf{dom}(\rho)}$ for the agents in $A \cup \mathsf{dom}(\rho)$ such that for all computations $\lambda \in \mathsf{out}(q, \sigma_{A \cup \mathsf{dom}(\rho)})$, it holds that $\mathcal{S}, \lambda[1] \models \varphi$;
- $\mathcal{S}, q \models \langle\langle A \rangle\rangle_\rho \Box \varphi$ iff there is a ρ-strategy $\sigma_{A \cup \mathsf{dom}(\rho)}$ for the agents in $A \cup \mathsf{dom}(\rho)$ such that for all computations $\lambda \in \mathsf{out}(q, \sigma_{A \cup \mathsf{dom}(\rho)})$, it holds that $\mathcal{S}, \lambda[i] \models \varphi$ for all positions $i \geq 0$;

- $\mathcal{S}, q \models \langle\langle A \rangle\rangle_\rho \psi \mathcal{U} \varphi$ iff there is a ρ-strategy $\sigma_{A \cup \mathsf{dom}(\rho)}$ for the agents in $A \cup \mathsf{dom}(\rho)$ such that for all computations $\lambda \in \mathsf{out}(q, \sigma_{A\cup\mathsf{dom}(\rho)})$, there is a position $i \geq 0$ such that $\mathcal{S}, \lambda[i] \models \varphi$ and $\mathcal{S}, \lambda[j] \models \psi$ for all positions j with $0 \leq j < i$.

If for some state q of some ATSN $\mathcal{S}$ it holds that $\mathcal{S}, q \models \psi$, then the ATLES-formula ψ is *true* at q, and $\mathcal{S}$ is called a *model* of ψ. An ATLES-formula is *satisfiable* if it has a model.

ATLES contains ATL as a fragment in which formulas exclusively use commitment functions that are undefined for all agents. For expressing properties in ATL, see Example 2.23 in Chapter 2. To better understand the additional expressive power of ATLES compared to ATL and the difference of ATLES to ATL*, consider the following example.

EXAMPLE 4.4. (EXPLICIT STRATEGIES). First, we illustrate how to state properties with ATLES that cannot be expressed in ATL. Figure 4.1 depicts parts of two structures each modelling a continuous process that grants a privilege after requests. The left structure is modelled by a cycle containing states at which a request is made and grants are given. In the right structure, however, the process has to leave the cycle to grant a privilege. The arrows depict system transitions. Using ATL*, we can specify the liveness

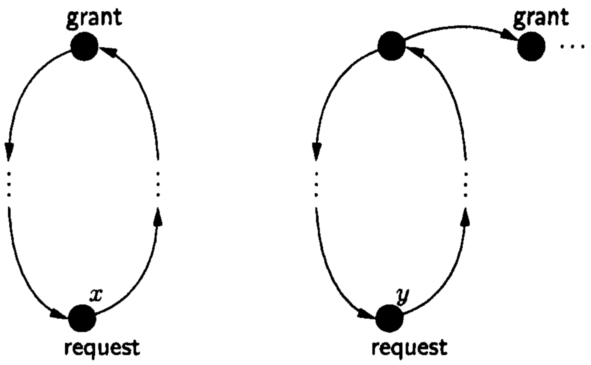


FIGURE 4.1. Two ATSNs for one agent.

property that the coalition A has a collective strategy to guarantee computations along which grants are not infinitely often postponed, and, thus, infinitely many grants are given. Thus, we have $x \models \langle\langle A \rangle\rangle \Box \Diamond \mathsf{grant}$. Notice that $\langle\langle A \rangle\rangle \Box \Diamond \mathsf{grant}$ does not hold at state y since, in the structure on the righthand side, only one grant can be given. In ATL, however, this property is not expressible. For instance, the similar ATL-formula $\langle\langle A \rangle\rangle \Box \langle\langle A \rangle\rangle \Diamond \mathsf{grant}$ states something different. The coalition A in the outer path quantifier allows computations along which no grant is given, and the agents in the nested path quantifier can select computations along which only one grant is given. This formula is true at both states x and y, i.e., this ATL-formula cannot detect the difference between x and y in the two structures. Notice that this does not mean that ATL cannot distinguish between x and y. However, using ATLES, we can express this liveness property. Using explicit names for strategies in the object language, we can fix the behaviour of agents by referring to the same strategy in different path quantifiers. In this case, we denote the collective strategy of the coalition A by the strategy term, say, ϱ_A and let the agents in A use the same strategy at both path quantifiers. We make sure that the strategy term ϱ_A corresponds to some appropriate strategy σ_A for the agents in A such that the system never leaves the cycles in Figure 4.1. Then the liveness property can be expressed with the ATLES-formula $\langle\!\langle A\rangle\!\rangle_{\{A\mapsto\varrho_A\}}\Box\langle\!\langle A\rangle\!\rangle_{\{A\mapsto\varrho_A\}}\Diamond\mathsf{grant}$. Notice that this formula holds at x but not at y, which means that ATLES can differentiate between the two structures in Figure 4.1 and overcomes a lack of expressivity in ATL.

The second example illustrates how we can state properties with ATLES that cannot be expressed even in ATL*. Such properties involve non-uniform choices, which can be described with ATLES using coalitions with committed and uncommitted agents. Figure 4.2 shows two structures for two agents each. The arrows denote the system transitions and the round boxes labelled with a or b denote the choices of agents a and b, respectively, at the states x and y. As illustrated in Figure 4.2, some successor states of x and y satisfy the proposition p and others not. Suppose we want to distinguish the

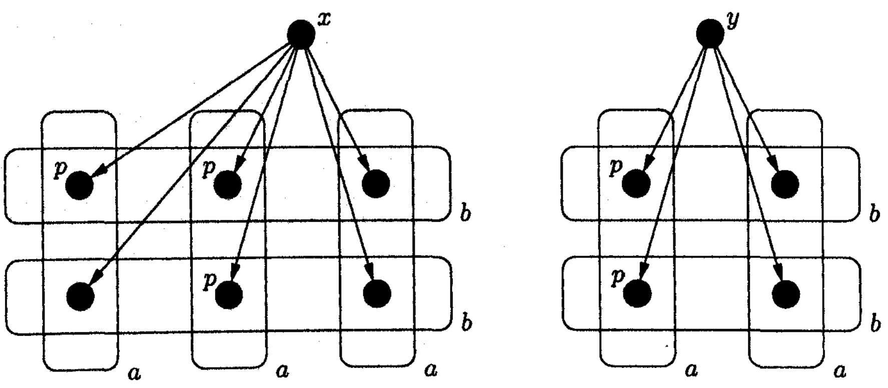


FIGURE 4.2. Overlapping choices in two ATSNs for two agents.

states x and y. Observe that all a-choices at y are uniform in the sense that all states of an a-choice either satisfy p or falsify p. At x, however, the a-choices are not uniform, because the left-most a-choice contains a state satisfying p and another state falsifying p. Using ATLES, the situation at x can be described as follows: Using an explicit name, we can denote the left-most a-choice at x with a strategy term for a, say, ϱ_a:

$$x \models \langle\!\langle a,b\rangle\!\rangle_{\{a\mapsto\varrho_a\}}\bigcirc p \wedge \langle\!\langle a,b\rangle\!\rangle_{\{a\mapsto\varrho_a\}}\bigcirc\neg p.$$

Notice that, using the name ϱ_a, we can refer to the same a-choice in different path quantifiers, where agent b selects different subsets of a's choice. In other words, in each path quantifier, agent b refines a's strategy ϱ_a in a different way. At the state y, however, we have that

$$y \not\models \langle\!\langle a,b\rangle\!\rangle_{\{a\mapsto\varrho_a\}}\bigcirc p \wedge \langle\!\langle a,b\rangle\!\rangle_{\{a\mapsto\varrho_a\}}\bigcirc\neg p$$

since no a-choice at y contains a p-state and a non-p-state. Consequently, ATLES can distinguish between the states x and y. On the other hand, ATL or even ATL* cannot describe the situations at x and y so closely as to distinguish the two states. For instance, consider the similar ATL-formula $\langle\langle a,b\rangle\rangle\bigcirc p \wedge \langle\langle a,b\rangle\rangle\bigcirc q$. It is readily checked that this formula is true at both states x and y. In fact, no ATL*-formula can distinguish x and y. To see this, notice that x and y are A-bisimilar for all $A \in \{a,b\}$; cf. Definition 2.25 of alternating bisimulation in Section 2.4.5 of Chapter 2. But then, x and y satisfy the same ATL*-formulas according to Theorem 2.26.

The final example exhibits some ATL*-formulas with Boolean combination of temporal expressions inside a path quantifier that cannot be expressed in ATLES. Figure 4.3 shows two structures for an agent a and some other agent, say, b. The round boxes denote the choices for a at the respective previous state, whereas the choices for agent b are not illustrated. Notice that any strategy that a selects at the states x and y will

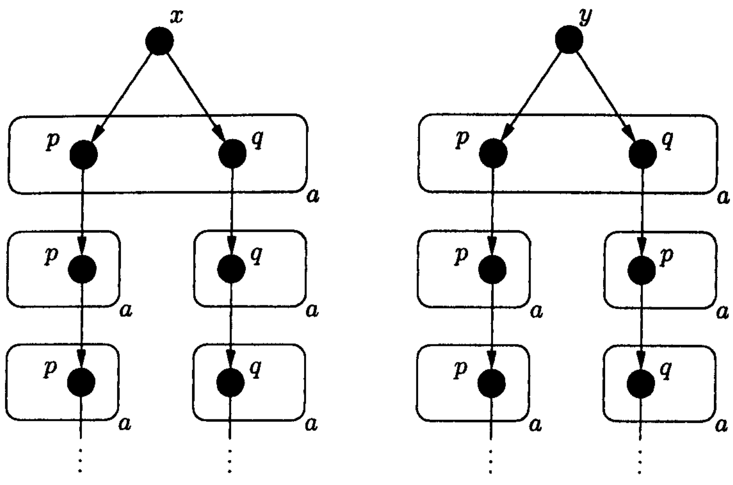


FIGURE 4.3. Computations in two ATSNs for two agents.

give rise to two computation paths since the a-choices at x and y contain two states. Consider the two computations determined by a at x. Using ATL*, we can express that they satisfy 'always p' or 'always q'. We have $x \models \langle\langle a\rangle\rangle(\Box p \vee \Box q)$. The situation is different at state y: The computation on the right starting at y does not satisfy 'always q'. Therefore, we have $y \not\models \langle\langle a\rangle\rangle(\Box p \vee \Box q)$. In ATLES, we cannot express $\langle\langle a\rangle\rangle(\Box p \vee \Box q)$ since Boolean combinations of temporal expressions inside a path quantifier are not available. The similar ATLES-formula $\langle\langle a\rangle\rangle_{\{a\mapsto\varrho_a\}}\Box p \vee \langle\langle a\rangle\rangle_{\{a\mapsto\varrho_a\}}\Box q$ does not express the same property: It requires that both computations satisfy 'always p' or 'always q'. Clearly, this is not the case at the states x and y. Anyway, observe that the two path quantifiers each select the same two computations since agent a uses both times her strategy ϱ_a. Let us consider another, similar formula: $\langle\langle a\rangle\rangle_{\{\}}\Box(p \vee q)$. Note that this formula belongs to the ATL-fragment of ATLES. It requires that the states of the computations selected by a satisfy p or q. It is readily checked that $\langle\langle a\rangle\rangle_{\{\}}\Box(p \vee q)$

is true at x and y. Consequently, both ATLES-formulas similar to the ATL*-formula $\langle\langle a\rangle\rangle(\Box p \vee \Box q)$ are not suited to differentiate between the states x and y.

⊣

4.1.2. Axiomatic system for ATLES. In this section, we present the axiomatic system for ATLES: Table 4.1 contains the axioms and inference rules. The notions of ATLES-provability and consistency are defined as usual. The axioms and the

(TAUT)	Propositional tautologies	
($\bot$)	$\neg\langle\langle A\rangle\rangle_\rho \bigcirc \bot$	
($\top$)	$\langle\langle A\rangle\rangle_\rho \bigcirc \top$	
(S)	$\langle\langle A\rangle\rangle_\rho \bigcirc\varphi \wedge \langle\langle B\rangle\rangle_\rho \bigcirc\psi \to \langle\langle A\cup B\rangle\rangle_\rho \bigcirc(\varphi\wedge\psi)$	where $A\cap B \subseteq \mathsf{dom}(\rho)$
(C1)	$\langle\langle A\rangle\rangle_\rho \bigcirc\varphi \to \langle\langle A\rangle\rangle_{\rho'} \bigcirc\varphi$	where $\rho' = \rho\cup\{a\mapsto \varrho_a\}$, $a\notin A$, $\varrho_a \in \Upsilon_a$
(C2)	$\langle\langle A\rangle\rangle_{\rho'} \bigcirc\varphi \to \langle\langle A\rangle\rangle_{\rho} \bigcirc\varphi$	where $\rho' = \rho\cup\{a\mapsto \varrho_a\}$, $a\in A$, $\varrho_a \in \Upsilon_a$
(C3)	$\langle\langle A\cup\{a\}\rangle\rangle_\rho \bigcirc\varphi \to \langle\langle A\rangle\rangle_\rho \bigcirc\varphi$	where $a \in \mathsf{dom}(\rho)$
(FP$_\Box$)	$\langle\langle A\rangle\rangle_\rho \Box\varphi \leftrightarrow \varphi \wedge \langle\langle A\rangle\rangle_\rho \bigcirc \langle\langle A\rangle\rangle_\rho \Box\varphi$	
(GFP$_\Box$)	$\langle\langle \emptyset\rangle\rangle_\rho \Box(\theta \to (\varphi\wedge \langle\langle A\rangle\rangle_\rho \bigcirc\theta)) \to \langle\langle \emptyset\rangle\rangle_\rho \Box(\theta \to \langle\langle A\rangle\rangle_\rho \Box\varphi)$	
(FP$_\mathcal{U}$)	$\langle\langle A\rangle\rangle_\rho \psi\,\mathcal{U}\,\varphi \leftrightarrow \varphi \vee (\psi \wedge \langle\langle A\rangle\rangle_\rho \bigcirc \langle\langle A\rangle\rangle_\rho \psi\,\mathcal{U}\,\varphi)$	
(LFP$_\mathcal{U}$)	$\langle\langle \emptyset\rangle\rangle_\rho \Box((\varphi \vee (\psi\wedge \langle\langle A\rangle\rangle_\rho \bigcirc\theta)) \to \theta) \to \langle\langle \emptyset\rangle\rangle_\rho \Box(\langle\langle A\rangle\rangle_\rho \psi\,\mathcal{U}\,\varphi \to \theta)$	
(Modus Ponens)	$\dfrac{\varphi, \varphi\to\psi}{\psi}$	
($\langle\langle \emptyset\rangle\rangle_\rho\Box$-Necessitation)	$\dfrac{\varphi}{\langle\langle \emptyset\rangle\rangle_\rho \Box\varphi}$	
($\langle\langle A\rangle\rangle_\rho\bigcirc$-Monotonicity)	$\dfrac{\varphi\to\psi}{\langle\langle A\rangle\rangle_\rho \bigcirc\varphi \to \langle\langle A\rangle\rangle_\rho \bigcirc\psi}$	

TABLE 4.1. An axiom system for ATLES.

inference rules were inspired by the axiomatisation of Coalition Logic in [195] and of ATL in [98] and extended to ATLES. Essentially, this involved adapting every ATL axiom by additionally parameterising each path quantifier with a commitment function. For the ATL axiom (S), we also needed to adapt its condition. Moreover, we added three new axioms (C1) to (C3) that characterise the expressivity of the commitment function. Intuitively, these three axioms express the following:

(C1): given a commitment ρ, a coalition A can still ensure φ at the next state after an agent outside of A commits to a strategy;

(C2): given a commitment ρ', a coalition A can still ensure φ at the next state after a member of A dismisses her commitment;

(C3): given a commitment ρ, after a committed agent a has left the coalition $A\cup\{a\}$, the remaining agents in A are still able to ensure φ at the next state.

In Section 4.5, we show completeness of the axiomatic system for ATLES in Table 4.1. In particular, we determine in Lemma 4.23 under which conditions the power of one coalition A, given some commitments ρ, can be "transferred" to the power of B, by assuming commitments ξ. That is, we characterise when a formula of the form $\langle\langle A\rangle\rangle_\rho \bigcirc\psi \to \langle\langle B\rangle\rangle_\xi \bigcirc\psi$ can be derived. Intuitively, the implication in Lemma 4.23 can be seen as a course of action where agents join or leave the coalition and agents take or dismiss commitments.

4.2. Expressivity

As demonstrated in Example 4.4, we have that ATLES and ATL* are incomparable with respect to expressivity: some formulas of ATL* cannot be expressed in ATLES, while some formulas of ATLES cannot be expressed in ATL*. But how exactly does ATLES's expressivity compare to that of ATL*? In the following, we aim to give an answer to that question.

Recall that ATL* allows for Boolean combinations and nesting of temporal operators inside a path quantifier; cf. Section 2.4.1 in Chapter 2. Some of these formulas can be translated into ATLES in a satisfiability preserving way. For instance, the ATL*-formula with nesting of the temporal operator next-time ($\bigcirc$) can be translated into ATLES as follows: for $n \geq 0$,

$$\langle\langle a,b\rangle\rangle \bigcirc^n \varphi = \underbrace{\langle\langle a,b\rangle\rangle_{\{a\mapsto\varrho_a, b\mapsto\varrho_b\}}\bigcirc \cdots \langle\langle a,b\rangle\rangle_{\{a\mapsto\varrho_a, b\mapsto\varrho_b\}}\bigcirc}_{n-\text{times}} \varphi$$

where ϱ_a, ϱ_b are fresh strategy terms for the agents a, b, respectively. Note that with the translation, we have an exponential blow-up in the formula size if n is coded in binary.

Here are some more ATL*-formulas with nesting of temporal operators that can be translated into ATLES:

$$\begin{array}{lcl}
\langle\langle a,b\rangle\rangle[\varphi\,\mathcal{U}(\psi\,\mathcal{U}\,\vartheta)] & = & \langle\langle a,b\rangle\rangle_{\{a\mapsto\varrho_a, b\mapsto\varrho_b\}}[\varphi\,\mathcal{U}(\langle\langle a,b\rangle\rangle_{\{a\mapsto\varrho_a, b\mapsto\varrho_b\}}(\psi\,\mathcal{U}\,\vartheta))] \\
\langle\langle a,b\rangle\rangle[\varphi\,\mathcal{U}(\bigcirc\psi)] & = & \langle\langle a,b\rangle\rangle_{\{a\mapsto\varrho_a, b\mapsto\varrho_b\}}[\varphi\,\mathcal{U}(\langle\langle a,b\rangle\rangle_{\{a\mapsto\varrho_a, b\mapsto\varrho_b\}}\bigcirc\psi)] \\
\langle\langle a,b\rangle\rangle\bigcirc(\varphi\,\mathcal{U}\,\psi) & = & \langle\langle a,b\rangle\rangle_{\{a\mapsto\varrho_a, b\mapsto\varrho_b\}}\bigcirc\langle\langle a,b\rangle\rangle_{\{a\mapsto\varrho_a, b\mapsto\varrho_b\}}(\varphi\,\mathcal{U}\,\psi).
\end{array}$$

To pinpoint the relationship between ATLES and ATL* more precisely, we now define a translation function $(\cdot)^*$ mapping some ATL*-formulas φ to formulas of ATLES in a satisfiability preserving way. To this end, recall Definition 2.20 of the ATL* syntax in Chapter 2. Observe that, other than in ATLES, in ATL* we can express Boolean combinations and nesting of temporal operators inside a path quantifier. We restrict the translation function to formulas in *negation normal form*, where negation only occurs in front of propositional variables or path quantifiers. To understand the following definition of the translation function, the distinction between ATL* state formulas and path formulas is important.

DEFINITION 4.5. (TRANSLATION FUNCTION). We define a partial function $(\cdot)^*_{(B,\xi)}$ on ATL* state and path formulas in negation normal form as follows, where B ranges over coalitions and ξ over commitment functions. For all state formulas p, φ, φ' and all path formulas ψ, ψ' in negation normal form:

$$\begin{array}{lcl}
(p)^*_{\langle B,\xi\rangle} & := & p, \text{ for atomic propositions } p \in \mathbf{\Pi}; \\
(\neg\varphi)^*_{\langle B,\xi\rangle} & := & \neg(\varphi)^*_{\langle B,\xi\rangle}; \\
(\varphi \vee \varphi')^*_{\langle B,\xi\rangle} & := & (\varphi)^*_{\langle B,\xi\rangle} \vee (\varphi')^*_{\langle B,\xi\rangle}; \\
(\langle\!\langle A\rangle\!\rangle\psi)^*_{\langle B,\xi\rangle} & := & (\psi)^*_{\langle A,\rho\rangle}, \text{ for commitment function } \rho \text{ with } \mathsf{dom}(\rho) = A \text{ and} \\
& & \text{the range of } \rho \text{ containing only fresh strategy terms;} \\
(\psi \wedge \psi')^*_{\langle B,\xi\rangle} & := & (\psi)^*_{\langle B,\xi\rangle} \wedge (\psi')^*_{\langle B,\xi\rangle}; \\
(\bigcirc\psi)^*_{\langle B,\xi\rangle} & := & \langle\!\langle B\rangle\!\rangle_\xi \bigcirc (\psi)^*_{\langle B,\xi\rangle}; \\
(\neg\bigcirc\psi)^*_{\langle B,\xi\rangle} & := & (\bigcirc\neg\psi)^*_{\langle B,\xi\rangle}; \\
(\psi\,\mathcal{U}\,\psi')^*_{\langle B,\xi\rangle} & := & \langle\!\langle B\rangle\!\rangle_\xi((\psi)^*_{\langle B,\xi\rangle}\,\mathcal{U}(\psi')^*_{\langle B,\xi\rangle}).
\end{array}$$

The following lemma establishes that the translation function is satisfiability preserving. The lemma can easily be shown by induction on the structure of ATL*-formulas for which $(\cdot)^*$ is defined; we leave the details to the reader.

LEMMA 4.6. *Let B be the empty coalition and ξ_0 the empty commitment function. For all ATL*-formulas φ in negation normal form such that the translation function $(\cdot)^*_{\langle B,\xi_0\rangle}$ is defined for φ, the following are equivalent:*

(a) *φ is satisfiable wrt. ATL*;*

(b) *$(\varphi)^*_{\langle B,\xi_0\rangle}$ is satisfiable wrt. ATLES.*

The translation function $(\cdot)^*$ determines only a fragment of ATL* that can be translated into ATLES in a satisfiability preserving way. This is not surprising since ATLES is not expressive enough to subsume full ATL* as we have already seen in Example 4.4. In the following, we exhibit two more examples of ATL*-formulas that cannot be translated. The first example deals with negated path quantifiers. Consider the ATL*-formula

$$\neg\langle\!\langle A\rangle\!\rangle\Box\Diamond\varphi$$

that expresses the negation of the liveness property from Example 4.4, i.e., the coalition A cannot guarantee that along all computations grants are not infinitely often postponed. At first sight, there seem to be two possibilities to translate that formula into ATLES:

(i) $\neg\langle\!\langle A\rangle\!\rangle_\rho\Box\langle\!\langle A\rangle\!\rangle_\rho\Diamond\varphi$, where the commitment function ρ maps each agent $a \in A$ to some a-strategy term $\rho(a) = \varrho_a$; and

(ii) $\neg\langle\!\langle A\rangle\!\rangle_{\rho'}\Box\langle\!\langle A\rangle\!\rangle_{\rho'}\Diamond\varphi$, where the commitment function ρ' maps each agent $b \in \Sigma \setminus A$ to some b-strategy term $\rho(b) = \varrho_b$.

It is readily checked that the ATLES-formulas in (i) and (ii) both express something different than the ATL*-formula above. The formula in (i) avoids quantification over all A-strategies by fixing the A-strategy in ρ. The problem with the formula in (ii) is the kind of strategy that is assigned to the agents outside of coalition A. The assigned strategies for the agents in $\Sigma \setminus A$ select a choice at each state instead of responding to the choice made by coalition A. Goranko and van Drimmelen [98] called such strategies for the agents in $\Sigma \setminus A$ that respond to given A-choices *co-strategies*. Another issue in (ii) is that the grand coalition Σ of all agents needs to be known.

The second example exhibits ATL*-formulas with disjunction of temporal expressions inside a path quantifier that cannot be translated into ATLES. For instance, ATLES cannot express the ATL*-formula

$$\langle\langle A\rangle\rangle(\psi \vee \psi')$$

stating that coalition A has a strategy to ensure computations on which ψ or ψ' holds. Notice that this formula is already expressible in the ATL-extension ATL$^+$, that allows for Boolean combinations of temporal operators inside path quantifiers. ATL$^+$ relates to ATL as CTL$^+$ to CTL; cf. Definition 2.11 of CTL$^+$ and Definition 2.20 of ATL in Chapter 2. However, we can translate ATL*-formulas of the form $\langle\langle\Sigma\rangle\rangle(\psi \vee \psi')$ into ATLES provided that the grand coalition Σ of all possible agents is known. Note that, since Σ is the set of all agents, the path quantifier $\langle\langle\Sigma\rangle\rangle$ selects exactly one computation path. We have that $\langle\langle\Sigma\rangle\rangle(\psi \vee \psi') \leftrightarrow \langle\langle\Sigma\rangle\rangle\psi \vee \langle\langle\Sigma\rangle\rangle\psi'$ is valid. By knowing Σ, we can select a single computation in ATLES as well by fixing the strategies for all agents in Σ with a commitment function, say, ρ (i.e., $\mathsf{dom}(\rho) = \Sigma$). Then $\langle\langle\Sigma\rangle\rangle(\psi \vee \psi')$ can be translated as

$$\langle\langle A\rangle\rangle_\rho\psi \vee \langle\langle A\rangle\rangle_\rho\psi'$$

where A is any coalition. Notice that the path quantifiers in both disjuncts select the same single computation path since every agent in Σ uses the same strategy specified in ρ.

We finish the discussion on ATLES's expressivity by showing that ATLES can describe important game-theoretical concepts which make this logic more suitable for modelling rational behaviour of agents. In [242], it was argued that CATL is suited to express properties of games such as Nash equilibrium and Pareto efficiency. Where CATL seems to provide for reasoning about *strategic* games, ATLES extends this to *extensive* games. Extensive games can be represented by a tree structure whose leave nodes indicate the payoff for the players. Such game trees can be associated with ATSNs. Using explicit names for strategies, we can fix a strategy for each player. ATLES allows now to explicitly reason about properties of strategies. In particular, ATLES can describe weakly dominated and dominated strategies. Moreover, we can characterise Nash Equilibria, backward induction and Pareto optimal strategies using ATLES. For more details, we refer to Walther, van der Hoek and Wooldridge [266].

4.3. ATLES Model Checking

In this section, we discuss two model checking algorithms for ATLES. Generally, the *model checking problem* is, given a formula φ and a model S, to determine whether a state of S that satisfies φ, or to compute the set of states in S that satisfy φ. When model checking an ATLES-formula, we have to take the strategies into account that come with an ATSN. However, it appears to be also an interesting problem to consider the possibility that strategies are not given as part of the input; cf. the 'model checking as planning' paradigm [**92**]. With this in mind, we now formulate two variations of the model checking problem for ATLES:

(a) *Model checking with given strategies*
Given an ATLES-formula φ, an ATSN $\mathcal{S} = \langle \Pi, \Sigma, Q, \{\Upsilon_a\}_{a\in\Sigma}, \pi, \delta, \|\cdot\| \rangle$ where the set $\Sigma = \{1, \dots, n\}$ contains n agents, and a state $q \in Q$, is φ satisfied at q in $\mathcal{S}$?

(b) *Model checking along with generating strategies*
Given an ATLES-formula φ, an ATS $\mathcal{S} = \langle \Pi, \Sigma, Q, \pi, \delta \rangle$, and a state $q \in Q$, are there strategies, one for each strategy term that occurs in φ, such that φ is satisfied at q in $\mathcal{S}$ augmented with these strategies?

For deciding variant (a) of the problem, we use a modified version of the symbolic model checking algorithm for ATL from [**15**]; see Figure 4.4. Consider an ATSN

```
function ATLES-$eval(\psi, \mathcal{S} = \langle \Pi, \Sigma, Q, \{\Upsilon_a\}_{a\in\Sigma}, \pi, \delta, \|\cdot\| \rangle)$ returns the extension $[\![\psi]\!]$ in $\mathcal{S}$
    case $\psi = p$:                          return $\pi(p)$
    case $\psi = \neg\vartheta$:              return $Q \setminus$ ATLES-$eval(\vartheta, \mathcal{S})$
    case $\psi = \vartheta_1 \vee \vartheta_2$:   return ATLES-$eval(\vartheta_1, \mathcal{S}) \cup$ ATLES-$eval(\vartheta_2, \mathcal{S})$
    case $\psi = \langle\!\langle A \rangle\!\rangle_\xi \bigcirc \vartheta$:   return $Pre'(A, \xi,$ ATLES-$eval(\vartheta, \mathcal{S}))$
    case $\psi = \langle\!\langle A \rangle\!\rangle_\xi \Box \vartheta$:   $\Delta_1 := Q$; $\Delta_2 =$ ATLES-$eval(\vartheta, \mathcal{S})$
                                              while $\Delta_1 \not\subseteq \Delta_2$ do $\Delta_1 := \Delta_2$
                                                  $\Delta_2 := Pre'(A, \xi, \Delta_1) \cap$ ATLES-$eval(\vartheta, \mathcal{S})$ od
                                              return $\Delta_1$
    case $\psi = \langle\!\langle A \rangle\!\rangle_\xi \vartheta_1 \mathcal{U} \vartheta_2$: $\Delta_1 := \emptyset$; $\Delta_2 =$ ATLES-$eval(\vartheta_2, \mathcal{S})$
                                              while $\Delta_2 \not\subseteq \Delta_1$ do $\Delta_1 := \Delta_1 \cup \Delta_2$
                                                  $\Delta_2 := Pre'(A, \xi, \Delta_1) \cap$ ATLES-$eval(\vartheta_1, \mathcal{S})$ od
                                              return $\Delta_1$
end-function
```

FIGURE 4.4. ATLES symbolic model checking (variant (a)).

$\mathcal{S} = \langle \Pi, \Sigma, Q, \{\Upsilon_a\}_{a\in\Sigma}, \pi, \delta, \|\cdot\| \rangle$, a state q in $\mathcal{S}$, and an ATLES-formula φ as input. Take Σ to be the set containing all agents occurring in φ, and, for each agent $a \in \Sigma$, let Υ_a be the set containing all a-strategy terms of φ. Notice that, for any agent a of φ, if $a \notin \Sigma$ or $\varrho_a \notin \Upsilon_a$ for some a-strategy term ϱ_a of φ, the ATSN $\mathcal{S}$ cannot be a model for φ. We denote with Ξ_φ the set of all commitment functions occurring in φ.

The algorithm computes, using a bottom-up approach, for each subformula ψ of φ, its extension $[\![\psi]\!]$ in $\mathcal{S}$, a set of states from $\mathcal{S}$ that all satisfy ψ. For computing the extension of formulas of the form $\langle\!\langle A \rangle\!\rangle_\xi \bigcirc \psi$, $\langle\!\langle A \rangle\!\rangle_\xi \Box \psi$ or $\langle\!\langle A \rangle\!\rangle_\xi \psi \mathcal{U} \vartheta$, we employ a modified pre-image operator Pre' that additionally accounts for the commitments of agents to strategies as specified in ξ. The function $Pre'(\cdots)$ maps a coalition A, a commitment function ξ, and a set Q' of states to the set $Pre'(A, \xi, Q')$ containing states q at which the coalition $A \cup \mathsf{dom}(\xi)$ has an A-choice wrt. ξ (i.e., a collective choice of the agents in $A \cup \mathsf{dom}(\xi)$ where the agents in $\mathsf{dom}(\xi)$ choose according to the strategy specified in ξ) to ensure the next state to lie in Q'. That is, at q, the agents a in $A \setminus \mathsf{dom}(\xi)$ can select a choice in $\delta(q,a)$, and the agents b in $\mathsf{dom}(\xi)$ select the choice $\sigma_b(q) \in \delta(q,b)$ according to the b-strategy $\sigma_b = ||\xi(b)||$ such that, for all possible choices made by the other agents in $\Sigma \setminus (A \cup \mathsf{dom}(\xi))$, the resulting successor state is in Q'. Formally, for all $A \subseteq \Sigma$, all $\xi \in \Xi_\varphi$ and all $Q' \subseteq Q$:

$$\begin{aligned} Pre'(A,\xi,Q') \quad := \quad & \{q \in Q \mid \text{there is a choice } Q_C \in \delta(q, A \cup \mathsf{dom}(\xi)) \\ & \qquad \text{such that } Q_C \subseteq Q' \\ & \qquad \text{and } Q_C \subseteq \textstyle\bigcap_{a \in \mathsf{dom}(\xi)} ||\xi(a)||(q)\}. \end{aligned}$$

This modification of the operator *Pre* does not affect the complexity of the model checking algorithm. Hence, the variant (a) of ATLES model checking is no more complex than model checking ATL.

THEOREM 4.7. *The variant (a) of the model checking problem for ATLES is* PTIME-*complete, and can be solved in time $\mathcal{O}(m \cdot \ell)$ for an ATSN with m transitions and an ATLES-formula of length ℓ.*

An algorithm deciding variant (b) of the ATLES model checking problem needs to generate the strategies for the strategy terms occurring in the input formula. However, we can make use of the algorithm for variant (a) as follows: we first non-deterministically guess the required strategies with which we augment the model. In the second step, we use the polynomial time algorithm from (a) to model-check the input formula on the augmented model. For ATLES with historyless strategies, we obtain the following complexity result.

THEOREM 4.8. *The variant (b) of the model checking problem for ATLES with historyless strategies is* NP-*complete in the number of transitions of the given ATS and in the length of the input formula.*

PROOF. The upper bound can easily be seen. Let φ be an ATLES-formula of length ℓ and S an ATS for the agents occurring in φ. In the first step, we guess a historyless strategy for at most ℓ strategy terms occurring in φ. Notice that strategies without history are of polynomial size in the number of transitions in S: such a strategy specifies one choice at every state of S. Thus, guessing a historyless strategy can be done in polynomial time in the number of transitions in S. The second step is in PTIME according to Theorem 4.7. Consequently, we obtain an NP algorithm.

In order to show the lower complexity bound, we reduce the well-known NP-hard satisfiability problem for propositional logic to variant (b) of the model checking problem for ATLES. For NP-hardness, it is sufficient to consider propositional logic formulas φ in conjunctive normal form only [60]. That is, φ is a conjunction of the form $\psi_1 \wedge \cdots \wedge \psi_m$ where each ψ_i (for $i = 1..m$) is a disjunction of the form $\vartheta^i_1 \vee \cdots \vee \vartheta^i_{m_i}$. Each ϑ^i_j (for $i = 1..m$ and $j = 1..m_i$) is a literal, i.e. a propositional variable or its negation. Let $p_1, \ldots, p_n$ be an enumeration of the propositional variables occurring in φ. Reserve, for each ϑ^i_j with $1 \leq i \leq m$ and $1 \leq j \leq m_i$, a fresh propositional variable $p(i,j)$. We now define the ATS $\mathcal{S}_\varphi = \langle \Pi, \Sigma, Q, \pi, \delta \rangle$ with

- $\Pi = \{p(i,j) \mid 1 \leq i \leq m \text{ and } 1 \leq j \leq m_i\}$;
- $\Sigma = \{a\}$;
- $Q = \{q_0, q_1, q'_1, \ldots q_n, q'_n\}$;
- for all $p(i,j) \in \Pi$, $\pi(p(i,j)) = \{q_k \mid \vartheta^i_j = p_k\} \cup \{q'_k \mid \vartheta^i_j = \neg p_k\}$;
- $\delta(q_i, a) = \delta(q'_i, a) = \{\{q_{i+1}\}, \{q'_{i+1}\}\}$, for all i with $0 \leq i < n$, $\delta(q_n, a) = \{\{q_n\}\}$ and $\delta(q'_n, a) = \{\{q'_n\}\}$.

For an illustration of $\mathcal{S}_\varphi$, see Figure 4.5. The number of transitions in $\mathcal{S}_\varphi$ is polynomial in the length of φ. To see that, notice that $\mathcal{S}_\varphi$ contains not more than twice times the length of φ many states with, for each state, at most two a-choices. We define the

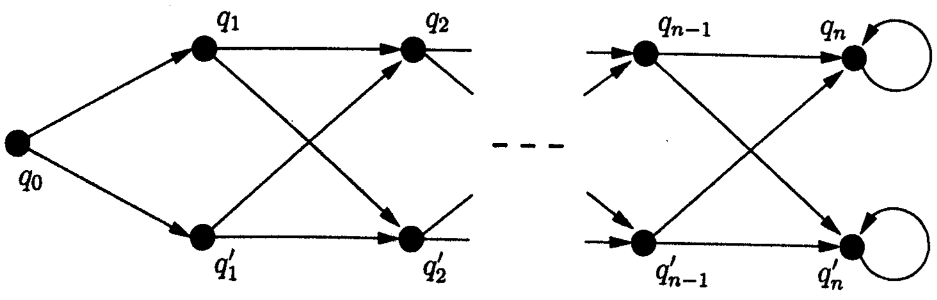


FIGURE 4.5. The ATS $\mathcal{S}_\varphi$ for one agent.

formula

$$\varphi_{\text{ATLES}} = \bigwedge_{i=1..m} \Big(\bigvee_{j=1..m_i} \langle\!\langle a \rangle\!\rangle_{\{a \mapsto \varrho_a\}} \Diamond p(i,j) \Big).$$

Notice that the length of φ_{ATLES} is polynomial in the length of φ. Moreover, observe that every path quantifier in φ_{ATLES} selects the same paths since agent a always chooses the strategy that corresponds to the strategy term ϱ_a.

The reduction works as follows: φ is satisfiable if, and only if, there is a strategy for agent a such that φ_{ATLES} is satisfied at q_0 in $\mathcal{S}_\varphi$ augmented with that strategy. To show the correctness of the reduction, we first show how to translate interpretations of propositional logic into strategies for a in $\mathcal{S}_\varphi$ and vice versa. If φ is satisfiable, there is an interpretation ι that assigns propositional variables p_i $(1 \leq i \leq n)$ in φ to truth values $\iota(p_i) \in \{0,1\}$. The notion of an interpretation can be lifted to formulas such that $\iota(\varphi) = 1$ if, and only if, φ is evaluated to true if its propositional variables are interpreted as specified in ι. Given an interpretation ι, we define an a-strategy $\sigma_a(\iota)$ in $\mathcal{S}_\varphi$ as follows: $\sigma_a(\iota)(q_n) = \{q_n\}$, $\sigma_a(\iota)(q'_n) = \{q'_n\}$, and for all i with $1 \leq i < n$,

$$\sigma_a(\iota)(q_{i-1}) = \sigma_a(\iota)(q'_{i-1}) = \begin{cases} \{q_i\} & \text{if } \iota(p_i) = 1 \\ \{q'_i\} & \text{if } \iota(p_i) = 0 \end{cases}.$$

For the reverse direction, we define, given an a-strategy σ_a in $\mathcal{S}_\varphi$, an interpretation $\iota(\sigma_a)$ as follows. We say a state $q \in Q$ is *σ_a-rooted* if there is a path $s_0, \ldots, s_k \in Q^*$ such that $s_0 = q_0$, $s_k = q$ and $\{s_i\} = \sigma_a(s_{i-1})$ for all i with $1 \leq i \leq k$. Then define, for all i with $1 \leq i \leq n$,

$$\iota(\sigma_a)(p_i) = \begin{cases} 1 & \text{if } q_i \text{ is } \sigma_a\text{-rooted} \\ 0 & \text{otherwise} \end{cases} .$$

We now show the correctness of the reduction. For the left-to-right direction, suppose φ is satisfiable. That is, there is an interpretation ι such that $\iota(\varphi) = 1$. This means that in all of φ's conjuncts ψ_i $(1 \leq i \leq m)$, there is a disjunct ϑ^i_j $(1 \leq j \leq m_i)$ such that $\iota(\vartheta^i_j) = 1$. It is readily verified that, by definition of $\mathcal{S}_\varphi$ and $\sigma_a(\iota)$, we have q_0 satisfies $\langle\!\langle a\rangle\!\rangle_{\{a \mapsto \varrho_a\}} \Diamond p(i,j)$ in the ATSN consisting of $\mathcal{S}_\varphi$ augmented with the strategy $\sigma_a(\iota)$ such that the denotation function $||\cdot||$ maps the strategy term ϱ_a to $||\varrho_a|| = \sigma_a(\iota)$. Hence, q_0 satisfies φ_{ATLES}. For the other direction from right to left, suppose there is an a-strategy σ_a such that φ_{ATLES} is satisfied at q_0 in the ATSN consisting of $\mathcal{S}_\varphi$ augmented with σ_a by setting $||\varrho_a|| = \sigma_a$. That is, for all i with $1 \leq i \leq m$, there is a j with $1 \leq j \leq m_i$ such that q_0 satisfies $\langle\!\langle a\rangle\!\rangle_{\{a \mapsto \varrho_a\}} \Diamond p(i,j)$. Then, by definition of $\iota(\sigma_a)$, we can easily see that $\iota(\sigma_a)(\vartheta^i_j) = 1$. This implies $\iota(\sigma_a)(\varphi) = 1$. Hence φ is satisfiable. □

4.4. Complexity of Reasoning in ATLES

This section investigates the computational complexity of the satisfiability problem for ATLES and completeness of the axiomatic system in Table 4.1. The complexity is settled at EXPTIME-complete and thus ATLES is no more complex than ATL. This is done by adapting the type elimination algorithm for ATL in Section 3.1.2 of Chapter 3; see also [**265**].

Since, in the definition of ATLES, we do not fix the number of agents in advance, some care is needed when formulating a satisfiability problem for ATLES. A similar consideration for ATL was done in Section 3.1.1 of Chapter 3. In particular, the range of semantic structures, over which a formula is to be interpreted, needs to be specified. Such a range can be determined by allowing for a certain number of agents to be present in the semantic structures. To see that allowing for different sets of agents to be present can influence satisfiability, take the following ATL-formula (adapted from [**195**, p.47]): $\neg\langle\!\langle a\rangle\!\rangle\bigcirc p \wedge \neg\langle\!\langle a\rangle\!\rangle\bigcirc q \wedge \langle\!\langle a\rangle\!\rangle\bigcirc(p \vee q)$. This formula expresses the fact that, in the next state, agent a cannot make p true, and cannot make q true; but it can make either p or q true. Now the question is whether this formula is satisfiable. The answer is that it is only satisfiable in a semantic structure for more than one agent, and not satisfiable with merely one available agent. Thus the number of agents present in a structure is important for determining satisfiability of a formula in this structure. With these concerns in mind, three variants of the satisfiability problem for ATL were suggested in Section 3.1.1 depending on the possibilities for the number of agents to occur in semantic structures. In this chapter, however, we concentrate only on one of these problems; the other two satisfiability problems can be reduced to it:

> *Satisfiability over formula-defined sets of agents:* Given an ATLES-formula φ, is φ satisfiable in a structure for exactly the agents which occur in φ?

The following theorem settles the complexity of the satisfiability problem for ATLES.

THEOREM 4.9. *The satisfiability problem for ATLES is* EXPTIME-*complete.*

The lower complexity bound carries over from the EXPTIME-hard fragment ATL; cf. Table 2.9 in Chapter 2 or see [**252, 265**]. For the upper bound, it is sufficient to show that ATLES satisfiability is in EXPTIME. The remainder of this section is devoted to presenting a decision procedure for ATLES that runs in exponential time and thus shows containment of ATLES satisfiability in EXPTIME. Correctness of the decision procedure follows from Lemma 4.17 below, while Lemma 4.18 establishes its exponential running time. The decision procedure implements an extended version of the type elimination construction for ATL from Section 3.1.2 in Chapter 3. The main issue that needs to be accounted for is that ATLES allows for commitment of agents to strategies explicitly in its syntax.

We begin our presentation of the type elimination decision procedure for ATLES with the definition of an extended closure that contains all formulas that are relevant for deciding the input formula. This and the following notion of types as specific subsets of the extended closure are adapted from Section 3.1.2 to additionally account for commitment functions.

DEFINITION 4.10 (Extended Closure for ATLES). Let ψ be an ATLES-formula and n the number of agents occurring in ψ. The *extended closure* $\mathsf{ecl}(\psi)$ *of* ψ is the smallest set which is closed under the following conditions:

- $\psi \in \mathsf{ecl}(\psi)$;
- $\mathsf{ecl}(\psi)$ is closed under subformulas;
- $\mathsf{ecl}(\psi)$ is closed under single negation;
- if $\langle\!\langle A\rangle\!\rangle_\rho\Box\varphi \in \mathsf{ecl}(\psi)$, then $\langle\!\langle A\rangle\!\rangle_\rho\bigcirc\langle\!\langle A\rangle\!\rangle_\rho\Box\varphi \in \mathsf{ecl}(\psi)$;
- if $\langle\!\langle A\rangle\!\rangle_\rho\varphi\,\mathcal{U}\,\vartheta \in \mathsf{ecl}(\psi)$, then $\langle\!\langle A\rangle\!\rangle_\rho\bigcirc\langle\!\langle A\rangle\!\rangle_\rho\varphi\,\mathcal{U}\,\vartheta \in \mathsf{ecl}(\psi)$.

Note that the cardinality of $\mathsf{ecl}(\psi)$ is linear in the length of ψ.

DEFINITION 4.11 (ATLES Type). Let ψ be an ATLES-formula. The set $\Psi \subseteq \mathsf{ecl}(\psi)$ is a *type for* ψ if the following conditions are satisfied:

(T1) $\varphi \vee \vartheta \in \Psi$ iff $\varphi \in \Psi$ or $\vartheta \in \Psi$, for all $\varphi \vee \vartheta \in \mathsf{ecl}(\psi)$;
(T2) $\varphi \in \Psi$ iff $\neg\varphi \notin \Psi$, for all $\neg\varphi \in \mathsf{ecl}(\psi)$;
(T3) $\langle\!\langle A\rangle\!\rangle_\rho\Box\varphi \in \Psi$ iff $\{\varphi, \langle\!\langle A\rangle\!\rangle_\rho\bigcirc\langle\!\langle A\rangle\!\rangle_\rho\Box\varphi\} \subseteq \Psi$, for all $\langle\!\langle A\rangle\!\rangle_\rho\Box\varphi \in \mathsf{ecl}(\psi)$;
(T4) $\langle\!\langle A\rangle\!\rangle_\rho\varphi\,\mathcal{U}\,\vartheta \in \Psi$ iff $\vartheta \in \Psi$ or $\{\varphi, \langle\!\langle A\rangle\!\rangle_\rho\bigcirc\langle\!\langle A\rangle\!\rangle_\rho\varphi\,\mathcal{U}\,\vartheta\} \subseteq \Psi$, for all $\langle\!\langle A\rangle\!\rangle_\rho\varphi\,\mathcal{U}\,\vartheta \in \mathsf{ecl}(\psi)$.

The set of all ATLES types for ψ is designated by Γ_ψ.

Note that the cardinality of Γ_ψ is exponential in the length of ψ.

Intuitively, an ATLES type describes a possible state q of a transition system using formulas from $\mathsf{ecl}(\psi)$ which are true at q. The decision procedure will use all ATLES types for an input formula to determine its decidability. The conditions (T1) to (T4) state restrictions on the formulas in a type.

Before proceeding with the construction, we introduce some auxiliary notions. As for ATL in Section 3.1.2, we use Σ_ψ to denote the set of all agents that occur in the formula ψ. Moreover, we distinguish between positive and negative next formulas of ATLES and assume that all next formulas in the extended closure $\mathsf{ecl}(\psi)$ are linearly ordered in a way that no negative next formula occurs before a positive one. We denote with $\sharp_\varphi$ the number of the next formula $\varphi \in \mathsf{ecl}(\psi)$ in this ordering; the numbering starts with 0. Since there are as many positive next formulas in $\mathsf{ecl}(\psi)$ as negative ones, we obtain the following enumeration $\varphi_0, \dots, \varphi_{k-1}$ with $\varphi_0, \dots, \varphi_{k/2-1}$ being positive next formulas and $\varphi_{k/2}, \dots, \varphi_{k-1}$ negative next formulas. Additionally, we assume that all strategy terms occurring in ψ are linearly ordered as well. Let ℓ be the number of strategy terms in ψ. Enumerate the strategy terms by $\varrho_1, \dots, \varrho_\ell$ and denote with $\sharp_{\varrho_i}$ the number $-i$, i.e., the negated number of ϱ_i's position in this ordering.

Intuitively, the type elimination algorithm eliminates those types that are not relevant for building an ATSN $\mathcal{S}$ in which the input formula ψ is satisfied at some state. The states of such an $\mathcal{S}$ are thought of as consisting of sequences of *n-tuples* where $n = |\Sigma_\psi|$. Each component of such an n-tuple corresponds to an agent and takes values from $-\ell$ to $k-1$ which in turn refer to one of the ℓ strategy terms or one of the k next formulas in $\mathsf{ecl}(\psi)$:

$$\underbrace{-\ell, \dots, -1,}_{\textbf{strategy terms}} \quad \underbrace{0, \dots, \frac{k}{2} - 1,}_{\textbf{positive next formulas}} \quad \underbrace{\frac{k}{2} + 1, \dots, k-1}_{\textbf{negative next formulas}} .$$

The set of all such n-tuples is denoted with $[(-\ell, k)/n]$, and the states of $\mathcal{S}$ will be sequences of n-tuples from $[(-\ell, k)/n]^*$. Supposing $q \in [(-\ell, k)/n]^*$ is a state of $\mathcal{S}$, then the set $\{q \cdot \vec{t} \mid \vec{t} \in [(-\ell, k)/n]\}$ contains its potential successor states, thus the choices in $\delta(q, a)$ are subsets of this set. Each state q of $\mathcal{S}$ corresponds to some type that was not eliminated and the construction of $\mathcal{S}$ is aiming at satisfying every formula from this type at q. In particular, q will have to satisfy a number of next formulas, i.e., the definition of δ that specifies the choices available at q is crucial. The transition function δ is defined as follows: For all agents a (ranging from 0 to $n-1$), define $\delta(q, a)$ to be the set of the choices

$$\{q \cdot \vec{t} \in [(-\ell, k)/n]^* \mid \vec{t} = (t_0, \dots, t_{n-1}) \text{ and } t_a = p\}$$

where p ranges over $\{0, \dots, k/2 - 1\}$ or $p = \sharp_\varrho$ for some a-strategy term ϱ that occurs in ψ. The idea here is that agent a makes an a-choice by "voting" for a positive next formula that she wants to be satisfied or for one of her strategies she may be committed to follow. Such votes are done via a's position t_a in tuples $\vec{t} = (t_0, \ldots, t_{n-1})$. Notice that this is implemented by the condition on the range of p: agent a can vote for one of the $k/2$ many positive next formulas by choosing a value from 0 to $k/2 - 1$, or a can vote for one of her strategies, say ϱ_a, by selecting the value $\sharp_{\varrho_a}$. For a coalition A and a commitment function ρ, an A-choice wrt. ρ is made by a collective vote of the entire set of agents $A \cup \mathsf{dom}(\rho)$ while accounting for the commitments of agents in $\mathsf{dom}(\rho)$ to certain strategies. We characterise A-choices wrt. ρ as follows: A subset $S_{\langle A,\rho\rangle} \subseteq [(-\ell, k)/n]$ of vectors is called an *A-voting set under commitment* ρ if there exists a mapping $\tau_\rho : A \to \{-\ell, \ldots, k/2 - 1\}$ with

- $\tau_\rho(a) = \sharp_{\rho(a)}$, for all $a \in \mathsf{dom}(\rho)$, and
- $\tau_\rho(a) \in \{0, \ldots, k/2 - 1\}$, for all $a \in A \setminus \mathsf{dom}(\rho)$

such that

$$S_{\langle A,\rho\rangle} = \{\vec{t} = (t_0, \ldots, t_{n-1}) \mid t_a = \tau_\rho(a), \text{ for all } a \in A\}.$$

Then we have that the choices in $\delta(q, A)$ are exactly the sets $\{q \cdot \vec{t} \mid \vec{t} \in S_{\langle A,\rho\rangle}\}$ where $S_{\langle A,\rho\rangle}$ ranges over the A-voting sets under any commitment ρ.

In order for q to satisfy a positive next formula $\langle\!\langle A\rangle\!\rangle_\rho \bigcirc \varphi$, the construction of $\mathcal{S}$ ensures that φ is satisfied at all states of an A-choice wrt. ρ. This A-choice corresponds to an A-voting set under commitment ρ where the agents in $A \setminus \mathsf{dom}(\rho)$ without commitment vote solely for the next formula $\langle\!\langle A\rangle\!\rangle_\rho \bigcirc \varphi$ and the other agents in $\mathsf{dom}(\rho)$ respectively vote for the strategies they committed to as described in ρ. Note that, if all agents in A have committed to a strategy in ρ, then there is exactly one A-voting set under commitment ρ. Moreover, any A-voting set under commitment ρ is also a B-voting set under ρ and vice versa, for any B with $(A \setminus \mathsf{dom}(\rho)) \subseteq B \subseteq (A \cup \mathsf{dom}(\rho))$.

Satisfying negative next formulas $\neg\langle\!\langle A\rangle\!\rangle_\rho \bigcirc \varphi$ at a state q is straightforward if $A \cup \mathsf{dom}(\rho) = \Sigma_\psi$: such next formulas merely state that *all* successors of q that can be reached when every committed agent a in $\mathsf{dom}(\rho)$ follows her strategy $\rho(a)$ and every other agent in $A \setminus \mathsf{dom}(\rho)$ freely makes a choice, satisfy $\neg\varphi$. For satisfying $\neg\langle\!\langle A\rangle\!\rangle_\rho \bigcirc \varphi$ with $A \cup \mathsf{dom}(\rho) \subset \Sigma_\psi$ at q, the subformula φ needs to be refuted at a state in every A-choice wrt. ρ. To this end, we pick a state from every such A-choice at which we falsify φ. Note that, by picking a state from every A-choice wrt. ρ, a state from every A'-choice wrt. ρ' is picked as well, where A' and ρ' satisfy

- $A' \cup \mathsf{dom}(\rho') \subseteq A \cup \mathsf{dom}(\rho)$, and
- $\rho'(a) = \rho(a)$, for all $a \in \mathsf{dom}(\rho')$.

But this is fine as $\neg\langle\!\langle A\rangle\!\rangle_\rho \bigcirc \varphi$ implies $\neg\langle\!\langle A'\rangle\!\rangle_{\rho'} \bigcirc \varphi$ under those restrictions. However, $\neg\langle\!\langle A\rangle\!\rangle_\rho \bigcirc \varphi$ does not imply any formula $\neg\langle\!\langle B\rangle\!\rangle_\xi \bigcirc \varphi$, where B and ξ are such that $B \cup \mathsf{dom}(\xi) \not\subseteq A \cup \mathsf{dom}(\rho)$ or $\xi(a) \neq \rho(a)$ for some $a \in \mathsf{dom}(\xi)$. Thus, in order not to automatically satisfy another negative next formula of the form $\neg\langle\!\langle B\rangle\!\rangle_\xi \bigcirc \varphi$ when picking elements in A-choices wrt. ρ, some care is needed such that the picked elements from such A-choices are not also elements of B-choices wrt. ξ, for any such B and ξ. Picking the right states that refute φ is implemented by a "refutation function". The following definition refines the refutation function in Definition 3.6 that was used for the ATL decision procedure in Section 3.1.2.

DEFINITION 4.12 (ATLES Refutation Function). Let ψ be an ATLES-formula, $n = |\Sigma_\psi|$, k the number of next formulas in $\mathsf{ecl}(\psi)$, and ℓ the number of strategy terms in ψ. Define a partial function

$$f : [(-\ell, k)/n] \times 2^{\Sigma_\psi} \rightarrow \{k/2, \dots, k-1\}$$

mapping vectors and coalitions of agents to numbers of negative next formulas: for each set of agents $B \subset \Sigma_\psi$, fix an agent $a_B \in \Sigma_\psi \setminus B$. Then set, for all $\vec{t} = (t_0, \dots, t_{n-1}) \in [(-\ell, k)/n]$ and $B \subseteq \Sigma_\psi$,

$$f(\vec{t}, B) := \begin{cases} \sharp_{\neg\langle\langle A\rangle\rangle_\rho \bigcirc\varphi} & \text{if } t_{a_B} = \sharp_{\neg\langle\langle A\rangle\rangle_\rho \bigcirc\varphi} \text{ with } A \cup \mathsf{dom}(\rho) = B \text{ and} \\ & \text{for all } a \in \Sigma_\psi,\ -\ell \leq t_a < k/2 \text{ iff } a \in B \\ \text{undefined} & \text{if there is no such } \neg\langle\langle A\rangle\rangle_\rho \bigcirc\varphi. \end{cases}$$

Intuitively, the role of the refutation function is explained as follows: Consider a commitment function ρ and a vector $\vec{t} = (t_0, \dots, t_{n-1})$ in which the agents comply with their commitment in ρ, i.e., $t_a = \sharp_{\rho(a)}$ for all $a \in \mathsf{dom}(\rho)$. Then $f(\vec{t}, B) = \sharp_{\neg\langle\langle A\rangle\rangle_\rho \bigcirc\varphi}$ with $B = A \cup \mathsf{dom}(\rho)$ means that, for every state q satisfying $\neg\langle\langle A\rangle\rangle_\rho \bigcirc\varphi$, the successor $q \cdot \vec{t}$ has to refute φ. Note that there may be more than one successor of q refuting φ: at least one element of each A-choice wrt. ρ. Formally, we need the following three properties of the refutation function:

(1) For all negative next formulas $\neg\langle\langle A\rangle\rangle_\rho \bigcirc\varphi \in \mathsf{ecl}(\psi)$ with $A \cup \mathsf{dom}(\rho) \subset \Sigma_\psi$ and all A-voting sets $S_{\langle A,\rho\rangle}$ under commitment ρ, there is a vector $\vec{t} \in S_{\langle A,\rho\rangle}$ such that $f(\vec{t}, A \cup \mathsf{dom}(\rho)) = \sharp_{\neg\langle\langle A\rangle\rangle_\rho \bigcirc\varphi}$. Recall that such A-voting sets correspond to A-choices wrt. ρ.
(2) For all vectors $\vec{t} = (t_0, \dots, t_{n-1}) \in [(-\ell, k)/n]$, we have that $f(\vec{t}, A \cup \mathsf{dom}(\rho)) = \sharp_{\neg\langle\langle A\rangle\rangle_\rho \bigcirc\varphi}$ implies $t_a \geq k/2$ for all $a \in \Sigma_\psi \setminus (A \cup \mathsf{dom}(\rho))$.
(3) For all vectors $\vec{t} \in [(-\ell, k)/n]$, there is at most a single set $B \subseteq \Sigma_\psi$ with $f(\vec{t}, B)$ defined.

It is readily verified that the refutation function f from Definition 4.12 indeed satisfies these properties.

The construction of a model for the input formula is more involved due to the presence of negated box formulas and until formulas which we call *eventualities*. This is because the satisfaction of eventualities needs to be witnessed by some state in a model and such witnesses can be "far away". We deal with eventualities as in the decision procedure for ATL in Section 3.1.2 with the help of so-called witness trees. For defining witness trees, the following three auxiliary notions are needed.

DEFINITION 4.13 (ϑ-vector, φ-tree, $\bigcirc$-matching). Let ψ be an ATLES-formula, $n = |\Sigma_\psi|$, k the number of next-formulas in $\mathsf{ecl}(\psi)$, and ℓ the number of strategy terms in ψ. For all next-formulas $\vartheta \in \mathsf{ecl}(\psi)$ with commitment function ρ parameterizing the outermost path quantifier of ϑ and all vectors $\vec{t} = (t_0, \ldots, t_{n-1}) \in [(-\ell, k)/n]$ with $t_a = \sharp_{\rho(a)}$ for each $a \in \mathsf{dom}(\rho)$,

- if $\vartheta = \langle\!\langle A \rangle\!\rangle_\rho \bigcirc \varphi$ and $t_a = \sharp_\vartheta$ for each $a \in A \backslash \mathsf{dom}(\rho)$, then $\vec{t}$ is called a *ϑ-vector*;
- if $\vartheta = \neg\langle\!\langle A \rangle\!\rangle_\rho \bigcirc \psi$ and $A \cup \mathsf{dom}(\rho) \subset \Sigma_\psi$, then $\vec{t}$ is called a *ϑ-vector* if $f(\vec{t}, A \cup \mathsf{dom}(\rho)) = \sharp_\vartheta$;
- if $\vartheta = \neg\langle\!\langle A \rangle\!\rangle_\rho \bigcirc \psi$ and $A \cup \mathsf{dom}(\rho) = \Sigma_\psi$, then $\vec{t}$ is called a *ϑ-vector.*

For all types $\Psi \in \Gamma_\psi$ and all vectors $\vec{t} = (t_0, \ldots, t_{n-1}) \in [(-\ell, k)/n]$, let $S_\Psi(\vec{t}) \subseteq \mathsf{ecl}(\psi)$ be the smallest set such that

(M1) if $\langle\!\langle A \rangle\!\rangle_\rho \bigcirc \varphi \in \Psi$ and $\vec{t}$ is a $\langle\!\langle A \rangle\!\rangle_\rho \bigcirc \varphi$-vector, then $\varphi \in S_\Psi(\vec{t})$, and
(M2) if $\neg\langle\!\langle A \rangle\!\rangle_\rho \bigcirc \varphi \in \Psi$ and $\vec{t}$ is a $\neg\langle\!\langle A \rangle\!\rangle_\rho \bigcirc \varphi$-vector, then $\neg\varphi \in S_\Psi(\vec{t})$.

Given a set M of labels, a *$\langle M, [(-\ell, k), n]\rangle$-tree* T is a mapping T from a finite prefix-closed subset of $[(-\ell, k)/n]^*$ to M. A *ψ-tree* is a $\langle \Gamma_\psi, [(-\ell, k)/n]\rangle$-tree in which every node is labelled with a type for ψ. A ψ-tree T is called *$\bigcirc$-matching* if, for all nodes α of T and all vectors $\vec{t} \in [(-\ell, k)/n]$, $\alpha \cdot \vec{t} \in \mathsf{dom}(T)$ implies $S_{T(\alpha)}(\vec{t}) \subseteq T(\alpha \cdot \vec{t})$.

Intuitively, a vector $\vec{t}$ is a $\langle\!\langle A \rangle\!\rangle_\rho \bigcirc \varphi$-vector if, for all nodes q satisfying $\langle\!\langle A \rangle\!\rangle_\rho \bigcirc \varphi$, the successor $q \cdot \vec{t}$ has to satisfy φ. The $\neg\langle\!\langle A \rangle\!\rangle_\rho \bigcirc \varphi$-vectors can be understood in an analogous way. This intuition is reflected in the conditions (M1) and (M2) and in the notion of $\bigcirc$-matching.

Equipped with the notions ϑ-vector, φ-tree, and $\bigcirc$-matching, we can now define witness trees and witness paths in order to testify the satisfaction of eventualities.

DEFINITION 4.14 (ATLES Witness Tree). Let ψ be an ATLES-formula, Γ a set of types for ψ, and $\Psi \in \Gamma$. A ψ-tree T is called a *witness-tree rooted at Ψ in Γ for a formula $\langle\!\langle A \rangle\!\rangle_\rho \varphi \mathcal{U} \vartheta$* if it satisfies the following properties:

(1) $T(\alpha) \in \Gamma$, for all nodes $\alpha \in \mathsf{dom}(T)$;
(2) T is $\bigcirc$-matching;
(3) $T(\varepsilon) = \Psi$;
(4) $\langle\!\langle A \rangle\!\rangle_\rho \varphi \mathcal{U} \vartheta \in T(\alpha)$, for all nodes $\alpha \in \mathsf{dom}(T)$;
(5) $\varphi \in T(\alpha)$, for all inner nodes $\alpha \in \mathsf{dom}(T)$;
(6) $\vartheta \in T(\alpha)$, for all leaf nodes $\alpha \in \mathsf{dom}(T)$;
(7) if $\alpha \in \mathsf{dom}(T)$, $\langle\!\langle A \rangle\!\rangle_\rho \bigcirc \langle\!\langle A \rangle\!\rangle_\rho \varphi \mathcal{U} \vartheta \in T(\alpha)$, $\vartheta \notin T(\alpha)$, and $\vec{t}$ is a $\langle\!\langle A \rangle\!\rangle_\rho \bigcirc \langle\!\langle A \rangle\!\rangle_\rho \varphi \mathcal{U} \vartheta$-vector, then $\alpha \cdot \vec{t} \in \mathsf{dom}(T)$.

T is called a *witness-tree rooted at Ψ in Γ for a formula $\neg\langle\!\langle A \rangle\!\rangle_\rho \Box \varphi$* if it satisfies the following properties:

(1) $T(\alpha) \in \Gamma$, for all nodes $\alpha \in \mathsf{dom}(T)$;
(2) T is $\bigcirc$-matching;
(3) $T(\varepsilon) = \Psi$;
(4) $\neg\langle\!\langle A \rangle\!\rangle_\rho \Box \varphi \in T(\alpha)$, for all nodes $\alpha \in \mathsf{dom}(T)$;
(5) $\neg\varphi \in T(\alpha)$, for all leaf nodes $\alpha \in \mathsf{dom}(T)$;

(6) if $\alpha \in \mathsf{dom}(T)$, $\neg\langle\!\langle A\rangle\!\rangle_\rho \bigcirc \langle\!\langle A\rangle\!\rangle_\rho \Box\varphi \in T(\alpha)$, $\neg\varphi \notin T(\alpha)$, and $\vec{t}$ is a $\neg\langle\!\langle A\rangle\!\rangle_\rho \bigcirc \langle\!\langle A\rangle\!\rangle_\rho \Box\varphi$-vector, then $\alpha \cdot \vec{t} \in \mathsf{dom}(T)$.

The presented decision procedure for ATLES employs a type elimination algorithm which relies essentially on the notion of *realizability*. Intuitively, a type Ψ is realizable in a set of types Γ if it is possible to satisfy all next-formulas in Ψ and to construct witness trees for all eventualities using only types from Γ.

DEFINITION 4.15 (ATLES Realizability). Let ψ be an ATLES-formula and Γ a set of types for ψ. A type $\Psi \in \Gamma$ is *ATLES realizable in* Γ if the following conditions are satisfied:

1. for all $\vec{t} \in [(-\ell, k)/n]$, there is a $\Psi' \in \Gamma$ such that $S_\Psi(\vec{t}) \subseteq \Psi'$;
2. for all $\langle\!\langle A\rangle\!\rangle_\rho \varphi\,\mathcal{U}\,\vartheta \in \Psi$, there is a $\langle\!\langle A\rangle\!\rangle_\rho \varphi\,\mathcal{U}\,\vartheta$-witness tree rooted at Ψ in Γ;
3. for all $\neg\langle\!\langle A\rangle\!\rangle_\rho \Box\varphi \in \Psi$, there is a $\neg\langle\!\langle A\rangle\!\rangle_\rho \Box\varphi$-witness tree rooted at Ψ in Γ.

```
1.  function ATLES–sat($\varphi$) returns 'Yes, ...' or 'No, ...'
2.     $m := 0$
3.     $\Delta_m := \Gamma_\varphi$
4.     do
5.        $m := m + 1$
6.        $\Delta_m := \{\Psi \in \Delta_{m-1} \mid \Psi$ is ATLES realizable in $\Delta_{m-1}\}$
7.     until $\Delta_m = \Delta_{m-1}$
8.     if $\varphi \in \Psi$, for some $\Psi \in \Delta_m$,
9.        then return 'Yes, $\varphi$ is satisfiable in an ATSN for $\Sigma_\varphi$.'
10.       else return 'No, $\varphi$ is not satisfiable.'
11. end-function
```

FIGURE 4.6. A type-elimination algorithm for ATLES.

The decision procedure for ATLES is presented as function ATLES–*sat*(φ) in Figure 4.6. The type-elimination algorithm starts with the set of all types Γ_φ for an input formula φ (line 2 and 3) and then repeatedly eliminates in a do–until loop the types that are not ATLES realizable (lines 4–7). Since there are only finitely many types to start with, the algorithm eventually leaves the loop with a set Δ_m of types, for some $m \geq 0$ (line 7). Notice that, at this point, Δ_m is a set of types for φ that are all ATLES realizable in Δ_m. The algorithm returns 'Yes, φ is satisfiable in an ATSN for Σ_φ' if the input formula φ is contained in some type of Δ_m (line 8 and 9); otherwise it returns 'No, φ is not satisfiable.' (line 8 and 10).

As for the decision procedure of ATL in Section 3.1.2, we use three lemmas to state that the described decision procedure for ATLES is correct and can be computed within exponential time in the length of the input formula. First, we ascertain that the procedure is effective by showing that the existence of the witness trees is decidable in exponential time.

LEMMA 4.16. *Let Γ be a set of types for an ATLES-formula ψ. Then the existence of witness trees in Γ can be decided in time exponential in the length of ψ.*

This Lemma corresponds to Lemma 3.10 about deciding the existence of witness trees for ATL in Section 3.1.2, and it can be proved in the same way.

Second, we determine soundness and completeness of the procedure.

LEMMA 4.17. *Let ψ be an ATLES-formula. Then the procedure returns 'Yes, the input formula ψ is satisfiable in an ATSN for Σ_ψ.' iff this is indeed the case.*

The proof of Lemma 4.17 is presented in the following section. It extends the proof of the corresponding Lemma 3.11 for ATL to ATLES. The difference between ATLES and ATL is the explicit commitment of agents to strategy terms in the syntax. To account for such commitments, every agent must be able to vote for her preferred strategy or for the strategy she has committed to. When showing soundness of the procedure, we construct a tree-model for the input formula ψ from the types that survived the elimination procedure. To enable agents to vote for the strategies they committed to, we appropriately extend the out-degree of this tree-model according to the number of strategy terms that occur in ψ. For showing completeness, we take an arbitrary model for ψ and prove that each type that corresponds to some states of this model survives the elimination procedure. In particular, we show that we can construct witness trees for the eventualities in the types by unraveling the model.

Finally, we establish exponential running time of the procedure.

LEMMA 4.18. *The described type elimination procedure for ATLES runs in exponential time.*

This lemma can be proved in the same way as Lemma 3.12, which shows that the decision procedure for ATL runs in exponential time.

4.4.1. Proof for ATLES Decision Procedure.

LEMMA 4.16. *Let φ be an ATLES-formula. Then the procedure returns 'Yes, the input formula φ is satisfiable in an ATSN for Σ_φ.' iff this is indeed the case.*

PROOF. Suppose φ is given. Let $\Sigma = \Sigma_\varphi$ and $n = |\Sigma_\varphi|$. Let k be the number of next formulas in $\mathsf{ecl}(\varphi)$, and ℓ the number of strategy terms in φ.

"$\Rightarrow$" (Soundness) Assume that the elimination procedure was started on input φ and returns 'Yes, the input formula φ is satisfiable'. Let $\Gamma = \{\Psi_0, \dots, \Psi_{m-1}\}$ be the computed set of types. Then all types of Γ are ATLES realizable in Γ and there is a type $\Psi \in \Gamma$ with $\varphi \in \Psi$. Our aim is to construct an ATSN that is a model of φ.

To this end, enumerate all eventualities in $\mathsf{ecl}(\varphi)$ by $\psi_0, \dots, \psi_{x-1}$. For each i with $i < x$ and each j with $j < m$, fix a φ-tree $T_{\langle\psi_i,\Psi_j\rangle}$ as follows:

- If $\psi_i \in \Psi_j$, then fix a ψ_i-witness tree T rooted at Ψ_j in Γ. Supplement all inner nodes of T with missing successors: for each inner node $\alpha \in \mathsf{dom}(T)$ and each $\vec{t} \in [(-\ell, k)/n]$, if $\alpha \cdot \vec{t} \notin \mathsf{dom}(T)$, then add it and set $T(\alpha \cdot \vec{t}) = \Psi$ for some

$\Psi \in \Gamma$ such that $S_{T(\alpha)}(\vec{t}) \subseteq \Psi$. Note that such a Ψ must exist by condition 1 of Definition 4.15. Let $T_{\langle\psi_i,\Psi_j\rangle}$ be the result of augmenting T in this way.

- If $\psi_i \notin \Psi_j$, then let $T_{\langle\psi_i,\Psi_j\rangle}$ be the tree comprised of the nodes $\{\varepsilon\} \cup [(-\ell,k)/n]$ such that $T_{\langle\psi_i,\Psi_j\rangle}(\varepsilon) = \Psi_j$ and, for each $\vec{t} \in [(-\ell,k)/n]$, $T_{\langle\psi_i,\Psi_j\rangle}(\vec{t}) = \Psi$ for some $\Psi \in \Gamma$ with $S_{\Psi_j}(\vec{t}) \subseteq \Psi$.

It is easy to see that all trees $T_{\langle\psi_i,\Psi_j\rangle}$ are $\bigcirc$-matching. To construct a model of φ, intuitively we do the following: we arrange the selected witness trees $T_{\langle\psi_i,\Psi_j\rangle}$ in an $x \times m$-matrix such that the rows range over the eventualities $\psi_0, \ldots, \psi_{x-1}$ and the columns over the types $\Psi_0, \ldots, \Psi_{m-1}$, and then we replace all leaf nodes by an 'arrow' from the leaf node's predecessor to the root of some other witness tree.

We now define the ATSN $\mathcal{S} = \langle \Pi, \Sigma, Q, \Upsilon_1, \ldots, \Upsilon_n, \pi, \delta, ||\cdot||\rangle$ that we then prove to be a model of φ. The sets Π and Σ contain exactly those propositions and agents that occur in the input formula φ. Moreover, $\Upsilon_1, \ldots, \Upsilon_n$ are the sets of strategy terms occurring in φ such that Υ_a contains the strategy terms for agent a, for all $a \in \Sigma$. For defining the set of states Q, fix symbols $\varepsilon_{i,j}$ with $i < x$ and $j < m$. Then set:

$$Q \;:=\; \{\varepsilon_{i,j}w \mid w \in \mathsf{dom}(T_{\langle\psi_i,\Psi_j\rangle}) \text{ is inner node}\}.$$

Next, the valuation π is easily defined: for $q = \varepsilon_{i,j}w \in Q$, set

$$\pi(q) \;:=\; T_{\langle\psi_i,\Psi_j\rangle}(w) \cap \Pi.$$

To define the transition function δ, we first define a successor function on Q: for all $q = \varepsilon_{i,j}w \in Q$ and $\vec{t} \in [(-\ell,k)/n]$, set

$$s_{\vec{t}}(q) \;:=\; \begin{cases} \varepsilon_{u,v} & \text{if } w \cdot \vec{t} \text{ is a leaf node of } T_{\langle\psi_i,\Psi_j\rangle}, \\ & u = i+1 \bmod x \text{ and } T_{\langle\psi_i,\Psi_j\rangle}(w \cdot \vec{t}) = \Psi_v; \\ q \cdot t & \text{if } w \cdot \vec{t} \text{ is an inner node of } T_{\langle\psi_i,\Psi_j\rangle}. \end{cases}$$

Now the definition of δ is straightforward: for all $q \in Q$ and $a \in \Sigma$, set $\delta(q,a)$ to be the set of the sets

$$\{s_{\vec{t}}(q) \mid \vec{t} = (t_0, \ldots, t_{n-1}) \in [(-\ell,k)/n] \text{ and } t_a = p\}$$

where $p \in \{0, \ldots, k/2-1\}$ or $p = \sharp_{\varrho_a}$, for some strategy term $\varrho_a \in \Upsilon_a$. To finish the definition of $\mathcal{S}$, it remains to define the denotation function $||\cdot||$. To this end, we first define, for all $a \in \Sigma$ and $\varrho \in \Upsilon_a$, a strategy σ_ϱ: for all $\lambda \in Q^*$ and $q \in Q$,

$$\sigma_\varrho(\lambda \cdot q) = \{s_{\vec{t}}(q) \mid \vec{t} = (t_0, \ldots, t_{n-1}) \text{ and } t_a = \sharp_\varrho\}.$$

Then $||\cdot||$ is defined as follows: for all $a \in \Sigma$ and $\varrho \in \Upsilon_a$,

$$||\varrho|| \;:=\; \sigma_\varrho.$$

To show that $\mathcal{S}$ is indeed a model of φ, we introduce some auxiliary notions:

For each strategy $\sigma_A = \{\sigma_a \mid a \in A\}$ for a set of agents $A \subseteq \Sigma$ and each sequence of states $\lambda \in Q^+$, we write $\sigma_A(\lambda)$ to denote the set of states $\bigcap_{a \in A} \sigma_a(\lambda)$. Observe that, by definition of strategies for single agents, we have $\sigma_A(\lambda \cdot q) \in \delta(q, A)$ for all $\lambda \in Q^+$ and $q \in Q$.

For every positive next-formula $\langle\langle A\rangle\rangle_\rho \bigcirc\psi \in \mathsf{ecl}(\varphi)$, the *$\langle\langle A\rangle\rangle_\rho \bigcirc\psi$-strategy* is the ρ-strategy $\sigma_{A\cup\mathsf{dom}(\rho)} = \{\sigma_a \mid a \in A \cup \mathsf{dom}(\rho)\}$ for the agents in $A \cup \mathsf{dom}(\rho)$ that is defined by setting, for all $a \in A \cup \mathsf{dom}(\rho)$, all $\lambda \in Q^*$ and $q \in Q$,

$$\sigma_a(\lambda \cdot q) := \begin{cases} ||\rho(a)||(\lambda \cdot q) & \text{if } a \in \mathsf{dom}(\rho) \\ \{s_{\vec{t}}(q) \mid \vec{t} = (t_0, \ldots, t_{n-1}) \text{ and } t_a = \sharp_{\langle\langle A\rangle\rangle_\rho \bigcirc\psi}\} & \text{if } a \in A \setminus \mathsf{dom}(\rho). \end{cases}$$

It is readily checked that we have

$$\sigma_{A\cup\mathsf{dom}(\rho)}(\lambda \cdot q) = \sigma_{A\cup\mathsf{dom}(\rho)}(q) = \bigcap_{\sigma_a \in \sigma_{A\cup\mathsf{dom}(\rho)}} \sigma_a(q)$$

and that

$$(*) \qquad \sigma_{A\cup\mathsf{dom}(\rho)}(q) = \{s_{\vec{t}}(q) \mid \vec{t} \in [(-\ell, k)/n] \text{ is a } \langle\langle A\rangle\rangle_\rho \bigcirc\psi\text{-vector}\}.$$

For every negative next-formula $\neg\langle\langle A\rangle\rangle_\rho \bigcirc\psi \in \mathsf{ecl}(\varphi)$, a *$\neg\langle\langle A\rangle\rangle_\rho \bigcirc\psi$-computation for a ρ-strategy $\sigma_{A\cup\mathsf{dom}(\rho)}$ rooted at a state $q \in Q$* is a computation $\lambda \in \mathsf{out}(q, \sigma_{A\cup\mathsf{dom}(\rho)})$ such that, for all positions $i \geq 0$, we have $\lambda[i+1] = s_{\vec{t}}(\lambda[i])$ for some $\neg\langle\langle A\rangle\rangle_\rho \bigcirc\psi$-vector $\vec{t} \in [(-\ell, k)/n]$.

CLAIM 4.17. *Let $\neg\langle\langle A\rangle\rangle_\rho \bigcirc\psi$ be a next-formula in $ecl(\varphi)$, $\sigma_{A\cup\mathsf{dom}(\rho)}$ a ρ-strategy, and $q \in Q$. Then there exists a $\neg\langle\langle A\rangle\rangle_\rho \bigcirc\psi$-computation for $\sigma_{A\cup\mathsf{dom}(\rho)}$ rooted at q.*

PROOF OF CLAIM Let $\neg\langle\langle A\rangle\rangle_\rho \bigcirc\psi$, $\sigma_{A\cup\mathsf{dom}(\rho)}$ and q be as in the claim. Inductively define a $\neg\langle\langle A\rangle\rangle_\rho \bigcirc\psi$-computation $\lambda \in Q^\omega$ for $\sigma_{A\cup\mathsf{dom}(\rho)}$ rooted at q as follows:

- $\lambda[0] := q$, and
- for each $i \geq 0$, $\lambda[i+1] := s_{\vec{t}}(\lambda[i])$ for some $\neg\langle\langle A\rangle\rangle_\rho \bigcirc\psi$-vector $\vec{t} \in [(-\ell, k)/n]$ such that $s_{\vec{t}}(\lambda[i]) \in \sigma_{A\cup\mathsf{dom}(\rho)}(\lambda[i])$.

In order to show that λ is well-defined, it remains to show that for each $i \geq 0$, there is a state $s_{\vec{t}}(\lambda[i]) \in \sigma_{A\cup\mathsf{dom}(\rho)}(\lambda[i])$ such that $\vec{t}$ is a $\neg\langle\langle A\rangle\rangle_\rho \bigcirc\psi$-vector. Distinguish two cases:

- $A \cup \mathsf{dom}(\rho) = \Sigma$. By Definition 4.13, all vectors $\vec{t} \in [(-\ell, k)/n]$ with $t_a = \sharp_{\rho(a)}$, for each $a \in \mathsf{dom}(\rho)$, are $\neg\langle\langle A\rangle\rangle_\rho \bigcirc\psi$-vectors. Since $\sigma_{A\cup\mathsf{dom}(\rho)}(\lambda[i]) \in \delta(\lambda[i], A)$, the set $\sigma_{A\cup\mathsf{dom}(\rho)}(\lambda[i])$ is non-empty. Thus, by the fact that $\sigma_{A\cup\mathsf{dom}(\rho)}$ is a ρ-strategy, any vector from $\sigma_{A\cup\mathsf{dom}(\rho)}(\lambda[i])$ is suitable.
- $A \cup \mathsf{dom}(\rho) \neq \Sigma$. By definition of δ and since $\sigma_{A\cup\mathsf{dom}(\rho)}(\lambda[i]) \in \delta(\lambda[i], A)$, we have that $\sigma_{A\cup\mathsf{dom}(\rho)}(\lambda[i]) = \{s_{\vec{t}}(\lambda[i]) \mid \vec{t} \in S_{\langle A,\rho\rangle}\}$ for some A-voting set $S_{\langle A,\rho\rangle}$ under commitment ρ. By property 1 of the refutation function used in Definition 4.13 for $\neg\langle\langle A\rangle\rangle_\rho \bigcirc\psi$-vectors, the voting set $S_{\langle A,\rho\rangle}$ contains a $\neg\langle\langle A\rangle\rangle_\rho \bigcirc\psi$-vector. Thus, there is a state $s_{\vec{t}}(\lambda[i]) \in \sigma_{A\cup\mathsf{dom}(\rho)}(\lambda[i])$ such that $\vec{t}$ is a $\neg\langle\langle A\rangle\rangle_\rho \bigcirc\psi$-vector.

◀

To denote the intended type of a state $q = \varepsilon_{i,j}w \in Q$, we set $t(q) := T_{\langle\psi_i,\Psi_j\rangle}(w)$. Using the construction of $\mathcal{S}$ and property 2 in Definition 4.14 for witness trees, it is straightforward to prove the following claim, which, intuitively, states that our ATSN is $\bigcirc$-matching:

CLAIM 4.18. *For all $q \in Q$ and $\vec{t} \in [(-\ell, k)/n]$, $S_{t(q)}(\vec{t}) \subseteq t(s_{\vec{t}}(q))$.*

The next claim establishes the property of $\mathcal{S}$ that is crucial for showing that it is a model of φ.

CLAIM 4.19. *For any state $q \in Q$ and any formula $\psi \in ecl(\varphi)$, $\psi \in t(q)$ iff $\mathcal{S}, q \models \psi$.*

The proof of this claim is as the proof for Claim 3.14 in the proof of Lemma 3.11 about the correctness of the ATL decision procedure in Section 3.1.2.

Since we have that $\varphi \in \Psi$ for some type $\Psi \in \Gamma$, there is a state $q \in Q$ such that $\psi \in t(q)$. Then it follows from Claim 4.19 that $\mathcal{S}, q \models \varphi$.

"$\Leftarrow$" (Completeness): Suppose the formula φ is satisfiable in an ATSN $\mathcal{S} = \langle \Pi, \Sigma, Q, \{\Upsilon_a\}_{a\in\Sigma}, \pi, \delta\rangle$ in a state $q_\varphi \in Q$. For each state $q \in Q$, let $t(q)$ be the type $\{\psi \in \mathsf{ecl}(\varphi) \mid \mathcal{S}, q \models \psi\}$. Denote with $\mathsf{types}(Q)$ the set of all types associated with some state in Q. We first establish the following claim:

CLAIM 4.20. *Let $q \in Q$ and $\vec{t} \in [(-\ell, k)/n]$. Then there is a state $q' \in Q$ such that $S_{t(q)}(\vec{t}) \subseteq t(q')$. Moreover, the following holds: if $\langle\langle A\rangle\rangle_\rho \bigcirc\psi \in t(q)$ and $\sigma_{A\cup\mathsf{dom}(\rho)}$ is a ρ-strategy for the agents in $A\cup\mathsf{dom}(\rho)$ such that for all computations $\lambda \in out(q, \sigma_{A\cup\mathsf{dom}(\rho)})$, we have $\mathcal{S}, \lambda[1] \models \psi$, then we can choose a state q' such that $q' \in \sigma_{A\cup\mathsf{dom}(\rho)}(q)$.*

PROOF OF CLAIM Let q and $\vec{t}$ be as in the claim. Also, select a formula $\langle\langle A\rangle\rangle_\rho\bigcirc\psi$ and a ρ-strategy $\sigma_{A\cup\mathsf{dom}(\rho)}$ as in the 'moreover' part of the claim. Note first that, by Definition 4.13 of vectors for negative next-formulas and by property 3 of the refutation function used in Definition 4.13, the vector $\vec{t}$ is a vector for at most a single formula $\neg\langle\langle B\rangle\rangle_\rho\bigcirc\psi'$ with $B \cup \mathsf{dom}(\rho) \subset \Sigma_\varphi$. Let

- $\langle\langle A_1\rangle\rangle_{\rho_1}\bigcirc\psi_1, \dots, \langle\langle A_x\rangle\rangle_{\rho_x}\bigcirc\psi_x$ be all positive next-formulas from $t(q)$ for which $\vec{t}$ is a vector; this includes the selected formula $\langle\langle A\rangle\rangle_\rho\bigcirc\psi$;
- $\neg\langle\langle B\rangle\rangle_\xi\bigcirc\psi'$ be the single negative next-formula from $t(q)$ with $B\cup\mathsf{dom}(\xi) \subset \Sigma_\varphi$ for which $\vec{t}$ is a vector, if such a formula exists;
- $\neg\langle\langle C_1\rangle\rangle_{\zeta_1}\bigcirc\psi''_1, \dots, \neg\langle\langle C_m\rangle\rangle_{\zeta_m}\bigcirc\psi''_m$ be all negative next-formulas from $t(q)$ with $C_i \cup \mathsf{dom}(\zeta_i) = \Sigma_\varphi$, for all i with $1 \le i \le m$.

Observe that, by Definition 4.13, the vector $\vec{t}$ is a vector for all negative next-formulas $\neg\langle\langle C_i\rangle\rangle_{\zeta_i}\psi''_i$ (with $1 \le i \le m$), so that $\neg\psi''_1, \dots, \neg\psi''_m \in S_{t(q)}(\vec{t})$. Next note that, by Definition 4.13 of vectors for positive next-formulas, we have the following three properties: for all $i \ne j$ with $1 \le i, j \le x$,

(i) $\rho_i(a) = \rho_j(a)$, for all $a \in \mathsf{dom}(\rho_i) \cap \mathsf{dom}(\rho_j)$;
(ii) $(A_i \setminus \mathsf{dom}(\rho_i)) \cap \mathsf{dom}(\rho_j) = \emptyset$;

(iii) $(A_i \setminus \mathsf{dom}(\rho_i)) \cap (A_j \setminus \mathsf{dom}(\rho_j)) = \emptyset$.

For all i with $1 \le i \le x$, let $\sigma_{A_i \cup \mathsf{dom}(\rho_i)}$ be a ρ_i-strategy for the agents in $A_i \cup \mathsf{dom}(\rho_i)$ such that for all computations $\lambda \in \mathsf{out}(q, \sigma_{A_i \cup \mathsf{dom}(\rho_i)})$, we have that $\mathcal{S}, \lambda[1] \models \psi_i$. Such a strategy exists since $\langle\!\langle A_i \rangle\!\rangle_{\rho_i} \bigcirc \psi_i \in t(q)$. For the selected formula $\langle\!\langle A \rangle\!\rangle_{\rho} \bigcirc \psi$ from the 'moreover' part of the claim, choose the selected ρ-strategy $\sigma_{A \cup \mathsf{dom}(\rho)}$ for the agents in $A \cup \mathsf{dom}(\rho)$. Let $A' = \bigcup_{i=1..x} A_i \cup \mathsf{dom}(\rho_i)$ and set $\sigma_{A'} = \bigcup_{i=1..x} \sigma_{A_i \cup \mathsf{dom}(\rho_i)}$ which is well-defined by properties (i) to (iii). Thus for all $\lambda \in \mathsf{out}(q, \sigma_{A'})$, we have $\mathcal{S}, \lambda[1] \models \psi_i$ for all i with $1 \le i \le x$.

Next, we select a computation $\lambda \in \mathsf{out}(q, \sigma_{A'})$: If there is no negative next-formula $\neg\langle\!\langle B \rangle\!\rangle_{\xi} \bigcirc \psi'$ with $B \cup \mathsf{dom}(\xi) \subset \Sigma_{\varphi}$ in $t(q)$, for which $\vec{t}$ is a vector, then choose an arbitrary element $\lambda \in \mathsf{out}(q, \sigma_{A'})$. Note that $\mathsf{out}(q, \sigma_{A'})$ is non-empty since $\delta(q, A)$ is non-empty, for all states q and all coalitions A. Otherwise, choose a $\lambda \in \mathsf{out}(q, \sigma_{A'})$ such that $\mathcal{S}, \lambda[1] \models \neg\psi'$. Such an element exists since, first, $\neg\langle\!\langle B \rangle\!\rangle_{\xi} \bigcirc \psi'$ is in $t(q)$ and, second, $\vec{t}$ being a vector for this formula implies (by Definition 4.13 of vectors for negative next-formulas and by property 2 of the refutation function used in Definition 4.13) that:

- $A_i \cup \mathsf{dom}(\rho_i) \subseteq B \cup \mathsf{dom}(\xi)$, for all i with $1 \le i \le x$, and
- for all $a \in \mathsf{dom}(\xi)$, there is a ρ_i (with $1 \le i \le x$) such that $\rho_i(a) = \xi(a)$.

Finally, we have $\mathcal{S}, \lambda[1] \models \neg\psi_i''$, for all i with $1 \le i \le m$, since $\neg\langle\!\langle C_i \rangle\!\rangle_{\zeta_i} \bigcirc \psi_i'' \in t(q)$ implies that $\mathcal{S}, \lambda'[1] \models \neg\psi_i''$ for any computation λ' rooted at q.

Summing up, we have shown that $S_{t(q)}(\vec{t}) \subseteq t(\lambda[1]) \in \mathsf{types}(Q)$. Thus, $\lambda[1]$ is the state whose existence is stated in the claim. ◀

In the following, it is shown that all types in $\mathsf{types}(Q)$ are ATLES realizable in $\mathsf{types}(Q)$. Let $q \in Q$ be a state. We have to check that the type $t(q)$ in $\mathsf{types}(Q)$ satisfies conditions 1 to 3 of Definition 4.15.

1. Let $\vec{t} \in [(-\ell, k)/n]$. We have to show that $S_{t(q)}(\vec{t}) \subseteq \Psi$ for some $\Psi \in \mathsf{types}(Q)$. Clearly, this is an immediate consequence of Claim 4.20 (the 'moreover' part is not needed).
2. Suppose $\langle\!\langle A \rangle\!\rangle_{\rho} \psi \,\mathcal{U}\, \vartheta \in t(q)$. It is our aim to construct a $\langle\!\langle A \rangle\!\rangle_{\rho} \psi \,\mathcal{U}\, \vartheta$-witness tree rooted at the type $t(q)$ in $\mathsf{types}(Q)$. Since $\mathcal{S}, q \models \langle\!\langle A \rangle\!\rangle_{\rho} \psi \,\mathcal{U}\, \vartheta$, there is a ρ-strategy $\sigma_{A \cup \mathsf{dom}(\rho)}$ for the agents in $A \cup \mathsf{dom}(\rho)$ such that for all computations $\lambda \in \mathsf{out}(q, \sigma_{A \cup \mathsf{dom}(\rho)})$, there is a position $i \ge 0$ such that $\mathcal{S}, \lambda[i] \models \vartheta$ and $\mathcal{S}, \lambda[j] \models \psi$ for all positions $j < i$. Similar to item (2) in the completeness direction of the proof of Lemma 3.11 about the correctness of the ATL decision procedure, we use $\sigma_{A \cup \mathsf{dom}(\rho)}$ to define a finite $\langle Q, [(-\ell, k), n] \rangle$-tree T. Since the nodes in T are labelled by states, the composition $t(T(\cdot))$ yields a finite φ-tree. Then it can easily be shown that $t(T(\cdot))$ is indeed the required $\langle\!\langle A \rangle\!\rangle_{\rho} \psi \,\mathcal{U}\, \vartheta$-witness tree rooted at $t(q)$ in $\mathsf{types}(Q)$ by showing that T satisfies properties 1 to 7 in Definition 4.14.
3. This case is similar to the previous one and left to the reader.

From $S, q_\varphi \models \varphi$, it follows that $\varphi \in t(q_\varphi)$. Then types(Q) is a set of types that are each ATLES realizable in types(Q) and $t(q_\varphi)$ is a type in types(Q) such that $\varphi \in t(q_\varphi)$. Let Δ be the set of types for φ computed by the type elimination algorithm. It is easy to see that types(Q) $\subseteq \Delta$. Hence, the algorithm returns 'Yes, the input formula φ is satisfiable'. □

4.5. Completeness of the axiom system for ATLES

In this section, we show that the axiomatic system for ATLES as presented in Section 4.1.2 is sound and complete. For proving completeness we use the type elimination algorithm for ATLES from Section 4.4. The basic structure of the proof is similar to the completeness proof of the axiomatic system for Computation Tree Logic that can be found in [69].

The following three schemes generate valid and provable implications which state properties of ATLES that we later use to show completeness: for all commitment functions ρ, we have

Regularity: $\vdash \langle\langle A\rangle\rangle_\rho \bigcirc\varphi \rightarrow \neg\langle\langle B\rangle\rangle_\rho \bigcirc\neg\varphi$, for $A \cap B \subseteq \mathsf{dom}(\rho)$;
Coalition-monotonicity: $\vdash \langle\langle A\rangle\rangle_\rho \bigcirc\varphi \rightarrow \langle\langle B\rangle\rangle_\rho \bigcirc\varphi$, for $A \subseteq B$;
Property (P): $\langle\langle A\rangle\rangle_\rho \bigcirc\varphi \wedge \neg\langle\langle A\rangle\rangle_\rho \bigcirc\neg\psi \rightarrow \neg\langle\langle \emptyset\rangle\rangle_\rho \bigcirc\neg(\varphi \wedge \psi)$.

Instances of Regularity are provable using axioms (S) and ($\bot$), and instances of Coalition-monotonicity by using (S) and ($\top$). Property (P) can be derived using (S) and the inference rule $\langle\langle A\rangle\rangle_\rho\bigcirc$-Monotonicity as follows:

(1) $\vdash$ $\varphi \wedge \psi \rightarrow \varphi \wedge \psi$ (instance of (TAUT))

(2) $\vdash$ $\varphi \wedge \neg(\varphi \wedge \psi) \rightarrow \neg\psi$ (using (1))

(3) $\vdash$ $\langle\langle A\rangle\rangle_\rho \bigcirc(\varphi \wedge \neg(\varphi \wedge \psi)) \rightarrow \langle\langle A\rangle\rangle_\rho \bigcirc\neg\psi$ (using (2) and $\langle\langle A\rangle\rangle_\rho\bigcirc$-Monotonicity)

(4) $\vdash$ $\langle\langle A\rangle\rangle_\rho \bigcirc\varphi \wedge \langle\langle \emptyset\rangle\rangle_\rho \bigcirc\neg(\varphi \wedge \psi) \rightarrow \langle\langle A\rangle\rangle_\rho \bigcirc(\varphi \wedge \neg(\varphi \wedge \psi))$ (instance of (S))

(5) $\vdash$ $\langle\langle A\rangle\rangle_\rho \bigcirc\varphi \wedge \langle\langle \emptyset\rangle\rangle_\rho \bigcirc\neg(\varphi \wedge \psi) \rightarrow \langle\langle A\rangle\rangle_\rho \bigcirc\neg\psi$ (using (3) and (4))

(6) $\vdash$ $\langle\langle A\rangle\rangle_\rho \bigcirc\varphi \wedge \neg\langle\langle A\rangle\rangle_\rho \bigcirc\neg\psi \rightarrow \neg\langle\langle \emptyset\rangle\rangle_\rho \bigcirc\neg(\varphi \wedge \psi)$(using (5))

The axiomatic system for ATL (see Table 2.8 in Chapter 2) contains the axiom (Σ): $\neg\langle\langle \emptyset\rangle\rangle_\rho \bigcirc\neg\varphi \rightarrow \langle\langle \Sigma\rangle\rangle_\rho \bigcirc\varphi$. Notice that ($\Sigma$) is not used in the axiomatic system for ATLES since the grand coalition Σ of all agents is not available in ATLES.

THEOREM 4.21. *The deductive system for ATLES is sound and complete.*

The proof works as follows. For soundness, the validity of the axioms and the inference rules is readily checked. We now explain how completeness of the deduction system for ATLES can be shown. This is done along the lines of the completeness proof of an axiomatic system for CTL that is based on a tableau algorithm for CTL; cf. [69]. Let ϑ be an ATLES-formula and suppose ϑ is valid. It is to show that ϑ is provable. We have that $\neg\vartheta$ is unsatisfiable. Run the type elimination algorithm for ATLES on the input formula $\neg\vartheta$. The algorithm starts with all types for $\neg\vartheta$ and repeatedly eliminates those types that are not ATLES realizable. Then it returns 'Yes, $\neg\vartheta$ is satisfiable in an ATSN for $\Sigma_{\neg\vartheta}$ iff there is a type that survived the elimination containing the input formula. By Lemma 4.17, the input formula is satisfiable iff the algorithm says so. But then, since $\neg\vartheta$ is unsatisfiable, all types containing $\neg\vartheta$ will be eliminated. Now we need the following lemma whose proof we sketch below.

LEMMA 4.22. *If a type Ψ is eliminated, then Ψ is inconsistent.*

Then, by Lemma 4.22, it follows, for all types Ψ with $\neg\vartheta \in \Psi$, that Ψ is inconsistent. Using propositional reasoning, we establish the following validity:

$$\vdash \neg\vartheta \leftrightarrow \bigvee\{\bigwedge \Psi' \mid \Psi' \in \Gamma \text{ with } \neg\vartheta \in \Psi'\}$$

where Γ is a set of consistent types for $\neg\vartheta$ remaining after the execution of the type elimination algorithm. Since there is no consistent type containing $\neg\vartheta$, we have $\vdash \neg\vartheta \rightarrow \bot$. But then $\vdash \neg\neg\vartheta$ and thus $\vdash \vartheta$ which finishes the proof of Theorem 4.21.

We now sketch the proof of Lemma 4.22; the full proof can be found in Section 4.5.1. The proof is by induction on the number of elimination rounds. In the induction base, the type elimination algorithm is initialized with the set of all types for $\neg\vartheta$. No type has been eliminated yet. In the induction step, we denote the current set of types with Δ. A type Ψ in Δ is eliminated if it is not ATLES realizable in Δ. The type Ψ is not ATLES realizable in Δ if, for instance, there is a formula $\langle\langle A\rangle\rangle_\rho \psi \mathcal{U} \varphi$ in Ψ but there is no $\langle\langle A\rangle\rangle_\rho \psi \mathcal{U} \varphi$-witness tree rooted at Ψ in Δ. Given this situation, we need to show that Ψ is inconsistent. To this end, we define a set V to be the set of types Ψ' from Δ such that either Ψ' contains $\langle\langle A\rangle\rangle_\rho \psi \mathcal{U} \varphi$ but there is no $\langle\langle A\rangle\rangle_\rho \psi \mathcal{U} \varphi$-witness tree rooted at Ψ' in Δ, or $\langle\langle A\rangle\rangle_\rho \psi \mathcal{U} \varphi$ is not contained in Ψ'. Obviously, we have Ψ to be in V. Given the set V, we define the formula θ describing V as:

$$\theta = \bigvee\{\bigwedge \Psi' \mid \Psi' \in V\}.$$

By the induction hypothesis, we know that all types not in Δ are inconsistent. Then we can describe the complement set $\Delta \setminus V$ of V as follows:

$$\neg\theta = \bigvee\{\bigwedge \Psi' \mid \Psi' \in \Delta \setminus V\}.$$

Notice that a type Ψ' in $\Delta \setminus V$ contains $\langle\langle A\rangle\rangle_\rho \psi \mathcal{U} \varphi$ and that there is a $\langle\langle A\rangle\rangle_\rho \psi \mathcal{U} \varphi$-witness tree rooted at Ψ' in Δ. Now we show

$$\vdash \theta \rightarrow \neg\varphi \wedge (\neg\psi \vee \neg\langle\langle A\rangle\rangle_\rho \bigcirc \neg\theta)$$

from which we can derive Ψ's inconsistency using the axioms (LFP$_{\mathcal{U}}$) and (FP$_\Box$) and the inference rule $\langle\langle \emptyset\rangle\rangle_\rho\Box$-Necessitation. To this end, we show that

$$\vdash \bigwedge \Psi \rightarrow \neg\varphi \wedge (\neg\psi \vee \neg\langle\langle A\rangle\rangle_\rho \bigcirc \bigwedge \Psi')$$

by constructing a $\langle\langle A\rangle\rangle_\rho \psi \mathcal{U} \varphi$-witness tree rooted at Ψ in Δ using the $\langle\langle A\rangle\rangle_\rho \psi \mathcal{U} \varphi$-witness tree rooted at Ψ' in Δ.

For showing that the constructed tree is indeed the desired $\langle\!\langle A\rangle\!\rangle_\rho \psi\,\mathcal{U}\,\varphi$-witness tree, we need the following lemma that formalises the relationship between next formulas with different coalitions and commitment functions by combining the axioms (C1), (C2) and (C3).

LEMMA 4.23. *Let A, B be two coalitions of agents and ρ, ξ two commitment functions. Then the following are equivalent:*

(a) $\vdash \langle\!\langle A\rangle\!\rangle_\rho \bigcirc\psi \to \langle\!\langle B\rangle\!\rangle_\xi \bigcirc\psi$, *for all ATLES-formulas ψ;*

(b) *the following four conditions are satisfied:*

(P0) $\rho(a) = \xi(a)$, *for all $a \in \mathsf{dom}(\rho) \cap \mathsf{dom}(\xi)$;*

(P1) $(\mathsf{dom}(\xi) \setminus \mathsf{dom}(\rho)) \cap A = \emptyset$;

(P2) $\mathsf{dom}(\rho) \setminus \mathsf{dom}(\xi) \subseteq B$;

(P3) $A \setminus B \subseteq \mathsf{dom}(\rho) \cap \mathsf{dom}(\xi)$.

Intuitively, the implication in Lemma 4.23 can be seen as a course of action where agents join or leave the coalition and agents take or dismiss commitments. Changing a commitment involves two steps: first, dismissing an old commitment and, second, taking a new commitment. Clearly, such actions might cause some changes in the formation of the coalition and in the state of the agents' commitments. However, the coalition will still be able to bring about ψ at the next state if the following four policies are obeyed: (P0) no committed agent changes her commitment; (P1) no member of the former coalition can take a commitment; (P2) all agents dismissing their commitments already are or have to become members of the coalition; and (P3) all agents leaving the coalition have got to be and to stay committed. For a set theoretic illustration of the coalitions A, B and the domains of ρ, ξ satisfying the conditions (P1) to (P3) see Figure 4.7. Notice that (P1) to (P3) correspond to the axioms (C1) to (C3), respectively.

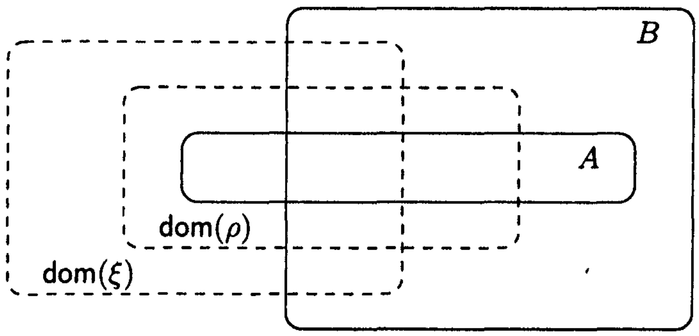


FIGURE 4.7. Venn diagram of coalitions A, B and domains of ρ, ξ.

We now present a detailed proof for Lemma 4.23.

PROOF. Let A, B, ρ, ξ and ψ be as in Lemma 4.23. For the direction from (b) to (a), suppose that conditions (P0) to (P3) are satisfied. In the following, we present a derivation of the implication $\langle\!\langle A\rangle\!\rangle_\rho \bigcirc\psi \to \langle\!\langle B\rangle\!\rangle_\xi \bigcirc\psi$ and thus show its provability.

We start in (1) with an instance of Coalition-monotonicity using the property $A \subseteq B \cup (\mathsf{dom}(\rho) \cap \mathsf{dom}(\xi))$ obtained from (P3). Notice that we use the operator $\upharpoonright$ to restrict the domain of a commitment function. For instance, the expression $\xi \upharpoonright C$ stands for the commitment function ξ whose domain is restricted to the agents in C.

(1) $\vdash \langle\!\langle A \rangle\!\rangle_{\zeta_0} \bigcirc \psi \to \langle\!\langle B \cup (\mathsf{dom}(\rho) \cap \mathsf{dom}(\xi)) \rangle\!\rangle_{\zeta_0} \bigcirc \psi$
(where $\zeta_0 = \rho \cup (\xi \upharpoonright B)$)

(2) $\vdash \langle\!\langle A \rangle\!\rangle_{\zeta_1} \bigcirc \psi \to \langle\!\langle A \rangle\!\rangle_{\zeta_0} \bigcirc \psi$
(where $\zeta_1 = \zeta_0 \upharpoonright \mathsf{dom}(\rho)$,
using (C1) since $(\mathsf{dom}(\xi) \setminus \mathsf{dom}(\rho)) \cap A = \emptyset$ by (P1))

(3) $\vdash \langle\!\langle B \cup (\mathsf{dom}(\rho) \cap \mathsf{dom}(\xi)) \rangle\!\rangle_{\zeta_0} \bigcirc \psi \to \langle\!\langle B \rangle\!\rangle_{\zeta_0} \bigcirc \psi$
(using (C3) since $\mathsf{dom}(\rho) \cap \mathsf{dom}(\xi) \subseteq \mathsf{dom}(\zeta_0)$)

(4) $\vdash \langle\!\langle B \rangle\!\rangle_{\zeta_0} \bigcirc \psi \to \langle\!\langle B \rangle\!\rangle_{\zeta_2} \bigcirc \psi$
(where $\zeta_2 = \zeta_0 \cup (\xi \upharpoonright C)$ with $C = \mathsf{dom}(\xi) \setminus B$,
using (C1) since $(\mathsf{dom}(\xi) \setminus B) \cap B = \emptyset$)

(5) $\vdash \langle\!\langle B \rangle\!\rangle_{\zeta_2} \bigcirc \psi \to \langle\!\langle B \rangle\!\rangle_{\zeta_3} \bigcirc \psi$
(where $\zeta_3 = \zeta_2 \upharpoonright \mathsf{dom}(\xi)$,
using (C2) since $\mathsf{dom}(\rho) \setminus \mathsf{dom}(\xi) \subseteq B$ by (P2))

(6) $\vdash \langle\!\langle A \rangle\!\rangle_{\zeta_1} \bigcirc \psi \to \langle\!\langle B \rangle\!\rangle_{\zeta_3} \bigcirc \psi$ (using (1) to (5))

Notice that, by (P0), the commitment functions $\zeta_0, \ldots, \zeta_3$ are well-defined. To see that in (6) we indeed derived the desired implication, we now show that $\zeta_1 = \rho$ and $\zeta_3 = \xi$.

$$\begin{aligned}
\zeta_1 &= \zeta_0 \upharpoonright \mathsf{dom}(\rho) \\
&= (\rho \cup (\xi \upharpoonright B)) \upharpoonright \mathsf{dom}(\rho) \\
&= \rho \cup (\xi \upharpoonright B) \upharpoonright \mathsf{dom}(\rho) \\
&= \rho
\end{aligned}$$

$$\begin{aligned}
\zeta_3 &= \zeta_2 \upharpoonright \mathsf{dom}(\xi) \\
&= (\zeta_0 \cup (\xi \upharpoonright C)) \upharpoonright \mathsf{dom}(\xi) \\
&= \zeta_0 \upharpoonright \mathsf{dom}(\xi) \cup (\xi \upharpoonright C) \upharpoonright \mathsf{dom}(\xi) \\
&= (\rho \cup (\xi \upharpoonright B)) \upharpoonright \mathsf{dom}(\xi) \cup \xi \upharpoonright C \\
&= \rho \upharpoonright \mathsf{dom}(\xi) \cup (\xi \upharpoonright B) \upharpoonright \mathsf{dom}(\xi) \cup \xi \upharpoonright (\mathsf{dom}(\xi) \setminus B) \\
&= \rho \upharpoonright \mathsf{dom}(\xi) \cup \xi \upharpoonright (B \cap \mathsf{dom}(\xi) \cup (\mathsf{dom}(\xi) \setminus B)) \\
&= \rho \upharpoonright \mathsf{dom}(\xi) \cup \xi \upharpoonright \mathsf{dom}(\xi) \\
&= \xi
\end{aligned}$$

Consider the other direction from (a) to (b). Let $\varphi = \langle\!\langle A \rangle\!\rangle_{\rho} \bigcirc \psi \to \langle\!\langle B \rangle\!\rangle_{\xi} \bigcirc \psi$. Suppose $\vdash \varphi$. Since the axiomatic system for ATLES is readily checked to be sound, we have that φ is valid. We reason by contradiction that each of the conditions (P0) to (P3) are satisfied. For (P0), assume $\rho(a) \neq \xi(a)$, for some agent $a \in \mathsf{dom}(\rho) \cap \mathsf{dom}(\xi)$. We describe a countermodel for φ as follows. Take an ATSN S satisfying $\langle\!\langle A\rangle\!\rangle_\rho \bigcirc\psi$ at a state q. Let agent a have exactly two choices Q_1 and Q_2 at q, where $Q_1 = \{q_1\}$ and $Q_2 = \{q_2\}$ such that $S, q_1 \models \psi$ and $S, q_2 \not\models \psi$. When following strategy $\rho(a)$ at state q, agent a chooses Q_1, and Q_2 when following $\xi(a)$. Clearly, we have $S, q \not\models \langle\!\langle B\rangle\!\rangle_\xi \bigcirc\psi$. For the other conditions (P1) to (P3), we can describe similar countermodels. □

4.5.1. Proofs for ATLES Completeness.

LEMMA 4.18. *If a type Ψ is eliminated, then Ψ is inconsistent.*

PROOF. Before proving the lemma, we recall and introduce some notation. Let $\neg\vartheta$ be the unsatisfiable ATLES-formula, on which we run the type elimination algorithm. Let Σ_ϑ the set of all agents that occur in ϑ, $n = |\Sigma_\vartheta|$ the number of those agents, k the number of next-formulas in $\mathsf{ecl}(\neg\vartheta)$ and ℓ the number of strategy terms in ϑ. Given a type Ψ' for $\neg\vartheta$ and a vector $\vec{t} \in [(-\ell, k)/n]$, let

- $\langle\!\langle A_1\rangle\!\rangle_{\rho_1} \bigcirc\varphi_1, \ldots, \langle\!\langle A_x\rangle\!\rangle_{\rho_x} \bigcirc\varphi_x$ be all positive next-formulas from Ψ' for which $\vec{t}$ is a vector (cf. Definition 4.13);
- $\neg\langle\!\langle B_1\rangle\!\rangle_{\xi_1} \bigcirc\varphi'_1, \ldots, \neg\langle\!\langle B_y\rangle\!\rangle_{\xi_y} \bigcirc\varphi'_y$ be all negative next-formulas from Ψ' for which $\vec{t}$ is a vector (cf. Definition 4.13).

The considered type Ψ' and vector $\vec{t}$ will be clear from the context, so we can avoid laborious notation. Note that, by property 3 of the refutation function used in Definition 4.13 vectors for negative next formulas, there is at most one formula $\neg\langle\!\langle B_i\rangle\!\rangle_{\xi_i} \bigcirc\varphi'_i$ (where $1 \leq i \leq y$) in Ψ' with $B_i \cup \mathsf{dom}(\xi_i) \subset \Sigma_\vartheta$, for which $\vec{t}$ is a vector. All the other formulas $\neg\langle\!\langle B_j\rangle\!\rangle_{\xi_j} \bigcirc\varphi'_j$ (with $j \neq i$ and $1 \leq j \leq y$) are such that $B_j \cup \mathsf{dom}(\xi_j) = \Sigma_\vartheta$.

Still considering Ψ' and $\vec{t}$, the coalitions $A_1, \ldots, A_x$ and $B_1, \ldots, B_y$ and the commitment functions $\rho_1, \ldots, \rho_x$ and $\xi_1, \ldots, \xi_y$ in the formulas from Ψ' as selected above will be used to abbreviate the following two conjunctions of positive next formulas:

$$\begin{aligned}
\Phi_{\langle\Psi',\vec{t}\rangle}(\psi) &= \bigwedge_{i=1..x} \langle\!\langle \Sigma_\vartheta \setminus A_i\rangle\!\rangle_{\rho_i} \bigcirc\psi, \\
\Phi^{\neg}_{\langle\Psi',\vec{t}\rangle}(\psi) &= \bigwedge_{i=1..y} \langle\!\langle B_i\rangle\!\rangle_{\xi_i} \bigcirc\psi.
\end{aligned}$$

For proving the lemma, we will frequently make use of the following claim.

CLAIM 4.19. *Let $\vec{t} \in [(-\ell, k)/n]$ be a vector, and Ψ and Ψ' two types from $\Gamma_{\neg\vartheta}$. Then it holds that:*

(a) *If $\langle\!\langle A\rangle\!\rangle_\rho \bigcirc\varphi \in \Psi$, $\varphi \notin \Psi'$, and $\vec{t}$ is a $\langle\!\langle A\rangle\!\rangle_\rho \bigcirc\varphi$-vector, then $\bigwedge\Psi \wedge \langle\!\langle \Sigma_\vartheta \setminus A\rangle\!\rangle_\rho \bigcirc \bigwedge\Psi'$ is inconsistent;*

(b) *If $\neg\langle\!\langle B\rangle\!\rangle_\xi \bigcirc\varphi' \in \Psi$, $\neg\varphi' \notin \Psi'$, and $\vec{t}$ is a $\neg\langle\!\langle B\rangle\!\rangle_\xi \bigcirc\varphi'$-vector, then $\bigwedge\Psi \wedge \langle\!\langle B\rangle\!\rangle_\xi \bigcirc \bigwedge\Psi'$ is inconsistent.*

PROOF OF CLAIM Let $\vec{t}$, Ψ and Ψ' be as in the claim. For Case (a), suppose $\langle\!\langle A\rangle\!\rangle_\rho \bigcirc \varphi \in \Psi$, $\varphi \notin \Psi'$, and that $\vec{t}$ is a $\langle\!\langle A\rangle\!\rangle_\rho \bigcirc \varphi$-vector. Consider the following abbreviated derivation:

(1) $\vdash \bigwedge \Psi \to \langle\!\langle A\rangle\!\rangle_\rho \bigcirc \varphi$ (since $\langle\!\langle A\rangle\!\rangle_\rho \bigcirc \varphi \in \Psi$)

(2) $\vdash \bigwedge \Psi' \to \neg\varphi$ (since $\neg\varphi \in \Psi'$ by (T2))

(3) $\vdash \langle\!\langle \Sigma_\vartheta \setminus A\rangle\!\rangle_\rho \bigcirc \bigwedge \Psi' \to \langle\!\langle \Sigma_\vartheta \setminus A\rangle\!\rangle_\rho \bigcirc \neg\varphi$ (using (2) and $\langle\!\langle A\rangle\!\rangle_\rho \bigcirc$-Monotonicity)

(4) $\vdash \bigwedge \Psi \wedge \langle\!\langle \Sigma_\vartheta \setminus A\rangle\!\rangle_\rho \bigcirc \bigwedge \Psi' \to \langle\!\langle A\rangle\!\rangle_\rho \bigcirc \varphi \wedge \langle\!\langle \Sigma_\vartheta \setminus A\rangle\!\rangle_\rho \bigcirc \neg\varphi$ (using (1) and (3))

(5) $\vdash \bigwedge \Psi \wedge \langle\!\langle \Sigma_\vartheta \setminus A\rangle\!\rangle_\rho \bigcirc \bigwedge \Psi' \to \langle\!\langle \Sigma_\vartheta\rangle\!\rangle_\rho \bigcirc (\varphi \wedge \neg\varphi)$ (using (4) and (S))

(6) $\vdash \bigwedge \Psi \wedge \langle\!\langle \Sigma_\vartheta \setminus A\rangle\!\rangle_\rho \bigcirc \bigwedge \Psi' \to \langle\!\langle \Sigma_\vartheta\rangle\!\rangle_\rho \bigcirc \bot$ (using (5))

(7) $\vdash \neg(\bigwedge \Psi \wedge \langle\!\langle \Sigma_\vartheta \setminus A\rangle\!\rangle_\rho \bigcirc \bigwedge \Psi')$ (using (6) and ($\bot$))

For Case (b), suppose $\neg\langle\!\langle B\rangle\!\rangle_\xi \bigcirc \varphi' \in \Psi$, $\neg\varphi' \notin \Psi'$ and that $\vec{t}$ is a $\neg\langle\!\langle B\rangle\!\rangle_\xi \bigcirc \varphi'$-vector. Then we derive:

(8) $\vdash \bigwedge \Psi \to \neg\langle\!\langle B\rangle\!\rangle_\xi \bigcirc \varphi'$ (since $\neg\langle\!\langle B\rangle\!\rangle_\xi \bigcirc \varphi' \in \Psi$)

(9) $\vdash \bigwedge \Psi' \to \varphi'$ (since $\varphi' \in \Psi'$ by (T2))

(10) $\vdash \langle\!\langle B\rangle\!\rangle_\xi \bigcirc \bigwedge \Psi' \to \langle\!\langle B\rangle\!\rangle_\xi \bigcirc \varphi'$ (using (9) and $\langle\!\langle A\rangle\!\rangle_\rho \bigcirc$-Monotonicity)

(11) $\vdash \bigwedge \Psi \wedge \langle\!\langle B\rangle\!\rangle_\xi \bigcirc \bigwedge \Psi' \to \neg\langle\!\langle B\rangle\!\rangle_\xi \bigcirc \varphi' \wedge \langle\!\langle B\rangle\!\rangle_\xi \bigcirc \varphi'$ (using (8) and (10))

(12) $\vdash \bigwedge \Psi \wedge \langle\!\langle B\rangle\!\rangle_\xi \bigcirc \bigwedge \Psi' \to \neg\langle\!\langle \emptyset\rangle\!\rangle_\xi \bigcirc \neg(\neg\varphi' \wedge \varphi')$ (using (11) and (P))

(13) $\vdash \bigwedge \Psi \wedge \langle\!\langle B\rangle\!\rangle_\xi \bigcirc \bigwedge \Psi' \to \neg\langle\!\langle \emptyset\rangle\!\rangle_\xi \bigcirc \top$ (using (12))

(14) $\vdash \neg(\bigwedge \Psi \wedge \langle\!\langle B\rangle\!\rangle_\xi \bigcirc \bigwedge \Psi')$ (using (13))

◀

Now we are ready to prove Lemma 4.22. The proof is by induction on the number of elimination rounds. In the induction base, the type elimination algorithm is initialized with the set of all types for $\neg\vartheta$. No type has been eliminated yet. Consider the induction step. Let Δ be the current set of types. A type $\Psi \in \Delta$ gets eliminated if it is not ATLES realizable in Δ. By Definition 4.15 of realizability, the type Ψ is not ATLES realizable in Δ if one of the following three conditions are satisfied:

(i) there is a vector $\vec{t} \in [(-\ell, k)/n]$ such that for all types $\Psi' \in \Delta$, it holds that $S_\Psi(\vec{t}) \not\subseteq \Psi'$;

(ii) $\langle\!\langle A\rangle\!\rangle_\rho \psi \mathcal{U} \varphi \in \Psi$ and there is no $\langle\!\langle A\rangle\!\rangle_\rho \psi \mathcal{U} \varphi$-witness tree rooted at Ψ in Δ;

(iii) $\neg\langle\!\langle A\rangle\!\rangle_\rho \Box\varphi \in \Psi$ and there is no $\neg\langle\!\langle A\rangle\!\rangle_\rho \Box\varphi$-witness tree rooted at Ψ in Δ.

First, consider Case (i). Suppose, for some vector $\vec{t} \in [(-\ell, k)/n]$, we have that $S_\Psi(\vec{t}) \not\subseteq \Psi'$ for all types $\Psi' \in \Delta$. We have to show that Ψ is inconsistent. We can derive the following.

$$(1)\ \vdash\ \top \leftrightarrow \bigvee\{\bigwedge \Psi' \mid \Psi' \in \Gamma_{\neg\vartheta}\}$$

$$(2)\ \vdash\ \top \leftrightarrow \bigvee\{\bigwedge \Psi' \mid \Psi' \in \Delta \text{ is consistent}\}$$

Notice that we can establish (1) by propositional reasoning. By the induction hypothesis, we have that all types in $\Gamma_{\neg\vartheta} \setminus \Delta$ are inconsistent. Thus, from (1) we can derive (2). Next, we show that $\bigwedge \Psi \wedge \Phi_{\langle\Psi,\vec{t}\rangle}(\bigwedge \Psi') \wedge \Phi^{\neg}_{\langle\Psi,\vec{t}\rangle}(\bigwedge \Psi')$ is inconsistent, for all types $\Psi' \in \Delta$. From $S_\Psi(\vec{t}) \not\subseteq \Psi'$, we have by Definition 4.13 that (a) $\varphi_i \in S_\Psi(\vec{t})$ and $\varphi_i \notin \Psi'$ for some i with $1 \leq i \leq x$, or (b) $\neg\varphi'_i \in S_\Psi(\vec{t})$ and $\neg\varphi'_i \notin \Psi'$ for some i with $1 \leq i \leq y$. Notice that we have (a) $\langle\!\langle A_i \rangle\!\rangle_{\rho_i} \bigcirc \varphi_i \in \Psi$ and (b) $\neg\langle\!\langle B_i \rangle\!\rangle_{\xi_i} \bigcirc \varphi'_i \in \Psi$. Then we obtain by Claim 4.19 that (a) $\bigwedge \Psi \wedge \langle\!\langle \Sigma_\vartheta \setminus A_i \rangle\!\rangle_{\rho_i} \bigcirc \bigwedge \Psi'$, or (b) $\bigwedge \Psi \wedge \langle\!\langle B_i \rangle\!\rangle_{\xi_i} \bigcirc \bigwedge \Psi'$ is inconsistent. Thus we can derive (3).

$$(3)\ \vdash\ \neg(\bigwedge \Psi \wedge \Phi_{\langle\Psi,\vec{t}\rangle}(\bigwedge \Psi') \wedge \Phi^{\neg}_{\langle\Psi,\vec{t}\rangle}(\bigwedge \Psi')), \text{ for every } \Psi' \in \Delta \text{ (by Claim 4.19)}$$

We need (2) and (3) in the following derivation, where we show that Ψ is inconsistent.

$$(4)\ \vdash\ \bigwedge\{\neg(\bigwedge \Psi \wedge \Phi_{\langle\Psi,\vec{t}\rangle}(\bigwedge \Psi') \wedge \Phi^{\neg}_{\langle\Psi,\vec{t}\rangle}(\bigwedge \Psi')) \mid \Psi' \in \Delta \text{ is consistent}\} \text{ (using (3))}$$

$$(5)\ \vdash\ \neg(\bigvee\{\bigwedge \Psi \wedge \Phi_{\langle\Psi,\vec{t}\rangle}(\bigwedge \Psi') \wedge \Phi^{\neg}_{\langle\Psi,\vec{t}\rangle}(\bigwedge \Psi') \mid \Psi' \in \Delta \text{ is consistent}\}) \text{ (using (4))}$$

$$(6)\ \vdash\ \neg(\bigwedge \Psi \wedge \bigvee\{\Phi_{\langle\Psi,\vec{t}\rangle}(\bigwedge \Psi') \wedge \Phi^{\neg}_{\langle\Psi,\vec{t}\rangle}(\bigwedge \Psi') \mid \Psi' \in \Delta \text{ is consistent}\}) \text{ (using (5))}$$

$$(7)\ \vdash\ \neg(\bigwedge \Psi \wedge \Phi_{\langle\Psi,\vec{t}\rangle}(\bigvee\{\bigwedge \Psi' \mid \Psi' \in \Delta \text{ is consistent}\}) \wedge$$
$$\Phi^{\neg}_{\langle\Psi,\vec{t}\rangle}(\bigvee\{\bigwedge \Psi' \mid \Psi' \in \Delta \text{ is consistent}\})) \text{ (using (6))}$$

$$(8)\ \vdash\ \neg(\bigwedge \Psi \wedge \Phi_{\langle\Psi,\vec{t}\rangle}(\top) \wedge \Phi^{\neg}_{\langle\Psi,\vec{t}\rangle}(\top)) \text{ (using (2) and (7))}$$

$$(9)\ \vdash\ \neg(\bigwedge \Psi) \text{ (using (8) and (T))}$$

By (9), we have that Ψ is inconsistent.

Second, suppose Case (ii) is satisfied. It is to show that Ψ is inconsistent. Let V be the set of types $\Psi' \in \Delta$ such that either $\langle\!\langle A \rangle\!\rangle_\rho \psi \mathcal{U} \varphi \in \Psi'$ but there is no $\langle\!\langle A \rangle\!\rangle_\rho \psi \mathcal{U} \varphi$-witness tree rooted at Ψ' in Δ, or $\langle\!\langle A \rangle\!\rangle_\rho \psi \mathcal{U} \varphi \notin \Psi'$. Note that $\Psi \in V$. Let

$$\theta = \bigvee\{\bigwedge \Psi' \mid \Psi' \in V\}.$$

By the induction hypothesis, all types in $\Gamma_{\neg\vartheta} \setminus \Delta$ are inconsistent. Thus we have that

$$\neg\theta = \bigvee\{\bigwedge \Psi' \mid \Psi' \in \Delta \setminus V\}.$$

In the following, it is shown that

$$\vdash \theta \rightarrow \neg\varphi \wedge (\neg\psi \vee \neg\langle\!\langle A \rangle\!\rangle_\rho \bigcirc \neg\theta).$$

Notice that $\neg\varphi \in \Psi'$ for all $\Psi' \in V$. To see that, suppose the contrary, i.e., $\neg\varphi \notin \Psi'$ for some $\Psi' \in V$. By (T2), we have $\varphi \in \Psi'$. But then, if $\langle\!\langle A \rangle\!\rangle_\rho \psi \mathcal{U} \varphi \in \Psi'$, a $\neg\vartheta$-tree containing only a root labelled with Ψ' would be a $\langle\!\langle A \rangle\!\rangle_\rho \psi \mathcal{U} \varphi$-witness tree rooted at Ψ' in Δ; contradicting $\Psi' \in V$. Otherwise, if $\langle\langle A\rangle\rangle_\rho \psi \mathcal{U} \varphi \notin \Psi'$, (T4) yields that $\varphi \notin \Psi'$; again a contradiction. Consequently, we have $\vdash \bigwedge \Psi' \to \neg\varphi$ for all $\Psi' \in V$, and thus $\vdash \theta \to \neg\varphi$. In order to show $\vdash \theta \to \neg\psi \vee \neg\langle\langle A\rangle\rangle_\rho \bigcirc \neg\theta$, it suffices to show $\vdash \bigwedge \Psi \to \neg\psi \vee \neg\langle\langle A\rangle\rangle_\rho \bigcirc \neg\theta$. Suppose by contradiction that $\nvdash \bigwedge \Psi \to \neg\psi \vee \neg\langle\langle A\rangle\rangle_\rho \bigcirc \neg\theta$, i.e., $\nvdash \neg(\bigwedge \Psi \wedge \psi \wedge \langle\langle A\rangle\rangle_\rho \bigcirc \neg\theta)$. Then we have, for some $\Psi' \in \Delta \setminus V$, that

$$\nvdash \neg(\bigwedge \Psi \wedge \psi \wedge \langle\langle A\rangle\rangle_\rho \bigcirc \bigwedge \Psi').$$

Fix such a Ψ'. Since $\Psi' \in \Delta \setminus V$, there is a $\langle\langle A\rangle\rangle_\rho \psi \mathcal{U} \varphi$-witness tree T rooted at Ψ' in Δ. Using T, we construct a $\langle\langle A\rangle\rangle_\rho \psi \mathcal{U} \varphi$-witness tree T' rooted at Ψ in Δ as follows:

- $T'(\varepsilon) := \Psi$;
- $T'(\vec{t}\cdot\alpha) := T(\alpha)$, for all $\langle\langle A\rangle\rangle_\rho \bigcirc \langle\langle A\rangle\rangle_\rho \psi \mathcal{U} \varphi$-vectors $\vec{t}$ and all nodes $\alpha \in \mathsf{dom}(T)$.

It can readily be checked that T' satisfies conditions 1 and 3 to 7 in Definition 4.14 of $\langle\langle A\rangle\rangle_\rho \psi \mathcal{U} \varphi$-witness trees. For conditions 4 and 5, notice that $\langle\langle A\rangle\rangle_\rho \psi \mathcal{U} \varphi \in \Psi$ by assumption and $\psi \in \Psi$ follows from (T4) by the fact that $\varphi \notin \Psi$. Consider condition 2, which requires that T' is $\bigcirc$-matching. Let $\vec{t}$ be a $\langle\langle A\rangle\rangle_\rho \bigcirc \langle\langle A\rangle\rangle_\rho \psi \mathcal{U} \varphi$-vector. Given Ψ and $\vec{t}$, let $j_{\langle\Psi,\vec{t}\rangle}$ be the number ranging over 1 to x such that $\langle\langle A\rangle\rangle_\rho \bigcirc \langle\langle A\rangle\rangle_\rho \psi \mathcal{U} \varphi$ is the $j_{\langle\Psi,\vec{t}\rangle}$-th formula in the enumeration of positive next formulas from Ψ for which $\vec{t}$ is a vector. Let $\chi_{\langle\Psi,\vec{t}\rangle}$ abbreviate the following negated conjunction:

$$\chi_{\langle\Psi,\vec{t}\rangle} = \neg(\bigwedge \Psi \wedge \bigwedge_{\substack{i=1..x \\ i\neq j_{\langle\Psi,\vec{t}\rangle}}} \langle\langle \Sigma_\vartheta \setminus A_i\rangle\rangle_{\rho_i} \bigcirc \bigwedge \Psi' \wedge \Phi^{\neg}_{\langle\Psi,\vec{t}\rangle}(\bigwedge \Psi')).$$

In the following, we show that

$$(\dagger) \qquad \nvdash \bigwedge \Psi \wedge \psi \wedge \langle\langle A\rangle\rangle_\rho \bigcirc \bigwedge \Psi') \text{ implies } \nvdash \chi_{\langle\Psi,\vec{t}\rangle}.$$

Then, since $\nvdash \neg(\bigwedge \Psi \wedge \psi \wedge \langle\langle A\rangle\rangle_\rho \bigcirc \bigwedge \Psi')$, we obtain from (†) that $\nvdash \chi_{\langle\Psi,\vec{t}\rangle}$. Together with Claim 4.19 and the fact that $\langle\langle A\rangle\rangle_\rho \bigcirc \langle\langle A\rangle\rangle_\rho \psi \mathcal{U} \varphi \in \Psi$ and $\langle\langle A\rangle\rangle_\rho \psi \mathcal{U} \varphi \in \Psi'$, we have $S_\Psi(\vec{t}) \subseteq \Psi'$, for every $\langle\langle A\rangle\rangle_\rho \bigcirc \langle\langle A\rangle\rangle_\rho \psi \mathcal{U} \varphi$-vector $\vec{t}$. By Definition 4.13 and the fact that the tree T is already $\bigcirc$-matching, we have that T' is $\bigcirc$-matching. Hence condition 2 is satisfied.

We now show, for every $\langle\langle A\rangle\rangle_\rho \bigcirc \langle\langle A\rangle\rangle_\rho \psi \mathcal{U} \varphi$-vector $\vec{t}$, the contrapositive of the implication (†). Suppose $\vdash \chi_{\langle\Psi,\vec{t}\rangle}$. Distinguish the following three cases:

(a) Suppose $\vdash \neg \bigwedge \Psi$. Obviously, this implies $\vdash \neg(\bigwedge \Psi \wedge \psi \wedge \langle\langle A\rangle\rangle_\rho \bigcirc \bigwedge \Psi')$.

(b) Suppose $\vdash \neg \bigwedge_{\substack{i=1..x \\ i\neq j_{\langle\Psi,\vec{t}\rangle}}} \langle\langle \Sigma_\vartheta \setminus A_i\rangle\rangle_{\rho_i} \bigcirc \bigwedge \Psi'$. That is $\vdash \neg\langle\langle \Sigma_\vartheta \setminus A_i\rangle\rangle_{\rho_i} \bigcirc \bigwedge \Psi'$ for some $i \neq j_{\langle\Psi,\vec{t}\rangle}$ with $1 \leq i \leq x$. Note that $\vec{t}$ is a $\langle\langle A\rangle\rangle_\rho \bigcirc \langle\langle A\rangle\rangle_\rho \psi \mathcal{U} \varphi$-vector and a $\langle\langle A_i\rangle\rangle_{\rho_i} \bigcirc \varphi_i$-vector, and that $\langle\langle A_i\rangle\rangle_{\rho_i} \bigcirc \varphi_i \neq \langle\langle A\rangle\rangle_\rho \bigcirc \langle\langle A\rangle\rangle_\rho \psi \mathcal{U} \varphi$. We show that the conditions (P0) to (P3) in Lemma 4.23 are satisfied:

   - By Definition 4.13 of $\vec{t}$ being a vector for both $\langle\langle A\rangle\rangle_\rho \bigcirc \langle\langle A\rangle\rangle_\rho \psi \mathcal{U} \varphi$ and $\langle\langle A_i\rangle\rangle_{\rho_i} \bigcirc \varphi_i$, we have that $\rho(a) = \rho_i(a)$, for all agents $a \in \mathsf{dom}(\rho) \cap \mathsf{dom}(\rho_i)$. This corresponds to (P0).

- By Definition 4.13 of $\vec{t}$ being both a $\langle\langle A\rangle\rangle_\rho \bigcirc \langle\langle A\rangle\rangle_\rho \psi \mathcal{U} \varphi$-vector and a $\langle\langle A_i\rangle\rangle_{\rho_i} \bigcirc \varphi_i$-vector, it holds that $(A \setminus \mathsf{dom}(\rho)) \cap \mathsf{dom}(\rho_i) = \emptyset$. We can convert this into its equivalent $(\mathsf{dom}(\rho_i) \setminus \mathsf{dom}(\rho)) \cap A = \emptyset$ which corresponds to (P1).
- Similarly to the property (P1) above, we can show $\mathsf{dom}(\rho) \setminus \mathsf{dom}(\rho_i) \subseteq \Sigma_\vartheta \setminus A_i$ which corresponds to (P2).
- Since $\langle\langle A_i\rangle\rangle_{\rho_i} \bigcirc \varphi_i \neq \langle\langle A\rangle\rangle_\rho \bigcirc \langle\langle A\rangle\rangle_\rho \psi \mathcal{U} \varphi$, it follows by Definition 4.13 of $\vec{t}$ being a vector for both of these formulas that $(A \setminus \mathsf{dom}(\rho)) \cap (A_i \setminus \mathsf{dom}(\rho_i)) = \emptyset$. This implies $A \cap A_i \subseteq \mathsf{dom}(\rho) \cap \mathsf{dom}(\rho_i)$. By the fact that $A \cap A_i = A \setminus (\Sigma_\vartheta \setminus A_i)$, we have $A \setminus (\Sigma_\vartheta \setminus A_i) \subseteq \mathsf{dom}(\rho) \cap \mathsf{dom}(\rho_i)$ which corresponds to (P3).

By Lemma 4.23, we obtain $\vdash \langle\langle A\rangle\rangle_\rho \bigcirc \bigwedge \Psi' \rightarrow \langle\langle \Sigma_\vartheta \setminus A_i\rangle\rangle_{\rho_i} \bigcirc \bigwedge \Psi'$. Together with the fact $\vdash \neg\langle\langle \Sigma_\vartheta \setminus A_i\rangle\rangle_{\rho_i} \bigcirc \bigwedge \Psi'$, it follows that $\vdash \neg\langle\langle A\rangle\rangle_\rho \bigcirc \bigwedge \Psi'$. Thus $\vdash \neg(\bigwedge \Psi \wedge \psi \wedge \langle\langle A\rangle\rangle_\rho \bigcirc \bigwedge \Psi')$.

(c) Suppose $\vdash \neg\Phi^{\neg}_{(\Psi,\vec{t})}(\bigwedge \Psi')$. This is equivalent to $\vdash \bigvee_{i=1..y} \neg\langle\langle B_i\rangle\rangle_{\xi_i} \bigcirc \bigwedge \Psi'$, i.e., we have $\vdash \neg\langle\langle B_i\rangle\rangle_{\xi_i} \bigcirc \bigwedge \Psi'$ for some i with $1 \leq i \leq y$. Note that $\vec{t} = (t_0, \ldots, t_{n-1})$ is a $\langle\langle A\rangle\rangle_\rho \bigcirc \langle\langle A\rangle\rangle_\rho \psi \mathcal{U} \varphi$-vector and a $\neg\langle\langle B_i\rangle\rangle_{\xi_i} \bigcirc \varphi'_i$-vector. We show that the conditions (P0) to (P3) in Lemma 4.23 are satisfied:

- By Definition 4.13 of $\vec{t}$ being a vector for both $\langle\langle A\rangle\rangle_\rho \bigcirc \langle\langle A\rangle\rangle_\rho \psi \mathcal{U} \varphi$ and $\neg\langle\langle B_i\rangle\rangle_{\xi_i} \bigcirc \varphi'_i$, we have that $\rho(a) = \xi_i(a)$, for all agents $a \in \mathsf{dom}(\rho) \cap \mathsf{dom}(\xi_i)$. This corresponds to (P0).
- We show $(\mathsf{dom}(\xi_i) \setminus \mathsf{dom}(\rho)) \cap A = \emptyset$ which corresponds to (P1). To this end, let $a \in \mathsf{dom}(\xi_i) \setminus \mathsf{dom}(\rho)$. By Definition 4.13 of $\neg\langle\langle B_i\rangle\rangle_{\xi_i} \bigcirc \varphi'_i$-vector $\vec{t}$, we have $-\ell \leq t_a < 0$. Then $a \notin A$ by Definition 4.13 of $\vec{t}$ also being a $\langle\langle A\rangle\rangle_\rho \bigcirc \langle\langle A\rangle\rangle_\rho \psi \mathcal{U} \varphi$-vector. Thus $(\mathsf{dom}(\xi_i) \setminus \mathsf{dom}(\rho)) \cap A = \emptyset$.
- Since $\vec{t}$ is a $\neg\langle\langle B_i\rangle\rangle_{\xi_i} \bigcirc \varphi'_i$-vector, we have, by property 2 of the refutation function used in Definition 4.13, that $\mathsf{dom}(\rho) \subseteq B_i \cup \mathsf{dom}(\xi_i)$. Then $\mathsf{dom}(\rho) \setminus \mathsf{dom}(\xi_i) \subseteq B_i$ which corresponds to (P2).
- Since $\vec{t}$ is a $\neg\langle\langle B_i\rangle\rangle_{\xi_i} \bigcirc \varphi'_i$-vector, property 2 of the refutation function used in Definition 4.13 yields that $A \subseteq B_i \cup \mathsf{dom}(\xi_i)$. Clearly, we have $A \setminus B_i \subseteq \mathsf{dom}(\xi_i)$. We now show that $A \setminus B_i \subseteq \mathsf{dom}(\rho)$. Thus $A \setminus B_i \subseteq \mathsf{dom}(\rho) \cap \mathsf{dom}(\xi_i)$ which corresponds to (P3). To this end, let $a \in A \setminus B_i$. Then $a \in \mathsf{dom}(\xi_i)$. By Definition 4.13 of $\vec{t}$ being a $\neg\langle\langle B_i\rangle\rangle_{\xi_i} \bigcirc \varphi'_i$-vector, it holds that $-\ell \leq t_a < 0$. But then $a \notin A \setminus \mathsf{dom}(\rho)$ by Definition 4.13 of $\vec{t}$ also being a $\langle\langle A\rangle\rangle_\rho \bigcirc \langle\langle A\rangle\rangle_\rho \psi \mathcal{U} \varphi$-vector. Thus $a \in \mathsf{dom}(\rho)$.

By Lemma 4.23, we get $\vdash \langle\langle A\rangle\rangle_\rho \bigcirc \bigwedge \Psi' \rightarrow \langle\langle B_i\rangle\rangle_{\xi_i} \bigcirc \bigwedge \Psi'$. Together with the fact $\vdash \neg\langle\langle B_i\rangle\rangle_{\xi_i} \bigcirc \bigwedge \Psi'$, it follows that $\vdash \neg\langle\langle A\rangle\rangle_\rho \bigcirc \bigwedge \Psi'$. Thus $\vdash \neg(\bigwedge \Psi \wedge \psi \wedge \langle\langle A\rangle\rangle_\rho \bigcirc \bigwedge \Psi')$.

We have established the implication (†). Consequently, T' is a $\langle\langle A\rangle\rangle_\rho \psi \mathcal{U} \varphi$-witness tree rooted at Ψ in Δ, which is in contradiction to $\Psi \in V$. Thus we have shown that

$\vdash \theta \rightarrow \neg\varphi \wedge (\neg\psi \vee \neg\langle\!\langle A\rangle\!\rangle_\rho \bigcirc \neg\theta)$ (0). In the following derivation, we show that Ψ is inconsistent.

(1) $\vdash$ $\varphi \vee (\psi \wedge \langle\!\langle A\rangle\!\rangle_\rho \bigcirc \neg\theta) \rightarrow \neg\theta$ (using (0))

(2) $\vdash$ $\langle\!\langle \emptyset\rangle\!\rangle_\rho \Box((\varphi \vee (\psi \wedge \langle\!\langle A\rangle\!\rangle_\rho \bigcirc \neg\theta)) \rightarrow \neg\theta)$ (using (1) and $\langle\!\langle \emptyset\rangle\!\rangle_\rho\Box$-Necessitation)

(3) $\vdash$ $\langle\!\langle \emptyset\rangle\!\rangle_\rho \Box(\langle\!\langle A\rangle\!\rangle_\rho \psi \mathcal{U} \varphi \rightarrow \neg\theta)$ (using (2) and (LFP$_\mathcal{U}$))

(4) $\vdash$ $\langle\!\langle A\rangle\!\rangle_\rho \psi \mathcal{U} \varphi \rightarrow \neg\theta$ (using (3) and (FP$_\Box$))

(5) $\vdash$ $\theta \rightarrow \neg\langle\!\langle A\rangle\!\rangle_\rho \psi \mathcal{U} \varphi$ (using (4))

(6) $\vdash$ $\bigwedge \Psi \rightarrow \theta$ (by definition of θ)

(7) $\vdash$ $\bigwedge \Psi \rightarrow \neg\langle\!\langle A\rangle\!\rangle_\rho \psi \mathcal{U} \varphi$ (using (5) and (6))

(8) $\vdash$ $\bigwedge \Psi \rightarrow \langle\!\langle A\rangle\!\rangle_\rho \psi \mathcal{U} \varphi$ (since $\langle\!\langle A\rangle\!\rangle_\rho \psi \mathcal{U} \varphi \in \Psi$)

(9) $\vdash$ $\bigwedge \Psi \rightarrow \bot$ (using (7) and (8))

Hence Ψ is inconsistent.

Finally, consider Case (iii). This case is shown similarly to Case (ii). Supposing Case (iii), we have $\neg\langle\!\langle A\rangle\!\rangle_\rho \Box\varphi \in \Psi$ and that there is no $\neg\langle\!\langle A\rangle\!\rangle_\rho \Box\varphi$-witness tree rooted at Ψ in Δ. It is to show that Ψ is inconsistent. Let V be the set of types $\Psi' \in \Delta$ such that either $\neg\langle\!\langle A\rangle\!\rangle_\rho \Box\varphi \in \Psi'$ but there is no $\neg\langle\!\langle A\rangle\!\rangle_\rho \Box\varphi$-witness tree rooted at Ψ' in Δ, or $\neg\langle\!\langle A\rangle\!\rangle_\rho \Box\varphi \notin \Psi'$. Note that $\Psi \in V$. Let

$$\theta = \bigvee\{\bigwedge \Psi' \mid \Psi' \in V\}.$$

By the induction hypothesis, all types in $\Gamma_{\neg\vartheta} \setminus \Delta$ are inconsistent. Thus we have

$$\neg\theta = \bigvee\{\bigwedge \Psi' \mid \Psi' \in \Delta \setminus V\}.$$

In the following, we show that

$$\vdash \theta \rightarrow \varphi \wedge \langle\!\langle A\rangle\!\rangle_\rho \bigcirc \theta.$$

Notice that $\varphi \in \Psi'$, for all $\Psi' \in V$. To see that, suppose the contrary, i.e., $\varphi \notin \Psi'$ for some $\Psi' \in V$. By (T2), we have $\neg\varphi \in \Psi'$ (or $\varphi' \in \Psi'$ if $\varphi = \neg\varphi'$). But then, if $\neg\langle\!\langle A\rangle\!\rangle_\rho \Box\varphi \in \Psi'$, a $\neg\vartheta$-tree containing only a root labelled with Ψ' would be a $\neg\langle\!\langle A\rangle\!\rangle_\rho \Box\varphi$-witness tree rooted at Ψ' in Δ; contradicting $\Psi' \in V$. Otherwise, if $\neg\langle\!\langle A\rangle\!\rangle_\rho \Box\varphi \notin \Psi'$, (T2) yields $\langle\!\langle A\rangle\!\rangle_\rho \Box\varphi \in \Psi'$, and (T3) that $\varphi \in \Psi'$; again a contradiction. Consequently, we have $\vdash \bigwedge \Psi' \rightarrow \varphi$ for all $\Psi' \in V$, and thus $\vdash \theta \rightarrow \varphi$. In order to show $\vdash \theta \rightarrow \langle\!\langle A\rangle\!\rangle_\rho \bigcirc \theta$, it suffices to show $\vdash \bigwedge \Psi \rightarrow \langle\!\langle A\rangle\!\rangle_\rho \bigcirc \theta$. Suppose by contradiction that $\nvdash \bigwedge \Psi \rightarrow \langle\!\langle A\rangle\!\rangle_\rho \bigcirc \theta$. Together with Regularity, it follows that $\nvdash \neg(\bigwedge \Psi \wedge \langle\!\langle \Sigma_\vartheta \setminus A\rangle\!\rangle_\rho \bigcirc \neg\theta)$. Note that $\neg\theta = \bigvee\{\bigwedge \Psi' \mid \Psi' \in \Delta \setminus V\}$. Then we have, for some $\Psi' \in \Delta \setminus V$, that

$$\nvdash \neg(\bigwedge \Psi \wedge \langle\!\langle \Sigma_\vartheta \setminus A\rangle\!\rangle_\rho \bigcirc \bigwedge \Psi').$$

Fix such a Ψ'. Since $\Psi' \in \Delta \setminus V$, there is a $\neg\langle\!\langle A\rangle\!\rangle_\rho\Box\varphi$-witness tree T rooted at Ψ' in Δ. Using T, we construct a $\neg\langle\!\langle A\rangle\!\rangle_\rho\Box\varphi$-witness tree T' rooted at Ψ in Δ as follows:

- $T'(\varepsilon) := \Psi$;
- $T'(\vec{t}\cdot\alpha) := T(\alpha)$, for all $\neg\langle\!\langle A\rangle\!\rangle_\rho\bigcirc\langle\!\langle A\rangle\!\rangle_\rho\Box\varphi$-vectors $\vec{t}$ and all $\alpha \in \mathsf{dom}(T)$.

It can readily be checked that T' satisfies conditions 1 and 3 to 6 in Definition 4.14 of $\neg\langle\!\langle A\rangle\!\rangle_\rho\Box\varphi$-witness trees. For condition 4, notice that $\neg\langle\!\langle A\rangle\!\rangle_\rho\Box\varphi \in \Psi$. Consider condition 2 that requires that T' is $\bigcirc$-matching. Let $\vec{t}$ be a $\neg\langle\!\langle A\rangle\!\rangle_\rho\bigcirc\langle\!\langle A\rangle\!\rangle_\rho\Box\varphi$-vector. Given Ψ and $\vec{t}$, let $j_{\langle\Psi,\vec{t}\rangle}$ be the number ranging from 1 to y such that $\neg\langle\!\langle A\rangle\!\rangle_\rho\bigcirc\langle\!\langle A\rangle\!\rangle_\rho\Box\varphi$ is the $k_{\langle\Psi,\vec{t}\rangle}$-th formula in the enumeration of positive next formulas from Ψ for which $\vec{t}$ is a vector. Let $\chi'_{\langle\Psi,\vec{t}\rangle}$ abbreviate the following negated conjunction:

$$\chi'_{\langle\Psi,\vec{t}\rangle} = \neg(\bigwedge\Psi \wedge \Phi_{\langle\Psi,\vec{t}\rangle}(\bigwedge\Psi') \wedge \bigwedge_{\substack{i=1..y\\ i\neq k_{\langle\Psi,\vec{t}\rangle}}} \langle\!\langle B_i\rangle\!\rangle_{\xi_i}\bigcirc\bigwedge\Psi').$$

In the following, we show that

$$(\ddagger) \qquad \nvdash \neg(\bigwedge\Psi \wedge \langle\!\langle \Sigma_\vartheta \setminus A\rangle\!\rangle_\rho\bigcirc\bigwedge\Psi') \text{ implies } \nvdash \chi'_{\langle\Psi,\vec{t}\rangle}.$$

Then, since $\nvdash \neg(\bigwedge\Psi \wedge \langle\!\langle \Sigma_\vartheta \setminus A\rangle\!\rangle_\rho\bigcirc\bigwedge\Psi')$, we obtain from ($\ddagger$) that $\nvdash \chi'_{\langle\Psi,\vec{t}\rangle}$. Together with Claim 4.19 and the fact that $\neg\langle\!\langle A\rangle\!\rangle_\rho\bigcirc\langle\!\langle A\rangle\!\rangle_\rho\Box\varphi \in \Psi$ and $\langle\!\langle A\rangle\!\rangle_\rho\Box\varphi \in \Psi'$, we have $S_\Psi(\vec{t}) \subseteq \Psi'$, for every $\neg\langle\!\langle A\rangle\!\rangle_\rho\bigcirc\langle\!\langle A\rangle\!\rangle_\rho\Box\varphi$-vector $\vec{t}$. By Definition 4.13 and the fact that the tree T is already $\bigcirc$-matching, we have that T' is $\bigcirc$-matching. Hence condition 2 is satisfied.

We now show, for every $\neg\langle\!\langle A\rangle\!\rangle_\rho\bigcirc\langle\!\langle A\rangle\!\rangle_\rho\Box\varphi$-vector $\vec{t}$, the contrapositive of the implication ($\ddagger$). Suppose $\vdash \chi'_{\langle\Psi,\vec{t}\rangle}$. Distinguish the following three cases:

(a) Suppose $\vdash \neg\bigwedge\Psi$. Obviously, this implies $\vdash \neg(\bigwedge\Psi \wedge \langle\!\langle \Sigma_\vartheta \setminus A\rangle\!\rangle_\rho\bigcirc\bigwedge\Psi')$.

(b) Suppose $\vdash \neg\Phi_{\langle\Psi,\vec{t}\rangle}(\bigwedge\Psi')$. This is equivalent to $\vdash\bigvee_{i=1..x}\neg\langle\!\langle \Sigma_\vartheta \setminus A_i\rangle\!\rangle_{\rho_i}\bigcirc\bigwedge\Psi'$, i.e., $\vdash \neg\langle\!\langle \Sigma_\vartheta \setminus A_i\rangle\!\rangle_{\rho_i}\bigcirc\bigwedge\Psi'$ for some i with $1 \leq i \leq x$. Note that $\vec{t} = (t_0, \ldots, t_{n-1})$ is a $\neg\langle\!\langle A\rangle\!\rangle_\rho\bigcirc\langle\!\langle A\rangle\!\rangle_\rho\Box\varphi$-vector and a $\langle\!\langle A_i\rangle\!\rangle_{\rho_i}\bigcirc\varphi_i$-vector. We show that the conditions (P0) to (P3) in Lemma 4.23 are satisfied:
   - By Definition 4.13 of $\vec{t}$ being a vector for both $\neg\langle\!\langle A\rangle\!\rangle_\rho\bigcirc\langle\!\langle A\rangle\!\rangle_\rho\Box\varphi$ and $\langle\!\langle A_i\rangle\!\rangle_{\rho_i}\bigcirc\varphi_i$, we have that $\rho(a) = \rho_i(a)$, for all agents $a \in \mathsf{dom}(\rho)\cap\mathsf{dom}(\rho_i)$. This corresponds to (P0).
   - Since $\vec{t}$ is a $\neg\langle\!\langle A\rangle\!\rangle_\rho\bigcirc\langle\!\langle A\rangle\!\rangle_\rho\Box\varphi$-vector, it holds, by property 2 of the refutation function used in Definition 4.13, that $\mathsf{dom}(\rho_i) \subseteq A \cup \mathsf{dom}(\rho)$. Thus $(\mathsf{dom}(\rho_i) \setminus \mathsf{dom}(\rho)) \cap (\Sigma_\vartheta \setminus A) = \emptyset$ which corresponds to (P1).
   - We show $\mathsf{dom}(\rho) \setminus \mathsf{dom}(\rho_i) \subseteq \Sigma_\vartheta \setminus A_i$ which corresponds to (P2). To this end, let $a \in \mathsf{dom}(\rho) \setminus \mathsf{dom}(\rho_i)$. By Definition 4.13 of $\neg\langle\!\langle A\rangle\!\rangle_\rho\bigcirc\langle\!\langle A\rangle\!\rangle_\rho\Box\varphi$-vector $\vec{t}$, we have $-\ell \leq t_a < 0$. But then $a \notin A_i$ by Definition 4.13 of $\langle\!\langle A_i\rangle\!\rangle_{\rho_i}\bigcirc\varphi_i$-vector $\vec{t}$. That is, $a \in \Sigma_\vartheta \setminus A_i$. Thus $\mathsf{dom}(\rho) \setminus \mathsf{dom}(\rho_i) \subseteq \Sigma_\vartheta \setminus A_i$.

- Since $\vec{t}$ is a $\neg\langle\langle A\rangle\rangle_\rho\bigcirc\langle\langle A\rangle\rangle_\rho\Box\varphi$-vector, we have $A_i \subseteq A \cup \mathsf{dom}(\rho)$ by property 2 of the refutation function used in Definition 4.13. Clearly, we have $A_i \setminus A \subseteq \mathsf{dom}(\rho)$. We now show that $A_i \setminus A \subseteq \mathsf{dom}(\rho_i)$. Then $A_i \setminus A \subseteq \mathsf{dom}(\rho)\cap\mathsf{dom}(\rho_i)$ and, equivalently, $(\Sigma_\vartheta\setminus A)\setminus(\Sigma_\vartheta\setminus A_i) \subseteq \mathsf{dom}(\rho)\cap\mathsf{dom}(\rho_i)$ which corresponds to (P3). To this end, let $a \in A_i \setminus A$. Then $a \in \mathsf{dom}(\rho)$. By Definition 4.13 of $\neg\langle\langle A\rangle\rangle_\rho\bigcirc\langle\langle A\rangle\rangle_\rho\Box\varphi$-vector $\vec{t}$, we have $-\ell \leq t_a < 0$. But then $a \notin A_i\setminus\mathsf{dom}(\rho_i)$ by Definition 4.13 of $\vec{t}$ also being a $\langle\langle A_i\rangle\rangle_{\rho_i}\bigcirc\varphi_i$-vector. Thus $a \in \mathsf{dom}(\rho_i)$.

By Lemma 4.23, we obtain $\vdash \langle\langle\Sigma_\vartheta\setminus A\rangle\rangle_\rho\bigcirc\bigwedge\Psi' \to \langle\langle\Sigma_\vartheta\setminus A_i\rangle\rangle_{\rho_i}\bigcirc\bigwedge\Psi'$. Together with the fact $\vdash \neg\langle\langle\Sigma_\vartheta\setminus A_i\rangle\rangle_{\rho_i}\bigcirc\bigwedge\Psi'$, it follows that $\vdash \neg\langle\langle\Sigma_\vartheta\setminus A\rangle\rangle_\rho\bigcirc\bigwedge\Psi'$. Thus $\vdash \neg(\bigwedge\Psi \wedge \langle\langle\Sigma_\vartheta\setminus A\rangle\rangle_\rho\bigcirc\bigwedge\Psi')$.

(c) Suppose $\vdash \neg\bigwedge_{\substack{i=1..y\\ i\neq k_{(\Psi,\vec{t})}}} \langle\langle B_i\rangle\rangle_{\xi_i}\bigcirc\bigwedge\Psi'$. That is, we have $\vdash \neg\langle\langle B_i\rangle\rangle_{\xi_i}\bigcirc\bigwedge\Psi'$ for some $i \neq k_{(\Psi,\vec{t})}$ with $1 \leq i \leq y$. Note that $\vec{t} = (t_0,\ldots,t_{n-1})$ is a $\neg\langle\langle A\rangle\rangle_\rho\bigcirc\langle\langle A\rangle\rangle_\rho\Box\varphi$-vector and a $\neg\langle\langle B_i\rangle\rangle_{\xi_i}\bigcirc\varphi'_i$-vector, and that $\neg\langle\langle B_i\rangle\rangle_{\xi_i}\bigcirc\varphi'_i \neq \neg\langle\langle A\rangle\rangle_\rho\bigcirc\langle\langle A\rangle\rangle_\rho\Box\varphi$. Distinguish four subcases:
(c1) $B_i \cup \mathsf{dom}(\xi_i) = \Sigma_\vartheta$ and $A \cup \mathsf{dom}(\rho) = \Sigma_\vartheta$;
(c2) $B_i \cup \mathsf{dom}(\xi_i) = \Sigma_\vartheta$ and $A \cup \mathsf{dom}(\rho) \subset \Sigma_\vartheta$;
(c3) $B_i \cup \mathsf{dom}(\xi_i) \subset \Sigma_\vartheta$ and $A \cup \mathsf{dom}(\rho) = \Sigma_\vartheta$;
(c4) $B_i \cup \mathsf{dom}(\xi_i) \subset \Sigma_\vartheta$ and $A \cup \mathsf{dom}(\rho) \subset \Sigma_\vartheta$.
We now show that, for Case (c2), the conditions (P0) to (P3) in Lemma 4.23 are satisfied. For cases (c1) and (c3), this can be done similarly.

- By Definition 4.13 of $\vec{t}$ being a vector for both $\neg\langle\langle A\rangle\rangle_\rho\bigcirc\langle\langle A\rangle\rangle_\rho\Box\varphi$ and $\neg\langle\langle B_i\rangle\rangle_{\xi_i}\bigcirc\varphi'_i$, we have that $\rho(a) = \xi_i(a)$, for all agents $a \in \mathsf{dom}(\rho) \cap \mathsf{dom}(\xi_i)$. This corresponds to (P0).
- Since $\vec{t}$ is a $\neg\langle\langle A\rangle\rangle_\rho\bigcirc\langle\langle A\rangle\rangle_\rho\Box\varphi$-vector, it holds $\mathsf{dom}(\xi_i) \subseteq A \cup \mathsf{dom}(\rho)$ by property 2 of the refutation function used in Definition 4.13. Thus $(\mathsf{dom}(\xi_i)\setminus\mathsf{dom}(\rho)) \cap (\Sigma_\vartheta\setminus A) = \emptyset$ which corresponds to (P1).
- Since $\mathsf{dom}(\rho) \subseteq \Sigma_\vartheta = B_i \cup \mathsf{dom}(\xi_i)$, we have $\mathsf{dom}(\rho)\setminus\mathsf{dom}(\xi_i) \subseteq B_i$ which corresponds to (P2).
- By $\Sigma_\vartheta = B_i \cup \mathsf{dom}(\xi_i)$, we have $\Sigma_\vartheta \setminus B_i \subseteq \mathsf{dom}(\xi_i)$. Clearly, this implies $(\Sigma_\vartheta\setminus A)\setminus B_i \subseteq \mathsf{dom}(\xi_i)$. We now show that $(\Sigma_\vartheta\setminus A)\setminus B_i \subseteq \mathsf{dom}(\rho)$. Then we have $(\Sigma_\vartheta\setminus A)\setminus B_i \subseteq \mathsf{dom}(\rho)\cap\mathsf{dom}(\xi_i)$ which corresponds to (P3). To this end, let $a \in (\Sigma_\vartheta\setminus A)\setminus B_i$. Then $a \in \mathsf{dom}(\xi_i)$. By Definition 4.13 of $\neg\langle\langle B_i\rangle\rangle_{\xi_i}\bigcirc\varphi'_i$-vector $\vec{t}$, it holds that $-\ell \leq t_a < 0$. But then $a \in A\cup\mathsf{dom}(\rho)$ by property 2 of the refutation function used in Definition 4.13 of $\vec{t}$ also being a $\neg\langle\langle A\rangle\rangle_\rho\bigcirc\langle\langle A\rangle\rangle_\rho\Box\varphi$-vector. Finally, since $a \notin A$, we have $a \in \mathsf{dom}(\rho)$.

Lemma 4.23 yields $\vdash \langle\langle\Sigma_\vartheta\setminus A\rangle\rangle_\rho\bigcirc\bigwedge\Psi' \to \langle\langle B_i\rangle\rangle_{\xi_i}\bigcirc\bigwedge\Psi'$. Together with the fact $\vdash \neg\langle\langle B_i\rangle\rangle_{\xi_i}\bigcirc\bigwedge\Psi'$, it follows that $\vdash \neg\langle\langle\Sigma_\vartheta\setminus A\rangle\rangle_\rho\bigcirc\bigwedge\Psi'$. Thus $\vdash \neg(\bigwedge\Psi \wedge \langle\langle\Sigma_\vartheta\setminus A\rangle\rangle_\rho\bigcirc\bigwedge\Psi')$.

Consider Case (c4). By property 3 of the refutation function f used in Definition 4.13 of vectors of negative next formulas, we have that there is at most a single coalition $B \subset \Sigma_\vartheta$ with $f(\vec{t}, B)$ defined. Thus we have that $A \cup \mathsf{dom}(\rho) = B_i \cup \mathsf{dom}(\xi_i)$ and, since f is a function, that $f(\vec{t}, A \cup \mathsf{dom}(\rho)) = f(\vec{t}, B_i \cup \mathsf{dom}(\xi_i))$. But then $\neg\langle\langle B_i\rangle\rangle_{\xi_i}\bigcirc\varphi'_i = \neg\langle\langle A\rangle\rangle_\rho\bigcirc\langle\langle A\rangle\rangle_\rho\Box\varphi$; a contradiction. Thus we can exclude (c4).

We have established the implication (‡). Consequently, T' is a $\neg\langle\langle A\rangle\rangle_\rho\Box\varphi$-witness tree rooted at Ψ in Δ, which is in contradiction to $\Psi \in V$. Thus we have shown that $\vdash \theta \to \varphi \wedge \langle\langle A\rangle\rangle_\rho\bigcirc\theta$ (0). With the following derivation we show that Ψ is inconsistent:

(1) $\vdash$ $\langle\langle\emptyset\rangle\rangle_\rho\Box(\theta \to \varphi \wedge \langle\langle A\rangle\rangle_\rho\bigcirc\theta)$ (using (0) and $\langle\langle\emptyset\rangle\rangle_\rho\Box$-Necessitation)

(2) $\vdash$ $\langle\langle\emptyset\rangle\rangle_\rho\Box(\theta \to \langle\langle A\rangle\rangle_\rho\Box\varphi)$ (using (1) and (GFP$_\Box$))

(3) $\vdash$ $\theta \to \langle\langle A\rangle\rangle_\rho\Box\varphi$ (using (2) and (FP$_\Box$))

(4) $\vdash$ $\bigwedge\Psi \to \theta$ (by definition of θ)

(5) $\vdash$ $\bigwedge\Psi \to \langle\langle A\rangle\rangle_\rho\Box\varphi$ (using (3) and (4))

(6) $\vdash$ $\bigwedge\Psi \to \neg\langle\langle A\rangle\rangle_\rho\Box\varphi$ (since $\neg\langle\langle A\rangle\rangle_\rho\Box\varphi \in \Psi$)

(7) $\vdash$ $\bigwedge\Psi \to \bot$ (using (5) and (6))

Hence Ψ is inconsistent. □

4.6. Conclusion

In this chapter, we introduced ATLES as an extension of ATL with explicit names for strategies. ATLES makes it possible to refer to the same strategy in different occurrences of path quantifiers, and, as a consequence, it possible to express in ATLES some properties that cannot be expressed even in ATL*. We argued that the expressivity of ATLES makes it suitable for reasoning about extensive games.

We showed that ATLES satisfiability is ExpTime-complete by extending the type elimination construction for ATL from Chapter 3. We presented a sound and complete axiomatisation for ATLES, the completeness proof of which is based on that type elimination construction. Moreover, we identified two variants of the model checking problem for ATLES and settled their computational complexities.

ATLES does neither fix the number of agents nor the available strategy terms in advance and thus contains the logics CATL [**242**], Coalition Logic [**195**], Action Logic [**38**], ATL [**265**], ATL$_\Sigma$ [**98**] and ATLES $_{\Sigma,\Upsilon}$ as fragments, where the parameter Σ is a fixed set of agents and the parameter Υ a fixed set of strategy terms. Thus the upper complexity bound for ATLES carries over to these fragments whose validities can, moreover, be derived using the axiomatic system for ATLES.

For future work, it is generally interesting to enrich the expressivity of ATLES while keeping its complexity low. We can model, e.g., *non-deterministic strategies* with "disjunction" of strategies: The formula $\langle\langle\rangle\rangle_{\{a\mapsto\varrho_1\vee\varrho_2\}}\Phi$ states that agent a uses the non-deterministic strategy $\varrho_1 \vee \varrho_2$, where, at each state, a chooses either according to strategy ϱ_1 or to strategy ϱ_2 to bring about the temporal expression Φ. Other interesting operators for composing strategies are thinkable. In the context of strategy composition, it seems relevant to mention the possibility of *partial strategies* that determine choices only at some states. Composed partial strategies can complement each other and we can model, e.g., agents who switch strategies after some system transitions. It would be intriguing to have an operator for *dynamic strategies* that evolve over time that can be used to model games more closely. Extending ATLES with "disjunction" of strategies and other operators for composing strategies enables us to succinctly express complex properties about strategic ability of agents.

A further suggestion for an ATL-variant with strategies in the object language is to allow for *strategy variables* and their quantification; cf. Borgo [**38**]. One idea is to introduce explicit *strategy quantifiers* and allow for formulas of the form $\exists x_a.\forall x_b.\exists x_c.\varphi$. In this way, we break the fixed quantification pattern of the ATL path quantifiers which, on the other hand, might easily render the resulting logic undecidable. An implicit way of quantifying strategy variables is by employing the quantification hidden in the ATL path quantifiers. For instance, in a formula of the form $\langle\langle a\rangle\rangle_{\{a:x_a,b:x_b\}}\Diamond(\langle\langle\rangle\rangle_{\{a\mapsto x_a\}}\Phi \wedge \langle\langle a\rangle\rangle_{\{b\mapsto x_b\}}\Phi')$, the strategy variable x_a for agent a is existentially quantified since a is a member of the coalition in the path quantifier and, since agent b is not in that coalition, x_b is universally quantified. Notice that this quantification corresponds to the semantics of path quantifier $\langle\langle a\rangle\rangle$. In the second and third path quantifier of this formula, it is demonstrated how a and b commit to x_a and x_b, respectively. Of course, this setting can be more refined, e.g., by specifying a range of strategies over which it is quantified.

CHAPTER 5

Cooperation and Transfer of Control

As we saw in Chapter 2, many logics for reasoning about the strategic abilities of agents and coalitions of agents in game-like multi-agent systems have been developed. However, these logics do not discuss the origins of an agent's abilities.

One exception is the MOCHA system for model checking coalitional power properties, in which powers are specified by defining, for every variable in a system, a unique agent that *controls* this variable [**17**]. Control, in this sense, means the unique ability to choose a value for this variable. Motivated by this observation, van der Hoek and Wooldridge [**249**] developed *Coalition Logic of Propositional Control* (CL-PC), a cooperation logic in which powers are specified by allocating every propositional variable to a unique agent in the system: the choices (and hence powers) available to a coalition then correspond to the possible assignments of truth or falsity that may be made to the variables under their control. The CL-PC modal expression $\Diamond_C \varphi$ means that coalition C can assign values to the variables under its control in such a way as to make φ true. Van der Hoek and Wooldridge gave a complete axiomatisation of CL-PC, and showed that the model checking and satisfiability problems for the logic are both PSPACE-complete. However, one drawback of CL-PC is that the power structures underpinning the logic – that is, the allocation of variables to agents – is *fixed*. Hence, ultimately, coalitional powers remain static in CL-PC.

In this chapter, we study a variant of CL-PC which allows us to reason about *dynamic* power structures. *Dynamic Coalition Logic of Propositional Control* (DCL-PC) extends CL-PC with dynamic logic operators as used by Harel, Tiuryn and Kozen [**107**], in which atomic actions are of the form $a_1 \leadsto_p a_2$, which is read as 'agent a_1 gives variable p to agent a_2'. The pre-condition of such an action is that variable p is in agent a_1's allocation of variables, and executing the program has the effect of *transferring control of variable p from agent a_1 to agent a_2*. Thus the dynamic component of DCL-PC is concerned with *delegating control* in systems, and by using the logic, we can reason about how the abilities of agents are affected by the transfer of control of variables in the system. Note that, as in conventional dynamic logic, atomic programs may be combined in DCL-PC with the usual sequential composition (';'), non-deterministic choice ('$\cup$'), test ('?') and iteration ('*') operations, to construct complex delegation programs. For example, the following DCL-PC-formula asserts that, if agent i gives either p or q to j, then j will be able to achieve φ:

$$[(i \leadsto_p j) \cup (i \leadsto_q j)]\Diamond_j \varphi.$$

The remainder of the chapter, is organized as follows. In Section 5.1, we give two alternative semantics for the logic: a "direct" semantics, in which models directly represent the allocation of propositional variables to the agents that control them, and a more conventional Kripke semantics. We prove that these two semantics are equivalent. The rationale for introducing two semantics is that, while the "direct" semantics is closer to the intended interpretation of the logic, the Kripke semantics is simpler for the purposes of, e.g., proving completeness. We give an axiomatisation of DCL-PC in Section 5.2, and show that this axiomatisation is complete (with respect to both semantics). Although one might expect that the additional dynamic logic operators in DCL-PC lead to higher a complexity than CL-PC wrt. model checking and satisfiability, we show in Section 5.3 that this is not the case. The satisfiability and model checking problems for DCL-PC are no more complex than the corresponding problems for CL-PC [**249**]: they are both PSPACE-complete. This result is, perhaps, of some technical interest in its own right, since EXPTIME-completeness seems to be the characteristic complexity of dynamic-like logics [**107**, pp.277–279]. Here, with "dynamic-like" logics we mean logics in which there is a modal operator and another operator representing the transitive closure of this operator [**33**, p.401]. Finally, we investigate the characterisation of *control* in DCL-PC. We distinguish between *first-order control* and *second-order control* in Section 5.4. While first-order control (as introduced by van der Hoek and Wooldridge [**249**]) is the ability to control some state of affairs by assigning values to variables, second-order control is the ability of an agent to exert control over the ability of other agents to control states of affairs. Agents and coalitions can exercise second-order control by transferring variables under their control to other agents. After informally discussing and introducing second-order control, we develop a logical characterisation of it, in the sense that we characterise the formulas over which an agent has second-order control. We conclude this chapter with some brief comments on related work.

5.1. The Logic DCL-PC

The language of DCL-PC is formed with respect to a set Σ of agents, and a set Π of propositional variables. DCL-PC extends classical propositional logic with "cooperation modalities" of the form $\Diamond_C \varphi$, where $C \subseteq \Sigma$ is a set of agents. These modalities are used to express *contingent ability* [**249**]: $\Diamond_C \varphi$ means that, under the assumption that the world remains otherwise unchanged, the set of agents C have the ability to achieve φ. As shown in [**249**], (and as defined below), stronger ability operators, roughly corresponding to Pauly's [**194**] Coalition Logic cooperation modality may be derived from these: we express the fact that the agents in coalition C have a choice such that, no matter what the agents outside C do, φ will become true, as $\langle\langle C \rangle\rangle_\alpha \varphi$ (the 'α' is for 'α-effectivity' [**194**, p.20]). Additionally, DCL-PC provides *dynamic delegation modalities*, of the form $[\delta]\varphi$, which means 'after the delegation program δ is executed, then φ will hold'. Delegation programs express the *transfer of control* between agents, and are built from a set of atomic delegation expressions of the form $i \rightsquigarrow_p j$, meaning 'agent i gives the control over propositional variable p to agent j'. Such a program can be executed if, and only if, agent i "owns" this variable, and the effect is to transfer ownership to j. These atomic programs may then be combined with the usual program constructs (;, ?, $\cup$, *) of regular dynamic logic [107], and of course, conventional program constructs such as while and if may be defined in terms of these. Executing such a program changes the distribution of ownership of propositional variables among the agents and, thus, the abilities of agents and coalitions. We can reason about the effect that such dynamic power allocation programs have on the abilities of agents and coalitions. For instance, the following formula states that it is possible for agent a to give away the control over its propositional variables in Π_a to agent b, non-deterministically choosing one variable at a time, until agent b is able to make φ true:

$$\langle \mathtt{while}\ \neg\Diamond_b\varphi\ \mathtt{do}\ \bigcup_{p\in\Pi_a} a \rightsquigarrow_p b\rangle\top.$$

5.1.1. DCL-PC Syntax. The syntax of DCL-PC is defined as follows.

DEFINITION 5.1. (DCL-PC SYNTAX). Let Π be a finite set of atomic propositions and Σ a finite set of agents. The set of DCL-PC-formulas φ and DCL-PC-delegation programs is simultaneously defined as given by the following BNF specification:

$$\begin{array}{rcccccc} \varphi & ::= & p & \mid \neg\varphi & \mid \varphi\vee\varphi & \mid \Diamond_C\varphi & \mid \langle\delta\rangle\varphi \\ \delta & ::= & a\rightsquigarrow_p b & \mid \delta;\delta & \mid \delta\cup\delta & \mid \delta^* & \mid \varphi? \end{array}$$

where p ranges over atomic propositions in Π, a, b range over the agents in Σ and C over finite subsets of Σ.

Logical truth ($\top$), falsehood ($\bot$) and the Boolean connectives ($\wedge$, $\rightarrow$ and $\leftrightarrow$) are defined as usual. We use $\Box_C\varphi := \neg\Diamond_C\neg\varphi$ and $[\delta]\varphi := \neg\langle\delta\rangle\neg\varphi$. Moreover, we use exclusive disjunction ($\bigtriangledown$), which is defined as follows: for any set Φ of DCL-PC-formulas,

$$\bigtriangledown_{\Phi} := \bigvee_{\varphi\in\Phi}\varphi \wedge \bigwedge_{\varphi_1\neq\varphi_2\in\Phi}\neg(\varphi_1\wedge\varphi_2).$$

We will also write $\varphi_1 \bigtriangledown \varphi_2 \bigtriangledown \cdots \bigtriangledown \varphi_n$ for $\bigtriangledown_{\varphi\in\Phi}$ with $\Phi = \{\varphi_1, \varphi_2, \ldots, \varphi_n\}$. With respect to delegation programs, while and if constructs are defined as follows [107, p.167]:

$$\begin{array}{rcl} \mathtt{if}\ \varphi\ \mathtt{then}\ \delta_1\ \mathtt{else}\ \delta_2 & := & ((\varphi?;\delta_1)\cup(\neg\varphi?;\delta_2)) \\ \mathtt{while}\ \varphi\ \mathtt{do}\ \delta & := & ((\varphi?;\delta)^*;\neg\varphi?) \end{array}$$

A DCL-PC-formula containing no modalities is said to be an *objective* formula.

Let Π_φ denote the set of propositional variables occurring in DCL-PC-formula φ, and let Σ_φ denote the set of all agents that occur in φ.

5.1.2. Direct Semantics for DCL-PC. We now introduce the first of two semantics for DCL-PC. Given a fixed, finite and non-empty set $\Sigma = \{1, \ldots, n\}$ of agents, and a fixed, finite and non-empty set Π of propositional variables, we say an *allocation* of Π to Σ is an indexed tuple $\xi = \langle\Pi_1, \ldots, \Pi_n\rangle$, where there is an indexed element Π_a for each agent $a \in \Sigma$, such that $\Pi_1, \ldots, \Pi_n$ forms a partition of Π (i.e., $\Pi = \bigcup_{a\in\Sigma}\Pi_a$

and $\Pi_a \cap \Pi_b = \emptyset$ for all $a \neq b \in \Sigma$). The intended interpretation of an allocation $\xi = \langle \Pi_1, \ldots, \Pi_n \rangle$ is that $\Pi_a \subseteq \Pi$ is the set of propositional variables under agent a's control. That is, agent a has freedom to allocate whatever Boolean values it sees fit to the members of Π_a.

DEFINITION 5.2. (DIRECT DCL-PC STRUCTURE). A *direct DCL-PC structure* is a tuple $\mathcal{M} = \langle \Pi, \Sigma, \xi_0, \theta \rangle$ where

- Π is a finite, non-empty set of *propositional variables*;
- $\Sigma = \{a_1, \ldots, a_n\}$ is a finite, non-empty set of *agents*;
- $\xi_0 = \langle \Pi_{a_1}, \ldots, \Pi_{a_n} \rangle$ is the *initial allocation* of Π to Σ, with the intended interpretation that Π_a is the subset of Π representing those variables under the control of agent $a \in \Sigma$; and
- $\theta : \Pi \to \{\text{tt}, \text{ff}\}$ is a *propositional valuation function*, which determines the initial truth value of every propositional variable.

Some additional notation is convenient in what follows. For any coalition $C \subseteq \Sigma$, we denote the *complement* of C, (i.e., $\Sigma \setminus C$) by $\overline{C}$. We will write Π_C for $\bigcup_{i \in C} \Pi_i$. For two valuations θ and θ', and a set of propositional variables $\Psi \subseteq \Pi$, we write $\theta = \theta'$ (mod Ψ) if θ and θ' differ at most in the propositional variables in Ψ, and we then say that θ and θ' are the *same modulo* Ψ. Given a direct structure $\mathcal{M} = \langle \Pi, \Sigma, \xi_0, \theta \rangle$ and a coalition C in $\mathcal{M}$, a *C-valuation* is a function:

$$\theta_C : \Pi_C \to \{\text{tt}, \text{ff}\}.$$

Thus a C-valuation is a partial propositional valuation function that assigns truth values to just the propositional variables controlled by the members of the coalition C. If $\mathcal{M} = \langle \Pi, \Sigma, \xi_0, \theta \rangle$ with $\xi_0 = \langle \Pi_1 \ldots, \Pi_n \rangle$ is a direct structure, C a coalition in $\mathcal{M}$, and θ_C a C-valuation, then by $\mathcal{M} \oplus \theta_C$ we mean the structure $\langle \Sigma, \Pi, \xi_0, \theta' \rangle$, where θ' is the valuation function defined as follows

$$\theta'(p) := \begin{cases} \theta_C(p) & \text{if } p \in \Pi_C \\ \theta(p) & \text{otherwise} \end{cases}$$

and all other elements of the structure are as in $\mathcal{M}$. Thus $\mathcal{M} \oplus \theta_C$ denotes the direct structure that is identical to $\mathcal{M}$ except that the values assigned by its valuation function to propositional variables controlled by members of C are determined by θ_C.

We define the *size* of a direct structure $\mathcal{M} = \langle \Pi, \Sigma, \xi_0, \theta \rangle$ to be $|\Pi| + |\Sigma|$; we denote the size of $\mathcal{M}$ by *size*($\mathcal{M}$).

To give a modal semantics to the dynamic logic constructs of DCL-PC, we must define, for every delegation program δ a binary relation R_δ over direct structures such that $(\mathcal{M}_1, \mathcal{M}_2) \in R_\delta$ iff $\mathcal{M}_2$ is a direct structure that may result from one possible execution of program δ at $\mathcal{M}_1$. We start by defining the relation $R_{a \leadsto_p b}$, for atomic delegation programs of the form $a \leadsto_p b$, i.e., agent a gives control of propositional variable p to agent b. Let $\mathcal{M} = \langle \Pi, \Sigma, \xi_0, \theta \rangle$ and $\mathcal{M}' = \langle \Pi', \Sigma', \xi_0', \theta' \rangle$ be two models with $\xi_0 = \langle \Pi_1, \ldots, \Pi_n \rangle$ and $\xi_0' = \langle \Pi_1', \ldots, \Pi_n' \rangle$. Then

$$(\mathcal{M}, \mathcal{M}') \in R_{a \leadsto_p b}$$

iff

(1) $p \in \Pi_a$ (agent a controls p to begin with)
(2) in case $a = b$:
  (a) $\mathcal{M} = \mathcal{M}'$ (agent a gives p to herself, with no change in the structure)
(3) in case $a \neq b$:
  (a) $\Pi_a' = \Pi_a \setminus \{p\}$ (agent a no longer controls p afterwards);
  (b) $\Pi_b' = \Pi_b \cup \{p\}$ (agent b controls p afterwards); and
  (c) all other components of $\mathcal{M}'$ are as in $\mathcal{M}$.

In order to define $\mathcal{M} \models^{\mathsf{d}} \varphi$, which means that φ is true in $\mathcal{M}$ under the direct semantics, we need to be able to determine what the interpretation of an arbitrary program is, on $\mathcal{M}$; we define this below. Notice that executing an atomic delegation program has *no effect* on the valuation function of a direct structure. Delegation programs *only* affect the distribution of propositional variables to agents.

For the remaining constructs of delegation programs, we define the program relations inductively, in terms of the relations for atomic delegation programs, as defined above. Let the composition of relations R_1 and R_2 be denoted by $R_1 \circ R_2$, and the reflexive transitive closure (ancestral) of relation R by R^*. Then the accessibility relations for complex programs are defined as follows [**107**, p.168]:

$$\begin{array}{rcl} R_{\delta_1;\delta_2} & := & R_{\delta_1} \circ R_{\delta_2}; \\ R_{\delta_1 \cup \delta_2} & := & R_{\delta_1} \cup R_{\delta_2}; \\ R_{\delta^*} & := & (R_\delta)^*; \\ R_{\varphi?} & := & \{(\mathcal{M}, \mathcal{M}) \mid \mathcal{M} \models^{\mathsf{d}} \varphi\}. \end{array}$$

We interpret formulas of DCL-PC with respect to direct structures, as introduced in Definition 5.2.

DEFINITION 5.3. (DCL-PC DIRECT SEMANTICS). Given a direct DCL-PC structure $\mathcal{M} = \langle \Pi, \Sigma, \xi_0, \theta \rangle$, the satisfaction relation $\models^{\mathsf{d}}$ is inductively defined as follows, where δ ranges over delegation programs, $C \subseteq \Sigma$ ranges over coalitions of agents, p over propositional variables in Π, and φ, ψ range over DCL-PC-formulas:

- $\mathcal{M} \models^{\mathsf{d}} p$ iff $\theta(p) = \mathrm{tt}$;
- $\mathcal{M} \models^{\mathsf{d}} \neg\varphi$ iff $\mathcal{M} \not\models^{\mathsf{d}} \varphi$;
- $\mathcal{M} \models^{\mathsf{d}} \varphi \vee \psi$ iff $\mathcal{M} \models^{\mathsf{d}} \varphi$ or $\mathcal{M} \models^{\mathsf{d}} \psi$;
- $\mathcal{M} \models^{\mathsf{d}} \Diamond_C \varphi$ iff there is a C-valuation θ_C such that $\mathcal{M} \oplus \theta_C \models^{\mathsf{d}} \varphi$;
- $\mathcal{M} \models^{\mathsf{d}} \langle \delta \rangle \varphi$ iff there is a direct structure $\mathcal{M}'$ such that $(\mathcal{M}, \mathcal{M}') \in R_\delta$ and $\mathcal{M}' \models^{\mathsf{d}} \varphi$.

A DCL-PC-formula φ is ${}^{\mathsf{d}}$-*satisfiable* iff there is a direct DCL-PC structure $\mathcal{M}$ such that $\mathcal{M} \models^{\mathsf{d}} \varphi$, and φ is ${}^{\mathsf{d}}$-*valid* if, and only if, for all direct DCL-PC structures $\mathcal{M}$, we have $\mathcal{M} \models^{\mathsf{d}} \varphi$. We write $\models^{\mathsf{d}} \varphi$ to indicate that φ is $^{\mathsf{d}}$-valid. A valid formula is also called *tautology*.

Where C is a coalition and φ is a formula of DCL-PC, we write $\mathit{controls}(C, \varphi)$ to mean that C can choose φ to be either true or false:

$$\mathit{controls}(C, \varphi) \;:=\; \Diamond_C \varphi \wedge \Diamond_C \neg\varphi \tag{5.1}$$

By using the $\mathit{controls}(\cdot, \cdot)$ construct, we can capture the distribution of propositional variables among the agents in a structure.

LEMMA 5.4. Let $\mathcal{M} = \langle \Pi, \Sigma, \xi_0, \theta \rangle$ be a direct DCL-PC structure, $a \in \Sigma$ an agent, $C \subseteq \Sigma$ a set of agents, and $p \in \Pi$ a propositional variable in $\mathcal{M}$. Then

(1) ([**249**]) $\mathcal{M} \models^{\mathsf{d}} \mathit{controls}(a, p)$ iff $p \in \Pi_a$;
(2) $\mathcal{M} \models^{\mathsf{d}} \mathit{controls}(C, p)$ iff $p \in \Pi_C$.

Can we characterise the formulas under control of a coalition C? We have that:

$$\text{If } \varphi \text{ is objective, then } \models^{\mathsf{d}} \Big(\bigwedge_{p \in \Pi_\varphi} \mathit{controls}(C, p)\Big) \rightarrow \mathit{controls}(C, \varphi) \tag{5.2}$$

Recall that φ is said to be objective if it does not contain any modalities. Property (5.2) is not true for arbitrary φ, for which $\langle i \leadsto_p j \rangle \top$ is a counterexample, no matter whether we define Π_i to be $\{p\}$ or $\emptyset$, which can be seen as follows. In fact, we have $\models^{\mathsf{d}} \neg \mathit{controls}(i, \langle i \leadsto_p j \rangle \top)$: independent of i owning p, exactly one of the two formulas $\langle i \leadsto_p j \rangle \top$ and $\neg \langle i \leadsto_p j \rangle \top$ is true. That is, $p \in \Pi_i$ in $\mathcal{M}$ if, and only if, $\mathcal{M} \models^{\mathsf{d}} \langle i \leadsto_p j \rangle \top$. Also we have that the reverse direction of the implication in the right hand side of (5.2) is not valid for objective φ: suppose $\mathcal{M} = \langle \Pi, \Sigma, \xi_0, \theta \rangle$ such that $\theta(q) = \mathrm{tt}$, $p \in \Pi_i$ and $q \notin \Pi_i$. Then, we have $\mathcal{M} \models^{\mathsf{d}} \mathit{controls}(i, p \wedge q) \wedge \neg(\mathit{controls}(i, p) \wedge \mathit{controls}(i, q))$: because q "happens" to be true in $\mathcal{M}$, i controls the conjunction $p \wedge q$, but not each of its conjuncts.

5.1.3. Kripke-style Semantics for DCL-PC.

For some purposes, it is more natural to formulate the semantics for DCL-PC using conventional Kripke structures [**51**, **33**]. Intuitively, given a set Σ of agents and a set Π of propositional variables, there will be a possible world for every possible allocation of the variables in Π to the agents in Σ and every possible propositional valuation function over Π. Between those worlds, there are basically two "orthogonal" accessibility relations (cf. Figure 5.1): a "horizontal" and a "vertical" one. First of all, we have, for each agent a, a "horizontal" relation R_a between two worlds u and v if agent a is able, given the valuation θ_u in u, to turn it into the valuation θ_v as described by v, just by choosing appropriate values for her variables. In what follows, we drop the symbolic distinction between worlds and valuations, i.e., we use θ for denoting a world and a valuation interchangeably. Notice that the "horizontal" relation does not affect the allocation ξ.

DEFINITION 5.5. (KRIPKE STRUCTURE FOR DCL-PC). A *Kripke structure for DCL-PC* is a tuple $\mathfrak{M} = \langle \Pi, \Sigma, \Theta, \{R_a\}_{a\in\Sigma}, \xi \rangle$ where

- Π is a finite, non-empty set of *propositional variables*;
- $\Sigma = \{a_1, \dots, a_n\}$ is a finite, non-empty set of *agents*;
- $\Theta = \{\theta \mid \theta \text{ is a valuation over } \Pi\}$ is a finite set of *states*;
- $R_a \subseteq \Theta \times \Theta$ is the *horizontal relation* for agent $a \in \Sigma$ such that $(\theta, \theta') \in R_a$ iff $\theta = \theta' \pmod{\Pi_a}$, i.e., R_a is an equivalence relation;
- $\xi = \langle \Pi_{a_1}, \dots, \Pi_{a_n} \rangle$ is an *allocation* of Π to Σ such that, for every agent $a \in \Sigma$, $\Pi_a \subseteq \Pi$ contains the propositional variables under the control of a.

The set of all Kripke models for DCL-PC over Σ and Π is denoted by $\mathcal{K}(\Sigma, \Pi)$.

It is important to realize that the sets Σ of agents and Π of variables are fixed, but the allocations of variables to agents may vary.

Secondly, the "vertical" accessibility relation is between pointed Kripke structures $(\mathfrak{M}, \theta)$ and $(\mathfrak{M}', \theta')$, where $\mathfrak{M} = \langle \Pi, \Sigma, \Theta, R_{a\in\Sigma}, \xi \rangle$ and $\mathfrak{M}' = \langle \Pi, \Sigma, \Theta, R_{a\in\Sigma}, \xi' \rangle$, which indicate a change of the allocation ξ to ξ'. Since such a change of allocation does not affect the current world, we have for such pairs that $\theta = \theta'$. Slightly abusing notation, we define $(\mathfrak{M}, \theta)(a \rightsquigarrow_p b)(\mathfrak{M}', \theta')$ exactly when $p \in \Pi_a$, and either $a = b$ and $\mathfrak{M} = \mathfrak{M}'$, or else $\Pi'_a = \Pi_b \setminus \{p\}$ and $\Pi'_b = \Pi_b \cup \{p\}$, and all the other sets Π_c remain the same.

The semantics over Kripke structures for DCL-PC is defined as follows; cf. [**248**].

DEFINITION 5.6. (DCL-PC SEMANTICS). Given a DCL-PC structure $\mathfrak{M} = \langle \Pi, \Sigma, \Theta, \{R_a\}_{a\in\Sigma}, \xi \rangle$, the satisfaction relation $\models^{\mathsf{K}}$ is inductively defined as follows, where θ, θ' range over valuations, $C \subseteq \Sigma$ ranges over coalitions of agents, a, b over agents in Σ, p over propositional variables, and φ, ψ range over DCL-PC-formulas:

- $\mathfrak{M}, \theta \models^{\mathsf{K}} p$ iff $\theta(p) = \mathrm{tt}$, for all propositions $p \in \Pi$;
- $\mathfrak{M}, \theta \models^{\mathsf{K}} \neg\varphi$ iff $\mathfrak{M}, \theta \not\models^{\mathsf{K}} \varphi$;
- $\mathfrak{M}, \theta \models^{\mathsf{K}} \varphi \vee \psi$ iff $\mathfrak{M}, \theta \models^{\mathsf{K}} \varphi$ or $\mathfrak{M}, \theta \models^{\mathsf{K}} \psi$;
- $\mathfrak{M}, \theta \models \Diamond_C \varphi$ iff there is a valuation θ' such that $(\theta, \theta') \in R_a$ for all agents $a \in C$, and $\mathfrak{M}, \theta' \models^{\mathsf{K}} \varphi$;
- $\mathfrak{M}, \theta \models \langle a \rightsquigarrow_p b \rangle \varphi$ iff there is a Kripke structure $\mathfrak{M}'$ for DCL-PC such that $(\mathfrak{M}, \theta)(a \rightsquigarrow_p b)(\mathfrak{M}', \theta)$ and $\mathfrak{M}', \theta \models^{\mathsf{K}} \varphi$;
- $\mathfrak{M}, \theta \models \langle \delta_1; \delta_2 \rangle \varphi$ iff $\mathfrak{M}, \theta \models \langle \delta_1 \rangle \langle \delta_2 \rangle \varphi$;
- $\mathfrak{M}, \theta \models \langle \delta_1 \cup \delta_2 \rangle \varphi$ iff $\mathfrak{M}, \theta \models \langle \delta_1 \rangle \varphi$ or $\mathfrak{M}, \theta \models \langle \delta_2 \rangle \varphi$;
- $\mathfrak{M}, \theta \models \langle \delta^* \rangle \varphi$ iff $\mathfrak{M}, \theta \models \langle \delta \rangle^n \varphi$ for some $n \geq 0$;
- $\mathfrak{M}, \theta \models \langle \psi? \rangle \varphi$ iff $\mathfrak{M}, \theta \models \psi$ and $\mathfrak{M}, \theta \models \varphi$.

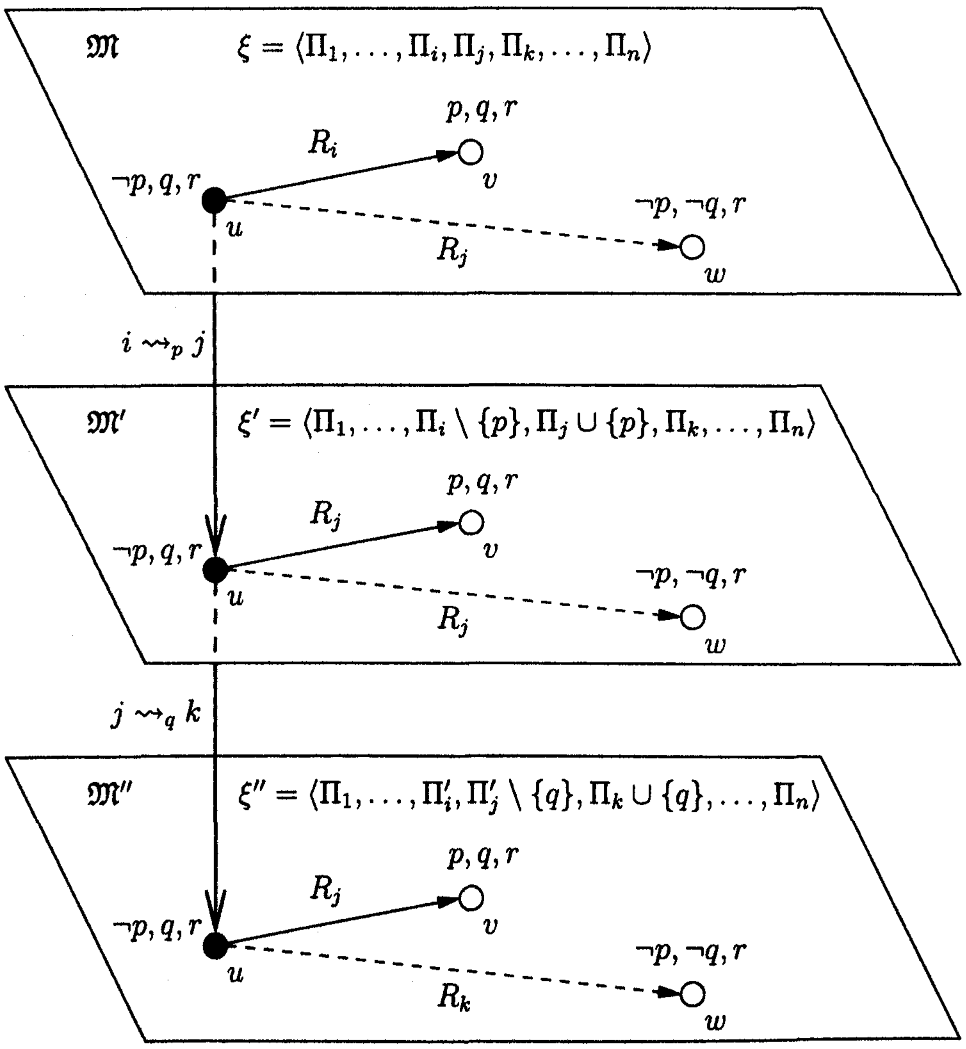
If, for some valuation θ and some Kripke structure $\mathfrak{M}$ for DCL-PC, it holds that $\mathfrak{M}, \theta \models \varphi$, then the DCL-PC-formula φ is *true* or *satisfied* at θ, and $\mathfrak{M}$ is called a *model* of φ.

Note that in fact, in the Kripke semantics, formulas are not interpreted in a structure together with a valuation only, but in the context of other structures (which are reached by the atomic program $i \rightsquigarrow_p j$). There are finitely many such Kripke structures for

DCL-PC, one for each ξ. Call this collection of models κ. In fact, this κ is the structure with respect to which formulas are interpreted. In that sense, there is only one Kripke structure for the language, it is κ. We will prove completeness with respect to this unique structure, in Section 5.2.

To better understand the Kripke-style semantics, consider the following example.

EXAMPLE 5.7. (LAYERS). This example demonstrates the Kripke semantics. Figure 5.1 depicts three Kripke structures $\mathfrak{M}$, $\mathfrak{M}'$ and $\mathfrak{M}''$, one in each of the three boxes. The structures only differ in the allocation of the propositional variables to the agents. The vertical arrows between states of different structures correspond to the vertical relation and the slightly thinner horizontal arrows between states within a structure correspond to the horizontal relations. The labelling of states and arrows is self-explanatory. In $\mathfrak{M}$, we have that at state u agent i has only control over the propositional variable



$\mathfrak{M}$
$\xi = \langle \Pi_1, \ldots, \Pi_i, \Pi_j, \Pi_k, \ldots, \Pi_n \rangle$
p, q, r
R_i
v
$\neg p, q, r$
u
$\neg p, \neg q, r$
R_j
w
$i \rightsquigarrow_p j$
$\mathfrak{M}'$
$\xi' = \langle \Pi_1, \ldots, \Pi_i \setminus \{p\}, \Pi_j \cup \{p\}, \Pi_k, \ldots, \Pi_n \rangle$
p, q, r
R_j
v
$\neg p, q, r$
u
$\neg p, \neg q, r$
R_j
w
$j \rightsquigarrow_q k$
$\mathfrak{M}''$
$\xi'' = \langle \Pi_1, \ldots, \Pi_i', \Pi_j' \setminus \{q\}, \Pi_k \cup \{q\}, \ldots, \Pi_n \rangle$
p, q, r
R_j
v
$\neg p, q, r$
u
$\neg p, \neg q, r$
R_k
w

FIGURE 5.1. Some Kripke models for DCL-PC.

p but not q and r: i can make p true and false, but i cannot change the truth value of q and r. That is, we have $p \in \Pi_i$ and $\{q, r\} \not\subseteq \Pi_i$. Using DCL-PC, we can express this with

$$\mathfrak{M}, u \models \mathit{controls}(i,p) \wedge \neg\mathit{controls}(i,q) \wedge \neg\mathit{controls}(i,r).$$

The situation for agent j at state u is different since she can control q but not p and r. That is, we have

$$\mathfrak{M}, u \models \neg\mathit{controls}(j,p) \wedge \mathit{controls}(j,q) \wedge \neg\mathit{controls}(j,r).$$

Now, using the delegation program $(i \leadsto_p j)$, agent i can pass the control over p on to agent j such that, after the execution of this program, i no longer controls p. This can be expressed as

$$\mathfrak{M}, u \models \langle i \leadsto_p j\rangle(\neg\mathit{controls}(i,p) \wedge \mathit{controls}(j,p)).$$

As Figure 5.1 illustrates, executing this program corresponds to travelling along the vertical relation to the same state u in the next structure $\mathfrak{M}'$. Notice the different allocation ξ' in $\mathfrak{M}'$ in which the variable p belongs to j but not i. Suppose that at this point agent j passes the control over q on to agent k. Again, executing $(j \leadsto_p k)$ corresponds to travelling to the next structure $\mathfrak{M}''$, where q belongs to k but no longer belongs to j. We have that

$$\mathfrak{M}', u \models \mathit{controls}(j,q) \wedge \langle j \leadsto_q k\rangle \mathit{controls}(k,q).$$

Note that for the sets Π'_i and Π'_j in the allocation ξ'' in $\mathfrak{M}''$ we have $\Pi'_i = \Pi_i \setminus \{p\}$ and $\Pi'_j = \Pi_j \cup \{p\}$ (cf. Figure 5.1). ⊣

The following lemma is easily established by induction on φ:

LEMMA 5.8. *For any fixed sets of agents Σ and propositional variables Π, the direct semantics and the Kripke semantics are equivalent, i.e., for any φ, any Kripke structure $\mathfrak{M} \in \mathcal{K}(\Sigma, \Pi)$ for DCL-PC with $\mathfrak{M} = \langle \Theta, R_{i\in\Sigma}, \xi\rangle$, and any direct structure $\mathcal{M} = \langle \Sigma, \Pi, \xi, \theta\rangle$:*

$$\mathcal{M} \models^{\mathsf{d}} \varphi \quad \textit{iff} \quad \mathfrak{M}, \theta \models^{\mathsf{K}} \varphi.$$

As usual, we define $\mathfrak{M} \models^{\mathsf{K}} \varphi$ as $\forall\theta : \mathfrak{M}, \theta \models^{\mathsf{K}} \varphi$, and $\models^{\mathsf{K}} \varphi$ as $\forall\mathfrak{M} : \mathfrak{M} \models^{\mathsf{K}} \varphi$.

5.1.4. Relationship to Strategic Logics. Although we will not use them in this chapter, we note that the basic modal operators of ATL (cf. Section 2.4.1) and Pauly's Coalition Logic [**194**] can be interpreted within the framework of DCL-PC. In fact, these operators can be defined in the fragment CL-PC [**249**] of DCL-PC, which does not allow for any dynamic operators such as $\langle i \leadsto_p j\rangle$. Notice that DCL-PC subsumes neither Coalition Logic nor ATL. We give a DCL-PC semantics to the modal operators of these logics by interpreting ability as in DCL-PC, namely, as having control over the truth values of propositional variables.

We start with formulas of Coalition Logic of the form $\langle\!\langle C\rangle\!\rangle_\alpha \varphi$, where '$\alpha$' stands for '$\alpha$-effectivity' [**194**, p.20]. In CL-PC, we can express such formulas as follows (where $\overline{C}$

denotes the complement of C):

$$\langle\!\langle C\rangle\!\rangle_\alpha \varphi \;:=\; \Diamond_C \Box_{\overline{C}} \varphi.$$

That is, $\langle\!\langle C\rangle\!\rangle_\alpha \varphi$ means that the agents in C can bring about a state at which φ is true, no matter what the agents outside C do. For more details, see [**249**]. ATL-formulas of the form $\langle\!\langle C\rangle\!\rangle\bigcirc\varphi$ can be defined in the same way:

$$\langle\!\langle C\rangle\!\rangle\bigcirc\varphi \;:=\; \Diamond_C \Box_{\overline{C}}\, \varphi.$$

Different to ATL, the logic DCL-PC does not provide explicit *temporal* operators, other than the implicit 'next'. However, the temporal operators of ATL 'always' ('$\Box$') and 'until' ('$\mathcal{U}$') can be reduced to the operator 'next-time' ('$\bigcirc$'), as follows. First, recall the following fixed-point schemes for the ATL 'always' and 'until' operators:

$$\begin{array}{lcl} \langle\!\langle C\rangle\!\rangle\Box\varphi & \leftrightarrow & \varphi \wedge \langle\!\langle C\rangle\!\rangle\bigcirc\langle\!\langle C\rangle\!\rangle\Box\varphi \\ \langle\!\langle C\rangle\!\rangle\varphi\,\mathcal{U}\,\psi & \leftrightarrow & \psi \vee (\varphi \wedge \langle\!\langle C\rangle\!\rangle\bigcirc\langle\!\langle C\rangle\!\rangle\varphi\,\mathcal{U}\,\psi) \end{array}$$

Now we can use the fact that the "horizontal" accessibility relation in Kripke models for DCL-PC is transitive to establish the equivalence:

$$\Diamond_C \Box_{\overline{C}}\, \varphi \leftrightarrow \Diamond_C \Box_{\overline{C}} (\Diamond_C \Box_{\overline{C}}\, \varphi)$$

Intuitively, what this equivalence tells us is that if a coalition in DCL-PC can achieve something in two steps, then *they can also achieve it in one step.*

With this equivalence, we can avoid repeated unfolding of the inductive schemes for the ATL operators 'always' and 'until'. That is, the remaining ATL operators can be defined in DCL-PC as follows:

$$\begin{array}{lcl} \langle\!\langle C\rangle\!\rangle\Box\varphi & := & \varphi \wedge \Diamond_C \Box_{\overline{C}}\, \varphi \\ \langle\!\langle C\rangle\!\rangle\varphi\,\mathcal{U}\,\psi & := & \psi \vee (\varphi \wedge \Diamond_C \Box_{\overline{C}}\, \psi) \end{array}$$

Giving ATL-formulas a DCL-PC semantics corresponds to viewing ability as having control over the truth values of propositional variables. In this setting, it appears that ATL reduces to a much simpler logic, as we have just demonstrated. Clearly, this does not mean that ATL can be reduced to DCL-PC as ATSs – the structures for ATL– cannot be translated into Kripke structures for DCL-PC.

5.2. A Complete Axiomatisation

A complete axiomatisation for the logic DCL-PC is given in Table 5.1. For the ease of exposition, we divide the axiomatisation into five categories as follows. While the 'Propositional Component' and the 'Rules of Inference' are straightforward, the 'Dynamic Component' is an immediate adaptation of Propositional Dynamic Logic (see [**107**]). The 'Control Axioms' are inherited from [**249**]. (The occurrence of $\ell(p)$ refers to a literal with atomic proposition p: it is either p or $\neg p$, with the obvious meaning for $\neg\ell(p)$.) Note that axiom (Allocation) specifies that every propositional variable is assigned to exactly one agent (i.e., we have *an* allocation), while in contrast, for the fixed allocation ξ that was assumed in [249], one could explicitly state that $controls(i,p)$, for every $p \in \Pi_i$.

For the 'Delegation & Control Axioms', ($\rightsquigarrow$-Permanence(atomic)) states that no program δ changes the valuation. From this, one easily extends this to arbitrary objective formulas (obtaining (Objective Permanence), see Theorem 5.9 below). The axiom (Persistence$_1$(control)) says that i's control over p is not affected when we move to another valuation, and axiom (Persistence$_2$(control)) specifies how i remains in control of p, even when a delegation program is executed: either the variable passed in that program is not p, or the delegating agent is not i. The axiom (Precondition(delegation)) expresses that agents can only give variables away that they possess, and, finally *func* says that the transition relation associated with an atomic delegation program is functional: at most one resulting world emerges.

The following theorem lists some properties of DCL-PC, where $controls(C,p)$ is as defined in equation (5.1) above.

THEOREM 5.9.

(1) *The schemes in Table 5.2 are derivable in DCL-PC.*
(2) *Moreover, from* [249] *we know that the axioms* (K_a), (T_a), (B_a) *and* (Effect$_a$) *have the coalitional counterparts* (K_C), (T_C), (B_C) *and* (Effect$_C$) *that are derivable for any coalition C.*
(3) $\vdash controls(C,p) \leftrightarrow \bigvee_{i\in C} controls(i,p)$.
(4) $\vdash controls(C,p) \rightarrow \Box_j controls(C,p)$, *i.e., the property (Persistence$_1$(control)) is also derivable when we replace agent i by an arbitrary coalition C.*

PROOF. The four properties in Theorem 5.9 can be shown as follows:

(1) This will follow after we have proven completeness: all the properties are valid. (Of course, this means that we cannot use this item in the completeness proof itself.)
(2) We refer to [249].
(3) The definition of $controls(C,p)$ is $\Diamond_C p \wedge \Diamond_C \neg p$. Let $C = \{a_1, a_2, \dots, a_C\}$. By axiom (Control$_a$), we have $\bigvee_{i\in C} controls(i,p) \rightarrow \bigvee_{i\in C}(\Diamond_i p \wedge \Diamond_i \neg p)$. By the contrapositive of axiom (T$_a$), we have $\varphi \rightarrow \Diamond_i \varphi$. We can apply this repeatedly for all agents in C, giving $\varphi \rightarrow \Diamond_{a_1}\Diamond_{a_2}\cdots\Diamond_{a_C}\varphi$. This is, according to (Comp$_\cup$), the same as $\varphi \rightarrow \Diamond_C \varphi$. (Note that we have now proven the contrapositive of (T$_C$).) This gives us $\bigvee_{i\in C} controls(i,p) \rightarrow \bigvee_{i\in C}(\Diamond_C\Diamond_i p \wedge \Diamond_C\Diamond_i \neg p)$. Using (Comp$_\cup$) again, we see that the consequent of this implication is equivalent to $\Diamond_C p \wedge \Diamond_C \neg p$. For the other direction, we first show (At-most(control)) of Table 5.2. From $\ell(p)$ we get, using axiom (T$_a$) and contraposition, $\Diamond_i \ell(p)$. Assuming moreover $\Diamond_i \neg\ell(p)$, with axiom (Control$_a$), gives $controls(i,p)$. From axiom (Allocation) we then obtain $\neg controls(j,p)$, for any agent $j \neq i$. Using

<u>Propositional Component</u>	
(TAUT)	Propositional tautologies
<u>Dynamic Component</u>	
(K_δ)	$[\delta](\varphi \to \psi) \to ([\delta]\varphi \to [\delta]\psi)$
(Union_δ)	$[\delta \cup \delta']\varphi \leftrightarrow ([\delta]\varphi \wedge [\delta']\varphi)$
(Comp_δ)	$[\delta; \delta']\varphi \leftrightarrow [\delta][\delta']\varphi$
(Test_δ)	$[\varphi?]\psi \leftrightarrow (\varphi \to \psi)$
(Mix_δ)	$(\varphi \wedge [\delta][\delta^*]\varphi) \leftrightarrow [\delta^*]\varphi$
(Ind_δ)	$(\varphi \wedge [\delta^*](\varphi \to [\delta]\varphi)) \to [\delta^*]\varphi$
<u>Control Axioms</u>	
(K_a)	$\Box_a(\varphi \to \psi) \to (\Box_a\varphi \to \Box_a\psi)$
(T_a)	$\Box_a\varphi \to \varphi$
(B_a)	$\varphi \to \Box_a\Diamond_a\varphi$
(Empty)	$\Box_\emptyset\varphi \leftrightarrow \varphi$
(Control_a)	$\mathit{controls}(a,p) \leftrightarrow (\Diamond_a p \wedge \Diamond_a\neg p)$
(Allocation)	$\bigwedge_{p\in\Pi}(\mathit{controls}(a_1,p) \,\nabla \cdots \nabla\, \mathit{controls}(a_n,p))$ where $\Sigma = \{a_1, \ldots, a_n\}$
(Effect_a)	$(\psi \wedge \ell(p) \wedge \mathit{controls}(a,p)) \to \Diamond_a(\psi \wedge \neg\ell(p))$ where $p \notin \Pi_\psi$, and ψ is objective
($\text{Comp}_\cup$)	$\Box_{C_1}\Box_{C_2}\varphi \leftrightarrow \Box_{C_1\cup C_2}\varphi$
<u>Delegation & Control Axioms</u>	
($\rightsquigarrow$-Permanence(atomic))	$\langle a \rightsquigarrow_p b\rangle\top \to ([a \rightsquigarrow_p b]q \leftrightarrow q)$
(Persistence_1(control))	$\mathit{controls}(a,p) \to \Box_b \mathit{controls}(a,p)$
(Persistence_2(control))	$\mathit{controls}(a,p) \to [b \rightsquigarrow_q c]\mathit{controls}(a,p)$ where $a \neq b$ or $p \neq q$
(Precondition(delegation))	$\langle a \rightsquigarrow_p b\rangle\top \to \mathit{controls}(a,p)$
(Delegation)	$\mathit{controls}(a,p) \to \langle a \rightsquigarrow_p b\rangle \mathit{controls}(b,p)$
(Func)	$\mathit{controls}(a,p) \to (\langle a \rightsquigarrow_p b\rangle\varphi \leftrightarrow [a \rightsquigarrow_p b]\varphi)$
<u>Rules of Inference</u>	
(Modus Ponens)	$\dfrac{\varphi, \varphi \to \psi}{\psi}$
(Necessitation)	$\dfrac{\varphi}{\Box\varphi}$ where $\Box = [\delta], [a \rightsquigarrow_p b], \Box_a$

TABLE 5.1. An axiom system for DCL-PC.

(Control_j), we get $\neg\Diamond_j p \vee \neg\Diamond_j\neg p$, i.e., $\neg\Diamond_j\ell(p) \vee \neg\Diamond_j\neg\ell(p)$. Since ($T_j$) gives us $\Diamond_j\ell(p)$, we obtain $\neg\Diamond_j\neg\ell(p)$, i.e., $\Box_j\ell(p)$.

Now suppose $\neg\bigvee_{i\in C}\mathit{controls}(i,p)$. By axiom (Allocation), we have that $\bigvee_{x\in\Sigma\setminus C}\mathit{controls}(x,p)$. This means that for one such x, we have $\Diamond_x p \wedge \Diamond_x\neg p$.

Now we do a case distinction based on $p \vee \neg p$. In the first case, we assume p, and derive, for all $i \in C$, that $\Box_i p$, and thus $\Box_C p$. Hence $\neg\Diamond_C \neg p$, from which we get $\neg controls(C,p)$. In the case of $\neg p$, we similarly have, for all $i \in C$, that $\Box_i \neg p$, which gives $\neg\Diamond_C \neg\neg p$, and again $\neg controls(C,p)$. All in all, no matter whether p or $\neg p$, we get $\neg controls(C,p)$.

(4) This is easy: in the previous item, we showed that $controls(C,p)$ means that $\bigvee_{i \in C} controls(i,p)$. Applying the axiom (Persistence$_1$(control)), we obtain $\bigvee_{i \in C} \Box_j controls(i,p)$. But since $controls(i,p) \rightarrow controls(C,p)$ if $i \in C$, we also have, for any $i \in C$, that $\Box_j controls(i,p) \rightarrow \Box_j controls(C,p)$ (use the rule (Necessitation) and (K$_j$)). This proves $\Box_j controls(C,p)$.

□

(At-least(control))	$(\ell(p) \wedge controls(a,p)) \rightarrow \Diamond_a \neg\ell(p)$
(At-most(control))	$\ell(p) \rightarrow (\Diamond_a \neg\ell(p) \rightarrow \Box_b \ell(p))$ where $(a \neq b)$
(Non-Effect$_a$)	$(\Diamond_a \ell(p) \wedge \neg controls(a,p)) \rightarrow \Box_a \ell(p)$
(Inverse)	$controls(a,p) \rightarrow (\varphi \leftrightarrow [a \rightsquigarrow_p b; b \rightsquigarrow_p a]\varphi)$
(Reverse)	$[a \rightsquigarrow_p b][c \rightsquigarrow_q d]\varphi \leftrightarrow [c \rightsquigarrow_q d][a \rightsquigarrow_p b]\varphi$ where $(b \neq c$ and $d \neq a)$ or $p \neq q$
(Persistence(non-control))	$\neg controls(a,p) \leftrightarrow \Box_b \neg controls(a,p)$
($\rightsquigarrow$-Objective Permanence)	$\langle a \rightsquigarrow_p b\rangle\top \rightarrow (\varphi \leftrightarrow [a \rightsquigarrow_p b]\varphi)$ where φ is objective
(Objective Permanence)	$\langle\delta\rangle\top \rightarrow (\varphi \leftrightarrow [\delta]\varphi)$ where φ is objective

TABLE 5.2. Some Theorems of DCL-PC.

Consider the language without dynamic delegation operators, in which we only have propositional logic with cooperation modalities $\Diamond_C$. Models for these are $\mathfrak{M}$, $\mathfrak{M}' \in \mathcal{K}(\Sigma, \Pi)$. In [249] it was shown that in that program-free language, every formula φ is equivalent to one without any occurrences of coalition operators. For instance, suppose that $\Pi_i = \{p, q\}$. Then a formula $\Diamond_i(\neg p \wedge r)$ is equivalent to $(p \wedge r) \vee (\neg p \wedge r)$ (we "read off" the current value of variable r outside i's control).

We now establish a similar result for the language including programs. Any world $(\mathfrak{M}, \theta)$ is completely characterised when we know which variables are true in it, and what the allocation of variables to agents is. In such a case, the truth of all objective formulas, formulas involving abilities and delegation programs is completely determined.

LEMMA 5.10. *Let φ be an arbitrary DCL-PC-formula and ζ be a conjunction of assertions of the form $controls(j,p)$ or $\neg controls(j,p)$. Then, in DCL-PC, we can derive*

$$\vdash \Diamond_C(\varphi \wedge \zeta) \leftrightarrow (\zeta \wedge \Diamond_C \varphi).$$

PROOF. Since by (Comp$_\cup$), for $C = \{a_1, a_2, \ldots, a_n\}$, we have $\Diamond_C\psi \leftrightarrow \Diamond_{a_1} \cdots \Diamond_{a_n}\psi$, it is sufficient to prove the claim for an individual agent i. Moreover, we can move all conjuncts of ζ out one by one, once we know that

$$\vdash \Diamond_i(\varphi \wedge \pm \mathit{controls}(j,p)) \leftrightarrow (\pm \mathit{controls}(j,p) \wedge \Diamond_i\varphi),$$

where $\pm\mathit{controls}(j,p)$ is either $\mathit{controls}(j,p)$ or $\neg\mathit{controls}(j,p)$. We do the reasoning for the non-negated case (the other one is similar): $\Diamond_i(\varphi \wedge \mathit{controls}(j,p))$ is equivalent to

$$\begin{array}{l}\big(\mathit{controls}(j,p) \wedge \Diamond_i(\varphi \wedge \mathit{controls}(j,p))\big)\\ \vee\big(\neg\mathit{controls}(j,p) \wedge \Diamond_i(\varphi \wedge \mathit{controls}(j,p))\big).\end{array}$$

However, by using the theorem (Persistence(non-control)) from Table 5.2 (which we derive below), we have for the second disjunct that $\big(\neg\mathit{controls}(j,p) \wedge \Diamond_i(\varphi \wedge \mathit{controls}(j,p))\big) \leftrightarrow \bot$. That concludes the proof.

For (Persistence(non-control)), the right-to-left direction follows immediately from (T$_j$). For the other direction, assume that $\neg\mathit{controls}(i,p)$. From (Allocation) we derive that

$$\mathit{controls}(1,p) \bigtriangledown \cdots \bigtriangledown \mathit{controls}(i-1,p) \bigtriangledown \mathit{controls}(i+1,p) \bigtriangledown \cdots \bigtriangledown \mathit{controls}(n,p),$$

and from this, by (Persistence$_1$(control)), we get $\bigvee_{k \neq i} \Box_j \mathit{controls}(k,p)$. For every $k \neq i$, we have $\mathit{controls}(k,p) \rightarrow \neg\mathit{controls}(i,p)$, which follows from (Allocation). Hence, using (Necessitation), we have $\Box_j(\mathit{controls}(k,p) \rightarrow \neg\mathit{controls}(i,p))$. Axiom (K$_j$) now yields $\Box_j\mathit{controls}(k,p) \rightarrow \Box_j\neg\mathit{controls}(i,p)$. Combining this with $\bigvee_{k \neq i} \Box_j \mathit{controls}(k,p)$, we obtain the desired conclusion $\Box_j\neg\mathit{controls}(i,p)$. $\Box$

Soundness of the axiom schemes in Figure 5.1 is readily checked. We now proceed to prove that the axiomatic system for DCL-PC in Figure 5.1 is complete. First, we introduce some notation.

DEFINITION 5.11. (VALUATION DESCRIPTION). Given the set of propositional variables Π, a *valuation description* π is a conjunction of literals (p or $\neg p$) over them such that every propositional variable in Π occurs in one literal. That is, for each propositional variable $p \in \Pi$, it holds that either $\pi \rightarrow p$, or $\pi \rightarrow \neg p$.

We denote the set of all valuation descriptions over Π with $\mathbb{P}$. Notice that, for each valuation θ, there is a $\pi_\theta \in \mathbb{P}$ such that

$$\begin{array}{rl}\pi_\theta = & \bigwedge\{p \mid p \in \Pi \text{ and } \theta(p) = \mathrm{tt}\}\\ & \wedge \bigwedge\{\neg p \mid p \in \Pi \text{ and } \theta(p) = \mathrm{ff}\}.\end{array}$$

DEFINITION 5.12. (ALLOCATION DESCRIPTION). Given the set of propositional variables Π and the set of agents Σ, an *allocation description* v is a conjunction of formulas of the form $\mathit{controls}(i,p)$ such that, for each propositional variable $p \in \Pi$, there is exactly one agent $i \in \Sigma$ such that $v \rightarrow \mathit{controls}(i,p)$.

We denote the set of all allocation descriptions with $\mathbb{A}$. Notice that allocations ξ and conjunction v correspond to each other: For each allocation $\xi = \langle \Pi_1, \dots, \Pi_n \rangle$ of the variables in Π over the agents in Σ, there is a $v_\xi \in \mathbb{A}$ such that

$$v_\xi = \bigwedge_{i \in \Sigma, p \in \mathbb{P}_i} controls(i, p).$$

Therefore, we refer to formulas v as allocation descriptions.

Given two allocation descriptions $v, v' \in \mathbb{A}$, we say $v(i \leadsto_p j)v'$ if the following three conditions are satisfied: $v \rightarrow controls(i,p)$, $v' \rightarrow controls(j,p)$, and v and v' agree on all other *control* expressions.

DEFINITION 5.13. (PROPOSITION DESCRIPTION). Let, for any allocation description v, $\mathbb{P}_v \subseteq \mathbb{P}$ be a set of valuation descriptions. Then, a formula of the form

$$\bigvee_{v \in \mathbb{A}} \left(\bigvee \mathbb{P}_v \wedge v \right) \tag{5.3}$$

will be called a *proposition description.*

Let us, for two valuation descriptions π and π', a coalition C, and an allocation description v, write $\pi \equiv \pi'$ (mod C, v) if the two conjunctions of literals π and π' only differ in the variables under control of C, which is determined by v. For instance, when $C = \{1, 2\}$ and $v = controls(1, p_1) \wedge controls(2, p_2) \wedge controls(3, p_3)$, then $(\neg p_1 \wedge p_2 \wedge p_3) \equiv (p_1 \wedge \neg p_2 \wedge p_3)$ (mod C, v).

We now first collect some facts about *valuation descriptions*, *allocation descriptions*, and *proposition descriptions*. Recall that, for any set Φ of DCL-PC-formulas, $\nabla_{\varphi \in \Phi}$ is used as shorthand for $\bigvee_{\varphi \in \Phi} \varphi \wedge \bigwedge_{\varphi_1 \neq \varphi_2 \in \Phi} \neg(\varphi_1 \wedge \varphi_2)$.

LEMMA 5.14. *Given the set $\mathbb{P}$ of valuation descriptions π, and the set $\mathbb{A}$ of allocation descriptions v, the following six items are satisfied:*

(1) $\vdash \nabla_{\pi \in \mathbb{P}} \; \pi$
(2) $\vdash \nabla_{v \in \mathbb{A}} \; v$
(3) $\vdash \varphi \rightarrow \bigvee_{v \in \mathbb{A}} (\varphi \wedge v)$
(4) *For all $v \in \mathbb{A}$ and all $\pi \in \mathbb{P}$: $\vdash v \rightarrow (\Diamond_C \pi \leftrightarrow \bigvee_{\pi' \equiv \pi \text{ (mod } C, v)} \pi')$.*
(5) $\vdash v(i \leadsto_p j)v'$ *implies* $\vdash (v \wedge \pi) \leftrightarrow \langle i \leadsto_p j \rangle (v' \wedge \pi)$.
(6) *Let n be the number of agents, and k the number of propositional variables. Then there are not more than $N(n,k) = 2^{2^{nk}}$ provably non-equivalent proposition descriptions.*

PROOF. (1) This follows from $(TAUT)$ and the definition of $\mathbb{P}$: the π's are mutually exclusive and cannot all be false.

(2) Item (2) is easily seen to be equivalent to the axiom (Allocation): (Allocation) implies $\bigvee_{v \in \mathbb{A}} v$, and, for every allocation description $v \in \mathbb{A}$, we have that v implies (Allocation).

(3) Item (3) is immediate from item (2) and axiom $(TAUT)$. In particular, using $(TAUT)$ we derive from $\vdash A \vee B$ that $C \rightarrow (C \wedge A) \vee (C \wedge B)$.

(4) Assume $\vdash \upsilon$. For the right-to-left direction, also assume that $\vdash \pi'$, for some valuation description π' with $\pi' \equiv \pi \pmod{C, \upsilon}$. This means that π' and π only differ in some variables $p_1, \ldots, p_m$ for which $controls(C, p_j)$ is implied by υ (for $j = 1...m$). Note that π' is an objective formula. We can write π' as $\psi_i \wedge \ell(p_i)$ (for $i = 1...m$), where ψ_i is as π' but with the literal $\ell(p_i)$ left out. Apparently, we have $\psi_1 \wedge \ell(p_1) \wedge controls(C, p_1)$. Since ψ_1 is objective, we can apply (Effect$_C$) to conclude $\Diamond_C(\psi_1 \wedge \neg\ell(p_1))$. Using Lemma 5.10, we derive $\Diamond_C(\upsilon \wedge \psi_1 \wedge \neg\ell(p_1))$. We can now rewrite ψ_1 to $\psi_2 \wedge \ell(p_2)$, and obtain $\Diamond_C(\psi_2 \wedge \ell(p_2) \wedge \neg\ell(p_1) \wedge \upsilon)$. We use (Effect$_C$) and Lemma 5.10 again and get $\Diamond_C\Diamond_C(\psi_2 \wedge \neg\ell(p_2) \wedge \neg\ell(p_1) \wedge \upsilon)$. By (Comp$_\cup$), this is the same as $\Diamond_C(\psi_2 \wedge \neg\ell(p_2) \wedge \neg\ell(p_1) \wedge \upsilon)$. We can repeat this process until we get $\Diamond_C(\psi_j \wedge \neg\ell(p_j) \wedge \cdots \wedge \neg\ell(p_2) \wedge \neg\ell(p_1) \wedge \upsilon)$. But, by definition of π', this implies $\Diamond_C\pi$.

We show the other direction from left to right by contrapositive: fix π and υ and assume $\vdash \neg \bigvee_{\pi' \equiv \pi \pmod{C,\upsilon}} \pi'$. We have to show that $\neg\Diamond_C\pi$. Let $Q(\pi, \upsilon) = \{\ell(p) \mid \vdash \pi \rightarrow \ell(p) \text{ and } \nvdash \upsilon \rightarrow controls(C, p)\}$ be the set of the literals over variables that are not under control of the agents in C at the allocation υ. Notice that all valuations $\pi' \equiv \pi \pmod{C, \upsilon}$ agree with π on the literals in $Q(\pi, \upsilon)$. We can use propositional reasoning to derive from $\neg \bigvee_{\pi' \equiv \pi \pmod{C,\upsilon}} \pi'$ that $\vdash \neg \bigwedge_{\ell(p) \in Q(\pi,\upsilon)} \ell(p)$. Using (T$_C$) and (K$_C$), we get $\vdash \bigvee_{\ell(p) \in Q(\pi,\upsilon)} \Diamond_C \neg\ell(p)$. From $\vdash \upsilon$, it follows, for each literal $\ell(p) \in Q(\pi, \upsilon)$, that $\vdash \neg controls(C, \ell(p))$, which, by equation (5.1), equals $\Diamond_C\neg\ell(p) \rightarrow \neg\Diamond_C\ell(p)$. But then, we can derive $\bigvee_{\ell(p) \in Q(\pi,\upsilon)} \neg\Diamond_C\ell(p)$. Using (K$_C$), we obtain $\vdash \neg\Diamond_C \bigwedge_{\ell(p) \in Q(\pi,\upsilon)} \ell(p)$. Hence, $\vdash \neg\Diamond_C\pi$.

(5) First of all, since $\upsilon(i \leadsto_p j)\upsilon'$, we know that $\upsilon \rightarrow controls(i, p)$. Hence, given υ, we have that all formulas $\langle i \leadsto_p j\rangle\psi$ and $[i \leadsto_p j]\psi$ are equivalent by axiom (Func). In particular, we have $\langle i \leadsto_p j\rangle\top$. Let us first show for literals $\ell(q)$ that $\ell(q) \leftrightarrow \langle i \leadsto_p j\rangle\ell(q)$. For negative literals $\neg q$, $\neg q \leftrightarrow \langle i \leadsto_p j\rangle\neg q$ equals $q \leftrightarrow [i \leadsto_p j]q$, which follows from axiom ($\leadsto$-Permanence(atomic)). For positive literals q, we use (Func) to obtain $q \leftrightarrow [i \leadsto_p j]q$, which again holds by ($\leadsto$-Permanence(atomic)). Now, given υ, we have $q \leftrightarrow \langle i \leadsto_p j\rangle q$ and $\neg q \leftrightarrow \langle i \leadsto_p j\rangle\neg q$. Then, for any valuation description π, we also have $\pi \leftrightarrow \langle i \leadsto_p j\rangle\pi$. It remains to show that $\upsilon \leftrightarrow \langle i \leadsto_p j\rangle\upsilon'$. From left to right, note that we have $controls(i, p) \rightarrow \langle i \leadsto_p j\rangle controls(j, p)$ by axiom (Delegation). For any other $controls$ expression implied by υ, this follows from (Persistence$_2$(control)) and (Func). Finally, consider the direction from right to left: $\langle i \leadsto_p j\rangle\upsilon' \rightarrow \upsilon$. We have to show that, first, $\langle i \leadsto_p j\rangle controls(j, p) \rightarrow controls(i, p)$, and, second, that $\langle i \leadsto_p j\rangle controls(h, q) \rightarrow controls(h, q)$, for each $controls(h, q)$ (with $q \neq p$) implied by υ'. The first part follows immediately from (Precondition(delegation)). For the second part, let h be an agent and $q \neq p$ a variable such that υ' implies $controls(h, q)$. Suppose $\langle i \leadsto_p j\rangle controls(h, q)$. By (Allocation), we have that $\langle i \leadsto_p j\rangle \bigwedge_{k \neq h} \neg controls(k, q)$. But then the contrapositive

of (Persistence$_2$(control)) yields $\bigwedge_{k \neq h} \neg controls(k,q)$. It follows by (Allocation) that $controls(h,q)$.

(6) Item 6 follows from a straightforward counting argument of proposition description formulas. The number of proposition descriptions depends on the cardinalities of the sets $\mathbb{P}$ and $\mathbb{A}$. Given the number $n = |\Sigma|$ of agents, and the number $k = |\Pi|$ of propositional variables, it is easy to see that there are 2^k valuation descriptions in $\mathbb{P}$, and n^k allocation descriptions in $\mathbb{A}$ (i.e., the number of ways we can distribute k variables over n agents). Observe that any proposition description formula is obtained by assigning a set of valuation descriptions π to each allocation description v. Hence, there are $2^{2^k \times n^k}$ proposition descriptions. Since $2^{2^k \times n^k} \leq 2^{2^{nk}}$, we obtain with $N(n,k) = 2^{2^{nk}}$ an upper bound for the number of different proposition description formulas.

□

We now present the main result of this section. We first formulate it, reflect briefly on it, and then give its proof.

THEOREM 5.15. *For every DCL-PC-formula φ, there are sets $\mathbb{P}_v(\varphi) \subseteq \mathbb{P}$ of valuation descriptions, one for each $v \in \mathbb{A}$, such that*

$$\vdash \varphi \leftrightarrow \bigvee_{v \in \mathbb{A}} \big(\bigvee \mathbb{P}_v(\varphi) \wedge v\big).$$

Thus, according to Theorem 5.15, we can find for each DCL-PC-formula an equivalent formula without dynamic operators such as $\langle i \leadsto_p j \rangle$. To show that our derivation system is good enough to establish that, is the main task of its proof. But let us first convince ourselves semantically that such a normal form makes sense. Remember that every Kripke structure $\mathfrak{M}$ for DCL-PC comes with its own allocation ξ. A formula $\langle i \leadsto_p j \rangle \psi$ is true at $\mathfrak{M}, \theta$, if ψ is true in a structure that is the same as $\mathfrak{M}$, but in which control over p is transferred from i to j. But this means that some formula ψ' must already be true at $\mathfrak{M}, \theta$, where i takes the role of j for as far as p is concerned. For instance, $\langle i \leadsto_p j \rangle (q \wedge \Diamond_j (p \wedge r))$ is true at $\mathfrak{M}, \theta$, if $q \wedge \Diamond_i (p \wedge r) \wedge controls(i,p)$ is true under the current allocation ξ. This formula has no reference to other "layers" anymore. Moreover, the resulting formula $q \wedge \Diamond_i (p \wedge r) \wedge controls(i,p)$ is equivalent to

$$\begin{gathered}(q \wedge ((p \wedge r) \vee (\neg p \wedge r)) \wedge controls(i,p) \wedge \neg controls(i,r)) \\ \vee \\ (q \wedge ((p \wedge r) \vee (\neg p \wedge r) \vee (p \wedge \neg r) \vee (\neg p \wedge \neg r)) \wedge controls(i,p) \wedge controls(i,r))\end{gathered}$$

PROOF. The proof is by induction on the norm $||\cdot||$ which counts the number of operators in an unfolded form of a formula. Formally, the norm $||\cdot||$ is defined on DCL-PC-formulas as follows:

$$\begin{array}{rcl} ||\top|| = ||p|| & := & 0, \text{ for any } p \in \Pi; \\ ||\neg\psi|| & := & 1 + ||\psi||; \\ ||\psi_1 \vee \psi_2|| & := & 1 + ||\psi_1|| + ||\psi_2||; \end{array}$$

$$\begin{array}{rcl}
\|\Diamond_C\psi\| & := & 1+\|\psi\|, \text{ for any } C\subseteq\Sigma;\\
\|[i\rightsquigarrow_p j]\psi\| & := & 1+\|\psi\|;\\
\|[\varphi?]\psi\| & := & 1+\|\neg\varphi\vee\psi\|;\\
\|[\delta_1\cup\delta_2]\psi\| & := & 1+\|[\delta_1]\psi\vee[\delta_2]\psi\|;\\
\|[\delta_1;\delta_2]\psi\| & := & 1+\|[\delta_1][\delta_2]\psi\|;\\
\|[\delta^*]\psi\| & := & 1+\|\bigwedge_{i=0..N}[\delta]^i\psi\|;
\end{array}$$

where $N=N(n,k)$ is the number defined in Lemma 5.14, item (6).

Let φ be a DCL-PC-formula. The induction base for the proof of Theorem 5.15 has two cases:

- $\varphi=\top$. We take $\mathbb{P}_v(\top)=\mathbb{P}$, for every $v\in\mathbb{A}$. By item (1) of Lemma 5.14, we have that $\bigvee\mathbb{P}$ is an objective tautology. Hence, $\bigvee_{v\in\mathbb{A}}\big(\bigvee\mathbb{P}_v(\top)\wedge v\big)$ is equivalent to $\bigvee_{v\in\mathbb{A}}v$, which in turn is equivalent to $\top$ by (2) of Lemma 5.14.
- $\varphi=p$, for $p\in\Pi$. We take $\mathbb{P}_v(p)=\{\pi\in\mathbb{P}\mid\vdash\pi\rightarrow p\}$, for every $v\in\mathbb{A}$. Clearly, p is equivalent to $\bigvee\mathbb{P}_v(p)$, and, using (2) of Lemma 5.14, to $\bigvee_{v\in\mathbb{A}}\big(\bigvee\mathbb{P}_v(p)\wedge v\big)$.

Consider the induction step.

- $\varphi=\neg\psi$. We set $\mathbb{P}_v(\neg\psi)=\mathbb{P}\setminus\mathbb{P}_v(\psi)$, for every $v\in\mathbb{A}$. This works, because of the following:

$$\neg\psi \leftrightarrow \neg\bigvee_{v\in\mathbb{A}}\big(\bigvee\mathbb{P}_v(\psi)\wedge v\big) \tag{5.4}$$

$$\leftrightarrow \bigwedge_{v\in\mathbb{A}}\big((\neg\bigvee\mathbb{P}_v(\psi))\vee\neg v\big) \tag{5.5}$$

$$\leftrightarrow \bigvee_{v\in\mathbb{A}}\big((\neg\bigvee\mathbb{P}_v(\psi))\wedge v\big) \tag{5.6}$$

$$\leftrightarrow \bigvee_{v\in\mathbb{A}}\big(\bigvee\mathbb{P}_v(\neg\psi)\wedge v\big) \tag{5.7}$$

All steps are purely propositional, except for the equivalence between (5.5) and (5.6) which we explain now. Let us abbreviate (5.5) to $\bigvee_{i\le k}(A_i\vee\neg B_i)$, then (5.6) is $\bigvee_{i\le k}(A_i\wedge B_i)$. Note that by Lemma 5.14, item (2), we derive $\bigvee_{i\le k}B_i$. Call the latter X. By propositional reasoning again, we have that $(5.5)\wedge X\rightarrow$ (5.6). For the other direction, note that Lemma 5.14, item (2) even guarantees $\Delta_{i\le k}B_i$. Call this Y. Now again we argue propositionally: by Y, at most one B_i can be true. Say it is B_j: all the others are false. Then, by (5.6), the only conjunction that is true is $(A_j\wedge B_j)$. But then, the disjunction $\bigvee_{i\le k}(A_i\vee\neg B_i)$ must hold as well: for $i=j$ this is because of A_j, for all the other indices it is because of B_j. Notice that, strictly speaking, we should not talk about true and false formulas here, only about derivable equivalences, but we know that for the propositional reasoning part, we may freely vary between derivability and validity.

- $\varphi = \psi_1 \vee \psi_2$. We set $\mathbb{P}_v(\psi_1 \vee \psi_2) = \mathbb{P}_v(\psi_1) \cup \mathbb{P}_v(\psi_2)$, for every $v \in \mathbb{A}$. For the following equivalences, we only need propositional reasoning:

$$\psi_1 \vee \psi_2 \leftrightarrow \bigvee_{v\in\mathbb{A}} (\bigvee \mathbb{P}_v(\psi_1) \wedge v) \vee \bigvee_{v\in\mathbb{A}} (\bigvee \mathbb{P}_v(\psi_2) \wedge v) \tag{5.8}$$

$$\bigvee_{v\in\mathbb{A}} ((\bigvee \mathbb{P}_v(\psi_1) \vee \bigvee \mathbb{P}_v(\psi_2)) \wedge v) \tag{5.9}$$

$$\bigvee_{v\in\mathbb{A}} (\bigvee \mathbb{P}_v(\psi_1 \vee \psi_2) \wedge v) \tag{5.10}$$

- $\varphi = \Diamond_C\psi$. For every $v \in \mathbb{A}$, we set $\mathbb{P}_v(\Diamond_C\psi) =$

$$\{\pi \in \mathbb{P} \mid \pi \equiv \pi' \pmod{C, v} \text{ for some } \pi' \in \mathbb{P}_v(\psi)\}$$

We can derive the following equivalences:

$$\Diamond_C\psi \leftrightarrow \Diamond_C \bigvee_{v\in\mathbb{A}} (\bigvee \mathbb{P}_v(\psi) \wedge v) \tag{5.11}$$

$$\leftrightarrow \bigvee_{v\in\mathbb{A}} \Diamond_C(\bigvee \mathbb{P}_v(\psi) \wedge v) \tag{5.12}$$

$$\leftrightarrow \bigvee_{v\in\mathbb{A}} [\Diamond_C(\bigvee \mathbb{P}_v(\psi)) \wedge v] \tag{5.13}$$

$$\leftrightarrow \bigvee_{v\in\mathbb{A}} (\bigvee \mathbb{P}_v(\Diamond_C\psi) \wedge v) \tag{5.14}$$

The equivalence in (5.11) holds by the induction hypothesis. Using (K_a), this is equivalent to (5.12) (for any diamond we have $\Diamond_C(\varphi \vee \psi) \leftrightarrow (\Diamond_C\varphi \vee \Diamond_C\psi)$). The equivalence of the latter and (5.13) is by Lemma 5.10.

It remains to show the equivalence of (5.13) and (5.14). We have that $\Diamond_C \bigvee \mathbb{P}_v(\psi) = \Diamond_C \bigvee_{\pi\in\mathbb{P}_v} \pi$. By ($K_C$) and ($\text{Comp}_\cup$), this formula is equivalent to $\bigvee_{\pi\in\mathbb{P}_v} \Diamond_C\pi$. Using item (4) in Lemma 5.14, we see that this is equivalent to $\bigvee_{\pi\in\mathbb{P}_v} \bigvee_{\pi' \mid \pi' \equiv \pi \pmod{C,v}} \pi'$. But this equals $\bigvee \mathbb{P}_v(\Diamond_C\psi)$ by definition of $\mathbb{P}_v(\Diamond_C\psi)$.

- $\varphi = [i \rightsquigarrow_p j]\psi$. We define $\mathbb{P}_v([i \rightsquigarrow_p j]\psi)$ as follows: for every $v \in \mathbb{A}$,

$$\mathbb{P}_v([i \rightsquigarrow_p j]\psi) = \begin{cases} \mathbb{P}_{v'}(\psi) & \text{if } v \rightarrow \mathit{controls}(i,p) \\ & \text{where } v(i \rightsquigarrow_p j)v' \\ \mathbb{P} & \text{otherwise} \end{cases}$$

To see that this yields a formula of the right form that is equivalent to $[i \rightsquigarrow_p j]\psi$, let us first partition $\mathbb{A}$ in $\mathbb{A}^+(i,p) = \{v \in \mathbb{A} \mid \vdash v \rightarrow \mathit{controls}(i,p)\}$ and $\mathbb{A}^-(i,p) = \{v \in \mathbb{A} \mid \vdash v \rightarrow \neg\mathit{controls}(i,p)\}$. Now consider the following derivable equivalences:

$$[i \rightsquigarrow_p j]\psi \quad \leftrightarrow \quad [i \rightsquigarrow_p j] \bigvee_{v' \in \mathbb{A}} (\bigvee \mathbb{P}_{v'}(\psi) \wedge v') \tag{5.15}$$

$$\leftrightarrow \quad \neg controls(i,p) \vee (controls(i,p) \wedge \langle i \rightsquigarrow_p j\rangle \bigvee_{v_0 \in \mathbb{A}} (\bigvee \mathbb{P}_{v_0}(\psi) \wedge v_0)) \tag{5.16}$$

$$\leftrightarrow \quad \bigvee_{v \in \mathbb{A}^-(i,p)} (\bigvee \mathbb{P} \wedge v) \vee \bigvee_{v \in \mathbb{A}^+(i,p)} (v \wedge \langle i \rightsquigarrow_p j\rangle \bigvee_{v_0 \in \mathbb{A}} (\bigvee \mathbb{P}_{v_0}(\psi) \wedge v_0)) \tag{5.17}$$

$$\leftrightarrow \quad \bigvee_{v \in \mathbb{A}} (\bigvee \mathbb{P}_v([i \rightsquigarrow_p j]\psi) \wedge v) \tag{5.18}$$

The equation in (5.15) holds by the induction hypothesis. It is equivalent to (5.16) by propositional reasoning, and changing from $[i \rightsquigarrow_p j]$ to $\langle i \rightsquigarrow_p j\rangle$ is allowed by (Func). The equivalence of (5.16) and (5.17) follows from the definition of $\mathbb{A}^+(i,p)$ and $\mathbb{A}^-(i,p)$ and the fact that $\bigvee \mathbb{P} \leftrightarrow \top$. In order to prove the equivalence between (5.17) and (5.18), it is sufficient to show that, for any fixed $v \in \mathbb{A}^+(i,p)$, the formula $v \wedge \langle i \rightsquigarrow_p j\rangle \bigvee_{v_0 \in \mathbb{A}}(\bigvee \mathbb{P}_{v_0}(\psi) \wedge v_0)$ is equivalent to $v \wedge \bigvee \mathbb{P}_{v'}(\psi)$, where $v(i \rightsquigarrow_p j)v'$. But this follows from Lemma 5.14, item (5), as follows. First of all, we can write $v \wedge \langle i \rightsquigarrow_p j\rangle \bigvee_{v_0 \in \mathbb{A}}(\bigvee \mathbb{P}_{v_0}(\psi) \wedge v_0)$ as $v \wedge \bigvee_{v_0 \in \mathbb{A}} \langle i \rightsquigarrow_p j\rangle(\bigvee \mathbb{P}_{v_0}(\psi) \wedge v_0)$. By the mentioned lemma, we know exactly which v_0 we need: it is v' for which $v(i \rightsquigarrow_p j)v'$ giving $v \wedge \langle i \rightsquigarrow_p j\rangle(\bigvee \mathbb{P}_{v'}(\psi) \wedge v')$. We can rewrite this into $v \wedge \langle i \rightsquigarrow_p j\rangle \bigvee\{\pi \wedge v' \mid \pi \in \mathbb{P}_{v'}(\psi)\}$, and then push the diamond $\langle i \rightsquigarrow_p j\rangle$ inside the disjunction to get $v \wedge \bigvee\{\langle i \rightsquigarrow_p j\rangle(\pi \wedge v') \mid \pi \in \mathbb{P}_{v'}(\psi)\}$. But then Lemma 5.14, item (5) yields $v \wedge \bigvee \mathbb{P}_{v'}(\psi)$. The other direction is similar: if $v \wedge \bigvee \mathbb{P}_{v'}(\psi)$ and $v(i \rightsquigarrow_p j)v'$, then, by Lemma 5.14, item (5), we get $v \wedge \langle i \rightsquigarrow_p j\rangle(\bigvee \mathbb{P}_{v'}(\psi) \wedge v')$, from which the desired result follows directly.

- $\varphi = [\psi'?]\psi$. By axiom (Test$_\delta$), $[\psi'?]\psi$ is equivalent to $\neg\psi' \vee \psi$, which has an equivalent formula of the right form by the induction hypothesis.
- $\varphi = [\delta_1;\delta_2]\psi$. By axiom (Comp$_\delta$), $[\delta_1;\delta_2]\psi$ is equivalent to $[\delta_1][\delta_2]\psi$, which has an equivalent formula of the right form by the induction hypothesis.
- $\varphi = [\delta_1 \cup \delta_2]\psi$. By axiom (Union$_\delta$), $[\delta_1 \cup \delta_2]\psi$ is equivalent to $[\delta_1]\psi \vee [\delta_2]\psi$, which has an equivalent formula of the right form by the induction hypothesis.
- $\varphi = [\delta^*]\psi$. Recall the number $N = N(n,k)$ as given in Lemma 5.14, item (6). Using axiom (Mix$_\delta$) and (K$_\delta$), we know that $[\delta^*]\psi$ is equivalent to $\psi \wedge [\delta][\delta^*]\psi$. Doing this N times, we obtain

$$\psi \wedge [\delta]\psi \wedge [\delta]^2\psi \wedge \cdots \wedge [\delta]^N\psi \wedge [\delta]^N[\delta^*]\psi.$$

By the induction hypothesis, we know that all except the last conjunct have an equivalent normal form. But since there are only N different such forms, the conjunct $[\delta]^N\psi$ must be equivalent to one of the earlier conjuncts $[\delta]^i\psi$ $(i < N)$. Now define $\lambda = \bigwedge_{i=0..N}[\delta]^i\psi$. We claim that

$$\lambda \leftrightarrow [\delta^*]\psi. \tag{5.19}$$

Then, this case follows since λ has an equivalent formula of the right form by the induction hypothesis.

The right-to-left direction of (5.19) is obvious since λ "is just the first part" of the "unraveling" $\psi \wedge [\delta]\psi \wedge [\delta][\delta]\psi \ldots$ of $[\delta^*]\psi$ using axioms (Mix$_\delta$) and (K$_\delta$). To show the other direction from left to right, it is sufficient to derive $\lambda \rightarrow [\delta^*]\lambda$, because, by the fact that ψ is just one of the conjuncts in λ, this immediately gives $\lambda \rightarrow [\delta^*]\psi$. To show the derivability of $\lambda \rightarrow [\delta^*]\lambda$, we will use (Ind$_\delta$). First of all, we show $\lambda \rightarrow [\delta]\lambda$. To see this, note that $[\delta]\lambda \leftrightarrow [\delta][\delta]^0\psi \wedge [\delta][\delta]^1\psi \wedge \cdots \wedge [\delta][\delta]^N\psi$. By the induction hypothesis, each conjunct $[\delta]^i\psi$ $(i \leq N)$ has a normal form, say, B_i. Then we obtain from λ a sequence $B_0, B_1, \ldots, B_N$ of N plus 1 formulas in normal form. But since there are at most N provably non-equivalent formulas by Lemma 5.14, item (6), we know that there is a B_j that equals a previous B_a with $a < j \leq N$. Let B_j be the first such repetition in this sequence. Notice that we have now $[\delta]^j\psi \equiv B_j = B_a \equiv [\delta]^a\psi$, and thus $[\delta]^k[\delta]^j\psi \equiv [\delta]^k[\delta]^a\psi$, for any $k \geq 0$. But then, it follows that the last conjunct $[\delta][\delta]^N\psi$ in $[\delta]\lambda$ is equivalent to $[\delta]^k[\delta]^a\psi$ (with $k = N - j$), which already appears in λ. Now that we have derived $\lambda \rightarrow [\delta]\lambda$, we apply (Necessitation) for $[\delta^*]$, and obtain $\vdash [\delta^*](\lambda \rightarrow [\delta]\lambda)$. Finally, by applying (Ind$_\delta$), we get $\lambda \rightarrow [\delta^*]\lambda$.

□

We know from Lemma 5.14 that there are only finitely many different normal forms: since every formula has such a normal form, there can be only finitely many non-equivalent formulas. We also know from the proof of Theorem 5.15 above that, for $[\delta^*]\psi$, we only have to consider a finite number of conjuncts $[\delta]^i\psi$ in its unraveling.

COROLLARY 5.16. *There are only finitely many pairwise non-equivalent formulas of DCL-PC. In fact, given the number n of agents, and k of propositional variables, we have:*

(1) $\vdash \bigwedge_{i \neq j \leq M}(\psi_i \leftrightarrow \psi_j) \rightarrow \bigvee_{i \leq M}\psi_i$, *and*
(2) $\vdash [\delta^*]\psi \leftrightarrow \bigwedge_{i \leq N}[\delta]^i\psi$,

where $M = 2^{nk}$ and $N = 2^{2^{nk}}$ (as defined in Lemma 5.14, item (6)).

Completeness of a derivation system with inference relation $\vdash$ with respect to a semantics means that every semantically valid formula is also provable: $\models \varphi \Rightarrow \vdash \varphi$. In order to prove completeness, often the contrapositive of this is shown: $\nvdash \varphi \Rightarrow \not\models \varphi$.

That is, every consistent formula $\neg\varphi$ has a model. A popular technique in modal logic is to construct a dedicated model (the canonical model, cf. [33]) for any consistent formula ψ. That canonical model is a "bridge" between syntax and semantics: it consists of all maximal consistent sets Φ (as worlds), and is constructed in such a way that membership of a formula in Φ and truth at the world corresponding to Φ coincide.

We can use Theorem 5.15 directly to generate a canonical model for a consistent formula ψ, as follows. First, determine its equivalent $\bigvee_{v\in\mathbf{A}}(\bigvee\mathbb{P}_v \wedge v)$. Pick any allocation description v for which $\mathbb{P}_v$ is not empty. Such a v exists by the fact that ψ is consistent. But now we have all the information needed to satisfy ψ: let the allocation ξ be such that is corresponds with v, i.e., such that $v = v_\xi$. Moreover, take any $\pi \in \mathbb{P}_v$, and take the valuation θ corresponding with π, i.e., for which $\pi_\theta = \pi$. It is immediate that we have, when we define $\mathfrak{M}$ as having this allocation function ξ, that $\mathfrak{M}, \theta \models^{\mathsf{K}} \bigvee_{v\in\mathbf{A}}(\bigvee\mathbb{P}_v \wedge v)$. Note that $\mathfrak{M}$ looks underspecified: we did not define what the accessibility relations look like. But indeed, our normal form theorem (i.e., Theorem 5.15) tells us that any reference to the programs for which these relations are the interpretation can be eliminated. Also note that as we constructed $\mathfrak{M}$, although $\bigvee_{v\in\mathbf{A}}(\bigvee\mathbb{P}_v \wedge v)$ is satisfied at θ, the equivalent formula ψ is strictly speaking not satisfied, since we have not specified what the models look like in "the other horizontal layers".

Alternatively, we can, given a set Σ of agents and a set Π of variables, conceive of the collection $\mathcal{K}(\Sigma, \Pi)$ of all Kripke models $\mathfrak{M} = \langle\Theta, R_{i\in\Sigma}, \xi\rangle$ over Σ and Π as the context to show completeness. Take an arbitrary consistent formula ψ. Build a maximal consistent set Γ "around it". This Γ contains several $\langle i \leadsto_p j\rangle\varphi$ and $\Diamond_C\varphi$ formulas, but it also contains one of the disjuncts of its normal form, say $\bigvee\mathbb{P}_v \wedge v$. One property of a maximal consistent set is that it then must contain, for exactly one $\pi \in \mathbb{P}_v$, the formula $\pi \wedge v$. Now, it is easy to see that within $\mathcal{K}(\Sigma, \Pi)$, the Kripke structure $\mathfrak{M}$ with allocation ξ such that $v = v_\xi$ and with valuation $\theta = \theta_\pi$ satisfies the normal form $\bigvee_{v\in\mathbf{A}}(\bigvee\mathbb{P}_v \wedge v)$, but also ψ itself, due to the soundness of the proof system, and hence of the way we obtained the normal form. As a result, we can directly interpret all subformulas of the form $\langle i \leadsto_p j\rangle\varphi$ and $\Diamond_C\varphi$ in the proper way, in $\mathfrak{M}, \theta$. We conclude:

THEOREM 5.17. *The axiomatic system DCL-PC is sound and complete with respect to both the Kripke and the direct semantics.*

5.3. Expressive Power and Complexity

We know from the results of van der Hoek and Wooldridge [249] that the model checking and satisfiability problems for CL-PC are PSPACE-complete, and since DCL-PC subsumes CL-PC, this implies a PSPACE-hardness lower bound on the corresponding problems for DCL-PC. The obvious question is then whether the dynamic constructs introduced in DCL-PC lead to a more complex decision problem – and in particular, whether DCL-PC satisfiability matches the EXPTIME-completeness of PDL satisfiability [**107**, pp.216–220]. In this section, we show that the complexity of the model checking and satisfiability problems are no worse than for CL-PC: they are both PSPACE-complete. (Notice that when we consider the model checking problem in this section, we consider the problem with respect to *direct* models, not Kripke models. Of course, with respect to satisfiability, it makes no difference: a formula is satisfiable with respect to direct models iff it is satisfiable wrt. Kripke models.)

```
1.  function $program\text{-}eval(\delta, \mathcal{M} = \langle \Sigma, \Pi, \xi, \theta \rangle, d)$ returns 'fail' or a structure over $\Sigma$, $\Pi$
2.    if $\delta = \varphi?$ then
3.      return $\mathcal{M}$ if DCL-PC$-eval(\varphi, \mathcal{M})$
4.          or 'fail' otherwise
5.    elsif $\delta = (i \rightsquigarrow_p j)$ then
6.      return $\langle \Sigma, \Pi, \xi', \theta \rangle$ if $p \in \Pi_i$
7.            where $\xi' = \xi$ if $i = j$
8.            otherwise $\xi = \langle \Pi_1, \ldots, \Pi_n \rangle$ and $\xi' = \langle \Pi'_1, \ldots, \Pi'_n \rangle$
9.            with $\Pi'_i = \Pi_i \setminus \{p\}$,
10.               $\Pi'_j = \Pi_j \cup \{p\}$, and
11.               $\Pi'_\ell = \Pi_\ell$ for all $\ell \leq n$, $\ell \neq i, j$
12.         or 'fail' otherwise
13.   elsif $\delta = (\delta_1; \delta_2)$ then
14.     return $program\text{-}eval(\delta_2, program\text{-}eval(\delta_1, \mathcal{M}, d), d)$
15.   elsif $\delta = (\delta_1 \cup \delta_2)$ then non-deterministically choose to either
16.     return $program\text{-}eval(\delta_1, \mathcal{M}, d)$
17.         or $program\text{-}eval(\delta_2, \mathcal{M}, d)$
18.   elsif $\delta = \delta'^*$ then
19.     return 'fail' if $d = 0$
20.         or otherwise (if $d > 0$) non-deterministically choose to either
21.           return $\mathcal{M}$
22.               or $program\text{-}eval((\delta'; \delta'^*), \mathcal{M}, d - 1)$
23. end-function
```

FIGURE 5.2. An algorithm for deciding $(\mathcal{M}, \mathcal{M}') \in R_\delta$.

Before proving PSPACE-completeness for DCL-PC model checking, consider some auxiliary notions first. A *program sequence* is a delegation program that is composed of atomic delegation programs, tests, and sequential composition only. A program δ *admits* a program sequence β if δ can be unfolded into β by recursively applying the following rules: For any atomic delegation program $(i \rightsquigarrow_p j)$, test $\varphi?$, and delegation programs δ_ℓ $(\ell \geq 0)$:

$$\begin{array}{rcl} (i \rightsquigarrow_p j) & \rightsquigarrow & (i \rightsquigarrow_p j); \\ \varphi? & \rightsquigarrow & \varphi?; \\ \delta_1; \delta_2 & \rightsquigarrow & \delta_1; \delta_2; \\ \delta_1 \cup \delta_2 & \rightsquigarrow & \delta_1 \text{ or } \delta_2; \\ \delta^* & \rightsquigarrow & \delta_1; \delta_2; \ldots; \delta_n, \text{ for some } n \geq 0, \\ & & \text{where } \delta_\ell = \delta, \text{ for all } \ell \leq n. \end{array}$$

The following two lemmas establish that membership in the accessibility relation R_δ for a delegation program δ can be decided in polynomial space.

LEMMA 5.18. *For all delegation programs δ' and all (direct) models $\mathcal{M}$ and $\mathcal{M}'$, $(\mathcal{M}, \mathcal{M}') \in R_{\delta'}$ implies that δ' admits a program sequence β of length at most exponential in the length of δ' such that $(\mathcal{M}, \mathcal{M}') \in R_\beta$. In fact, the length of β can be limited to $2^{|\delta'|^3}$.*

PROOF. Let δ', $\mathcal{M}$ and $\mathcal{M}'$ be as in the lemma. The proof is by induction on the structure of δ'. The only interesting case is where $\delta' = \delta^*$; the other cases are straightforward. Suppose $(\mathcal{M}, \mathcal{M}') \in R_{\delta^*}$. Since $R_{\delta^*} = (R_\delta)^*$, there is a sequence $\mathcal{M}_0, \ldots, \mathcal{M}_n$, $n > 0$, of models such that $\mathcal{M}_0 = \mathcal{M}$, $\mathcal{M}_n = \mathcal{M}'$ and $(\mathcal{M}_{i-1}, \mathcal{M}_i) \in R_\delta$, for each i with $1 \leq i \leq n$. By the transitivity of R_{δ^*}, we can assume that the sequence $\mathcal{M}_0, \ldots, \mathcal{M}_n$ is such that $\mathcal{M}_i \neq \mathcal{M}_j$, for all i, j with $1 \leq i < j \leq n$, i.e., the sequence of models contains no loops. The induction hypothesis yields that δ admits program sequences $\beta_1, \ldots, \beta_n$ such that β_i is of length at most $2^{|\delta|^3}$ and $(\mathcal{M}_{i-1}, \mathcal{M}_i) \in R_{\beta_i}$ for each i with $1 \leq i \leq n$. But then $\beta = \beta_1; \beta_2; \ldots; \beta_n$ is a program sequence admitted by δ^* such that $(\mathcal{M}, \mathcal{M}') \in R_{\delta^*}$. In the following, it is shown that β has the required length. Note that all models reachable from $\mathcal{M} = \langle \Sigma, \Pi, \xi, \theta \rangle$ via R_{δ^*} only differ in the allocation ξ of propositional variables in Π to the agents in Σ. More precisely, they differ in the allocation of propositional variables to agents that occur in δ. Thus there are at most ℓ^m such reachable models, where ℓ is the number of propositional variables occurring in δ and m the number of agents occurring in δ. Notice that n does not exceed ℓ^m; otherwise the sequence $\mathcal{M}_0, \ldots, \mathcal{M}_n$ contains loops contradicting the assumption. Together with the fact that $\ell^m \leq |\delta|^{|\delta|} \leq 2^{|\delta|^2}$, an upper bound for the length of β can be given as follows:

$$\begin{aligned} |\beta| = |\beta_1; \beta_2; \ldots; \beta_n| &\leq n \times \sup\{|\beta_i| : 1 \leq i \leq n\} + n \\ &\leq 2^{|\delta|^2} \times 2^{|\delta|^3} + 2^{|\delta|^2} \\ &= 2^{|\delta|^2 + |\delta|^3} + 2^{|\delta|^2} \\ &\leq 2^{(|\delta|+1)^3} \\ &\leq 2^{|\delta^*|^3}. \end{aligned}$$

□

LEMMA 5.19. *For all programs δ and all (direct) models $\mathcal{M}$ and $\mathcal{M}'$, the membership problem $(\mathcal{M}, \mathcal{M}') \in R_\delta$ can be decided in* PSPACE.

PROOF. Let δ be a program and let $\mathcal{M}$, $\mathcal{M}'$ be two (direct) models. Consider the following algorithm that decides $(\mathcal{M}, \mathcal{M}') \in R_\delta$ by using the function *program−eval*$(\cdots)$ in Figure 5.2:

1. Set $d = 2^{|\delta|^3}$.
2. If *program−eval*$(\delta, \mathcal{M}, d) = \mathcal{M}'$, then return '$(\mathcal{M}, \mathcal{M}') \in R_\delta$', and 'No' otherwise.

To see that this algorithm is correct, it is shown that $program{-}eval(\delta, \mathcal{M}, d) = \mathcal{M}'$ iff $(\mathcal{M}, \mathcal{M}') \in R_\delta$. For the direction from left to right, it is readily checked that $program{-}eval(\delta, \mathcal{M}, d) = \mathcal{M}'$ implies the existence of a program sequence β admitted by δ of length at most $|\delta| \times d$ such that $(\mathcal{M}, \mathcal{M}') \in R_\beta$. Clearly, $R_\beta \subseteq R_\delta$ and thus $(\mathcal{M}, \mathcal{M}') \in R_\delta$. Consider the direction from right to left. From $(\mathcal{M}, \mathcal{M}') \in R_\delta$, it follows by Lemma 5.18 that there is a program sequence β admitted by δ of length at most $2^{|\delta|^3}$ such that $(\mathcal{M}, \mathcal{M}') \in R_\beta$. Step 1 ensures that the value of d is such that $2^{|\delta|^3} \leq |\delta| \times d$. Then it is obvious by construction of the algorithm that the non-deterministic choices in the lines 15 and 20 of Figure 5.2 yield that $program{-}eval(\delta, \mathcal{M}, d) = \mathcal{M}'$. Notice that the algorithm terminates since the recursive calls in the lines 14, 16 and 17 are applied on strict subprograms only and the recursive call in Line 22 is followed by the one in Line 14 while the parameter d limits the recursion depth.

The above algorithm can be run in polynomial space. To see that, notice that the function DCL-PC$-eval(\cdots)$, which is called in Line 3, can be computed in polynomial space and that the parameter d is encoded in binary. Moreover, the stack of an algorithm computing the function $program{-}eval(\cdots)$ can be limited to a size polynomial in the length of δ. Note that the stack only needs to store the currently evaluated program and the programs at the backtracking points, which are introduced at the nested function call in Line 14. But since this nested function call is applied on strict subprograms, there are only linearly many backtracking points needed at a time. Although the algorithm is non-deterministic, it follows from the well-known fact NPSPACE equals PSPACE **[217]** that it runs in PSPACE. □

REMARK 5.20. The fact that NPSPACE equals PSPACE **[217]** tells us that the non-deterministic algorithm from Lemma 5.19 can be turned into a deterministic one. Such transformations are well-known. ⊣

Using the previous two lemmas, we can now prove the following.

THEOREM 5.21. *The model checking problem DCL-PC (wrt. direct structures) is* PSPACE*-complete.*

PROOF. Given that DCL-PC subsumes the PSPACE-hard logic CL-PC, we only need to prove the upper bound. Consider the function DCL-PC$-eval(\cdots)$ in Figure 5.3. Soundness is obvious by construction. First note that the algorithm is strictly analytic: recursion is always on a sub-formula of the input. That the algorithm is in PSPACE follows from the fact that the loops at lines 10–12 and 15-18 involve, in the first case simply binary counting with the variables Π_C, and in the second simply looping through all direct models over Σ and Π: we do not need to store these models once they are checked, and so this can be done in polynomial space. Finally, Lemma 5.19 yields that the check $(\mathcal{M}, \mathcal{M}') \in R_\delta$ on Line 16 can be done in polynomial space. □

Now, we make use of the following result (the proof of which is identical to the equivalent result proved in **[249]**).

```
1. function DCL-PC-$eval(\varphi, \mathcal{M} = \langle \Pi, \Sigma, \xi_0, \theta \rangle)$ returns tt or ff
2.    if $\varphi \in \Pi$ then
3.       return $\theta(\varphi)$
4.    elsif $\varphi = \neg\psi$ then
5.       return not DCL-PC-$eval(\psi, \mathcal{M})$
6.    elsif $\varphi = \psi_1 \vee \psi_2$ then
7.       return DCL-PC-$eval(\psi_1, \langle \Pi, \Sigma, \xi_0, \theta \rangle)$
8.          or DCL-PC-$eval(\psi_2, \langle \Pi, \Sigma, \xi_0, \theta \rangle)$
9.    elsif $\varphi = \Diamond_C \psi$ then
10.      for each $C$-valuation $\theta_C$
11.            if DCL-PC-$eval(\psi, \langle \Pi, \Sigma, \xi_0, \theta \rangle \oplus \theta_C)$ then return tt
12.      end-for
13.      return ff
14.   elsif $\varphi = \langle \delta \rangle \psi$ then
15.      for each structure $\mathcal{M}'$ over $\Sigma$, $\Pi$
16.         if $(\mathcal{M}, \mathcal{M}') \in R_\delta$ then
17.            if DCL-PC-$eval(\varphi, \mathcal{M}')$ then return tt
18.      end-for
19.      return ff
20. end-function
```

FIGURE 5.3. A model checking algorithm for DCL-PC.

LEMMA 5.22. *If a DCL-PC-formula φ is satisfiable, then it is satisfied in a (direct) structure $\mathcal{M}$ such that size$(\mathcal{M}) = |\Pi_\varphi| + |\Sigma_\varphi| + 1$.*

We can now prove the following.

THEOREM 5.23. *The satisfiability problem for DCL-PC is* PSPACE*-complete.*

PROOF. Given a formula φ, loop through each direct structure $\mathcal{M}$ containing Π_φ and Σ_φ such that *size*$(\mathcal{M}) = |\Pi_\varphi| + |\Sigma_\varphi| + 1$, and if $\mathcal{M} \models^{\mathsf{d}} \varphi$ then return 'Yes'. If we have considered all such models, return 'No'. By Theorem 5.21, we can check whether $\mathcal{M} \models^{\mathsf{d}} \varphi$ in polynomial space. □

Notice that the PSPACE complexity for checking satisfiability depends upon the fact that models for DCL-PC are concise, and that hence we can loop through them all in polynomial space (we do not need to keep track of the models after they have been considered).

REMARK 5.24. In Section 5.1.4, we gave interpreted formulas of ATL under the semantics of DCL-PC. Notice that, by interpreting ATL-formulas in models for DCL-PC, we can reduce the satisfiability problem for ATL from EXPTIME-complete [265] to PSPACE-complete. Of course, this is not a *general* result, in the sense that we are assuming a specific interpretation for ability in terms of propositional variables distributed to agents in the system. But, for this special case, it seems ATL has a simpler satisfiability problem. ⊣

5.4. Characterising Control

One of the main concerns in the original study of CL-PC [**249**] was to investigate the logical characterisation of *control*: the extent to which we could characterise, in the logic, what states of affairs agents could reliably control. Control was distinguished from ability in the sense that, for example, no agent could be said to control a tautology, even if one might be prepared to concede that an agent would have the ability to bring about a tautology. The starting point for the study of control in [**249**] was the $\mathit{controls}(i,p)$ construct: as we have already seen, such an expression will be true if, and only if, the variable p is under the control of agent i. This led to an analysis and characterisation of the types of formulas that an agent could be said to control. The type of control studied in [**249**] derives from the ability of agents to choose values for the propositional variables under their control. Let us refer to this type of control, where an agent is directly able to exert some influence over some state of affairs by assigning values to its variables, as *first-order control*. In this section, we undertake a similar study of control in the richer setting of DCL-PC. Here, however, we have a second type of control, which derives from the ability to *transfer control of variables to other agents*. Thus, for example, if i controls p, she also "has the power" to ensure for instance $\mathit{controls}(j,p)$, where j is an agent different from i. This control is expressed through the delegation modality: $\langle i \rightsquigarrow_p j\rangle \mathit{controls}(j,p)$. We refer to this type of control as *second-order control*. We will see that these types of "control" are indeed rather orthogonal. For instance, $\langle i \rightsquigarrow_p j\rangle \Diamond_j \varphi$ (i can give p to j, who then can achieve φ) and $\Diamond_{i,j}\varphi$ (i and j can cooperate, to achieve φ) are logically incomparable. For example, taking $\varphi = \langle j \rightsquigarrow_p i\rangle\top$ gives

$$\models^{\mathsf{d}} \mathit{controls}(i,p) \rightarrow (\langle i \rightsquigarrow_p j\rangle\varphi \land \neg\Diamond_{i,j}\varphi)$$

while for $\varphi = \langle i \rightsquigarrow_p j\rangle\top$ and assuming $i \neq j$, we have

$$\models^{\mathsf{d}} \mathit{controls}(i,p) \rightarrow (\neg\langle i \rightsquigarrow_p j\rangle\varphi \land \Diamond_{i,j}\varphi).$$

However, if the goal is an objective formula, we can relate atomic control and delegation, as we will shortly see.

To begin our study, consider the delegation program

$$\mathtt{give}_i := \bigcup_{p\in\Pi} \left(\mathit{controls}(i,p)?; \bigcup_{j\in\Sigma} i \rightsquigarrow_p j\right). \tag{5.20}$$

Then $\langle\mathtt{give}_i\rangle\varphi$ would express that i has a way to give one of her propositional variables to one of the agents (possibly herself) in such a way that consequently φ holds. Thus, $\langle\mathtt{give}_i^*\rangle\varphi$ means that i can distribute her variables among the agents in such a way that afterwards φ holds. Hence, when reasoning about i's power, the strongest that she can achieve is any φ for which $\Diamond_i\varphi \lor \langle\mathtt{give}_i^*\rangle\varphi$, expressing that i can achieve φ by either choosing an appropriate value for her variables, or by distributing her variables over Σ in an appropriate way. Note that both $\Diamond_i\varphi$ and $\langle\mathtt{give}_i^*\rangle\varphi$ imply

$\langle \mathtt{give}_i^* \rangle \Diamond_i \varphi$, and hence any φ for which $\langle \mathtt{give}_i^* \rangle \Diamond_i \varphi$ holds can be seen as what i can achieve on her own. We will come back to the program $\mathtt{give}_i$ below.

The program **give** can be generalised to incorporate coalitions that can give away variables, and those that can receive: let

$$\mathtt{give}_{C \rightsquigarrow D} := \bigcup_{i \in C} \bigcup_{p \in \Pi} \big(\mathit{controls}(i,p)?; \bigcup_{j \in D \cup \{i\}} \langle i \rightsquigarrow_p j \rangle\big). \tag{5.21}$$

This program $\mathtt{give}_{C \rightsquigarrow D}$ lets an arbitrary agent i from the coalition C either give any of her variables p to an arbitrary member of the coalition D, or do nothing (i.e., give them to herself). Now, for objective formulas φ, we have the following, where i is a dedicated agent from C:

$$\Diamond_C \varphi \leftrightarrow \langle \mathtt{give}^*_{C \rightsquigarrow \{i\}} \rangle \Diamond_i \varphi.$$

In words: the agents in the coalition C can choose values for their variables such that φ, if and only if they have a way to give all their variables to the dedicated agent i, who then can achieve φ. Note that we are in general not able to eliminate all occurrences of $\Diamond$'s, since this is the only way to express first-order control, i.e., to reason about a "different valuation".

For some examples in the language without delegation, we refer to [**249**], especially to the example 'Bach or Stravinsky' (Example 2.4 in [**249**]). Before looking at two examples of control in a dynamic setting, note that $\Diamond_\Sigma$ allows the following equation, for any objective formula φ:

$$\varphi \text{ is consistent } \Leftrightarrow \models^d \Diamond_\Sigma \varphi \tag{5.22}$$

This equation states that the grand coalition Σ can achieve any satisfiable objective formula and vice versa.

EXAMPLE 5.25. (FAIRNESS). Suppose we have n agents: $1, \ldots, n$. Each controls a flag r_i $(i = 1...n)$ to indicate that they desire control over a particular resource, modelled with a variable p. It is not that they want p to be true or false every now and then, (which could be taken care of by a central agent), but rather, they want to *control* p eventually. Let $+_n$ denote addition modulo n, and, similarly, $-_n$ subtraction modulo n. Let **skip** denote $\top?$, i.e., a test on a tautology. Consider the following program:

```
grant_req($i$) :=  if  $\neg controls(i,p)$  then  skip else
                   if  $r_{i+_n 1}$             then  $(i \rightsquigarrow_p i +_n 1)$ else
                   ...                             ...
                   if  $r_{i+_n(n-1)}$          then  $(i \rightsquigarrow_p i +_n (n-1))$ else skip
```

The program $\mathtt{grant\_req}(i)$ makes agent i pass on the resource p whenever she has it and somebody else needs it, where the need is checked in the order starting with the agent with the next $+_n$ index. Note that the "use" of this variable p, i.e., making it true or false, is not encoded in our program constructs. Now consider the program:

$$\mathtt{pass\_on}(i,j) := \mathtt{grant\_req}(i); \ldots ; \mathtt{grant\_req}(j -_n 1).$$

The program pass_on(i, j) will pass control over the variable p to agent j, provided that initially r_j is set and one of the agents in the sequence $i, i +_n 1, \ldots, j -_n 1$ owns it. This can be expressed as follows:

$$r_j \wedge \mathit{controls}(\{i, i +_n 1, \ldots, j -_n 1\}, p) \rightarrow [\mathtt{pass\_on}(i, j)]\mathit{controls}(j, p).$$

Now we have:

$$\begin{aligned} r_i \rightarrow \ & \big(\langle \mathtt{pass\_on}(i +_n 1, i)\rangle \mathit{controls}(i, p) \\ & \wedge [\mathtt{pass\_on}(i +_n 1, i)]\mathit{controls}(i, p)\big). \end{aligned}$$

That is, if agent i flags a request r_i for source p, then, after the program pass_on$(i +_n 1, i)$ has been executed, i will be under control of p. ⊣

Notice that the previous example "freely" passes on a variable along chains of agents, thereby taking for granted that they can control that variable on the fly, and making it true or false at will. In the following example, control over a variable is not only important, but also the truth of some side conditions involving them.

EXAMPLE 5.26. (CLIENT-SERVER-SCENARIO). We have a scenario with three agents: two clients c_1 and c_2, and a server s. The server always has control over one of the propositional variables p_1 and p_2, in particular s wants to guarantee that those variables are never true simultaneously. At the same time, c_1 and c_2 want to ensure that at least one of the variables p_i $(i = 1, 2)$ is true, where variable p_i belongs to client c_i. We can describe the invariant of the system with the formula Inv:

$$Inv := \bigvee_{i=1,2} \mathit{controls}(s, p_i) \ \wedge \ \bigvee_{i=1,2} \mathit{controls}(c_i, p_i)$$

Consider the following delegation program β:

$$\begin{aligned} \beta := \ & \big((\mathit{controls}(s, p_1))? \ ; \ s \rightsquigarrow_{p_1} c_1 \ ; \ c_2 \rightsquigarrow_{p_2} s) \\ & \cup(\mathit{controls}(s, p_2))? \ ; \ s \rightsquigarrow_{p_2} c_2 \ ; \ c_1 \rightsquigarrow_{p_1} s)\big)^*. \end{aligned}$$

This says that an arbitrary number of times one variable p_i is passed from the server to the client c_i, and another variable p_j $(i \neq j)$ from the client c_j to the server.

Using Inv and β, we can describe the whole scenario as follows:

$$\begin{aligned} & Inv \rightarrow [\beta]Inv \\ & Inv \rightarrow [\beta]\big(\Diamond_s \neg(p_1 \wedge p_2) \wedge \Diamond_{\{c_1, c_2\}}(p_1 \vee p_2)\big) \end{aligned}$$

⊣

In **[248]**, van der Hoek and Wooldridge gave a general characterisation of the types of formulas that agents and coalitions could control, and our aim is now to undertake the same study for DCL-PC. It will appear that this can be done on a local and a global level, but we will also see that the notion of control that we inherited from **[249]**, has a natural generalisation in our context.

The next corollary establishes a result concerning the characterisation of control. If follows easily from the truth definition of the Kripke semantics defined in Section 5.1.3, and from Theorem 5.15 and Theorem 5.17.

COROLLARY 5.27. *Let $\mathfrak{M}$ be a Kripke structure, C any coalition $C \subseteq \Sigma$ with $C \neq \Sigma$, and let φ be ranging over DCL-PC formulas. Then it follows that:*

(1) *For no φ, do we have $\models^{\mathsf{K}}$ controls(C, φ); and*
(2) *$\models^{\mathsf{K}}$ controls(Σ, φ) iff the formula $\bigvee_{v \in \Upsilon}(\bigvee \Pi_v(\varphi) \wedge v)$, to which φ is equivalent according to Theorem 5.15, is such that for no allocation description v, $\mathbb{P}_v(\varphi) = \mathbb{P}$ or $\mathbb{P}_v(\varphi) = \emptyset$.*

PROOF. (1) In order for *controls*(C, φ) to be valid under the Kripke semantics, it has to be true at all worlds in all Kripke structures. Take any Kripke structure $\mathfrak{M} = \langle \Theta, R_{i \in \Sigma}, \xi \rangle$ for which the allocation $\xi = \langle \Pi_1, \Pi_2, \ldots, \Pi_n \rangle$ is such that $\Pi_j = \emptyset$, for all $j \in C$. We then have $\mathfrak{M}, \theta \models^{\mathsf{K}} \Diamond_C \varphi$ iff for some world $\theta' \in \Theta$ with $\theta' = \theta \pmod{\Pi_C}$, it holds that $\mathfrak{M}, \theta' \models^{\mathsf{K}} \varphi$. But, since $\Pi_C = \emptyset$, the only such θ' is θ itself, so that we cannot have $\mathfrak{M}, \theta \models^{\mathsf{K}} \Diamond_C \varphi \wedge \Diamond_C \neg \varphi$. Hence, $\mathfrak{M}, \theta \not\models^{\mathsf{K}}$ *controls*(C, φ).

(2) First we prove the left-to-right direction by contraposition. Let φ be equivalent to $\bigvee_{v \in \Upsilon}(\bigvee \Pi_v(\varphi) \wedge v)$, which we have by Theorem 5.15. Suppose, for some allocation description v, that $\mathbb{P}_v(\varphi) = \mathbb{P}$. This means, for every Kripke structure $\mathfrak{M} = \langle \Theta, R_{i \in \Sigma}, \xi_v \rangle$ and for every valuation θ, that $\mathfrak{M}, \theta \models^{\mathsf{K}} \varphi$. Consequently, $\mathfrak{M}, \theta \models^{\mathsf{K}} \Diamond_\Sigma \varphi$. However, the agents in Σ can only change θ, but not the current allocation ξ_v. Since $\mathbb{P}_v(\varphi) = \mathbb{P}$, Σ cannot choose a valuation that falsifies φ, i.e., $\mathfrak{M}, \theta \models^{\mathsf{K}} \neg \Diamond_\Sigma \neg \varphi$. Similarly, if $\mathbb{P}_v(\varphi) = \emptyset$, we have $\mathfrak{M}, \theta \models^{\mathsf{K}} \Diamond_C \neg \varphi$, for each θ. But, given v, Σ cannot choose a valuation satisfying φ on the current allocation described by v, i.e., $\mathfrak{M}, \theta \models^{\mathsf{K}} \neg \Diamond_\Sigma \varphi$. Hence, in either case we have $\not\models^{\mathsf{K}}$ *controls*(Σ, φ).

Consider the other direction from right to left. Suppose φ is equivalent to $\bigvee_{v \in \Upsilon}(\bigvee \Pi_v(\varphi) \wedge v)$ while, for no v, the set $\mathbb{P}_v(\varphi)$ is either $\mathbb{P}$ or $\emptyset$. Let $\mathfrak{M} = \langle \Theta, R_{i \in \Sigma}, \xi \rangle$ be a Kripke structure. Remember that v_ξ is the allocation description corresponding to the allocation ξ. By the fact that $\mathbb{P}_{v_\xi} \neq \emptyset$, there is a valuation description $\pi \in \mathbb{P}_{v_\xi}$ such that the corresponding valuation θ_π satisfies φ at $(\mathfrak{M}, \theta_\pi)$. But then, Σ can choose θ_π in order to satisfy φ, and thus we have $\mathfrak{M}, \theta \models^{\mathsf{K}} \Diamond_\Sigma \varphi$, for any θ. Similarly, we have $\mathfrak{M}, \theta \models^{\mathsf{K}} \Diamond_\Sigma \neg \varphi$ which follows by the fact that $\mathbb{P}_{v_\xi} \neq \mathbb{P}$. Hence, we have $\models^{\mathsf{K}}$ *controls*(Σ, φ).

□

One may ask for a more local characterisation of what a coalition controls: for which Kripke models $\mathfrak{M}$ and valuations θ do we have $\mathfrak{M}, \theta \models^{\mathsf{K}}$ *controls*(C, φ)? For this notion of control, the answer can be immediately read off from Theorem 5.28, to be given shortly. That theorem is about a more general notion: to recover a characterisation result for the current notion *controls*(i, φ), we would only need the items (1b) and (2b) of Theorem 5.28.

The notion of control discussed so far is that taken from **[249]**, where we here have lifted the characterisation results to our richer language. However, as is clear from our discussion earlier in this section, a more appropriate notion of control of an individual i in our language might be obtained using the program $\mathbf{give}_i^*$, where $\mathbf{give}_i$ is defined in (5.20). Note that $\Diamond_i\varphi \to \langle\mathbf{give}_i^*\rangle\Diamond_i\varphi$ is valid, and hence that $\langle\mathbf{give}_i^*\rangle$ seems a more general way to reason about i's control: it is about what i can achieve *both by toggling its propositional variables and delegating some of them.* One can easily discuss this at the coalitional level, by lifting Definition (5.20) to the case of coalitions C as it was suggested in (5.21) with $\mathbf{give}_{C\rightsquigarrow D}$. However, we stick to the individual case here for simplicity. Let us therefore define

$$(5.23) \qquad CONTROLS(i,\varphi) \;:=\; \big(\langle\mathbf{give}_i^*\rangle\Diamond_i\varphi \wedge \langle\mathbf{give}_i^*\rangle\Diamond_i\neg\varphi\big)$$

This definition says that agent i controls a formula φ iff there is a way to distribute her propositional variables over the agents such that after i makes appropriate choices for her remaining variables, φ holds, but there is also a way of distributing her variables that enables her to enforce $\neg\varphi$. From the validity of $\Diamond_i\varphi \to \langle\mathbf{give}_i^*\rangle\Diamond_i\varphi$, we infer that $controls(i,\varphi)$ implies $CONTROLS(i,\varphi)$. Notice that the implication the other way around is not valid since $controls(i,\cdot)$ can never be true for control of other agents over variables. For example, $CONTROLS(i, controls(j,p))$ holds iff $p \in \Pi_i$. From this, we know that $controls(i,p) \to CONTROLS(i, controls(j,p))$ is a theorem, which basically says that when having control over a variable, you can freely choose to keep it or pass it on. However, $controls(i,p) \to controls(i, controls(j,p))$ is not valid, we even have $controls(i,p) \to \neg controls(i, controls(j,p))$: once agent i owns p, she cannot choose to keep p or to pass it on by only toggling her propositional variables.

Before we state our characterisation result, we introduce some more notation. For any two Kripke models $\mathfrak{M} = \langle\Theta, R_{i\in\Sigma}, \xi\rangle$ and $\mathfrak{M}' = \langle\Theta, R_{i\in\Sigma}, \xi'\rangle$ in $\mathcal{K}(\Sigma,\Pi)$ and for any agent i, we say that $\mathfrak{M} \geq_i \mathfrak{M}'$ if the allocations $\xi = \langle\Pi_1,\ldots,\Pi_i,\ldots,\Pi_n\rangle$ and $\xi' = \langle\Pi_1',\ldots,\Pi_i',\ldots,\Pi_n'\rangle$ are such that $\Pi_i' \subseteq \Pi_i$ and, for all $j \neq i$, $\Pi_j \subseteq \Pi_j'$. That is, $\mathfrak{M}'$ is obtained from $\mathfrak{M}$ by executing $\mathbf{give}_i^*$. In such a case, we also say $\xi \geq_i \xi'$.

For each valuation description $\pi \in \mathbb{P}$, let θ_π be the valuation that is described by π, and, for each allocation description $v \in \mathbb{A}$, let ξ_v be the allocation described by v, and let Π_i^v be the set of propositional variables controlled by agent i in ξ_v.

THEOREM 5.28. *Let φ be a DCL-PC-formula with $\varphi \equiv \bigvee_{v\in\mathbb{A}}\big(\bigvee\mathbb{P}_v(\varphi) \wedge v\big)$, as given by Theorem 5.15. Let $\mathfrak{M} = \langle\Theta, R_{i\in\Sigma}, \xi\rangle$ be a Kripke structure of $\mathcal{K}(\Sigma,\Pi)$, and θ a world in $\mathfrak{M}$. Then, for each agent $i \in \Sigma$,*

$$\mathfrak{M},\theta \models^{\mathsf{K}} CONTROLS(i,\varphi)$$

iff the following two conditions are satisfied:

(1) *There is a $v \in \mathbb{A}$ and a $\pi \in \mathbb{P}_v(\varphi)$ such that*
   (a) *$\xi \geq_i \xi_v$, and*
   (b) $\theta_\pi = \theta \pmod{\Pi_i^v}$.
(2) *There is a $v \in \mathbb{A}$ and a $\pi \in \mathbb{P} \setminus \mathbb{P}_v(\varphi)$ such that*

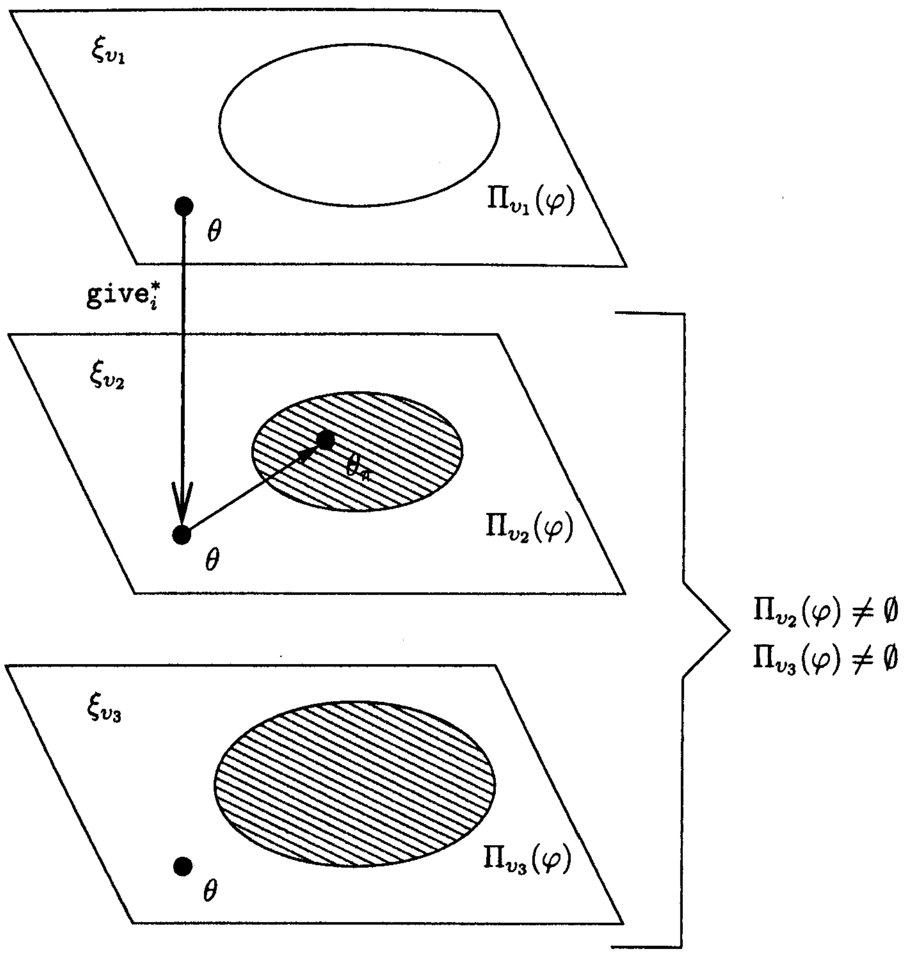


FIGURE 5.4. Illustration of $\varphi \equiv \bigvee_{v \in \mathbf{A}} \left(\bigvee \mathbb{P}_v(\varphi) \wedge v \right)$.

(a) $\xi \geq_i \xi_v$, *and*
(b) $\theta_\pi = \theta \pmod{\Pi_i^v}$.

We first demonstrate the requirements (1) and (2) of Theorem (5.28). Suppose that θ satisfies $(p \wedge \neg q \wedge r)$, agent $i = 1$ owns p in $\mathfrak{M}$, agent 2 owns q and r, and agent 3 has no propositional variables in $\mathfrak{M}$. First of all, to see why item (1b) is needed, we have to guarantee that for some $\mathfrak{M}', \theta'$, it holds that $\mathfrak{M}', \theta' \models^{\mathsf{K}} \Diamond_1 \varphi$. That means that, even after 1 has given away some of her atoms (resulting in some allocation ξ'), she still should be able to make φ true. This is possible for $\varphi = (\neg p \wedge \neg q \wedge r)$: agent 1 could simply stay within the current allocation ξ and just make p false. However, this is not possible for $\varphi = (\neg p \wedge \neg q \wedge r \wedge \mathit{controls}(3, p))$ since, once 1 has delegated control of p to 3, agent 1 cannot make p false anymore. Moreover, an agent i can only give atoms away, so any structure with allocation ξ_v that makes it possible for her to satisfy φ should be one for which $\xi \geq_i \xi_v$, which explains item (1a). Item (2a) has exactly the same motivation, and requirement (2b) is easily understood to be similar to (1b), once one realizes that the normal form of $\neg\varphi$ can be expressed in terms of the normal form of φ as follows:

$$\neg\varphi \equiv \bigvee_{v\in\mathbf{A}} \big(\bigvee(\mathbb{P}\setminus\mathbb{P}_v(\varphi)) \wedge v\big).$$

For a simple illustrating example, suppose there are only two allocations ξ_1 and ξ_2, and φ is equivalent to

$$\begin{array}{ll} & ((p\wedge q)\vee(p\wedge\neg q))\wedge\xi_1 \\ \vee & ((\neg p\wedge q)\vee(p\wedge\neg q))\wedge\xi_2. \end{array}$$

Note that φ's normal form describes the valuations on ξ_1 and ξ_2 where φ is satisfied. The normal form of $\neg\varphi$ is complementary to the one of φ in the sense that it describes the valuations on ξ_1 and ξ_2 where φ is falsified:

$$\begin{array}{ll} & ((\neg p\wedge\neg q)\vee(\neg p\wedge q))\wedge\xi_1 \\ \vee & ((\neg p\wedge\neg q)\vee(p\wedge q))\wedge\xi_2. \end{array}$$

PROOF. We illustrate our proof with a pictorial story that shows why the requirements 1 and 2 of the theorem are both sufficient and necessary. Given that φ is equivalent to $\bigvee_{v\in\Upsilon}(\bigvee\Pi_v(\varphi)\wedge v)$ by Theorem 5.15, semantically this means that φ corresponds to a collection of the "shaded areas", as depicted in Figure 5.4. Now, for $CONTROLS(i,\varphi)$ to be true at a world θ in a Kripke structure $\mathfrak{M}=\langle\Theta, R_{i\in\Sigma},\xi\rangle$, agent i has to be able to move *inside* such a shaded area, and to move *outside* it as well. But moving inside a shaded area, means being able to first go to a structure with allocation ξ_v, and then to a world θ_π within that structure such that the valuation description π is in $\mathbb{P}_v$. Notice that i can move to allocation ξ_v only by delegating control over her variables to other agents (hence the requirement $\xi\geq_i\xi_v$), and i can move to valuation θ_π only by toggling her remaining variables in Π_i^v at ξ_v (hence the condition $\theta_\pi=\theta \pmod{\Pi_i^v}$). This shows that Condition 1 is equivalent to $\mathfrak{M},\theta\models^{\mathsf{K}}\langle\mathtt{give}_i^*\rangle\Diamond_i\varphi$. Accordingly, Condition 2 corresponds to being able to move *outside* of a shaded area in Figure 5.4. Semantically, this means being able to first go to a structure with allocation $\xi_{v'}$, and then to a world $\theta_{\pi'}$ within that structure such that the valuation description π' is not in $\mathbb{P}_{v'}$. Consequently, Condition 2 is equivalent to $\mathfrak{M},\theta\models^{\mathsf{K}}\langle\mathtt{give}_i^*\rangle\Diamond_i\neg\varphi$, which finishes the proof. □

5.5. Conclusion

In this chapter, we have built upon the logic CL-PC of strategic cooperative ability, in which the control that agents have over their environment is represented by assigning them specific propositional variables, for which the agents that "own" them can determine their truth value. We extended the syntax of this logic with dynamic operators to this logic, thus obtaining the language DCL-PC in which one can directly reason about what agents and coalitions of agents can achieve by setting their assigned variables, or by giving the control over them to others. We gave two different but equivalent semantics for this language – a direct and a more conventional Kripke semantics – and provided a complete axiomatisation for them. The key property that establishes the proof of completeness for DCL-PC's axiomatic system is the fact that every formula in the language is provably equivalent to a normal form: a disjunction of conjunctions of literals over propositional variables p and assertions of the form $controls(i, p)$. We also investigated the complexity of the model checking and satisfiability problems for DCL-PC, and showed that these problems are no worse than for the program-free fragment CL-PC: they are both PSPACE-complete. We demonstrated that we can interpret ability in ATL as in CL-PC or DCL-PC – as having control over the truth values of propositional variables – by expressing ATL-formulas in DCL-PC, which implies a simpler satisfiability problem for ATL in this setting.

We demonstrated that, for the special case where ability in ATL is interpreted as in CL-PC or DCL-PC– as having control over the truth values of propositional variables – this implies a simpler satisfiability problem for ATL.

Although other researchers have begun to develop formal systems for reasoning about delegation (e.g., [**163**]), to the best of our knowledge DCL-PC is the first such system to have a rigorous semantics, and a complete axiomatisation. Also, the emphasis in [**163**] is on decentralized "trust management", in which roles like that of a requester, credentials and an authorizer are distinguished. In the work presented here, the emphasis is more on what coalitions can achieve, if they are allowed to hand over control over propositional variables.

There are several avenues for further development of this work. First of all, it is interesting to add the assignments that the agents can perform to the dynamic actions they can perform, so that the two dimensions of what agents can achieve become projected in one dimension. Secondly, in many realistic systems, property (5.22) may be too general: often, we want to specify that the overall system satisfies some constraints. For this, it seems appropriate, however, not only to reason about what agents *can* achieve, but also about what they *should* guarantee. The framework of *Social Laws* [**225, 187, 244**] could be set to work in order to express that under certain conditions, an agent will not set a certain propositional variable to 'true', or that she will not pass on control over a certain variable to a specific agent, or that the overall system behaves in such a way that every agent gets a fair chance to trigger a specific variable (i.e., use a specific resource) infinitely often.

CHAPTER 6

Conclusion

We conclude this work by summarizing the contributions, stating open problems and showing possible directions for further research.

Chapter 2 contains two contributions. First, we investigated a quantitative extension of CTL: *Real-Time Computational Tree Logic* (RTCTL) [**68**]. In addition to the CTL-operators, RTCTL provides for qualitative temporal assertions, quantitative operators for expressing real-time constraints. We proposed a new method for polynomially reducing satisfiability in RTCTL (whose numerical parameters are coded in binary) to satisfiability in the same logic with numbers coded in unary. The reduction uses new propositional variables that serve as the bits of a binary counter measuring distances. We reproved the original result of Emerson *et al.* [**68**] that RTCTL is in EXPTIME. As second contribution in this chapter, we suggested a framework for reasoning about strategic ability in MASs of resource-bounded agents: We introduced *Alternating-time Temporal Logic with Bounded Memory* (ATLBM), a family of ATL-variants that account for agents with limited amount of memory for storing strategies. We argued that the setting of agents with limited memory is closer to real-world applications. Compared to ATL, each path quantifier $\langle\langle C\rangle\rangle^n$ in ATLBM is additionally parameterised with a natural number n, which determines the amount of memory available to the coalition C. We defined and illustrated the variant ATLBM$^m_{ir}$ for agents under incomplete information and imperfect recall and started to characterise its expressive power by presenting some axioms.

In Chapter 3, we defined three variations of the satisfiability problem for ATL, where the set of agents that can occur formulas is not fixed externally in advance. We showed that all three variants of the satisfiability problem for ATL are EXPTIME-complete by means of an type elimination construction. Moreover, we showed that this result can be extended to ATEL with epistemic operators for individual knowledge, common and distributed knowledge. In this way, we proved that ATEL is EXPTIME-complete as well, which means that adding these epistemic operators to ATL does not yield an increase in computational complexity. For future work, it would be interesting to investigate the complexity of ATEL-variants that capture various desired and reasonable interactions between knowledge and time. Another interesting issue to address is the precise complexity of ATL* satisfiability which, as yet, lies between 2-EXPTIME and 3-EXPTIME.

In Chapter 4, we introduced *Alternating-time Temporal Logic with Explicit Strategies* (ATLES), an extension of ATL with explicit names for strategies in the object language.

The ATL path quantifiers are extended with the possibility to bind agents to strategy names. ATLES makes it possible to refer to the same strategy in different occurrences of path quantifiers, and, as a consequence, it becomes possible to express some properties in ATLES that cannot even be expressed in ATL*. We showed that ATLES satisfiability is EXPTIME-complete by extending the type elimination construction for ATL. We presented a sound and complete axiomatisation for ATLES, the completeness proof of which is based on that type elimination construction. Moreover, we identified two variants of the model-checking problem for ATLES depending on whether strategies are given as part of the input or strategies have to be generated. We settled the computational complexities of the model checking problems at PTIME-complete and NP-complete, respectively. For future work, we presented various suggestions of how to extend ATLES and thus enrich its expressivity. By introducing appropriate operators for composing strategies, we can model, e.g., *non-deterministic strategies* or *dynamic strategies*. Another idea is to introduce *strategy variables* that can be quantified.

In Chapter 5, we introduced *Dynamic Coalition Logic of Propositional Control* (DCL-PC), as an extension of CL-PC [**249**] with dynamic logic operators. In DCL-PC, every propositional variable is allocated to a unique agent in the system. For an agent having *control* over her propositional variable means the unique ability to determine the truth value for this variable. The dynamic component in DCL-PC allows agents to *delegate control* of variables to other agents. By using DCL-PC, we can reason about how the abilities of agents are affected by the transfer of control in the system. We presented a complete axiomatisation of DCL-PC and showed that the satisfiability problem and the model checking problem for DCL-PC are both PSPACE-complete and, thus, no more complex than the corresponding problems for CL-PC. A possible further development of DCL-PC is to additionally account for reasoning about agent's obligations. The framework of *Social Laws* [**225, 187, 244**] could be set to work in order to express that under certain conditions, an agent will not set a certain propositional variable to 'true', or that she will not pass on control over a certain variable to a specific agent, or that the overall system behaves in such a way that every agent gets a fair chance to trigger a specific variable (i.e., use a specific resource) infinitely often.

Bibliography

[1] AGC 2007: The international workshop on Agent-based Grid Computing. `http://recerca.ac.upc.edu/conferencies/AGC2007/`.

[2] AiML: Advances in Modal Logic. `http://www.aiml.net`.

[3] M4M: Methods for Modalities. `http://m4m.loria.fr`.

[4] M. Adler and N. Immerman. An n! lower bound on formula size. *ACM Transactions on Computational Logic*, 4(3):296–314, 2003.

[5] T. Ågotnes. *A Logic of Finite Syntactic Epistemic States*. PhD thesis, Department of Informatics, University of Bergen, 2004.

[6] T. Ågotnes. A note on syntactic characterization of incomplete information in ATEL. In S. van Otterloo, P. McBurney, W. van der Hoek, and M. Wooldridge, editors, *Proceedings of KAG'04: the 1st Workshop on Knowledge and Games*, Liverpool, UK, July 2004.

[7] T. Ågotnes, V. Goranko, and W. Jamroga. Alternating-time temporal logic with irrevocable strategies. To appear in proceedings of TARK'07: the 11th Conference on Theoretical Aspects of Rationality and Knowledge, Brussels, Belgium, June 25–27, 2007.

[8] M. Aiello, J. van Benthem, and G. Bezhanishvili. Reasoning about space: the modal way. *Journal of Logic and Computation*, 13(6):889–920, 2003.

[9] M. Aiello, J. van Benthem, and I. Pratt-Hartmann, editors. *The Logic of Space*. 2007. To appear.

[10] C. Alchourrón, P. Gärdenfors, and D. Makinson. On the logic of change. *Journal of Symbolic Logic*, 50:510–530, 1985.

[11] N. Alechina and B. Logan. Ascribing beliefs to resource bounded agents. In *Proceedings of AAMAS'02: the 1st international joint conference on Autonomous agents and multiagent systems, Bologna, Italy*, pages 881–888, New York, NY, USA, 2002. ACM Press.

[12] N. Alechina, B. Logan, and M. Whitsey. A complete and decidable logic for resource-bounded agents. In *Proceedings of AAMAS'04: the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems, New York*, pages 606–613, Washington, DC, USA, 2004. IEEE Computer Society.

[13] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. In *Proceedings of FOCS'97: the 38th Annual IEEE Symposium on Foundations of Computer Science*, pages 100–109, Washington, DC, USA, October 1997. IEEE Computer Society.

[14] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Lecture Notes in Computer Science*, 1536:23–60, 1998.

[15] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49(5):672–713, 2002.

[16] R. Alur, T. A. Henzinger, O. Kupferman, and M. Y. Vardi. Alternating Refinement Relations. In *Proceedings of CONCUR'98: the 9th International Conference on Concurrency Theory*, pages 163–178, London, UK, 1998. Springer-Verlag.

[17] R. Alur, T. A. Henzinger, F. Y. C. Mang, S. Qadeer, S. K. Rajamani, and S. Taşiran. Mocha: Modularity in model checking. In *Proceedings of CAV'98: the 10th International Conference on Computer-aided Verification, (LNCS Volume 1427)*, pages 521–525, London, UK, 1998. Springer-Verlag.

[18] A. Ankolenkar, M. Burstein, J. R. Hobbs, O. Lassila, D. L. Martin, D. McDermott, S. A. McIlraith, S. Narayanan, M. Paolucci, T. R. Payne, and K. Sycara. DAML-S: Web service description for the semantic web. In *Proceedings of ISWC'02: the 1st International Semantic Web Conference, Sardinia, Italy, June, 2002*, LNCS. Springer Verlag, 2003.

[19] S. Artemov and L. Beklemishev. Provability logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, chapter 25, pages 189–360. Springer Verlag, Dortrecht, 2nd edition, 2004.

[20] R. Aumann and A. Brandenburger. Epistemic conditions for Nash equilibrium. *Econometrica*, 63(5):1161–1180, 1995.

[21] R. Aumann and S. Hart. *Handbook of Game Theory with Economic Applications*, volume 1. Elsevier, 1992.

[22] S. Azhar, G. Peterson, and J. Reif. On multiplayer non-cooperative games of incomplete information: Part 1 – decision algorithms. Technical Report TR CS–1991–37, Duke University, Durham, NC, USA, 1991.

[23] S. Azhar, G. Peterson, and J. Reif. On multiplayer non-cooperative games of incomplete information: Part 2 – lower bounds. Technical report, Duke University, Durham, NC, USA, 1991.

[24] F. Baader, D. Calvanese, D. L. McGuiness, D. Nardi, and P. Patel-Schneider. *The Description Logic Handbook: Theory, implementation and applications.* Cambridge University Press, 2003.

[25] F. Baader, S. Ghilardi, and C. Tinelli. A new combination procedure for the word problem that generalizes fusion decidability results in modal logics. In D. Basin and M. Rusinowitch, editors, *Proceedings of IJCAR'04: the 2nd International Joint Conference on Automated Reasoning*, Lecture Notes in Artificial Intelligence. Springer-Verlag, 2004.

[26] F. Baader, S. Ghilardi, and C. Tinelli. A new combination procedure for the word problem that generalizes fusion decidability results in modal logics. *Information and Computation*, 204(10):1413–1452, 2006.

[27] F. Baader, C. Lutz, H. Sturm, and F. Wolter. Fusions of description logics and abstract description systems. *Journal of Artificial Intelligence Research (JAIR)*, 16:1–58, 2002.

[28] A. Baltag. STS: A structural theory of sets. *Logic Journal of the IGPL*, 7(4):481–515, 1999.

[29] J. Barwise and L. S. Moss. *Vicious Circles.* CSLI publications, Stanford, 1996.

[30] N. Belnap and M. Perloff. Seeing to it that: A canonical form for agentives. In H. E. Kyburg, R. P. Loui, and G. N. Carlson, editors, *Knowledge Representation and Defeasible Reasoning*, pages 167–190. Kluwer, Boston, 1990.

[31] N. D. Belnap, M. Perloff, and M. Xu. *Facing the Future: Agents and Choices in Our Indeterminist World.* Oxford University Press, USA, January 2001.

[32] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic Web. *Scientific American*, 284(5), 2001.

[33] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic.* Number 53 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2001.

[34] P. Blackburn, J. van Benthem, and F. Wolter, editors. *Handbook of Modal Logic, Volume 3.* Studies in Logic and Practical Reasoning. Elsevier Science, December 2006.

[35] W. J. Blok. On the degree of incompleteness of modal logics (abstract). *Bulletin of the Section of Logic of the Polish Academy of Sciences*, 7:167–175, 1978.

[36] W. J. Blok. On the degree of incompleteness of modal logics and the covering relation in the lattice of modal logics. Technical Report 78-07, Department of Mathematics, University of Amsterdam, Amsterdam, The Netherlands, 1978.

[37] W. J. Blok. An axiomatization of the modal theory of the veiled recession frame. *Studia Logica*, 38:37–47, 1979.

[38] S. Borgo. Coalitions in action logic. In M. Veloso, editor, *Proceedings of IJCAI'07: the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India*, pages 1822–1827, 2007.

[39] C. Boutilier. Toward a logic for qualitative decision theory. In J. Doyle, E. Sandewall, and P. Torasso, editors, *Proceedings of KR'94: the 4th International Conference on Principles of Knowledge Representation and Reasoning*, pages 75–Ű86, San Mateo, CA, USA, 1994. Morgan Kaufmann publishers Inc.

[40] M. E. Bratman. *Intention, Plans, and Practical Reason.* Harvard University Press, Cambridge, Mass., USA, 1987.

[41] J. Broersen, A. Herzig, and N. Troquard. Embedding alternating-time temporal logic in strategic STIT logic of agency. *Journal of Logic and Computation*, 16(5):559–578, 2006.

[42] J. Broersen, A. Herzig, and N. Troquard. From coalition logic to STIT. *Electronic Notes in Theoretical Computer Science*, 157(4):23–35, 2006.

[43] J. Broersen, A. Herzig, and N. Troquard. A STIT-extension of ATL. In M. Fisher, W. van der Hoek, B. Konev, and A. Lisitsa, editors, *Proceedings of JELIA'06: 10th European Conference on Logics in Artificial Intelligence, Liverpool, UK*, volume 4160 of *Lecture Notes in Computer Science*, pages 69–81. Springer, 2006.

[44] M. C. Browne, E. M. Clarke, and O. Grümberg. Characterizing finite Kripke structures in propositional temporal logic. *Theoretical Computer Science*, 59(1–2):115–131, 1988.

[45] J. R. Büchi and L. H. Landweber. Solving sequential conditions by finite-state strategies. *Transactions of the American Mathematical Society*, 138:295–311, 1969.

[46] R. M. Burstall. Program proving as hand simulation with a little induction. In J. L. Rosenfeld, editor, *Proceedings of IFIP Congress, Stockholm, Sweden*, pages 308–312, Amsterdam, 1974.

[47] B. ten Cate. The expressivity of XPath with transitive closure. In S. Vansummeren, editor, *Proceedings of PODS'06: the 25th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 26-28, 2006, Chicago, Illinois, Maryland, USA*, pages 328–337. ACM, 2006.

[48] B. ten Cate and M. Marx. Axiomatizing the logical core of XPath 2.0. In T. Schwentick and D. Suciu, editors, *Proceedings of ICDT'07: the 11th International Conference of Database Theory, Barcelona, Spain, January 10-12, 2007*, volume 4353 of *Lecture Notes in Computer Science*, pages 134–148. Springer, 2007.

[49] A. K. Chandra, D. Kozen, and L. J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, 1981.

[50] A. K. Chandra and P. M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *Proceedings of STOC'77: the 9th Annual ACM Symposium on Theory of Computing*, pages 77–90, New York, NY, USA, 1977. ACM Press.

[51] B. F. Chellas. *Modal Logic: An Introduction.* Cambridge University Press, Cambridge, UK, 1980.

[52] L. Cholvy and C. Garion. An attempt to adapt a logic of conditional preferences for reasoning with contrary-to-duties. *Fundamenta Informatica*, 48(2–3):183–204, 2001.

[53] H. H. Clark and C. R. Marshall. Definite reference and mutual knowledge. In A. K. Joshe, B. L. Webber, and I. A. Sag, editors, *Elements of Discourse Understanding*, pages 10–63. Cambridge University Press, Cambridge, UK, 1981.

[54] E. Clarke, O. Grumberg, and D. Peled. *Model Checking.* MIT Press, 1999.

[55] E. Clarke and B.-H. Schlingloff. Model checking. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume II, chapter 24, pages 1635–1790. Elsevier Science Publishers B.V., 2001.

[56] E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In D. Kozen, editor, *Proceedings of Workshop on Logic of Programs,*

Yorktown Heights, New York, May, 1981, volume 131 of *Lecture Notes in Computer Science*, pages 52–71, London, UK, 1982. Springer-Verlag.

[57] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.

[58] R. Cleaveland and B. Steffen. A linear-time model-checking algorithm for the alternation-free modal mu-calculus. *Formal Methods in System Design*, 2(2):121–147, 1993.

[59] P. R. Cohen and H. J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42(2-3):213–261, 1990.

[60] S. A. Cook and D. G. Mitchell. Finding hard instances of the satisfiability problem: A survey. In D. Du, J. Gu, and P. M. Pardalos, editors, *Satisfiability Problem: Theory and Applications*, volume 35 of *DIMACS: Series in Discrete Mathematics and Theoretical Computer Science*, pages 1–17, 1997.

[61] M. Cresswell. A Henkin completeness theorem for T. *Notre Dame Journal of Formal Logic*, 8:186–196, 1967.

[62] M. Dam. CTL* and ECTL* as fragments of the modal μ-calculus. *Theoretical Computer Science*, 126(1):77–96, April 1994.

[63] L. de Alfaro, T. A. Henzinger, and R. Majumdar. From verification to control: Dynamic programs for omega-regular objectives. In *Proceedings of LICS'01: the 16th Annual IEEE Symposium on Logic in Computer Science, June 16-19, 2001, Boston, MA, USA*, pages 279–290, 2001.

[64] K. S. Decker. Distributed problem solving techniques: A survey. *IEEE Transactions on Systems, Man, and Cybernetics*, 17(5):729–740, 1987.

[65] K. S. Decker. *Task environment centered simulation*, pages 105–128. AAAI Press/MIT Press, Cambridge, MA, USA, 1998.

[66] V. Dignum, J.-J. C. Meyer, and H. Weigand. Towards an organizational model for agent societies using contracts. In *Proceedings of AAMAS'02: the 1st International Joint Conference on Autonomous Agents and Multiagent Systems, Bologna, Italy*, pages 694–695, New York, NY, USA, 2002. ACM Press.

[67] H. N. Duc. *Resource-Bounded Reasoning about Knowledge*. PhD thesis, Leipzig University, Germany, 2001.

[68] E. Emerson, A. Mok, A. Sistla, and J. Srinivasan. Quantitative temporal reasoning. *Real-Time Systems*, 4:331–352, 1992.

[69] E. A. Emerson. Temporal and modal logic. In *Handbook of Theoretical Computer Science (vol. B): Formal Models and Semantics*, pages 995–1072. MIT Press, Cambridge, MA, USA, 1990.

[70] E. A. Emerson. Model checking and the mu-calculus. In N. Immerman and P. G. Kolaitis, editors, *Proceedings of a DIMACS Workshop: Descriptive Complexity and Finite Models, January 14-17, 1996, Princeton University, NY, USA*, volume 31 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 185–214. American Mathematical Society, 1996.

[71] E. A. Emerson and E. M. Clarke. Characterizing correctness properties of parallel programs using fixpoints. In *Proceedings of the 7th Colloquium on Automata, Languages and Programming*, volume 85 of *LNCS*, pages 169–181, London, UK, 1980. Springer-Verlag.

[72] E. A. Emerson and J. Y. Halpern. Decision procedures and expressiveness in the temporal logic of branching time. In *Proceedings of STOC'82: the 14th annual ACM symposium on Theory of Computing, San Francisco, CA, USA*, pages 169–180, New York, NY, USA, 1982. ACM Press.

[73] E. A. Emerson and J. Y. Halpern. "Sometimes" and "not never" revisited: On branching versus linear time temporal logic. *Journal of the ACM*, 33(1):151–178, January 1986.

[74] E. A. Emerson and C. S. Jutla. The complexity of tree automata and logics of programs (extended abstract). In *Proceedings of FOCS'88: the 29th Annual Symposium on Foundations of Computer

Science, 24-26 October 1988, White Plains, New York, USA, pages 328–337. IEEE Computer Society Press, 1988.

[75] E. A. Emerson and C. S. Jutla. Tree automata, μ-calculus and determinacy. In *Proceedings of FOCS'91: the 32nd Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, October 1-4, 1991*, pages 368–377, Los Alamitos, CA, USA, 1991. IEEE Computer Society Press.

[76] E. A. Emerson and C. S. Jutla. The complexity of tree automata and logics of programs. *SIAM Journal on Computing*, 29(1):132–158, 2000.

[77] E. A. Emerson and C.-L. Lei. Modalities for model checking: branching time logic strikes back. *Science of Computer Programming*, 8(3):275–306, 1987.

[78] L. Esakia. Diagonal constructions, Löb's formula, and Cantor's scattered spaces. In *Logical and Semantical Investigations*, pages 128–Ũ143, Tbilisi, Metsniereba, 1981. Academy Press. In Russian.

[79] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning About Knowledge*. The MIT Press, Cambridge, MA, USA, 1995.

[80] R. Fagin, J. Y. Halpern, and M. Y. Vardi. What can machines know?: On the properties of knowledge in distributed systems. *Journal of the ACM*, 39(2):328–376, 1992.

[81] R. Fagin and M. Y. Vardi. Knowledge and implicit knowledge in a distributed environment: preliminary report. In J. Y. Halpern, editor, *Proceedings of TARK'86: the 1st Conference on Theoretical Aspects of Reasoning about Knowledge, Monterey, CA, USA*, pages 187–206, San Francisco, CA, USA, 1986. Morgan Kaufmann Publishers Inc.

[82] K. Fine. An incomplete logic containing S4. *Theoria*, 40:23–29, 1974.

[83] K. Fine. Normal forms in modal logic. *Notre Dame Journal of Formal Logic*, 16:229–234, 1975.

[84] M. Fitting. *Proof Methods for Modal and Intuitionistic Logic*. Reidel, 1983.

[85] I. O. for Standardization. *Information technology – Syntactic metalanguage – Extended BNF*. ISO, Geneva, Switzerland, 1996. ISO/IEC 14977:1996(E).

[86] J. Forrester. *Being Good and Being Logical: Philosophical Groundwork for a New Deontic Logic*. M.E. Sharpe, New York, USA and London, UK, 1996.

[87] T. French. *Bisimulation quantifiers for modal logics*. PhD thesis, University of Western Australia, 2006.

[88] D. Gabbay, A. Kurucz, F. Wolter, and M. Zakharyaschev. *Many-Dimensional Modal Logics: Theory and Applications*, volume 148 of *Studies in Logic*. Elsevier, 2003.

[89] D. Gabbay, A. Pnueli, S. Shelah, and J. Stavi. On the temporal analysis of fairness. In *Proceedings of POPL'80: the 7th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Las Vegas, Nevada, USA*, pages 163–173, New York, NY, USA, 1980. ACM Press.

[90] D. M. Gabbay. An irreflexivity lemma with applications to axiomatizations of conditions on tense frames. In U. Mönnich, editor, *Aspects of Philosophical Logic: Some Logical Forays into Central Notions of Linguistics and Philosophy*, pages 67–89. Reidel, Dordrecht, Germany, 1981.

[91] P. Gärdenfors. *Knowledge in flux: modeling the dynamics of epistemic states*. MIT Press, Cambridge, MA, USA, 1988.

[92] F. Giunchiglia and P. Traverso. Planning as model checking. In S. Biundo and M. Fox, editors, *Proceedings of ECP'99: the 5th European conference on planning, Durham, 8-10 September, 1999*, volume 1809 of *Recent Advances in AI Planning (LNAI)*, pages 1–20, Berlin, Germany, 1999. Springer-Verlag.

[93] K. Gödel. Eine Interpretation des intuitionistischen Aussagenkalküls. *Ergebnisse eines mathematischen Kolloquiums*, 4:39–40, 1933.

[94] A. T. Goldberg. Knowledge-based programming: a survey of program design and construction techniques. *IEEE Transactions on Software Engineering*, 12(7):752–768, 1986.

[95] R. Goldblatt. Mathematical modal logic: a view of its evolution. *Journal of Applied Logic*, 1(5-6):309–392, 2003.

[96] V. Goranko and W. Jamroga. Comparing semantics of logics for multi-agent systems. *Synthese*, 139(2):241–280, 2004.

[97] V. Goranko, W. Jamroga, and G. van Drimmelen. Axiomatic systems for Alternating-time Temporal Epistemic Logics (extended abstract). In *Proceedings of LOFT'04: the 6th Conference on Logic and the Foundations of Game and Decision Theory, July 16–18, 2004, Leipzig Graduate School of Management, Germany.* IEEE Computer Society, 2004.

[98] V. Goranko and G. van Drimmelen. Complete axiomatization and decidability of Alternating-time Temporal Logic. *Theoretical Computer Science*, 353(1-3):93–117, 2006.

[99] E. Grädel. Why are modal logics so robustly decidable? In G. Paun, G. Rozenberg, and A. Salomaa, editors, *Current Trends in Theoretical Computer Science. Entering the 21st Century*, pages 393–408. World Scientific, River Edge, NJ, USA, 2001.

[100] J. Y. Halpern and R. Fagin. Modelling knowledge and action in distributed systems. In *Proceedings of Concurrency'88: the International Conference on Concurrency*, pages 18–32, London, UK, 1988. Springer-Verlag.

[101] J. Y. Halpern and Y. Moses. A guide to the modal logics of knowledge and belief: Preliminary draft. In A. K. Joshi, editor, *Proceedings of IJCAI'85: the 9th International Joint Conference on Artificial Intelligence, Los Angeles, CA, August, 1985*, pages 480–490, San Francisco, CA, USA, 1985. Morgan Kaufmann Publishers Inc.

[102] J. Y. Halpern and Y. Moses. Knowledge and common knowledge in a distributed environment. *Journal of the ACM*, 37(3):549–587, 1990.

[103] J. Y. Halpern and Y. Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54(3):319–380, 1992.

[104] J. Y. Halpern and R. Pucella. Modeling adversaries in a logic for security protocol analysis. In A. E. Abdallah, P. Ryan, and S. Schneider, editors, *Proceedings of FASec'02: the 1st International Conference on Formal Aspects of Security, London, UK, December 2002*, volume 2629 of *Lecture Notes in Computer Science*, pages 115–132. Springer, 2003.

[105] A. Harding, M. Ryan, and P.-Y. Schobbens. Approximating ATL* in ATL. In A. Cortesi, editor, *Proceedings of VMCAI'02: the 3rd International Workshop on Verification, Model Checking, and Abstract Interpretation, Venice, Italy, January 21-22, 2002, Revised Papers*, volume 2294 of *Lecture Notes in Computer Science*, pages 289–301, London, UK, 2002. Springer-Verlag.

[106] D. Harel. Dynamic logic. In D. M. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic, Volume II*, pages 496–604. D. Reidel Publishers, Dordrecht, 1984.

[107] D. Harel, J. Tiuryn, and D. Kozen. *Dynamic Logic*. MIT Press, Cambridge, MA, USA, 2000.

[108] P. Harrenstein, W. van der Hoek, J.-J. C. Meyer, and C. Witteveen. A modal characterization of nash equilibrium. *Fundamenta Informaticae*, 57(2–4):281–321, 2003.

[109] M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, 32(1):137–161, 1985.

[110] A. Herzig and N. Troquard. Knowing how to play: Uniform choices in logics of agency. In G. Weiss and P. Stone, editors, *Proceedings of AAMAS'06: the 5th International Joint Conference on Autonomous Agents and Multi Agent Systems, Hakodate, Japan, May 8-12, 2006*, pages 209–216. ACM Press, May 2006.

[111] J. Hintikka. *Knowledge and Belief, an introduction to the logic of the two notions.* Cornell University Press, Ithaca (NY) and London, 1962.

[112] W. Hodges. *Model Theory.* Cambridge University Press, Cambridge, 1993.

[113] B. A. Huberman and S. H. Clearwater. A multi-agent system for controlling building environments. In V. R. Lesser and L. Gasser, editors, *Proceedings of ICMAS'95: the 1st International*

Conference on Multiagent Systems, June 12-14, 1995, San Francisco, CA, USA, pages 171–176. The MIT Press, 1995.

[114] M. Jago. *Logics for Resource-Bounded Agents.* PhD thesis, University of Nottingham, UK, 2006.

[115] W. Jamroga. Some remarks on alternating temporal epistemic logic. In B.Dunin-Keplicz and R. Verbrugge, editors, *Proceedings of FAMAS'03: International Workshop on Formal Approaches to Multi-Agent Systems, Warsaw, Poland, April 12, 2003*, special issue of Fundamenta Informaticae, pages 133–140. IOS Press, 2004.

[116] W. Jamroga. Strategic planning through model checking of ATL formulae. In L. Rutkowski, J. H. Siekmann, R. Tadeusiewicz, and L. A. Zadeh, editors, *Proceedings of ICAISC'04: the 7th International Conference, Zakopane, Poland, June 7-11, 2004*, volume 3070 of *Artificial Intelligence and Soft Computing, Lecture Notes in Artificial Intelligence*, pages 879–884. Springer-Verlag, 2004.

[117] W. Jamroga. On the relationship between playing rationally and knowing how to play: A logical account. Technical Report IfI-06-01, Institut für Informatik, Technische Universität Clausthal, Germany, January 2006.

[118] W. Jamroga. On the relationship between playing rationally and knowing how to play: A logical account. In *Proceedings of KI'06: the 29th German Conference on Artificial Intelligence, June 14-19, 2006, Bremen, Germany*, Lecture Notes in Artificial Intelligence. Springer-Verlag, 2006. To appear.

[119] W. Jamroga and T. Ågotnes. Constructive knowledge: What agents can achieve under incomplete information. Technical Report IfI-05-10, Institut für Informatik, Technische Universität Clausthal, Germany, October 2005.

[120] W. Jamroga and T. Ågotnes. What agents can achieve under incomplete information. In G. Weiss and P. Stone, editors, *Proceedings of AAMAS'06: the 5th International Joint Conference on Autonomous Agents and Multi-Agent Systems: Hakodate, Japan, May 8-12, 2006*, pages 232–234. ACM Press, May 2006.

[121] W. Jamroga and J. Dix. Do agents make model checking explode (computationally)? In M. Pechoucek, P. Petta, and L. Z. Varga, editors, *Proceedings of CEEMAS'05: the 4th International Central and Eastern European Conference on Multi-Agent Systems, Budapest, Hungary, September 15-17, 2005*, volume 3690 of *Lecture Notes in Computer Science*, pages 398–407. Springer-Verlag, 2005.

[122] W. Jamroga and J. Dix. Model checking strategic abilities of agents under incomplete information. In M. Coppo, E. Lodi, and G. M. Pinna, editors, *Proceedings of ICTCS'05: the 9th Italian Conference on Theoretical Computer Science*, volume 3701 of *Lecture Notes in Computer Science*, pages 295–308. Springer-Verlag, 2005.

[123] W. Jamroga and J. Dix. Model checking abilities of agents: A closer look. Technical Report IfI-06-02, Institut für Informatik, Technische Universität Clausthal, Germany, February 2006.

[124] W. Jamroga and J. Dix. Model checking abilities of agents: A closer look. *Theory of Computing Systems*, 2007. To appear.

[125] W. Jamroga and W. van der Hoek. Agents that know how to play. *Fundamenta Informaticae*, 63(2–3):185–219, 2004.

[126] W. Jamroga, W. van der Hoek, and M. Wooldridge. Intentions and strategies in game-like scenarios. In C. Bento, A. Cardoso, and G. Dias, editors, *Proceedings of EPIA'05: 3rd Workshop on Multi-Agent Systems: Theory and Applications (MASTA'05) - a workshop of EPIA 2005*, volume 3808 of *Lecture Notes in Computer Science*, pages 512–523. Springer-Verlag, 2005.

[127] W. Jamroga, W. van der Hoek, and M. Wooldridge. Intentions and strategies in game-like scenarios. Technical Report IfI-05-08, Institut für Informatik, Technische Universität Clausthal, Germany, August 2005. ISSN: 1860-8477.

[128] D. Janin and I. Walukiewicz. On the expressive completeness of the propositional mu-calculus with respect to monadic second order logic. In U. Montanari and V. Sassone, editors, *Proceedings of CONCUR'96: the 7th International Conference on Concurrency Theory, Pisa, Italy*, volume 1119, pages 263–277. Springer-Verlag, 1996.

[129] N. R. Jennings, J. Corera, and I. Laresgoiti. Developing industrial multi-agent systems (invited paper). In *Proceedings of ICMAS'95: 1st International Conference on Multi-Agent Systems*, pages 423–430, San Francisco, USA, 1995.

[130] J. Johannsen and M. Lange. CTL$^+$ is complete for double exponential time. In J. Baeten, J. Lenstra, J. Parrow, and G. Woeginger, editors, *Proceedings of ICALP'03: the 30th International Colloquium on Automata, Languages and Programming*, volume 2719 of *LNCS*, pages 767 – 775, Eindhoven, NL, 2003. Springer.

[131] G. Jonker. Feasible strategies in alternating-time temporal epistemic logic, 2003. Master's thesis, University of Utrech, The Netherlands.

[132] B. Jónsson and A. Tarski. Boolean algebras with operators, part I. *American Journal of Mathematics*, 73:891–Ű939, 1952.

[133] B. Jónsson and A. Tarski. Boolean algebras with operators, part II. *American Journal of Mathematics*, 74:127–162, 1952.

[134] H. Kamp. *Tense Logic and the Theory of Linear Order.* PhD thesis, University of California, Los Angeles, 1968.

[135] S. Kanger. The morning star paradox. *Theoria*, 23:1–11, 1957.

[136] S. Kanger. *Provability in Logic.* Almqvist and Wiksell, Stockholm, 1957.

[137] H. Katsuno and A. Mendelzon. On the difference between updating a knowledge base and revising it. In J. F. Allen, R. Fikes, and E. Sandewall, editors, *KR'91: Principles of Knowledge Representation and Reasoning*, pages 387–394. Morgan Kaufmann Publishers Inc., San Mateo, CA, USA, 1991.

[138] D. E. Knuth. Backus normal form vs. Backus Naur form. *Communications of the ACM*, 7(12):735–736, 1964.

[139] K. Konolige. *A Deduction Model of Belief.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1986.

[140] D. Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27(3):333–354, 1983.

[141] D. Kozen and R. Parikh. A decision procedure for the propositional μ-calculus. In E. M. Clarke and D. Kozen, editors, *Proceedings of the Workshop Logics of Programs, Carnegie Mellon University, Pittsburgh, PA, USA, June 6-8, 1983*, volume 164 of *Lecture Notes in Computer Science*, pages 313–325. Springer, 1983.

[142] S. Kripke. A completeness theorem in modal logic. *Journal of Symbolic Logic*, 24:1–14, 1959.

[143] S. Kripke. Semantical analysis of modal logic I: Normal modal propositional calculi. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 9:67–96, 1963.

[144] O. Kupferman and O. Grumberg. Buy one, get one free!!! *Journal of Logic and Computation*, 6(4):523–539, 1996.

[145] O. Kupferman, M. Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. *Journal of the ACM*, 47(2):312–360, 2000.

[146] K. Kuratowski. Sur l'operation a de l'analysis situs. *Fundamenta Mathematicae*, 3:181–199, 1922.

[147] R. E. Ladner. The computational complexity of provability in systems of modal propositional logic. *SIAM Journal on Computing*, 6(3):467–480, 1977.

[148] L. Lamport. "Sometime" is sometimes "not never": on the temporal logic of programs. In *Proceedings of POPL'80: the 7th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 174–185, New York, NY, USA, 1980. ACM Press.

[149] J. Lang, L. V. D. Torre, and E. Weydert. Utilitarian desires. *Autonomous Agents and Multi-Agent Systems*, 5(3):329–363, 2002.

[150] F. Laroussinie. About the expressive power of CTL combinators. *Information Processing Letters*, 54(6):343–345, 1995.

[151] F. Laroussinie, N. Markey, and P. Schnoebelen. Model checking CTL$^+$ and FCTL is hard. In *Proceedings of FoSSaCS'01: the 4th International Conference on Foundations of Software Science and Computation Structures*, pages 318–331, London, UK, 2001. Springer-Verlag.

[152] D. Lehmann. Knowledge, common knowledge and related puzzles (extended summary). In *Proceedings of PODC'84: the 3rd Annual ACM Symposium on Principles of Distributed Computing, Vancouver, BC, Canada, August 27-29, 1984*, pages 62–67, New York, NY, USA, 1984. ACM Press.

[153] G. W. Leibniz. Essais de Théodicée sur la bonté de Dieu, la liberté de l'homme et l'origine du mal (Theodicy Essay on the Benevolence of God, the Free will of man, and the Origin of Evil), 1710. In French.

[154] E. Lemmon. Algebraic semantics for modal logics, parts I and II. *Journal of Symbolic Logic*, pages 46–65 and 191Ū–218, 1966.

[155] E. Lemmon and D. Scott. *The "Lemmon Notes": An Introduction to Modal Logic*. Blackwell, Oxford, UK, 1977.

[156] Y. Lespérance. A formal account of self-knowledge and action. In N. S. Sridharan, editor, *Proceedings of IJCAI'89: the 11th International Joint Conference on Artificial Intelligence, Detroit, MI, USA, August 1989*, pages 868–874, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.

[157] Y. Lesperance. *A formal theory of indexical knowledge and action*. PhD thesis, University of Toronto, Toronto, Ontario, Canada, 1991.

[158] H. J. Levesque. Foundations of a functional approach to knowledge representation. *Artificial Intelligence*, 23(2):155–212, 1984.

[159] C. I. Lewis. *A Survey of Symbolic Logic*. University of California Press, Berkley, 1918.

[160] C. I. Lewis and C. H. Langford. *Symbolic Logic*. Dover, NY, USA, 1932.

[161] D. Lewis. Attitudes de dicto and de se. *The Philosophical Review*, 88(4):513–543, 1979.

[162] D. K. Lewis. *Convention: A Philosophical Study*. Harvard University Press, 1969.

[163] N. Li, B. N. Grosof, and J. Feigenbaum. Delegation logic: A logic-based approach to distributed authorization. *ACM Transactions on Information and System Security*, 6(1):128–171, 2003.

[164] M. Ljungberg and A. Lucas. The OASIS air-traffic management system. In *Proceedings of PRICAI'92: the 2nd Pacific Rim International Conference on Artificial Intelligence*, Seoul, Korea, 1992.

[165] A. Lomuscio and F. Raimondi. Model checking knowledge, strategies, and games in multi-agent systems. In *Proceedings of AAMAS'06: the 5th International Joint Conference on Autonomous Agents and Multiagent Systems, Hakodate, Japan*, pages 161–168, New York, NY, USA, 2006. ACM Press.

[166] A. Lomuscio and M. J. Sergot. A formalisation of violation, error recovery, and enforcement in the bit transmission problem. *Journal of Applied Logic*, 2(1):93–116, 2004.

[167] C. Lutz, D. Walther, and F. Wolter. Quantitative temporal logics: PSPACE and below. In *Proceedings of TIME'05: the 12th International Symposium on Temporal Representation and Reasoning*, pages 138–146, Washington, DC, USA, 2005. IEEE Computer Society.

[168] C. Lutz, D. Walther, and F. Wolter. Quantitative temporal logics over the reals: PSPACE and below. *TIME'05 special issue of Information and Computation*, 205(1):99–123, 2007.

[169] D. Makinson. On some completeness theorems in modal logic. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 12:379–384, 1966.

[170] M. J. Mataric. Issues and approaches in the design of collective autonomous agents. *Robotics and Autonomous Systems*, 16(2–4):321–331, 1995.

[171] J. McCarthy, M. L. Minsky, N. Rochester, and C. Shannon. A proposal for the Dartmouth reseach project on Artificial Intelligence. `http://www-formal.stanford.edu/jmc/history/dartmouth/dartmouth.html`, August 1955.

[172] J. C. C. McKinsey and A. Tarski. The algebra of topology. *Annals of Mathematics*, 45:141–191, 1944.

[173] K. McMillan. *Symbolic model checking - an approach to the state explosion problem.* PhD thesis, SCS, Carnegie Mellon University, 1992.

[174] J.-J. C. Meyer. A different approach to deontic logic: Deontic logic viewed as a variant of dynamic logic. *Notre Dame Journal of Formal Logic*, 29(1):109–136, 1987.

[175] J.-J. C. Meyer and W. van der Hoek. *Epistemic Logic for Computer Science and Artificial Intelligence.* Number 41 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge, UK, 1995.

[176] P. Milgrom. An axiomatic characterization of common knowledge. *Econometrica*, 49(1):219–222, 1981.

[177] R. Milner. *A Calculus of Communicating Systems.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1982.

[178] R. Montague. Pragmatics. In R. Klibansky, editor, *Contemporary Philosophy: a Survey*, pages 102–122. La Nuova Italia Editrice, Florence, 1968.

[179] R. Montague. Universal grammar. *Theoria*, 36:373-398, 1970.

[180] G. E. Moore. Cramming more components onto integrated circuits. *Electronics*, 38:114–117, 1965.

[181] G. E. Moore. Cramming more components onto integrated circuits. In *Readings in computer architecture*, pages 56–59. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2000.

[182] R. C. Moore. A formal theory of knowledge and action. In J. R. Hobbs and R. C. Moore, editors, *Formal Theories of the Commonsense World.* Ablex Publishing, Norwood, NJ, USA, 1985.

[183] L. Morgenstern. A first order theory of planning, knowledge, and action. In *Proceedings of the 1986 Conference on Theoretical aspects of reasoning about knowledge*, pages 99–114, San Francisco, CA, USA, 1986. Morgan Kaufmann Publishers Inc.

[184] L. Morgenstern. Knowledge preconditions for actions and plans. In *Distributed Artificial Intelligence*, pages 192–199. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[185] L. Morgenstern. Knowledge and the frame problem. *International Journal of Expert Systems*, 3(4):309–343, 1990.

[186] Y. Moses, D. Dolev, and J. Y. Halpern. Cheating husbands and other stories: A case study of knowledge, action, and communication. *Distributed Computing*, 1(3):167–176, 1986.

[187] Y. Moses and M. Tennenholtz. Artificial social systems. *Computers and AI*, 14(6):533–562, 1995.

[188] D. Niwiński. Fixed point characterization of infinite behavior of finite-state systems. *Theoretical Computer Science*, 189(1–2):1–69, 1997.

[189] T. J. Norman and C. Reed. Group delegation and responsibility. In *Proceedings of AAMAS'02: the 1st International Joint Conference on Autonomous Agents and Multiagent Systems, Bologna, Italy, July 15-19, 2002*, pages 491–498. ACM, 2002.

[190] M. J. Osborne and A. Rubinstein. *A Course in Game Theory.* The MIT Press, Cambridge, MA, USA, 1994.

[191] O. Pacheco and F. Santos. Delegation in a role-based organization. In A. Lomuscio and D. Nute, editors, *Proceedings of DEON'04: 7th International Workshop on Deontic Logic in Computer Science, Madeira, Portugal, May 26-28, 2004.*, volume 3065 of *Lecture Notes in Computer Science*, pages 209–227. Springer, 2004.

[192] C. Papadimitriou. *Computational Complexity.* Addison-Wesley, Englewood, Cliffs, NJ, 1994.

[193] D. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, *Proceedings of the 5th GI-Conference on Theoretical Computer Science, Karlsruhe, Germany*, pages 167–183, London, UK, 1981. Springer-Verlag.

[194] M. Pauly. *Logic for Social Software.* PhD thesis, University of Amsterdam, 2001. ILLC Dissertation Series 2001-10.

[195] M. Pauly. A logical framework for coalitional effectivity in dynamic procedures. *Bulletin of Economic Research*, 53(4):305–324, October 2001.

[196] M. Pauly. A modal logic for coalitional power in games. *Journal of Logic and Computation*, 12(1):149–166, February 2002.

[197] M. Pauly and R. Parikh. Game logic - an overview. *Studia Logica*, 75(2):165–182, 2003.

[198] M. Pauly and M. Wooldridge. Logic for mechanism design — a manifesto. In *Proceedings of GTDT: the 2003 Workshop on Game Theory and Decision Theory in Agent Systems*, Melbourne, Australia, 2003.

[199] G. L. Peterson and J. H. Reif. Multiple-person alternation. In *Proceedings of FOCS'79: the 20th Annual Symposium on Foundations of Computer Science, 29-31 October 1979, San Juan, Puerto Rico*, pages 348–363. IEEE Computer Society Press, 1979.

[200] A. Pnueli. The temporal logic of programs. In *Proceedings of FOCS'77: the 18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October-November 2, 1977*, pages 46–57. IEEE Computer Society Press, 1977.

[201] A. Pnueli and R. Rosner. Distributed reactive systems are hard to synthesize. In *Proceedings of FOCS'90: the 31st Annual Symposium on Foundations of Computer Science, 22-24 October 1990, St. Louis, Missouri, USA*, volume II, pages 746–757. IEEE Computer Society Press, 1990.

[202] V. R. Pratt. Semantical considerations on floyd-hoare logic. In *Proceedings of FOCS'76: the 17th IEEE Symposium on Foundations of Computer Science, Houston, Texas, October 25-27, 1976*, pages 109–121, 1976.

[203] A. N. Prior. *Time and Modality.* Oxford University Press, 1957.

[204] A. N. Prior. *Past, Present and Future.* Oxford University Press, 1967.

[205] A. N. Prior. *Papers on Time and Tense.* Oxford University Press, 1969. New Edition 2003.

[206] W. Quine. Quantifiers and propositional attitudes. *Journal of Philosophy*, 53(5):177–187, March 1956.

[207] M. O. Rabin. Decidability of second-order theories and finite automata on infinite trees. *Transactions of the American Mathematical Society*, 141:1–Ū35, 1969.

[208] M. O. Rabin. *Automata on Infinite Objects and Church's Problem.* American Mathematical Society, Boston, MA, USA, 1972.

[209] A. S. Rao and M. P. Georgeff. Modeling rational agents within a BDI-architecture. In J. Allen, R. Fikes, and E. Sandewall, editors, *Proceedings of KR'91: the 2nd International Conference on Principles of Knowledge Representation and Reasoning*, pages 473–484, San Mateo, CA, USA, 1991. Morgan Kaufmann publishers Inc.

[210] A. S. Rao and M. P. Georgeff. Decision procedures for bdi logics. *Journal of Logic and Computation*, 8(3):293–342, 1998.

[211] J. H. Reif. The complexity of two-player games of incomplete information. *Journal of Computer and Systems Sciences*, 29(2):274–301, 1984.

[212] M. Reynolds. An axiomatization of full computation tree logic. *Journal of Symbolic Logic*, 66(3):1011–1057, September 2001.

[213] R. Rosner. *Modular Synthesis of Reactive Systems.* PhD thesis, Weizmann Institute of Science, Rehovot, Israel, 1992.

[214] A. Rubinstein. *Modeling Bounded Rationality.* The MIT Press, December 1997.

[215] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, Englewood, Cliffs, NJ, 1995.

[216] H. Sahlqvist. Completeness and correspondence in the first and second order semantics for modal logic. In S. Kanger, editor, *Proceedings of the 3rd Scandinavian Logic Symposium, Uppsala, 1973,* pages 110–143, Amsterdam, 1975.

[217] W. J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences,* 4(2):177–192, 1970.

[218] S. Schewe and B. Finkbeiner. Satisfiability and finite model property for the alternating-time μ-calculus. In *Proceedings of CSL'06: the 15th Annual Conference on Computer Science Logic.* Springer Verlag, 2006.

[219] P.-Y. Schobbens. Alternating-time logic with imperfect recall. *Electronic Notes in Theoretical Computer Science,* 85(2):82–93, April 2004.

[220] U. M. Schwuttke and A. G. Quan. Enhancing performance of cooperating agents in real-time diagnostic systems. In *Proceedings of IJCAI'93: 13th International Conference on Artificial Intelligence, Menlo Park, CA, USA,* pages 332–337, 1993.

[221] D. Scott. Advice on modal logic. In K. Lambert, editor, *Philosophical Problems in Logic,* pages 143–173. Reidel, 1970.

[222] D. Scott and J. de Bakker. A theory of programs. 1969.

[223] K. Segerberg. An essay in classical modal logic. *Filosofiska Studier,* 13, 1971.

[224] V. Shehtman. Modal logics of domains of the real plane. *Studia Logica,* 42:63–80, 1983.

[225] Y. Shoham and M. Tennenholtz. On social laws for artificial agent societies: Off-line design. *Artificial Intelligence,* 73(1-2):231–252, 1995.

[226] R. Solovay. Provability interpretations of modal logic. *Israel Journal of Mathematics,* 25:287–304, 1976.

[227] E. Spaan. *Complexity of Modal Logics.* PhD thesis, Department of Mathematics and Computer Science, University of Amsterdam, 1993.

[228] K. Spackman. Managing clinical terminology hierarchies using algorithmic calculation of subsumption: Experience with SNOMED-RT. *Journal of the American Medical Informatics Association (JAMIA),* 2000.

[229] R. Stevens, C. Goble, I. Horrocks, and S. Bechhofer. Building a bioinformatics ontology using OIL. *IEEE Transactions on Information Technology and Biomedicine,* 6, 2002.

[230] C. Stirling. Modal and temporal logics. In S. Abramsky, D. Gabbay, and T. Maibaum, editors, *Handbook of Logic in Computer Science,* pages 477–563. Oxford University Press, 1992.

[231] P. Stone and M. M. Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots,* 8(3):345–383, 2000.

[232] K. Sycara. Multiagent systems. *AI Magazine,* 10(2):79–93, 1998.

[233] A. Tarski. Der Aussagenkalkül und die Topologie. *Fundamenta Mathematika,* 31:103–134, 1938.

[234] S. Thomason. An incompleteness theorem in modal logic. *Theoria,* 40:30–34, 1974.

[235] A. Troelstra and H. Schwichtenberg. *Basic Proof Theory.* Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2000.

[236] J. van Benthem. *Modal Correspondence Theory.* PhD thesis, Mathematisch Instituut & Instituut voor Grondslagenonderzoek, University of Amsterdam, 1976.

[237] J. van Benthem. *Modal Logic and Classical Logic.* Bibliopolis, Napoli, Italy, 1983.

[238] J. van Benthem. Games in dynamic-epistemic logic. *Bulletin of Economic Research,* 53(4):219–248, 2001.

[239] J. van Benthem. Games in dynamic-epistemic logic. In *Proceedings of LOFT'04: the 6th Conference on Logic and the Foundations of Game and Decision Theory, July 16–18, 2004, Leipzig Graduate School of Management, Germany.* IEEE Computer Society, 2004.

[240] J. van Benthem and J. Bergstra. Logics of transition systems. *Journal of Logic, Language, and Information*, pages 247–284, 1995.

[241] J. van Benthem and A. ter Meulen. *Handbook of Logic and Language*. Elsevier, Amsterdam, The Netherlands, 1997.

[242] W. van der Hoek, W. Jamroga, and M. Wooldridge. A logic for strategic reasoning. In *Proceedings of AAMAS'05: the 4th international joint conference on Autonomous agents and multiagent systems*, pages 157–164, New York, NY, USA, 2005. ACM Press.

[243] W. van der Hoek and J.-J. C. Meyer. A complete epistemic logic for multiple agents–combining distributed and common knowledge. In M. Bacharach, L. Gerard-Varet, P. Mongin, and H. Shin, editors, *Epistemic Logic and the Theory of Games and Decisions*, pages 35–68. Kluwer, Dortrecht, 1997.

[244] W. van der Hoek, M. Roberts, and M. Wooldridge. Knowledge and social laws. In F. Dignum, V. Dignum, S. Koenig, S. Kraus, M. Singh, and M. Wooldridge, editors, *Proceedings of AAMAS'05: the 4th International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 674–681, New York, USA, 2005. ACM Inc.

[245] W. van der Hoek and M. Wooldridge. Tractable multiagent planning for epistemic goals. In *Proceedings of AAMAS'02: the 1st International Conference on Autonomous Agents and Multiagent Systems, Bologna, Italy, July 15-19, 2002*, pages 1167–1174, New York, NY, USA, 2002. ACM Press.

[246] W. van der Hoek and M. Wooldridge. Cooperation, knowledge, and time: Alternating-time temporal epistemic logic and its applications. *Studia Logica*, 75(1):125–157, 2003.

[247] W. van der Hoek and M. Wooldridge. Towards a logic of rational agency. *Logic Journal of the IGPL*, 11(2):135–159, 2003.

[248] W. van der Hoek and M. Wooldridge. On the dynamics of delegation, cooperation, and control: A logical account. In F. Dignum, V. Dignum, S. Koenig, S. Kraus, M. Singh, and M. Wooldridge, editors, *Proceedings of AAMAS'05: the 4th International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 701–708, New York, USA, 2005. ACM Inc.

[249] W. van der Hoek and M. Wooldridge. On the logic of cooperation and propositional control. *Artificial Intelligence*, 64:81–119, 2005.

[250] R. van der Meyden. Common knowledge and update in finite environments. *Information and Computation*, 140(2):115–157, 1998.

[251] H. van Ditmarsch. *Knowledge Games*. PhD thesis, University of Groningen, Groningen, The Netherlands, 2000.

[252] G. van Drimmelen. Satisfiability in alternating-time temporal logic. In *Proceedings of LICS'03: the 18th Annual IEEE Symposium on Logic in Computer Science, 22-25 June 2003, Ottawa, Canada*, pages 208–217, Washington, DC, USA, 2003. IEEE Computer Society.

[253] B. van Linder, W. van der Hoek, and J.-J. C. Meyer. Formalising abilities and opportunities of agents. *Fundamanta Informaticae*, 34(1–2):53–101, 1998.

[254] S. van Otterloo and G. Jonker. On epistemic temporal strategic logic. In *Proceedings of LCMAS'04: 2nd International Workshop on Logic and Communication in Multi-Agent Systems: Nancy, France, August 16-20, 2004*, pages 35–45, 2004.

[255] S. van Otterloo, W. van der Hoek, and M. Wooldridge. Knowledge as strategic ability. *Electronic Notes in Theoretical Computer Science*, 85(2):1–23, 2004.

[256] M. Vardi and L. Stockmeyer. Improved upper and lower bounds for modal logics of programs. In *Proceedings of STOC'85: the seventeenth annual ACM symposium on Theory of computing, Providence, RI, USA*, pages 240–251, New York, NY, USA, 1985. ACM Press.

[257] M. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, 1994.

[258] M. Y. Vardi. On the complexity of bounded-variable queries. In *Proceedings of PODS'95: the 14th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 22-25, 1995, San Jose, California, USA*, pages 266–276. ACM Press, 1995.

[259] M. Y. Vardi. Why is modal logic so robustly decidable? In N. Immerman and P. G. Kolaitis, editors, *Proceedings of a DIMACS Workshop: Descriptive Complexity and Finite Models, January 14-17, 1996, Princeton University, NY, USA*, volume 31 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 149–184. American Mathematical Society, 1996.

[260] M. Y. Vardi and P. Wolper. Automata-theoretic techniques for modal logic of programs. *Journal of Computer and System Sciences*, 32(2):183–221, April 1986.

[261] J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior.* Princeton University Press, Princeton, NJ, USA, 1944.

[262] G. H. von Wright. *An Essay in Modal Logic.* North Holland Publishing Company, Amsterdam, The Netherlands, 1951.

[263] G. H. von Wright. *Norm and Action: A logical Inquiry.* Routledge and Kegan Paul, London, UK, 1963.

[264] D. Walther. ATEL with Common and Distributed Knowledge is ExpTime-complete. In *Proceedings of M4M-4: the 4th Workshop on Methods for Modalities, Humbolt University, Berlin, December 1-2, 2005*, 2005.

[265] D. Walther, C. Lutz, F. Wolter, and M. Wooldridge. ATL satisfiability is indeed Exptime-complete. *Journal of Logic and Computation*, 16:765–787, 2006.

[266] D. Walther, W. van der Hoek, and M. Wooldridge. Alternating-time temporal logic with explicit strategies. To appear in proceedings of TARK'07: the 11th Conference on Theoretical Aspects of Rationality and Knowledge, Brussels, Belgium, June 25–27, 2007.

[267] I. Walukiewicz. Completeness of Kozen's axiomatisation of the propositional mu-calculus. In *Proceedings of LICS'95: the 10th Annual IEEE Symposium on Logic in Computer Science, June 26-29, 1995, San Diego, California, USA*, pages 14–24. IEEE Computer Society, 1995.

[268] T. Wilke. CTL$^+$ is exponentially more succinct than CTL. In *Proceedings of FSTTCS'99: the 19th Conference on Foundations of Software Technology and Theoretical Computer Science, Chennai, India, December 13-15, 1999*, pages 110–121, London, UK, 1999. Springer-Verlag.

[269] T. Wilke. Alternating tree automata, parity games, and modal μ-calculus. *Bulletin of the Belgium Mathematical Society*, 8(2):359–391, May 2001.

[270] M. Wooldridge. *Reasoning about Rational Agents.* The MIT Press, July 2000.

[271] M. Wooldridge. *An Introduction to Multi-agent Systems.* John Wiley and Sons Ltd, March 2002.

[272] M. Wooldridge and W. van der Hoek. Time, knowledge, and cooperation: Alternating-time temporal epistemic logic and its applications. In *Proceedings of the 5th International Conference on Coordination Models and Languages*, page 4. Springer-Verlag, 2002.

List of Tables

List of Figures

Index

Symbols