Social Laws for Multi-Agent Systems

Thesis submitted in accordance with the requirements of the
University of Liverpool for the degree of Doctor in Philosophy

by

Mark Jonathan Roberts

Defended March 28, 2007

Abstract

The aim of this thesis is to investigate formal frameworks for social laws in multi-agent systems. These frameworks are used for describing, reasoning about and ultimately verifying the properties of such systems. In this thesis we show how Alternating-time Temporal Logic (ATL) provides an elegant and powerful framework within which to express and understand social laws for multiagent systems. We show that the *effectiveness*, *feasibility*, and *synthesis* problems for social laws may naturally be framed as ATL model checking problems, and that as a consequence, existing ATL model checkers may be applied to these problems. We also show that the complexity of the feasibility problem in our framework is no more complex in the general case than that of the corresponding problem in the Shoham-Tennenholtz framework (it is NP-complete). We show how our basic framework can easily be extended to permit social laws in which constraints on the legality or otherwise of some action may be explicitly required. Next, we introduce the notion of knowledge. The result is a framework in which we can, for example, express that a desirable property (objective) of a social law is that one agent has the ability to bring about a certain type of knowledge in another agent, or that if one agent knows something, then it should behave in a certain way. Next, we reason about agents being able to choose whether to act socially or not. We construct a logical language, based on Alternating Temporal Epistemic Logic (ATEL), in order to reason about whether or not agents are acting in a social manner. We introduce Social Belief, which is belief based on what *should* be true according to the social laws. We also introduce the notion of a socially necessary fact, which is a fact that will inevitably be true if all the agents in the system act in a social manner. Finally, we try to capture similar properties in an alternative approach and try to relate these two approaches.

Acknowledgements

This thesis would not have been possible without the help and support I have received from numerous people...

First, I am extremely grateful for all the academic support provided to me by my supervisors, Mike Wooldridge and Wiebe van der Hoek. They have given me lots of help and guidance along the way, without which, my thesis would not have been possible. I would also like to thank Clare Dixon, who as my advisor, has helped greatly with my PhD. I would like to thank Clare Dixon and Tim Norman for acting as examiners in my Viva Voce.

Second, thanks to the Department of Computer Science at the University of Liverpool for funding my PhD. Also, thanks to AgentLink for supporting my attendance at the AAMAS conference in Utrecht.

Third, I would like to thank all my colleagues in the Department of Computer Science. Special thanks to Justin Wang who has been a great friend here at Liverpool and provided me with lots of help and support.

Finally, I would like to thank my girlfriend Jyoti, and my parents for all their encouragement and support.

This thesis was submitted to the Faculty of Science in the University of Liverpool on January 29, 2007, and was successfully defended on March 28, 2007. It was typeset using LaTeX.

Contents

Part I

Introduction

Chapter 1

Introduction

Multi-agent systems has become a very prominent area of Computer Science research in recent years [63]. The area of multi-agent systems was originally a part of Distributed Artificial Intelligence (DAI) [8, 18, 36], which is a subfield of Artificial Intelligence (AI) [6, 20, 21, 44]. An agent is defined to be a software entity capable of flexible, autonomous behaviour in some environment, to act on behalf of its owner to achieve specified goals (adapted from [63]). A multi-agent system is a system comprising two or more interacting agents. Coordination is one of the fundamental problems in multi-agent systems. The coordination problem is that of *managing the inter-dependencies between the activities of agents [63].* One approach to solving this problem is that of "Social Laws" [47, 46, 48, 35, 49, 50]. A social law is a restriction on the behaviour of agents in such a way that some overall societal goal is achieved. The aim of this thesis is to investigate formal frameworks for social laws in multi-agent systems. These frameworks are used for describing, reasoning about and ultimately verifying the properties of such systems.

This chapter discusses the motivations for developing formal frameworks for social laws, and existing social laws frameworks that we can base our work on. We then outline our own approach to social laws in multi-agent systems. Finally, the structure of the remainder of the thesis is outlined.

1.1 Motivation

The area of coordination is very important in multi-agent systems. Given the possibility of different agents' activities interacting, some form of coordination mechanism will be essential for the agents to coexist. The coordination mechanism itself can vary in design from the one extreme of having a centralised control entity to the other extreme of having each individual agent negotiate with each other every time there is a potential conflict. We call these two extremes tightly coupled and loosely coupled, respectively. The former case has a large number of interdependencies across the agents and there is a high risk that changes in one agent will create unanticipated changes within others. The latter case is far more flexible, but this flexibility comes at a cost – a more complex problem at run-time.

Various types of coordination mechanisms have been researched. These include Partial Global Planning [13], a coordination technique that allows separate AI systems to reason about their roles and responsibilities as part of group problem solving, coordination through joint intentions [26], an explicit model of joint problem solving, and coordination by mutual modelling [19], a coordination technique where agents build a model of the other agents (beliefs, intentions etc) and coordinate their activites around predictions made from this model. Also, in [41], Pretschner et al. describe the fundamentals of distributed usage control of data. Due to the amount of personal data collected and enhanced network capabilities, the distribution of data could be potentially uncontrolled. To rectify this, they introduce usage control requirements which are classed as either provisions or obligations. Provisions represent access control requirements, while obligations are concerned with requirements on the future that the data consumer must adhere to. A requirement is said to be enforceable if mechanisms can be employed such that all executions of the system satisfy the requirement. Finally, and most important to this thesis, is the area of coordination by social laws [48]. The area of social laws has proved to be a popular approach in trying to solve the coordination problem. A social law constrains the behaviour of agents in a society in such a way that some societal goal is achieved. The approach is said to be an intermediate approach between the two extremes outlined above.

Social laws originate from social sciences [11]. The social laws paradigm in Multi-Agent systems was first introduced by Moses, Shoham and Tennenholtz [47, 35, 50]. Shoham et al investigated a number of issues surrounding the development and use of social laws, including the computational complexity of their synthesis (the problem of finding and *implementing* an effective social law), and the possibility of the development of social laws or conventions by the agents within the system themselves. So, as such, existing frameworks for social laws have been investigated. However, we can see scope for much more research to be carried out in the area. Firstly, there exists no *explicit* notion of *what* the social laws are in place to achieve. Social laws are implemented to achieve some *objective*, and the way this objective is achieved is by the restriction on the behaviour of the agents. Secondly, existing frameworks have used logics primarily to derive axioms in order to capture static properties of systems and social laws. This leaves scope for logics to be used in order to express dynamic properties. The notion of cooperation lends itself well to coordination scenarios, as in a group of agents cooperating to achieve some coordination objective. So a cooperation logic would be ideally suited. Temporal aspects of coordination could also be considered. Also, a very important notion is that of knowledge. Agents use knowledge about their states and the states of other agents in order to follow social laws. Furthermore, knowledge of laws themselves could be used, e.g., only a certain group of agents may know a certain law. Finally, most existing frameworks have made the assumption that all agents in the system will follow the social laws in place. The possibility of agents violating social laws is interesting and could be investigated within the setting of a formal framework.

1.2 Background

The first stage in researching social laws in multi-agent systems is to look at any existing work in the field. There are two main approaches to the design of social laws. These are:

- *Offline design*, whereby social laws are designed offline and hardwired into the agents. Examples of this approach are [48, 35].
- *Emergence at run-time*, whereby social laws evolve within a society of agents. Examples of this approach include [62, 50, 27].

The first approach will generally be simpler to implement and the system designer will have a greater degree of direct control over the system functionality. However, not all characteristics of a system are necessarily known at the time of design; for example, open systems such as the Internet where interacting software entities interact through publicly available interfaces. Also, the goals of agents might be constantly changing, requiring the agents to be re-programmed, which would be costly and inefficient.

Existing work in the emergence of social laws at run-time is largely empirical. Most of the work in this area is experimental work to find out the best strategy update function to use, where the goal is for all the agents in the system to converge on the same strategy. The strategy update function uses feedback from interactions with other agents in the environment and updates the agent's strategy accordingly, based on pre-defined rules. Findings from these experiments have been analysed and presented in works by Shoham and Tennenholtz [50], Walker and Wooldridge [62] and Kittock [27].

The offline design of social laws has received far more attention. The main idea behind this approach is that a social law is a set of constraints. These constraints forbid the agents from performing a specified action in one or more specified environment states. This is seen as a restriction on the behaviour of agents. Formal frameworks for social laws have been created and computational problems which arise have been addressed. Researchers have also looked at the design problem of social laws, which involves striking the right balance between being overly restrictive and overly liberal. This has led to work into the notion of *minimality*, which is essentially a social law which constrains the agents *just enough* to achieve their goals, while leaving them with maximal individual freedom.

1.3 Our Approach to Social Laws for MAS

In this thesis we will only look at the offline design of social laws. The work consists of three parts. The first part is a formal model of social laws. Here we model a system as an Alternating Transition System, which is essentially a concurrent game structure. In this model, a social law consists of two parts:

1. An *objective*.
2. A *behavioural constraint*.

The notion of an objective is first made explicit in our framework of social laws. The objective of a social law is what the society aims to achieve by adopting the social law. An *effective* social law is one which achieves its objective. The behavioural constraint is the *means* by which the objective is achieved. In our framework we can express objectives that refer to time and cooperation in a rich logic known as Alternating-time Temporal Logic (ATL). Also, in this framework we can *automatically verify* objectives of social laws, in other words, see if certain laws are indeed effective. We also investigate the computational problems which arise as a result of our framework.

The second part of the work involves incorporating the notion of knowledge into the framework. We claim that knowledge is important in any non-trivial multi-agent system. Coordination scenarios will require agents to use knowledge of their states and other agents' states in order to follow laws. Also we can imagine situations when information must become known to a group of agents, or conversely the information must remain private to a group of agents. So here the knowledge itself is under the control of the social laws, and, as such, the social laws can ensure certain desirable knowledge properties ensue. We extend the framework in a natural way using Alternating Temporal Epistemic Logic, which is essentially ATL with knowledge operators.

The third and final piece of work involves reasoning about agents acting in an anti-social manner. We give agents the choice of whether to follow the social laws or not. We are able to specify whether agents are forced to follow the social laws or not in the object language itself. This gives rise to a new notion of belief, which we call *Social Belief* and is a belief based on the assumption of correct social behaviour. Finally, we look at an alternative approach for expressing similar properties and try to relate the two.

1.4 Structure of Thesis

The structure of the rest of the thesis is in three parts. The first part, consisting of chapters 2 and 3, is a literature survey. Chapter 2 is a survey of logics used in multi-agent systems. We look at Alternating-time Temporal Logic (ATL), Epistemic Logic, Alternating Temporal Epistemic Logic (ATEL), BDI Logic, and Deontic Logic. Some of these logics will then be used later in the thesis. Chapter 3 is a survey of the social laws literature. It looks briefly at the emergence of social laws at run-time and goes on to cover three approaches to the offline design of social laws.

Chapters 5, 6 and 7 are the main body of research as they describe in detail our formal framework of social laws in multi-agent systems. Chapter 4 introduces the semantic structures on which our formal framework is based and defines ATL over these structures. In Chapter 5 our notion of a social law is introduced along with three computational problems which arise from it. Chapter 6 adds in the notion of knowledge to our framework by naturally extending the semantic structures and the logic used to express objectives of social laws. Different types of knowledge properties are identified and many properties of knowledge are investigated in an example. Chapter 7 extends the framework further by adding in the possibility of agents acting in an anti-social manner. Many properties are investigated in this framework, including informational properties. Finally, in Chapter 8 we try to capture similar properties in an alternative framework and find direct equivalences between the two.

The final part of the thesis (Chapter 9) presents some conclusions.

1.5 Publications in this Thesis

Portions of this thesis include previously published work. Chapters 4 and 5 include material from [52] and Chapter 6 includes material from [51].

Part II

Background

Chapter 2

Logics for Multi-Agent Systems

Logics are widely used in multi-agent systems for formally specifying, reasoning about and verifying properties of such systems. Logics consist of a well-defined syntax, which specifies what the well-formed formulae of the logic are, a well-defined semantics, used to give a formal meaning to the operators of the logic, and a well-defined proof theory, consisting of axioms and rules of inference, which determine some mechanical procedure to derive formulae. Using logics for reasoning about multi-agent systems allows properties of systems to be expressed in a rigorous, mathematical way, and at the same time removes ambiguity.

In this chapter we first introduce Alternating-time Temporal Logic (ATL), a popular temporal logic of cooperation [2]. We then go on to introduce epistemic logic, which is the logic of knowledge [16]. Then we show how ATL can be extended to include knowledge operators to form Alternating Temporal Epistemic Logic (ATEL) [57]. Then we introduce BDI logic, the logic for expressing an agent's beliefs, desires and intentions [42]. Finally, we look at denontic logic, the field of logic concerned with permissions and obligations [60].

2.1 Alternating-time Temporal Logic (ATL)

Temporal logic formulae are usually evaluated in either *linear-time* or *branching-time*. The former assumes that there is only one path of the system, so at each moment there is only one possible future. The latter assumes that time has a branching, tree-like nature, so each moment in time may split into alternate courses representing different possible futures. Branching-time logics allow explicit existential or universal quantification over all paths. Alternating-time Temporal Logic (ATL), however, allows a more subtle quantification over the possible outcomes of the system. As such, ATL can be understood as a generalisation of the well-known branching time temporal logic CTL [14], in which path quantifiers are replaced by *cooperation* *modalities*. A cooperation modality $\langle\langle G\rangle\rangle\varphi$, where G is a coalition of agents, expresses the fact that the coalition G can cooperate to ensure that φ; more precisely, that there exists a strategy profile for G such that by following this strategy profile, G can ensure φ, no matter what the other agents in the system do. ATL formulae comprise of a cooperation modality proceeded with a temporal logic formula using the following temporal operators: "$\Box$" means "now and forever more", "$\Diamond$" means "either now or at some point in the future", "$\mathcal{U}$" means "until", and "$\bigcirc$" means "in the next state". So ATL allows the powers of agents and coalitions of agents to be expressed in a temporal manner. Formally, the set of ATL formulae, formed with respect to a set of agents Ag, and a set of primitive propositions Φ, is given by the following grammar:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle\langle G\rangle\rangle\bigcirc\varphi \mid \langle\langle G\rangle\rangle\Box\varphi \mid \langle\langle G\rangle\rangle\varphi\,\mathcal{U}\,\varphi$$

where $p \in \Phi$ is a propositional variable and $G \subseteq Ag$ is a set of agents. We identify two fragments of ATL: the $\bigcirc$-fragment is the class of ATL-formulae containing only the $\bigcirc$ temporal operator and propositional variables; *propositional logic* formulae are those containing no cooperation modalities. Consider the following informal examples in order to illustrate the logic:

$$\langle\langle 1,2\rangle\rangle\Box\neg\mathit{fail}$$

This ATL formula says that agents 1 and 2 have a strategy to cooperate in such a way to ensure the system never enters a *fail* state, no matter what the other agents in the system do. Next, consider:

$$\langle\langle G\rangle\rangle\Diamond\mathit{goal}$$

This formula expresses the fact that there exists a strategy profile for the coalition G, such that if G follow this profile, then the system is guaranteed to reach a state where *goal* holds. Finally, consider:

$$\langle\langle 1\rangle\rangle\neg\mathit{enter}\,\mathcal{U}\,\mathit{permission}$$

This says that agent 1 has a strategy so that *enter* will remain false until *permission* becomes true. This could capture a scenario where an agent is not allowed to enter some resource until permission has been granted.

Now that we have seen the logical language itself, we introduce the semantic structures that ATL is based upon. These structures are called Alternating Transition Systems (ATSs) and are essentially concurrent game structures. These systems may be in any of a finite set Q of possible *states*. Systems are populated by a set Ag of *agents*; a *coalition* of agents is simply a set $G \subseteq Ag$, and the set of all agents is known as the *grand coalition*.

Definition 1 *An* ATS *is a tuple* $S = \langle Q, Ag, \tau, \Phi, \pi\rangle$ *with the following components:*

- Q *is a finite, non-empty set of* states;
- $Ag = \{1, \ldots, n\}$ *is a finite, non-empty set of* agents;
- $\tau : Q \times Ag \rightarrow 2^{2^Q}$ *is the choice function, which maps states and agents to the choices available to these agents. So* $\tau(q, i)$ *is a set of choices available to agent i when the system is in state q. The choice function yields a unique result when applied to the set of all agents, so for every state* $q \in Q$ *and every set* $Q_1, \ldots, Q_n$ *of choices* $Q_i \in \tau(q, i)$, *the intersection* $Q_1 \cap \ldots \cap Q_n$ *is a singleton. This imposes a transition function that yields, in every* $q \in Q$, *a unique outcome* $\tau(q, 1) \cap \ldots \cap \tau(q, n)$;
- Φ *is a finite, non-empty set of* atomic propositions; *and*
- $\pi : Q \rightarrow 2^{\Phi}$ *is an interpretation function, which gives the set of primitive propositions satisfied in each state: if* $p \in \pi(q)$, *then this means that the propositional variable p is satisfied (equivalently, true) in state q.*

We denote the set of sequences over Q *by* Q^*, *and the set of non-empty sequences over* Q *by* Q^+.

We now give some further definitions before we can precisely define the semantics of ATL. For two states $q, q' \in Q$ and an agent $i \in Ag$, q' is an i-successor of q if there exists a set $Q' \in \tau(q, i)$ such that $q' \in Q'$. So here, q' is said to be an i-successor of q, if q' is a possible outcome of one of the choices available to i in state q. If $succ(q, i)$ denotes the set of i successors to state q, then q' is simply a *successor* to q if for all agents $i \in Ag$, we have $q' \in succ(q, i)$. A *computation* is an infinite sequence of states $\lambda = q_0, q_1, \ldots$ such that for all $u > 0$, the state q_u is a successor of q_{u-1}. A computation $\lambda \in Q^+$ starting in state q is referred to as a *q-computation*; if $u \in \mathbb{N}$, then we denote by $\lambda[u]$ the component indexed by u in λ (thus $\lambda[0]$ denotes the first element, $\lambda[1]$ the second, and so on).

We define a sequence of states, $\overrightarrow{s}$, as follows: if q is a state, then q is a sequence; if $\overrightarrow{s}$ is a sequence and q is a state, then $\overrightarrow{s}; q$ is a sequence. We define a function to return the length of a sequence of states as follows: $length(q) = 1, length(\overrightarrow{s}; q) = length(\overrightarrow{s}) + 1$.

A strategy σ_i for an agent $i \in Ag$ is a total function: $\sigma_i : Q^+ \rightarrow 2^Q$ which must satisfy the constraint that $\sigma_i(\overrightarrow{z}; q) \in \tau(q, i)$ for all $\overrightarrow{z} \in Q^*$ and $q \in Q$. So a strategy for an agent i takes a non-empty sequence of states and maps this to a set of choices for agent i. The constraint ensures that the strategy leads to choices that are consistent with the choice function $\tau(q, i)$ for agent i in state q. We also define a special type of strategy, called a *memoryless strategy*, where agents only consider the current state as a basis of their choices. A strategy is memoryless if it satisfies the following property: If $\sigma_i(\overrightarrow{z}; q) = \sigma_i(\overrightarrow{z}'; q)$ for all $q \in Q$ and $\overrightarrow{z}, \overrightarrow{z}' \in Q^*$, then we simply write $\sigma_i(q)$. A *strategy profile* for a coalition $G = \{1, \ldots, k\} \subseteq Ag$ is a tuple of strategies $\langle \sigma_1, \ldots, \sigma_k \rangle$, one for each agent $i \in G$. We denote by Σ_G the set of all strategy profiles for coalition $G \subseteq Ag$; if $\sigma_G \in \Sigma_G$ and $i \in G$, then we denote i's component of σ_G by σ_G^i. Given a strategy profile $\sigma_G \in \Sigma_G$ and state $q \in Q$, let $out(\sigma_G, q)$ denote the set of possible states that may result by the members of the coalition G acting as defined by their components of σ_G for one step from q:

$$out(\sigma_G, q) = \left\{ q' \mid \forall h \in Ag \setminus G, \exists Q_h \in \tau(q, h) : q' \in \bigcap_{g \in G} \sigma_g(q) \cap \bigcap_{h \in Ag \setminus G} Q_h \right\}$$

Notice that, for any grand coalition strategy profile σ_{Ag} and state q, the set of possible states $out(\sigma_{Ag}, q)$ will be singleton.

Given a strategy profile $\sigma_G \in \Sigma_G$ and state $q \in Q$, let $comp(\sigma_G, q)$ denote the set of possible computations that may result by the members of the coalition G acting as defined by their components of σ_G starting from q. The set $comp(\sigma_G, q)$ will contain all the possible q-computations that G can enforce by following strategies in σ_G:

$$comp(\sigma_G, q) = \{\lambda \mid \lambda[0] = q \text{ and } \forall u \in \mathbb{N} : \lambda[u+1] \in out(\sigma_G, \lambda[u])\}.$$

Again, notice that for any grand coalition strategy profile σ_{Ag} and state q, the set $comp(\sigma_{Ag}, q)$ will be singleton.

We can now give the rules defining the satisfaction relation "$\models$" for ATL, which holds between pairs of the form S, q (where S is an ATS and q is a state in S), and formulae of ATL:

$S, q \models p$ iff $p \in \pi(q)$ (where $p \in \Phi$);

$S, q \models \neg\varphi$ iff $S, q \not\models \varphi$;

$S, q \models \varphi \vee \psi$ iff $S, q \models \varphi$ or $S, q \models \psi$;

$S, q \models \langle\langle G \rangle\rangle \bigcirc \varphi$ iff $\exists \sigma_G \in \Sigma_G$, such that $\forall \lambda \in comp(\sigma_G, q)$, we have $S, \lambda[1] \models \varphi$;

$S, q \models \langle\langle G \rangle\rangle \Box \varphi$ iff $\exists \sigma_G \in \Sigma_G$, such that $\forall \lambda \in comp(\sigma_G, q)$, we have $S, \lambda[u] \models \varphi$ for all $u \in \mathbb{N}$;

$S, q \models \langle\langle G \rangle\rangle \varphi \mathcal{U} \psi$ iff $\exists \sigma_G \in \Sigma_G$, such that $\forall \lambda \in comp(\sigma_G, q)$, there exists some $u \in \mathbb{N}$ such that $S, \lambda[u] \models \psi$, and for all $0 \leq v < u$, we have $S, \lambda[v] \models \varphi$.

We say that φ is *valid in S*, denoted $S \models \varphi$, if $S, q \models \varphi$ for all $q \in Q$. φ is simply *valid*, denoted $\models \varphi$, if it is valid in S for all ATSs S. The remaining classical logic connectives ("$\wedge$",

"$\rightarrow$", "$\leftrightarrow$") are assumed to be defined as abbreviations in terms of $\neg, \vee$, in the conventional manner. Also, $\langle\langle G\rangle\rangle \Diamond \varphi$ is shorthand for $\langle\langle G\rangle\rangle \top \mathcal{U} \varphi$. For readability, we omit set brackets in cooperation modalities, for example writing $\langle\langle 1\rangle\rangle$ instead of $\langle\langle\{1\}\rangle\rangle$.

Two cooperation modalities play a special role and are worth singling out for special attention. The cooperation modality $\langle\langle\rangle\rangle$ ("the empty set of agents can cooperate to...") asserts that its argument is true on all computations, and thus acts like CTL's universal path quantifier "A". Similarly, the cooperation modality $\langle\langle Ag\rangle\rangle$ asserts that its argument is satisfied on at least one computation, and thus acts like the CTL path quantifier "E".

In order to illustrate ATL-formulae, we give a simple example adapted from [1], page 50.

Example 1 (The train example) *We refer to this system as S_1. There are two trains, one of which (E) is Eastbound, the other of which (W) is Westbound, each occupy their own circular track. At one point, both tracks pass through a narrow tunnel – a crash will occur if both trains are in the tunnel at the same time.*

We model each train $i \in Ag = \{E, W\}$ as an automaton that can be in one of three states: "$away_i$" (the initial state of the train); "$waiting_i$" (waiting to enter the tunnel); and "in_i" (the train is in the tunnel). Each train $i \in \{E, W\}$ has two actions available. They can either move or idle. Idling causes no change in the train's state. If a train i moves while it is $away_i$, then it goes to a $waiting_i$ state; moving while $waiting_i$ causes a transition to an in_i state; and finally, moving while in_i causes a transition to $away_i$ as long as the other train was not in the tunnel, while if both trains are in the tunnel, then they have crashed, and are forced to idle indefinitely. Initially, both trains are away. Figure 2.1 illustrates the overall structure and transitions of the train system.

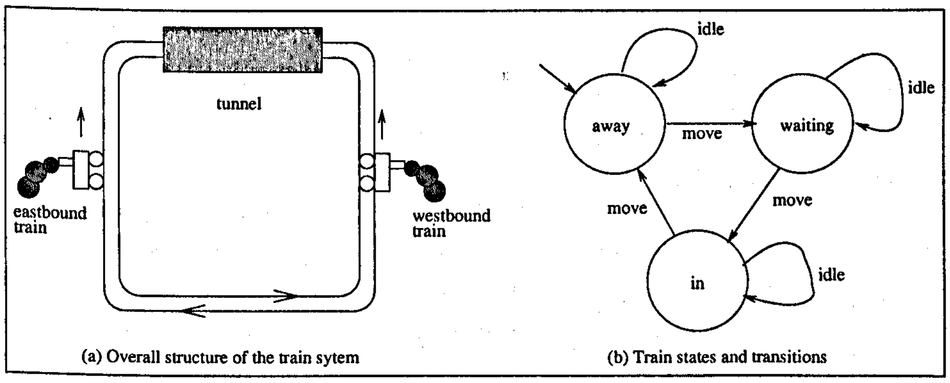

Figure 2.1: The train system.

The overall state of the system at any given time can be characterised by the propositional variables $\{away_E, away_W, waiting_E, waiting_W, in_E, in_W\}$, where these variables have the obvi-

ous interpretation. The system can be in one of nine possible states at any one time. These states along with their interpretations and the choices available to each agent are given in Figure 2.2.

q	$\pi(q)$	$\tau(q,E)$	$\tau(q,W)$
q_0	$\{away_E, away_W\}$	$\{\{q_0,q_1\},\{q_3,q_5\}\}$	$\{\{q_0,q_3\},\{q_1,q_5\}\}$
q_1	$\{away_E, waiting_W\}$	$\{\{q_1,q_2\},\{q_5,q_6\}\}$	$\{\{q_1,q_5\},\{q_2,q_6\}\}$
q_2	$\{away_E, in_W\}$	$\{\{q_2,q_0\},\{q_6,q_3\}\}$	$\{\{q_2,q_6\},\{q_0,q_3\}\}$
q_3	$\{waiting_E, away_W\}$	$\{\{q_3,q_5\},\{q_4,q_7\}\}$	$\{\{q_3,q_4\},\{q_5,q_7\}\}$
q_4	$\{in_E, away_W\}$	$\{\{q_4,q_7\},\{q_0,q_1\}\}$	$\{\{q_4,q_0\},\{q_7,q_1\}\}$
q_5	$\{waiting_E, waiting_W\}$	$\{\{q_5,q_6\},\{q_7,q_8\}\}$	$\{\{q_5,q_7\},\{q_6,q_8\}\}$
q_6	$\{waiting_E, in_W\}$	$\{\{q_6,q_3\},\{q_8,q_4\}\}$	$\{\{q_6,q_8\},\{q_3,q_4\}\}$
q_7	$\{in_E, waiting_W\}$	$\{\{q_7,q_8\},\{q_1,q_2\}\}$	$\{\{q_7,q_1\},\{q_8,q_2\}\}$
q_8	$\{in_E, in_W\}$	$\{q_8\}$	$\{q_8\}$

Figure 2.2: States and choices in the train example.

In order to demonstrate how strategies work in ATSs, we will consider the following example strategy. Firstly, for the eastbound train (E), imagine the strategy σ_E is as follows: move, idle, idle, move, idle, idle, etc. So E is always going to move, then idle for two time steps, then move again and so on. So every state where the index is a multiple of 3, E will move (e.g. q_3, q_6, q_9, etc).

So now, given the strategy for the eastbound train outlined above, $\sigma_E(\vec{s};q)$ is defined as follows:

If *length*$(\vec{s};q)$ mod $3 = 1$

1: $q = q_0$, then, $\sigma_E(\vec{s};q) = \{q_3,q_5\}$

2: $q = q_3$, then, $\sigma_E(\vec{s};q) = \{q_4,q_7\}$

3: and so on, for all remaining states.

Else

1: $q = q_0$, then, $\sigma_E(\vec{s};q) = \{q_0,q_1\}$

2: $q = q_3$, then, $\sigma_E(\vec{s};q) = \{q_3,q_5\}$

3: and so on, for all remaining states.

Secondly, we give an example strategy for the westbound train (W). The strategy, σ_W, is as follows: idle, move, idle, move, etc. So W is always going to alternate between moving and idling, starting off by idling. So, given this strategy, $\sigma_W(\vec{s};q)$ is defined as follows:

If $length(\vec{s}; q) \bmod 2 = 1$

1: $q = q_0$, then, $\sigma_W(\vec{s}; q) = \{q_0, q_3\}$
2: $q = q_1$, then, $\sigma_W(\vec{s}; q) = \{q_1, q_5\}$
3: and so on, for all remaining states.

Else

1: $q = q_0$, then, $\sigma_W(\vec{s}; q) = \{q_1, q_5\}$
2: $q = q_1$, then, $\sigma_W(\vec{s}; q) = \{q_2, q_6\}$
3: and so on, for all remaining states.

If we take the intersection $\sigma_W(\vec{s}; q) \cap \sigma_E(\vec{s}; q)$, we get the next state from q that the strategy would lead to.

Now we are able to express some desirable properties that we would like to hold in the train system. We will look at liveness and safety properties [32]. Liveness properties are those properties where something good should *eventually* happen, while safety properties ensure that something bad will be prevented from happening. First we can express the following liveness property:

$$S_1, q \models wait_E \rightarrow \langle\langle\rangle\rangle \Diamond in_E$$

This property expresses the fact that if the eastbound train is waiting to enter the tunnel, then no matter what the agents in the system do, at some point in the future it will be in the tunnel. This is a desirable liveness property, however, there is nothing to guarantee that this property holds, since the eastbound train could idle indefinitely in the waiting state. Finally, we express the following safety property:

$$S_1, q \models \langle\langle\rangle\rangle \Box \neg(in_E \wedge in_W)$$

This property expresses that no matter what the agents do, it will always be the case that they are never in the tunnel at the same time. Again, this is only a desirable property and it does not hold in the system S_1, as both trains can always move in the waiting state.

Now we introduce the model checking problem for ATL [9]. Model checking is used to verify formal systems, such as ATSs. Model checking tools take a system, called the model, programmed in some specific language for that model checker and a specification usually written in a temporal logic. The model checker then checks to see if the specification is true in the model, in which case the specification passes, otherwise a counter example is given to show why the specification failed. We make use of an ATL model checker called MOCHA [1].

MOCHA takes as input a system programmed in the REACTIVE MODULES language and a specification written in ATL. More precisely, the *model checking problem* for ATL is the problem of determining, for any given ATL formula φ, ATS S, and state q in S, whether or not $S, q \models \varphi$. If S is an ATS and φ is a formula then we say that φ is *initially satisfied* in S if $S, q_0 \models \varphi$, where q_0 is a pre-defined initial state. A formula φ is *satisfiable* if there is some ATS S and state q in S such that $S, q \models \varphi$. The *satisfiability problem* for ATL is the problem of determining, for any given ATL formula, whether this formula is satisfiable or not. The results for satisfiablity in ATL are given by the following theorem:

Theorem 1 ([2, 40, 12]) *The satisfiability problem for* ATL *is* EXPTIME*-complete [12], while the satisfiability problem for the* $\bigcirc$*-fragment of* ATL *is proven to be* PSPACE*-complete [40]. The model checking problem for "full"* ATL *can be solved in time* $O(|Q| \cdot |\varphi|)$*, where* $|Q|$ *is the number of states, and* $|\varphi|$ *is the size of the formula to be checked [2].*

Finally, we briefly mention ATL*, as we refer to it later in this thesis. ATL* is a more expressive variant of ATL, in which temporal operators and cooperation modalities are allowed to be intermingled in formulae. ATL requires cooperation modalities to be immediately proceeded by a temporal operator. This requirement is not present in ATL*. For example, the formula $\langle\langle i \rangle\rangle \Diamond \Box \varphi$ is perfectly acceptable in ATL*. For more details, consult [2].

2.2 Epistemic Logic

Epistemic logic is the logic of knowledge. It was originally studied in philosophy [25], but has grown to applications in computer science for reasoning about communication protocols and more recently in AI in order to reason about the knowledge and belief of software agents. The main operator, K, means "*it is known that...*". Epistemic logic is used to model what agents know. Several different types of knowledge exist, such as the knowledge of individual agents in a group, and group knowledge, such as common knowledge and distributed knowledge. We will first look at knowledge of individual agents in a group. Throughout this section we make use of the following references [16, 53].

2.2.1 Knowledge of individual agents in a group

We will now define the language of knowledge of individual agents in a group of k agents.

Formally, the set of knowledge formulae, formed with respect to a set of agents Ag, and a set of primitive propositions Φ, is the language L, given by the following grammar:

$$\varphi \ ::= \ p \mid \neg\varphi \mid \varphi \vee \varphi \mid K_i\varphi$$

where $p \in \Phi$ is a propositional variable and $1 \leq i \leq k$ is an agent.

The propositional variables in the language are used to express facts about the state of the world. The formula $K_i\varphi$ means that agent i *knows* φ, where φ will be some fact. For example, if we use φ to denote the fact that "it is raining in Liverpool", by $K_i\varphi$, we are expressing that agent i knows that it is raining in Liverpool.

We will now define the semantics of knowledge of individual agents in a group. If we look at the formula $K_i\varphi$, this asserts that agent i knows φ. But how is the notion of knowledge captured in the model? To capture knowledge, *possible worlds semantics* [24] are used. Given an agent, i, and a situation s, i has doubt about the true nature of the real world in s. So agent i considers several worlds possible, known as *epistemic alternatives* to s. If φ is true in all epistemic alternatives t that i considers possible, then i *knows* φ in s. For example, let φ denote "it is raining in Liverpool", as before, but now let ψ denote "it is raining in New York". If the current state I'm in is where I am in Liverpool and it is raining in Liverpool then there are two possible worlds for me: w_1 in which $(\varphi \wedge \psi)$ is true and w_2 in which $(\varphi \wedge \neg\psi)$ is true. In all possible worlds φ holds true, therefore I know φ. See Figure 2.3 for an illustration.

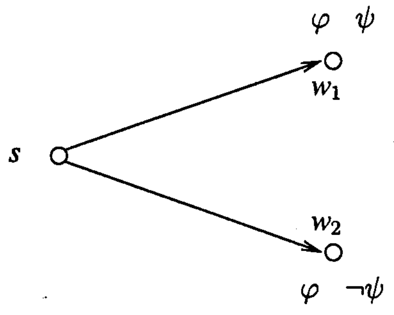

Figure 2.3: Kripke model

We will now define the semantics in a formal manner. As explained above, the semantics of knowledge use the idea of possible worlds. To define the semantics of our modal language L in a formal manner we introduce a Kripke structure: A Kripke structure (model) is a tuple $S = \langle Q, \pi, \sim_1, \ldots, \sim_k \rangle$ where:

- Q is a non-empty set of states,
- $\pi : Q \rightarrow 2^{\Phi}$ is a truth assignment to the propositional atoms in each state,
- $\sim_i \subseteq Q \times Q$ is an *epistemic accessibility* relation for $1 \leq i \leq k$.

We now define the satisfaction relation "$\models$" between pairs of the form S, q (where S is a Kripke model and q is a state in S) and formulae in our modal language L:

$S, q \models p$ iff $p \in \pi(q)$

$S, q \models \varphi \vee \psi$ iff $S, q \models \varphi$ or $S, q \models \psi$

$S, q \models \neg\varphi$ iff $S, q \not\models \varphi$

$S, q \models K_i\varphi$ iff for all q' such that $q \sim_i q'$, $S, q' \models \varphi$

Validity and satisfiability are defined as before. The accessibility relations, $\sim_i$, are often equivalence relations.

We now look at properties of this logic that are valid in every Kripke model. For a property φ, this is denoted by $\models \varphi$.

Definition 2 *Let φ, ψ be formulae in L, and let K_i be an epistemic operator for $1 \leq i \leq k$. The following properties hold:*

- $\models K_i\varphi \wedge K_i(\varphi \rightarrow \psi) \rightarrow K_i\psi$
- $\models (K_i\varphi \wedge K_i\psi) \rightarrow K_i(\varphi \wedge \psi)$
- $\models \varphi \Rightarrow \models K_i\varphi$
- $\models \varphi \rightarrow \psi \Rightarrow \models K_i\varphi \rightarrow K_i\psi$

So, these are a few example properties that are valid in every Kripke model. The first property says that knowledge is closed under consequences. The second property says that knowing φ and knowing ψ implies that you know the conjunction of the two. The third property says that agents know all validities. Finally, the last property says that if φ implies ψ is valid, then knowing φ implies you know ψ is also a validity.

The full axiom system $S5$ is often regarded as *the* standard epistemic logic. It establishes the exact properties of the notion K_i. In the system $S5$, the accessibility relations, $\sim_i$, are equivalence relations. The $S5$ system is defined as follows:

Definition 3 *The standard epistemic logic $S5$, where we have an operator K_i for every $1 \leq i \leq k$, is comprised of axioms $A1$, $A2$, $A3$, $A4$, and $A5$ below, and the derivation rules $R1$ and $R2$. The corresponding axioms and rules are given as:*

$A1$ *any axiomatisation for propositional logic*

$A2$ $(K_i\varphi \wedge K_i(\varphi \rightarrow \psi)) \rightarrow K_i\psi$

$A3$ $K_i\varphi \rightarrow \varphi$

$A4$ $K_i\varphi \rightarrow K_iK_i\varphi$

$A5$ $\neg K_i\varphi \rightarrow K_i\neg K_i\varphi$

$R1$ $\vdash \varphi, \vdash \varphi \rightarrow \psi \Rightarrow \ \vdash \psi$

$R2$ $\vdash \varphi \Rightarrow \ \vdash K_i\varphi, \mathit{for\ all}\ 1 \leq i \leq k$

The basic epistemic logic $\mathcal{K}$, consists of axioms $A1$, $A2$, and the derivation rules $R1$ and $R2$. In addition to the basic system $\mathcal{K}$, $\mathcal{S}5$ has axioms $A3, A4$ and $A5$. $A3$ says that knowledge is assumed to be *veridical*, meaning that agents do not know falsities. $A4$ is known as positive introspection, which means if agents know something, they know that they know it. Finally, $A5$ is known as negative introspection, meaning if agents do not know something, they know that they do not know it.

2.2.2 Group Knowledge

In this subsection we introduce group knowledge – the knowledge of multiple agents in a group. We have a group of k agents $\{1, \ldots, k\}$. First we introduce the notion of "Everybody knows...". This is written as $E\varphi$, meaning everybody in the group knows φ and is defined informally as follows:

$$E\varphi = K_1\varphi \wedge \ldots \wedge K_k\varphi$$

Next, we introduce the notion of common knowledge. Common knowledge is said to be the knowledge that "any fool" knows. It is knowledge that everybody knows, that everybody knows that everybody knows, and so on. This would intuitively be defined as:

$$C\varphi = \varphi \wedge E\varphi \wedge EE\varphi \wedge EEE\varphi \wedge \ldots$$

As infinite conjunctions are not allowed in the language of epistemic logic, this is not a formal definition.

Finally, we introduce distributed knowledge (or implicit knowledge). Distributed knowledge is knowledge that is implicit within a group of agents. No one particular agent may have knowledge of a fact φ, but together the knowledge of the fact φ may be present within the group of agents. If the agents could communicate, the fact φ could become explicit. For example, say no one in the group actually knows ψ. However, agent 1 knows φ $(K_1\varphi)$ and agent 2 knows that φ implies ψ $(K_2(\varphi \rightarrow \psi))$. This leads to distributed knowledge of ψ being present in the group of agents $(D\psi)$.

We will now give formal semantics of group knowledge. First we must extend our language L with all the group notions.

Formally, the set of knowledge formulae, formed with respect to a set of agents, and a set of primitive propositions Φ, is the language L', given by the following grammar:

$$\varphi \ ::= \ p \mid \neg\varphi \mid \varphi \vee \varphi \mid K_i\varphi \mid E\varphi \mid C\varphi \mid D\varphi$$

where $p \in \Phi$ is a propositional variable and $1 \leq i \leq k$ is an agent.

Now we can give formal semantics of L'. As before, we have a Kripke structure $S = \langle Q, \pi, \sim_1, \ldots, \sim_k \rangle$. Now we need to give the semantics of the new operators. For a group of k agents, we denote the union of their accessibility relations by $\sim^E$, so $\sim^E = \left(\bigcup_{1 \leq i \leq k} \sim_i\right)$. The transitive closure of $\sim^E$ is denoted by $\sim^C$.

The semantics of the E operator (everyone knows) are given as follows:

$$S, q \models E\varphi \text{ iff for all } q' \text{ such that } q \sim^E q' : S, q' \models \varphi$$

So, in a state q everyone knows φ ($E\varphi$), if and only if in all epistemic alternatives, φ is true. Here, the epistemic alternatives are not just for one agent, but for all the agents combined.

The common knowledge operator, C, is defined as follows:

$$S, q \models C\varphi \text{ iff for all } q' \text{ such that } q \sim^C q' : S, q' \models \varphi$$

This is defined in a similar way to "everyone knows" above, but here the relation that defines the epistemic alternatives is the transitive closure of $\sim^E$.

Finally, for distributed knowledge, D, we have:

$$S, q \models D\varphi \text{ iff } S, q' \models \varphi \text{ for all } q' \text{ such that } (q, q') \in \sim_1 \cap \ldots \cap \sim_k$$

In a state q, there is distributed knowledge of φ ($D\varphi$), if and only if for all epistemic alternatives, φ is true. The epistemic alternatives here are defined under the relation which is the intersection of all the agents' accessibility relations. So the worlds that the agents would consider possible are those which every agent has as an epistemic alternative in isolation.

In order to illustrate knowledge formulae we will introduce a simple example called The Muddy Children Problem, taken from [16]:

Example 2 *A number, say n, of children are standing in a circle around their father. There are $k(1 \leq k \leq n)$ children with mud on their heads. The children can see each other but they cannot see themselves. In particular they do not know if they themselves have mud on their heads. There is no communication between the children. The children all attended a course on epistemic logic and they can reason with this in a perfect way. Furthermore, they are perfectly* *honest and do not cheat. Now Father says aloud: "There is at least one child with mud on its head. Will all the children who know they have mud on their heads please step forward?" In case $k > 1$, no child steps forward. Father repeats his question. If $k > 2$, again the children show no response. This procedure is repeated until, after the k-th time Father has asked the same question, all muddy children miraculously step forward.*

In order to understand why all the children with mud on their heads suddenly step forward after the k-th announcement, we will look at the cases where $k = 1$ and $k = 2$. Firstly, when $k = 1$ (only one muddy child), after the father makes the announcement, it becomes common knowledge that at least one of the children has mud on their forehead. So, since each muddy child can only see $k - 1$ muddy children, the child who is muddy will see no muddy children and will thus deduce that he/she must be the muddy child and step forward. In the case where $k = 2$, call the two muddy children m_1 and m_2. After the first announcement has been made by the father, m_2 doesn't know whether he/she is muddy, as only one muddy child can be seen by m_2, namely m_1. But if m_1 couldn't see any other muddy children, m_1 would have stepped forward. After the father has made the second announcement, m_2 deduces that he/she must have a muddy forehead from the fact that m_1 didn't step forward after the first announcement was made. m_1 also reasons in exactly the same way as m_2, and hence both children now step forward.

Now we will analyse how the knowledge in the group changes after each announcement the father makes. We will take the case where we have three children, named 1, 2 and 3. S is the Kripke model representing the puzzle where $n = 3$. A state is a tuple $\langle x, y, z\rangle$ made up of proposition atoms $m_i (i = 1, 2, 3)$, with the interpretation that child i has a muddy forehead. For example, consider the state $\langle 0, 1, 0\rangle$, here the proposition m_2 holds, meaning child 2 has a muddy forehead. We use $\psi(j)$ to denote the fact that there are at least j muddy children. We are going to analyse the situation in (S, q) where $q = \langle 0, 1, 1\rangle$. Before the first announcement is made, we have the following knowledge properties present:

1. $S, q \models \neg(K_1 m_1 \vee K_1 \neg m_1)$
   child 1 does not know whether it is muddy;

2. $S, q \models K_1 m_2 \wedge K_1 m_3$
   child 1 knows that 2 and 3 are both muddy;

3. $S, q \models K_2(m_3 \wedge \neg K_3 m_3) \wedge K_2(\neg m_1 \wedge \neg K_1 \neg m_1)$
   child 2 knows that 3 is muddy without knowing it and 1 is not muddy without knowing it;

4. $S, q \models E\psi(1) \wedge \neg Em_1 \wedge \neg Em_2 \wedge \neg Em_3$
   Everyone knows that there is at least one muddy child, but not who the child is;

5. $S, q \models \neg C\psi(1)$
   It is not common knowledge that at least one child is muddy.

These properties can be verified by consulting Figure 2.4. The solid lines represent epistemic alternatives of child 1, the fine dotted lines represent epistemic alternatives of child 2, and finally, child 3's are represented by the coarse dotted lines. There are also reflexive arrows at each state in the diagram which we omit for clarity.

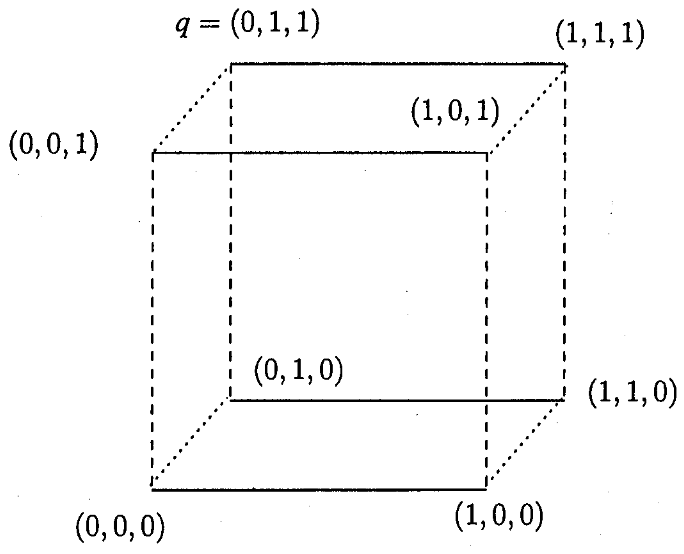

Figure 2.4: Kripke structure S showing 2 out of 3 children muddy, before the first announcement

After the first announcement from the father is made, we encounter an updated model S' (see Figure 2.5), where the children's knowledge is altered:

1. Items: 1, 2, 3 and 4 from before are not altered;

2. $S', q' \models C\psi(1)$
   It is now common knowledge that at least one of the children is muddy.

So now Figure 2.4 is truncated as none of the agents consider $(0,0,0)$ to be a possible world. The new structure we get is depicted by Figure 2.5.

After the second announcement, again we encounter an updated model S'' (see Figure 2.6), where the following knowledge properties hold:

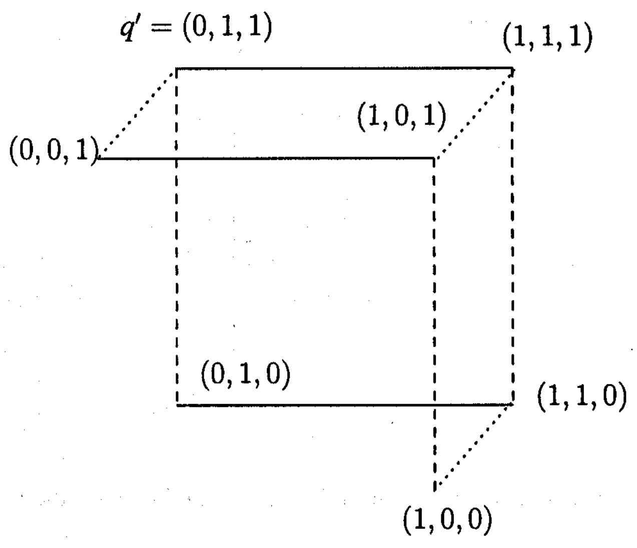

Figure 2.5: Kripke structure S' showing 2 out of 3 children muddy, after the first announcement

1. $S'', q'' \models K_2 m_2 \wedge K_3 m_3$
   child 2 knows he/she is muddy, as does 3;

2. $S'', q'' \models E(m_2 \wedge m_3)$
   everyone knows that children 2 and 3 are muddy;

3. $S'', q'' \models C\psi(2)$
   it is common knowledge that at least two children are muddy.

Now Figure 2.5 is truncated. The only epistemic alternative is of child 1 who still does not know if he/she is muddy. The epistemic alternatives of the other two children would be shown by drawing a reflexive arc at q''.

2.3 Alternating-time Temporal Epistemic Logic

Alternating-time Temporal Epistemic Logic (ATEL) is an extension of ATL by van der Hoek and Wooldridge [57] in which knowledge modalities, such as those described in the previous chapter, are combined with ATL. The resultant logic is a very powerful and succinct language for expressing properties of multiagent systems. ATEL has exactly the same cooperation modalities and temporal operators as ATL, but additionally, has the following knowledge operators:

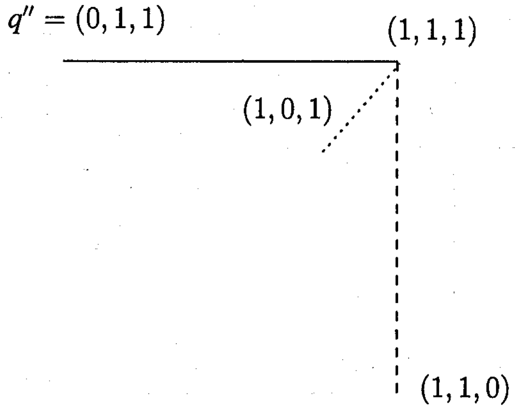

Figure 2.6: Kripke structure S'' showing 2 out of 3 children muddy, after the final announcement

$K_i\varphi$ ("agent i knows φ"), $E_G\varphi$ ("everyone in coalition G knows φ"), and $C_G\varphi$ ("it is common knowledge to everyone in the coalition G that φ").

Formally, the set of ATEL formulae, formed with respect to a set of agents Ag, and a set of primitive propositions Φ, is given by the following grammar:

$$\begin{aligned}\varphi \quad ::= \quad & p \mid \neg\varphi \mid \varphi \vee \varphi \mid K_i\varphi \mid E_G\varphi \mid C_G\varphi \mid \\ & \langle\langle G\rangle\rangle \bigcirc \varphi \mid \langle\langle G\rangle\rangle \Box \varphi \mid \langle\langle G\rangle\rangle \varphi \mathcal{U} \varphi\end{aligned}$$

where $p \in \Phi$ is a propositional variable, $G \subseteq Ag$ is a set of agents, and $i \in Ag$ is an agent.

We will now give some informal examples to illustrate the language and the type of formulae that can be expressed. By $cs = x$ we denote the value of the code to the safe is x. So $K_1(cs = x)$ means that agent 1 knows the code to the safe is x. The following formula expresses that if agent 1 knows the code to the safe, he can communicate this to agent 2, thus making agent 2 know the code to the safe:

$$K_1(cs = x) \rightarrow \langle\langle 1\rangle\rangle \Diamond K_2(cs = x)$$

If we take ψ to be the situation where the safe door is open, then we can express the following:

$$\langle\langle 1\rangle\rangle \Diamond \psi \rightarrow K_1(cs = x)$$

This formula says that knowing the code to the safe is x, is a *necessary* requirement for being able to open the safe door at some point in the future.

We can also have formulae which contain group knowledge. For example:

$$E_G\varphi \rightarrow \langle\langle G\rangle\rangle\Diamond\psi$$

This expresses the fact that if everyone in coalition G knows φ, then G have a strategy to achieve ψ at some point in the future. We can imagine a scenario where a group of people all have to go to the same meeting. If everyone knows the meeting is at a specific time, then the group has a strategy to arrive at the meeting on time.

We will now introduce the semantic structures that ATEL is based upon. These structures are very similar to the Alternating Transition Systems introduced earlier, but now we have an epistemic accessibility relation for each agent in the system. These structures are called Alternating Epistemic Transition Systems (AETSs) and are defined as follows.

Definition 4 *An* AETS *is a tuple* $S = \langle Q, Ag, \sim_1, \ldots, \sim_n, \tau, \Phi, \pi\rangle$ *with all components as before in the* ATSs, *with the addition of:*

- $\sim_i \subseteq Q \times Q$ *is an epistemic accessibility relation for each agent* $i \in Ag$. *Each* $\sim_i$ *must be an equivalence relation.*

The truth definition of ATEL formulae is the same as for ATL and the epistemic operators are defined in the same way as in Epistemic Logic in the previous section.

So now, not only can we refer to the powers of agents and coalitions of agents over time, but we can also refer to epistemic properties. With ATEL we can formulate properties where agents have the ability to communicate information which alters the knowledge of the agents, also we can formulate properties where certain knowledge is a necessary requirement in order to achieve some goal. Also, we can formulate properties where certain information must remain private to an agent or group of agents. This logic is very expressive and we will make use of it later in the thesis.

2.4 BDI

A common approach to talking about the behaviours of agents is the *intentional stance*. Rather than being concerned with the internal structure and operation of agents, the intentional stance abstracts away from such level of detail and attributes agents with human-like *attitudes*, such as believing, wanting (desiring) etc [58]. Such attitudes are used in *folk psychology* [64] to make predictions about human behaviour. The philosopher Daniel Dennett coined the phrase *intentional system* to describe entities whose behaviour can be predicted by attributing such attitudes [10].

The *belief-desire-intention* model of Rao and Georgeff [42] is a well-known approach to reasoning about rational agents from the intentional stance. The BDI model uses the three important attitudes of *belief, desire* and *intention* in order to reason about agents. An agent's beliefs are what the agent believes to be true in the current state based on the (possibly) limited information available. An agent's desires (or goals) correspond to *what* the agent *wants* to achieve. In general these desires can be inconsistent with one another, but implemented BDI systems require desires to be consistent with one another and these consistent desires are often called goals. Also, an agent would not be expected to achieve all of its desires, given resource limitations. This is where the agent's intentions come into play. The intentions are formed from a filtering process of the agent's desires. Only a sub-set of the agent's desires are chosen to become intentions. Once intentions are formed, the agent need not deliberate about what to do, its resources can simply be allocated to realising its intentions.

We will now introduce BDI logic itself. Our presentation is based on [58]. The formalism is similar to Computational Tree Logic (CTL*) [14], but there are additional modal operators for representing the beliefs, desires and intentions of the agents. These are Bel, Des and Intend, respectively. For example (Bel $i\,\varphi$) means agent i believes φ. Worlds in this logic are modelled using a temporal structure (called a *time tree*) with a branching time future and a single past. A time point in a world is called a *situation*. Events transform one situation (time point) to another. As shown below in Figure 2.7, an agent can choose to execute e_2 or e_3 in time point t_1, causing the associated transition to a different situation, t_2 or t_3 respectively. To summarise

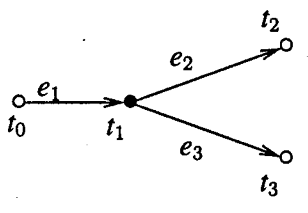

Figure 2.7: A BDI world

the key semantic structures in the logic, T is a set of time points, and all evolutions of the system possible are given by the binary relation $R \subseteq T \times T$. A world (over T and R) is then a pair $\langle T', R' \rangle$, where $T' \subseteq T$ is a non-empty set of time points, and $R' \subseteq R$ is a branching time structure on T'. W is the set of all worlds over T. A *situation* is a world, w, at a specific time, t, and is given by the pair $\langle w, t \rangle$ where $w \in W$ and $t \in T_w$ (T_w is the set of time points in the world w). For a given world $w \in W$, the set of all situations in w is denoted by S_w. The relationship of one world being a *sub-world* of another is defined as follows: w' is a sub-world of w if both worlds have the same structure and truth assignment of formulae, but w' has fewer paths than

w (i.e. w' is a sub-tree of world w).

As with CTL, a distinction is made between *state* and *path* formulae. State formulae are evaluated at a specific time point in a time tree, whereas path formulae are evaluated over a specific path in a time tree. We have the same path quantifiers as with CTL: E (called optional) where a formula is true in at least one path branching from the current time point, and A (called inevitable) where a formula is true on all paths branching from the current time point. We have the standard temporal operators which are used in both state and path formulae. In order to model the beliefs, desires and intentions we have belief accessibility relations $\mathcal{B}$, $\mathcal{D}$, and $\mathcal{I}$. These are modelled as functions that assign to every agent a relation over situations and are defined as follows:

- $B_t^w(i) = \{w' \mid \langle w, t, w' \rangle \in \mathcal{B}(i)\}$
- $D_t^w(i) = \{w' \mid \langle w, t, w' \rangle \in \mathcal{D}(i)\}$
- $I_t^w(i) = \{w' \mid \langle w, t, w' \rangle \in \mathcal{I}(i)\}$

where $B_t^w(i)$ denotes the set of worlds accessible to agent i from situation $\langle w, t \rangle$, and likewise for $D_t^w(i)$, $I_t^w(i)$. The semantics of the belief, desire and intention modalities are given below:

- $\langle w, t \rangle \models (\mathsf{Bel}\ i\ \varphi)$ iff $\langle w', t \rangle \models \varphi$ for all $w' \in B_t^w(i)$;
- $\langle w, t \rangle \models (\mathsf{Des}\ i\ \varphi)$ iff $\langle w', t \rangle \models \varphi$ for all $w' \in D_t^w(i)$;
- $\langle w, t \rangle \models (\mathsf{Intend}\ i\ \varphi)$ iff $\langle w', t \rangle \models \varphi$ for all $w' \in I_t^w(i)$.

The logic requires that desires are compatible with beliefs so that the agents do not have desires which are believed to be unachievable. This property is known as *strong realism* [42] and requires that the agent should believe it can optionally achieve its desires. To this end, for every belief-accessible world w at time t, there must exist a desire-accessible sub-world of w at time t. Desire-intention compatibility is achieved in a similar way.

The different belief, desire and intention accessible worlds represent different possible scenarios for the agent. The agent has doubt about the state of the actual world. The agent believes one of its belief-accessible worlds to be the actual world. If the actual world was b_1, then the agent's desires would be the corresponding desire-accessible world, d_1, and the agent's intentions would be the corresponding intention-accessible world, i_1.

In order to illustrate the belief, desire and intention relations we have formulated an example in Figure 2.8. Here the agent is faced with the choice of going swimming, going running or going to McDonalds, where the agent's goal is to keep fit. First we look at the belief worlds accessible to the agent. As b_1 is the only accessible belief world, the agent believes that going swimming will make him happy and fit, going to McDonalds will make him happy but unfit, and finally going running will make him unhappy but fit. Next we look at the agent's desire worlds. Notice in d_1 and d_2, f is true inevitably in both worlds, therefore the agent's desire is to be fit. Finally, looking at the intention worlds we see that the agent has fixed upon one option, both outcomes of which will make the agent fit, thus achieving his goal. Therefore, the intention the agent has fixed upon is to be fit, and in order to do this the agent has chosen to go swimming, so now the agent will focus his attention to achieving this and will not consider any of the previous options which have been ruled out. Now we can formally define the situation

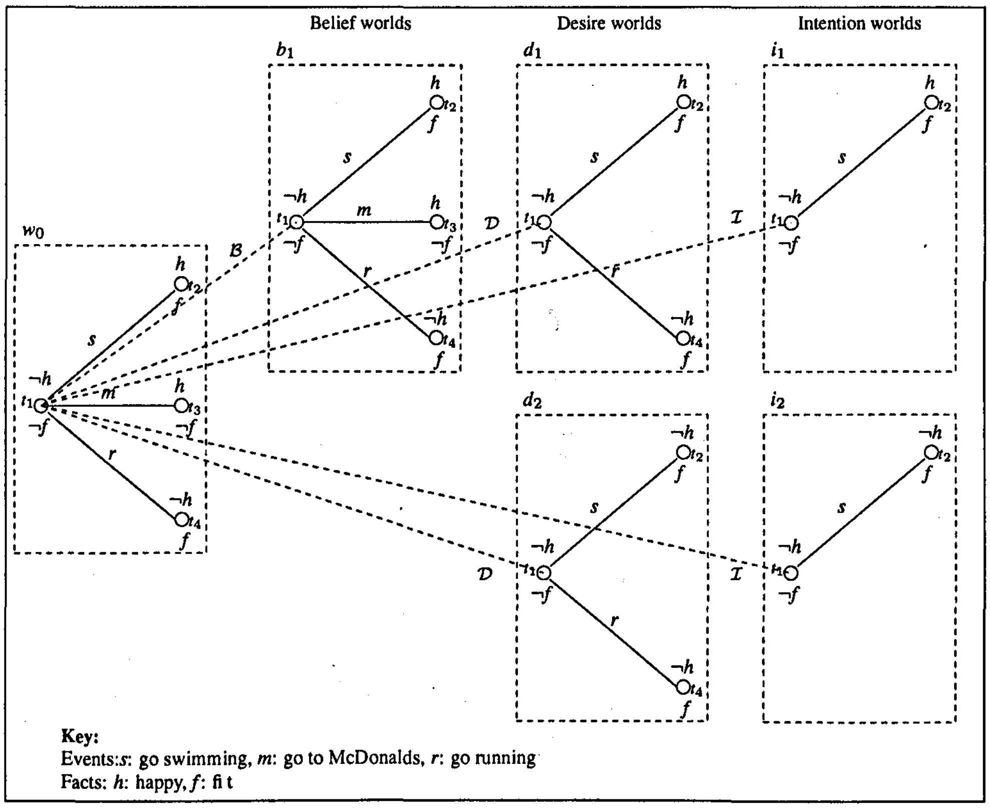

Figure 2.8: BDI relations

$\langle w_0, t_1 \rangle$ using BDI logic. If we call the agent 1, we can formulate the following:

$$\langle w_0, t_1 \rangle \models (\text{Des } 1\, A\Diamond f) \wedge (\text{Intend } 1\, A\Diamond f)$$

This says that agent 1 desires that it is inevitable for him to eventually become fit and that he also intends that it is inevitable for him to eventually become fit.

BDI logic has recently been extended to take into account the capabilities of individual agents [37]. Padgham and Lambrix define capability as *the ability to react rationally towards achieving a particular goal.* A capability to achieve X is understood as the agent having at least one plan to achieve X. Using such capabilities could be advantageous for finding plans that are a possibility for responding to a specific event. Not all plans need be looked at, only those plans with the capability relevant to responding to the particular event. It is suggested that capabilities could also be used in defining agents' roles. If agents dynamically change their roles, it is expected that their behvaiour will also change. A capability could specify and implement the things that an agent could do within a particular role. Then in the case of agents dynamically changing roles, it would simply be a case of appropriate capabilities being activated or de-activated. It is important to note that these capabilities do not come about because the agents have certain roles, rather the capabilities are inherent to the agents physical competence and pursuing a role may lead to expectations regarding the use of such capabilities. Capabilities may also help agents in cooperating. If an agent observes an event that itself does not have the capability to respond to, it could pass on the event to another agent who is believed to have the capability for dealing with the event. The semantics of capabilities are constructed using capability-accessible worlds. An agent is capable of achieving φ, if φ is true in all capability-accessible worlds. Goals and intentions are limited by capabilities.

2.5 Deontic Logic

Deontic logic is the branch of modal logics used for formally specifying normative behaviour. It originates from philosophy [60], but has more recently become an area of interest amongst Computer Scientists and AI researchers [34]. Deontic norms are sentences which create obligations and permissions. Norms do not describe how the world is, but they prescribe how the world *should* be. Deontic logic has modal operators for normative specification such as prohibition, permission and obligation.

Many systems of deontic logic have been formalised since Ernst Mally first tried to capture deontic notions in 1926 [31]. However, it can be argued that the first "real" system of deontic logic was proposed by von Wright in 1951 [60]. This system is known as the *standard system of deontic logic* or KD. In this section we will only be concerned with the KD system.

First we must introduce the deontic logic operators. There are three of these operators: O means obligated, P means permitted and F means forbidden. So, for example, $P\varphi$ means that φ is permitted. Formally, deontic logic formulae are formed with respect to the following grammar given in BNF:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid O\varphi \mid P\varphi \mid F\varphi$$

where p is a propositional variable.

Now we present the KD system as a normal modal logic consisting of the following rules and axioms:

(KD0) All (or enough) tautologies of Propositional Calculus

(KD1) $O(\varphi \rightarrow \psi) \rightarrow (O\varphi \rightarrow O\psi)$

(KD2) $O\varphi \rightarrow P\varphi$

(KD3) $P\varphi \equiv \neg O \neg\varphi$

(KD4) $F\varphi \equiv \neg P\varphi$

(KD5) Modus Ponens: $\varphi, \varphi \rightarrow \psi$ then ψ

(KD6) O-necessitation: φ then $O\varphi$

(KD1) is known as the K-axiom and holds for any modal necessity operator. It states that obligations are closed under logical implication. (KD2) is the D-axiom and says that if something is obliged then it is permitted. (KD3) shows how permission is the dual of obligation. (KD4) says that if something is forbidden that is the same as not permitted. (KD5) is modus ponens and finally (KD6) is O-necessitation where if φ is an established theorem then $O\varphi$ can be derived.

There are many theorems of the KD system, a few of which are considered as paradoxes. For example, consider Ross's paradox [43]:

$$O\varphi \rightarrow O(\varphi \vee \psi)$$

To understand why this is a paradox, consider the case where φ denotes "return the library book" and ψ denotes "keep the library book". The theorem then reads, "If one is obliged to return the library book, then one is obliged to either return the library book or keep the library book". Intuitively this does not seem to make sense. If one is obliged to do something, then they are obliged to do it. This is saying, if one is obliged to do something, they are either obliged to do it or they are obliged to do something that is contrary to the first action. Another paradox present in the system is:

$$\neg\varphi \rightarrow (\varphi \rightarrow O\psi)$$

If we use the same interpretation of φ and ψ then this theorem says, "If I do not return the library book, then returning the library book commits me to keeping the book".

The semantics of the standard system of deontic logic are based on possible worlds semantics [28], as with epistemic logic and BDI logic in the previous sections. Again, we have a set of worlds, Q, and a truth assignment function, π, assigning truth to primitive propositions in each world. We have a relation $R \subseteq Q \times Q$ representing the possible worlds accessible in each state. Here the possible worlds are the "perfect alternative worlds"; worlds in which all norms are fulfilled. There are modal operators O, F and P, meaning "obliged", "forbidden" and "permitted", respectively.

Formally, we have a Kripke structure $S = \langle Q, \pi, R \rangle$ where:

- Q is a non-empty set of states,
- π is the interpretation function which assigns truth to primitive propositions in each world,
- $R \subseteq Q \times Q$ is an accessibility relation, where R is taken to be serial in every state.

In a system S and a state q we define the semantics of the modal operators as follows:

$S, q \models O\varphi$ iff for all q' such that $qRq', S, q' \models \varphi$

$S, q \models P\varphi$ iff there exists q' such that $qRq', S, q' \models \varphi$

$S, q \models F\varphi$ iff for all q' such that $qRq', S, q' \not\models \varphi$

The formula $O\varphi$ means that φ is obligated. If $O\varphi$ is true in state q, then φ is true in all states related to q. The states related to q represent perfect alternatives to q. So if φ holds in all perfect alternatives to the current world, it is obligated. There is no way φ could be false by adhering to the norms, therefore it *has* to be true. The operator P is the dual of O. $P\varphi$ means that φ is permitted. If $P\varphi$ is true in state q, then there exists *some* perfect alternative to q where φ holds. Finally, $F\varphi$ means φ is forbidden. If $F\varphi$ is true in q then there exist no perfect alternatives where φ holds.

Now we are in a position to give an example to illustrate the deontic modal operators. Using the library example we introduced earlier, let φ denote "borrow library book", ψ denote "return library book by return date" and γ denote "keep library book indefinitely". Now we can formulate the following:

$$\varphi \rightarrow O\psi$$

this says, "If you borrow a library book, you are obliged to return the library book by the return date". This means the following will also be true:

$$O\psi \rightarrow F\neg\psi$$

this says, "if you are obliged to return the library book by the return date then you are forbidden from not returning the library book by the return date". Finally we can also formulate the following:

$$\varphi \rightarrow F\gamma$$

this says, "if you borrow a library book, you are forbidden from keeping the book indefinitely".

Many systems of deontic logic have been constructed since the standard system of deontic logic (KD system). For examples of such systems, see the following references: [61, 4, 59, 33]. Also, in [45], Sergot presents a generalisation of the Kanger-Lindahl theory of normative positions, where a combination of deontic logic and a logic of action/agency are used to give a formal account of obligations, duties, rights and other complex normative concepts. A normative 'position' is a mapping of the complete space of all logically possible relations between two agents with respect to some act-type (type of action). The language is propositional logic augmented with modal operators O (obligation), P (permission), and relativised modal operators $E_i, E_j, \ldots$ for act expressions, where $i, j, \ldots$ are the names of individual agents. So, for example, OE_iF says 'i ought to bring it about that F'. Given the truth of OE_iF in some state, the obligations and permissions of the other agents with respect to bringing about F are analysed. Say the only other agent is j, some possibilities might be OE_jF, $PE_jF \wedge \neg OE_jF$ and $\neg PE_jF$. All of these possibilities are examined systematically using 'simple types of rights relations'. There are the normative 'positions' which express the normative rights of one agent in relation to the rights of another agent.

2.6 Summary

This chapter has introduced a number of modal logics which are widely used for reasoning about multi-agent systems. The first logic introduced was Alternating-time Temporal Logic (ATL) and the systems the semantics are based upon, alternating transition systems (ATSs). A full formal semantics for the logic were given and the logic was illustrated by way of an example. We explained model checking and precisely defined the model checking problem in ATL. We then went on to introduce epistemic logic, the logic of knowledge. We started off by only looking at the knowledge of individual agents in a group and then went on to look at group knowledge. We demonstrated epistemic logic using the well known muddy children problem. Next we introduced Alternating-time Temporal Epistemic Logic, an extension to ATL, in which ATL is enriched with epistemic operators. We gave some simple examples to illustrate the expressiveness of the logic. We then went on to introduce BDI logic and demonstrated it using an example. Finally, deontic logic was introduced. The KD system was presented along with the semantics, some theorems and some example formulae.

All of the logics we have looked at in this chapter have been modal logics. ATL and ATEL naturally have lots in common, as ATEL is just an extension of ATL. Both of these logics can be used to express the powers of agents and coalitions of agents using cooperation modalities. Also, they are both temporal logics with identical temporal operators. The difference between the two is that ATEL has epistemic operators to refer to the knowledge of agents and groups of agents. ATL is very important in this thesis as it provides a logical basis for our frameworks of social laws. ATEL and epistemic logic also have something in common; they both use epistemic operators and can refer to the knowledge of agents and groups of agents. However, epistemic logic cannot refer to time or to the powers of agents and coalitions of agents. Epistemic logic, ATEL, BDI logic and deontic logic all use the idea of possible worlds to give semantics to their modal operators. Finally, we note that epistemic logic and BDI both have something in common. BDI has a modal operator for specifying the beliefs of an agent. This can be seen as *similar* to the knowledge operator, K_i, of epistemic logic. However, the distinction between these is that in epistemic logic, agents can only believe things which are actually true, whereas in BDI, the beliefs of agents are not necessarily true.

Chapter 3

Social Laws for Multi-Agent Systems

A social law is a restriction on the behaviour of agents, to ensure they can work individually in a mutually compatible manner in order to fulfil their individual goals. Social laws work by prohibiting the performance of certain actions in certain situations (states). For example, if two mobile robot agents wish to pass through a door from opposite directions only big enough to allow one of the robots to pass at a time, the robots will need to coordinate their activities. Without coordination, the two robots will likely collide, or at best a deadlock situation will occur, in which both robots wait indefinitely for the other robot to use the door. This type of situation could easily be avoided with the use of social laws. An example social law for this situation would be to always give priority to robots leaving the building. Assuming all agents abide by the social laws in place, the robot leaving the building need not consider waiting for the other robot to use the door first, as this would not be a legal configuration of the system. The robot leaving the building would simply pass through the door first, leaving the other robot free to pass through the door.

The social laws paradigm is an intermediate approach to the design of multi-agent systems, between the two extremes of having totally centralised control and a purely decentralised approach, where agents need to negotiate each time there is a potential conflict. It is suggested in [35] that agents should be designed to act individually, but their actions should be restricted so that their behaviour is mutually compatible. This involves striking the right balance between being overly liberal and overly restrictive. This trade-off is known as the Golden Mean Problem [35].

The effect of a social law is twofold. Firstly, it restricts the freedom of agents by reducing the number of actions available to them. This may reduce the number of goals the agent is able to attain, but crucially, it also reduces the on-line decision making burden on agents. For example, in the social law described above, the robot entering the room knows that it must wait for the robot leaving the room to use the door first. Knowing this means that the robot entering the room need not consider situations in which it uses the door first. Secondly, due to the fact that the social law also restricts the freedom of all the *other* agents in the environment, each agent might actually be able to achieve more goals, due to the social laws removing the possibility of conflicting actions.

There are two ways in which social laws can come to exist in a system. These are: offline design, whereby social laws are designed offline and hardwired into the agents [48, 35], and emergence at run-time, whereby social laws evolve within a society of agents [62, 50, 27]. The latter approach has been investigated because of the disadvantages associated with the offline design of social laws. Not all characteristics of a system are necessarily known at the time of design; for example, open systems. Also, the goals of agents might be constantly changing, requiring the agents to be re-programmed, which would be costly and inefficient. Instead, it would be beneficial if the society of agents could exhibit flexible behaviour, and converge on new social laws dynamically, as and when they are required.

Now we have introduced the social laws paradigm as a solution to the coordination problem, the rest of this chapter is structured as follows: In Section 3.1, we discuss the approach of "emergence of social laws at run-time" and present various strategy update functions. We also survey experimental results obtained by using these functions and explain their significance. In Section 3.2, we give details of the offline design of social laws and present various frameworks. Firstly, we introduce Artificial Social Systems, which define a multi-agent system as a set of non-deterministic automata. Secondly, we introduce the social laws framework, as proposed by Shoham and Tennenholtz. We then look at ways to choose between different useful social laws: minimality and simplicity are the suggested criteria. We finish by summarising the chapter in Section 3.3.

3.1 Emergence at Run-time

In this approach, the possibility of conventions (social laws) emerging from within a group of agents is investigated. The study of social laws emerging from within a system of agents can be thought of as first beginning with the study of *Social Conventions* in social philosophy. In [23], social conventions are said to have the following two characteristics, (i) they result somehow from the interdependency of actions, and (ii) they appear to come about by chance within the bounds of some functional description. For example, in the UK we drive on the left, but in terms of the functional description of avoiding collisions, we could equally just as well drive on the right, as in the USA. A central issue in social philosophy has been to explain what causes conventions *to*, and how conventions *do*, emerge, stabilise, and in some cases change or deteriorate [23]. In [29], Lewis employed game theory as means of studying social conventions in a formal manner. Lewis defines a convention as follows: A regularity R in the behaviour of members of a population P when they are agents in a recurrent situation S is a *convention* if and only if it is true that, and it is common knowledge in P that, in almost any instance of S among members of P,

(1) almost everyone conforms to R;

(2) almost everyone expects almost everyone else to conform to R;

(3) almost everyone has approximately the same preferences regarding all possible combinations of actions;

(4) almost everyone prefers that any one more conform to R, on condition that almost everyone conforms to R;

(5) almost everyone would prefer that any one more conforms to R', on condition that almost everyone conforms to R',

where R' is some possible regularity in the behaviour of members of P in S, such that almost no one in almost any instance of S among members of P could conform to both R' and to R [29]. It is difficult to see if this definition is consistent with the two characteristics (i) and (ii) above. With R described in a game theoretic setting, conventions can be characterised by the same behaviour as *proper coordination equilibrium.* The notion of proper coordination equilibrium is best illustrated in a pure-coordination game such as the one illustrated in Figure 3.1 below. In this game, player 1 and player 2 repeatedly face the symmetric and simultaneous strategy

		player 2	
		a	b
player 1	a	**1,1**	0,0
	b	0,0	**1,1**

Figure 3.1: a pure-coordination game

choice between a and b. The Nash Equilibria of the game are marked in bold. A strategy profile is a proper coordination equilibrium, if: (a) each agent likes this profile *better than* any other strategy profile he could have reached, given the choice of the other player (Nash Equilibria), and (b) if more than one Nash Equilibria exists in the game [29].

Lewis' definition of convention leads to a fundamental question: How do specific regularities like these *emerge* rather than the others which could have possibly emerged? This question is addressed by adopting a *non-cooperative contest* approach to the game-theoretic models used in analysing social conventions. This means games are interpreted such that players do not have the possibility of making binding commitments or engage in pre-play communication [5]. Now, analysing why specific regularities emerge rather than others is a case of analysing how strategic rational players may come to coordinate their choices repeatedly on particular outcomes of the multiple available proper coordination equilibria in a non-contest game.

Unfortunately, game theory does not provide us with any specific answers to the analysis of the question above. Any proper coordination equilibrium in a non-cooperative contest game, such as above, is by definition, just one out of multiple Nash Equilibria in the game. This means the general Nash Equilibrium selection problem applies. To deal with this, Lewis devised his own theory. He argued that *salience* could be used to explain the emergence and stability of social conventions. According to Lewis, agents "will tend to pick the salient as the last resort, when they have no stronger grounds for choice", and that this tendency is a matter of common knowledge up to some level [29]. Furthermore, once a salient strategy has been chosen, this will set a precedent for future strategies, and as everyone expects almost everyone else to conform to R, stability is ensured.

Lewis argues that conventions are norms, by definition. To quote Lewis,

> Any convention is, by definition, a norm for which there is some presumption that one ought to conform to... one is expected to conform, and failure to conform tends to evoke unfavourable responses from others... These are bad consequences, and my interest in avoiding them strengthens my conditional preference for conforming [29], page 99.

More recently, work in the area of the emergence of social laws has investigated various mechanisms by which a society of agents can reach a global agreement, using only locally available information, in an experimental setting. An agent must decide which convention (social law) to adopt based solely on its interaction with other agents (e.g. feedback from games). So, how do agents decide which conventions to adopt? They make use of a *strategy update function.* Such a function uses feedback from the interactions with other agents in the environment, and based on certain rules, updates the strategy accordingly. So the key problem in the design of a strategy update function is to bring the society to a global convergence, and to do this as efficiently as possible, when all agents in the society make use of this function.

In order to have a better understanding of the problem, consider the following scenario taken from [46], called the *tee shirt game*, where here the tee shirts represent strategies:

> Consider a group of agents, each of which has two tee shirts: one red and one blue. The agents - who have never met previously, and who have no prior knowledge of

each other - play a game, the goal of which is for *all* the agents to end up wearing the same coloured tee shirt. Initially, each agent wears a red or blue tee shirt selected randomly. The game is played in a series of rounds. On each round, every agent is paired up with exactly one other agent; pairs are selected at random. Each pair gets to see the colour of the tee shirt the other is wearing - no other information or communication between the agents is allowed. After a round is complete, every agent is allowed to either stay wearing the same coloured tee shirt, or to swap to the other colour.

The above scenario is a very simplified example where there are only two strategies present. The idea is the same, that all agents should reach a global agreement on which strategy to adopt. In this scenario, this corresponds to all the agents wearing the same colour tee shirt. There are some important points to note about this scenario: No global view is possible here. Agents base their decision of whether to change strategy (tee shirt) purely on their memory of past interactions with other agents.

The possibility of social laws emerging at run-time is investigated in [62, 50, 27], where a number of strategy update functions have been experimented with. These are general strategy update functions, not designed specifically for a particular scenario. For example, two of the strategy update functions experimented with are as follows:

Simple majority: In this strategy update function, agents will change to another strategy if they have observed it being adopted by more agents than their current strategy. If there are multiple strategies being adopted more than the current strategy, the strategy that is being adopted by the *most* agents is chosen.

Highest cumulative reward: In this function, an agent will change to a new strategy if the total payoff obtained by adopting this new strategy is greater than the payoff obtained with the current strategy, in the same finite period of time. For this function to work, an agent must be able to see the payoff resulting from performing each of the strategies. Payoff in this sense would be obtained as in a game setting such as the well known coordination and cooperation games, where the choice of whether to cooperate or defect results in different payoffs, depending on the choice of the other agent.

Many other strategy update functions have been evaluated, but all strategy update functions take the same form, whereby the agent's current strategy is changed to another strategy based on some rule.

In [62], three different performance measures were used in order to compare the strategy update functions. The performance measures are *average convergence at time n*, *average number of strategy changes per interaction*, and *maximum number of strategy changes*.

Convergence here is the fraction of agents using the most popular strategy at time n. If the convergence is 1.0 at a given time in a given run, this means every agent has chosen the same strategy. An optimal strategy update function would reach a convergence in the shortest possible time. The average number of strategy changes per interaction indicates how frequently the agents changed strategy. As changing from one strategy to another generally incurs a cost, lower values for this performance measure are preferable. Finally, the maximum number of strategy changes is simply the maximum number of strategy changes that any agent incurs. Again, lower values of this performance measure would be better.

In [50], the *efficiency of convergence* was tested with the Highest Cumulative Reward strategy update function under various conditions. The conditions tested are the effects of: update frequency, memory restarts, co-varying memory size and update frequency, and finally, limited memory windows. Update frequency refers to how often the update function is used. Here, by update, this does not mean that agents necessarily *change* their strategies, just that the strategy update function is used. It is suggested that updating their strategies at every possible point may cause "thrashing" in the system. However, if the updating of strategies is delayed too much, agents may be prevented from updating their strategies, even when it is appropriate to do so. Memory restarts have the effect of wiping the previous history of the agents. However, the current strategies are not forgotten, as these are the strategies the agents will now start with. The effects of memory size and update frequency have been investigated together to see if these two parameters interact. Finally, limited memory windows is where each agent will only keep a limited window into its past experiences and base the HCR rule on that window alone.

It is apparent that most of the work to date in this area is purely experimental. Many different strategy update functions have been tested, and these have been tested under various conditions. Experimental results have been given and analysed by researchers. These results are arguably not very significant, and the fact that some of the results were unexpected, shows that there is still a lot more research to be done in this area.

3.2 Offline Design

There has been a lot more research into the *offline* design of social laws, and hence this paradigm will receive more coverage in this chapter. This approach looks at the possibility of social laws being designed offline and hardwired into the agents for use at run-time. This approach is often favoured due to its simplicity to implement and the fact that it gives the system designer a greater degree of direct control over the functionality of the system. Also, from a computational point of view, it shifts the computationally hard problem of finding a social law from run-time to design time. It is preferable to spend longer at the design stage for more efficient results at run-time.

It will be useful to introduce an example that can be referred to in each of the approaches. This example is an adaptation of the train example introduced earlier:

Example 3 *In this scenario, there is a train on a circular track, which at one point crosses a road. The place where the track crosses the road (level crossing) is controlled by gates operated by a gate controller agent. If the train moves on to the level crossing while the gates are closed, the train will be in a crash situation, in which it has crashed into a car on the crossing. Also, if the gates close while the train is on the crossing, a crash will occur. We are interested in social laws that can prevent such situations from arising, at the same time as ensuring that the liveness goal that the train will eventually enter the crossing is achieved.*

This system consists of two agents called t and g, where t is the agent representing the train and g is the gate controller agent. The train can be in one of three states: "away" (the initial state of the train), "waiting" (waiting to use the crossing) and "oncrossing" (the train is on the crossing). The gates can only be in two states, either "open" (gates are open to the train) or "closed" (gates are closed to the train). The train has two actions available to it: $move_t$ *and* $idle_t$*. The* $idle_t$ *action is the identity, which causes no change in the train's state. If the train executes a* $move_t$ *action while it is away, then it goes to a waiting state; executing a* $move_t$ *while waiting causes a transition to the oncrossing state; and finally, executing a* $move_t$ *while oncrossing causes a transition to away, as long as the gates were not in the closed state at the same time as the train was in the oncrossing state, as, if this is the case, the train is said to have crashed and is forced to* $idle_t$ *indefinitely. The gates controller also has two actions available to it:* $gates_g$ *and* $idle_g$*. As with the train, this* $idle_g$ *action causes no change in the state of the gates. The* $gates_g$ *action causes the position of the gates to be toggled, i.e., performing* $gates_g$ *when the gates are closed will result in the gates being open and vice versa. Initially, the train is away and the gates are closed.*

The system can be in one of six different states and the overall state of the system at any given time can be characterised in terms of the following five propositional variables: $\{away_t, waiting_t, oncrossing_t, closed_g, open_g\}$*. The overall structure of the level crossing system is illustrated in Figure 3.2; the models of the t and g agents are illustrated in Figure 3.3.*

3.2.1 Moses & Tennenholtz

An artificial social system, as proposed by Moses and Tennenholtz in [35], is defined as *a set of restrictions on agents' behaviours in a multi-agent environment.* Its purpose is to allow agents to act individually, while at the same time ensuring that their behaviour is mutually compatible. It is suggested as an intermediate approach between the two extremes of having

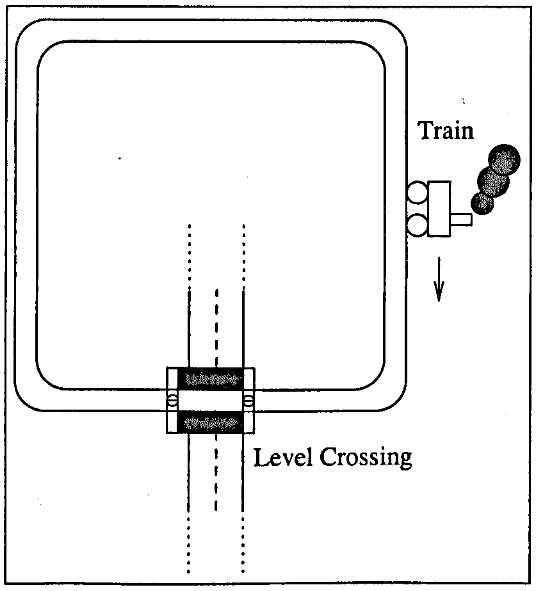

Figure 3.2: The Level Crossing system.

a centralised control entity, and having a purely decentralised approach. In the former case, as the number of agents and tasks grows, the costs in communication, synchronisation, and processing grow dramatically. More importantly, the system depends crucially on one central element. In the latter case, if the cost of conflict is high, or if conflict resolution is difficult, a purely decentralised approach is arguably less desirable. In this section, we will first present the idea of artificial social systems in the simple automata theoretic framework. This will allow the idea to be presented in a simple setting and for the computational problem of finding a social law to be addressed. We go on to present the computational problem of designing a social law, known as the Golden Mean Problem. Finally, we present the deontic modal logic and how it can be used for reasoning about artificial social systems.

Social Automata

The first task in the study of artificial social systems is to introduce the model of multi-agent activity. The model proposed by Moses and Tennenholtz [35] is a system of dependent automata (DA system). Such a system consists of n agents, where $n > 1$. Each agent can be in one of a finite set of local states L_i at any time. The overall configuration of the system is denoted by a tuple of each of the agent's local states $\langle s_1, ..., s_n \rangle$; $C_0 \subseteq L_1 \times \ldots \times L_n$ is a set of initial configurations. The possible actions agent i is able to perform in any state s, is denoted by $A_i(s)$. The

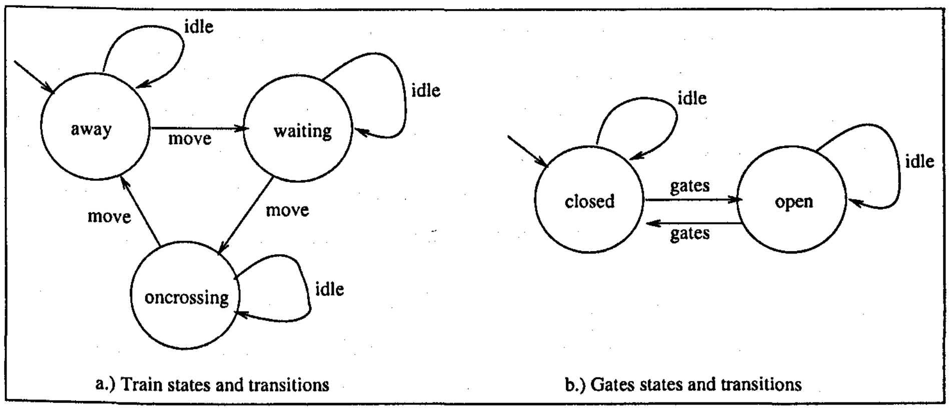

Figure 3.3: States and Transitions.

tuple $\langle \alpha_1, ..., \alpha_n \rangle$ of actions denotes the actions the different agents perform at a given point, and is known as their *joint action*. The set of joint actions is denoted $Ac_1 \times \ldots \times Ac_n$ and $Ac : \bigcup_{i \in Ag} Ac_i$ is the set of actions of the system. A plan for agent i is a function $p(s)$ which for every state s of agent i, returns a particular action α which i will perform in s, where $\alpha \in A_i(s)$. A plan is said to *guarantee* a particular goal if, no matter what the other agents in the system do, by following this plan the goal will be achieved. Goal states are defined as sets $G_i \subseteq L_i$, for $i = 1, ..., n$ and intuitively each goal state $s^g \in G_i$ corresponds to the state of affairs where agent i has achieved some goal denoted by g. A local state s of agent i is called an *initial state* (denoted by s^i) if s is in one of the configurations in C_0. Thus, a DA system is a $(2n+3)$-tuple $\langle L_1, ..., L_n, C_0, Ac, A_1, ..., A_n, \tau \rangle$ where:

- L_i is a set of local states for agent i;
- $C_0 \subseteq L_1 \times ... \times L_n$ is a set of initial configurations;
- Ac is a set of actions;
- $A_i : L_i \rightarrow 2^{Ac}$ is a function which for each local state of agent i, defines the set of actions that agent i can perform in this state; and
- $\tau : (L_1 \times \ldots \times L_n) \times (Ac_1 \times \ldots \times Ac_n) \rightarrow (L_1 \times \ldots \times L_n)$ is a transition function mapping configurations and joint actions into configurations.

A DA system is said to be *social* if, for every initial state s^i and goal state s^g, it is computationally feasible for an agent to devise, on-line, an efficient plan that guarantees to attain the goal s^g state when starting in the initial state s^i [35]. That is to say that no matter what the initial state of the agent is, an agent should be able to reach its goal state regardless of the actions of the other agents and the initial states of the other agents. A standard DA system will be modified by a *social law*. A social law restricts the set of actions that each agent can perform, so a social law, sl, for a DA system S, is a tuple $\langle A'_1, A'_2, ..., A'_n \rangle$ consisting of functions $A'_i : L_i \rightarrow 2^{Ac}$ satisfying $A'_i(s) \subseteq A_i(s)$ for every agent i and state $s \in L_i$. To implement the social law sl, we replace the functions A_i by the restricted functions A'_i, and obtain a new system, S^{sl}.

Example 4 *Recall the level crossing scenario introduced earlier. A social law we wish to formulate is, if the train is on the level crossing, it will not idle there. This is done by forbidding the train from idling, when it is on the crossing. If s_1 is the local state where $oncrossing_t$ is true, then in the system S:*

$$A_t(s_1) = \{idle_t, move_t\}$$

In the constrained system S^{sl}, the function A_t is replaced by:

$$A'_t(s_1) = \{move_t\}$$

Secondly, the computational issues associated with the design of social laws in such a setting need to be addressed. The main computational problem relates to finding a social law, if one exists, so that agents can devise plans from an initial state, that will eventually reach some goal state. Given a DA system $S = \langle L_1, ..., L_n, C_0, Ac, A_1, ..., A_n, \tau \rangle$, with the standard interpretation as defined above, the size of such a system is defined to be $|Ac| + max_i|L_i|$[1]. The problem is to find a social law sl, such that in the new system S^{sl}, given any agent i and any initial state $s_0 \in L_i$ and goal $s_i^g \in G_i$, there exists a plan $p^g : L_i \rightarrow Ac$ that is guaranteed to reach s_i^g starting from s_0.

It is shown that:

Theorem 2 ([35]) *Let $n \geq 2$ be a constant. Given a DA system S with n agents, the problem of finding a social law sl, such that in S^{sl} each agent can devise plans for reaching each goal state from each initial state, if such a law sl exists, is NP-complete.*

The authors of [35] argue that this result should not be interpreted as a highly negative one. Although a problem that is proved to be NP-complete is usually evidence that the problem is hard to solve, and that no efficient algorithms exist for solving it, in this setting there are also positive implications of the result. Firstly, the fact that the problem is NP-complete means that the *verification* of the design process can be done efficiently. This is because the process of designing a social law corresponds to guessing a social law and associated plans, all of which

[1] $max_i|L_i|$ denotes the maximum size out of all the sets of local states for each agent.

can be encoded in polynomial space and verified in polynomial time. Secondly, it is suggested that, the designer's ability to solve NP-hard problems in an offline setting is far greater than the agent's ability to solve such problems online. This means that a trial and error procedure might be feasible in the design stage.

Designing Social Laws

Social laws have the effect of restricting each agent's strategies (actions). By doing so, it may seem that, as individual agents have fewer strategies they can adopt, they will be able to attain fewer goals. However, to an individual agent, his strategies may well be limited, but so are the strategies of the *other* agents in the system. As a result, agents might actually be able to attain more goals than without the social laws in place. In designing social laws, it is important to strike just the right balance between being overly restrictive and overly liberal. On the one hand, if you restrict the strategies of agents too much, they will be able to attain fewer goals due to a lack of options; whereas on the other hand, if the laws are not restrictive enough, many conflicting strategies may be able to be performed, again resulting in the attainment of fewer goals. It is suggested in [35] that *a social system should strike the right balance: It should restrict the allowable behaviours of the various agents enough to serve the different agents in a good manner.* This problem is referred to as the *golden mean problem.*

In [35], a variant of the basic golden mean problem is considered in another basic model called a *one-shot social game.* This model consists of a set S of possible physical strategies identical for all agents and a set G_{soc} of socially acceptable goals (goals which the social system allows the agents to attain). With each goal $g \in G_{soc}$ and agent i we associate a payoff function $u_g(i)$ that assigns to each joint strategy in S^n a value between 0 and 1. It is assumed that social restrictions on the strategies are similar for all agents. It is also assumed that the value of the payoff function for an agent depends only on its current goal and the strategies executed. Hence, the authors claim it is possible to refer to only the payoff functions of the first agent and drop the agent's number from the notation of the payoff function, without loss of generality. Finally, an "efficiency parameter", $0 \leq \epsilon \leq 1$, is given. Now the basic golden mean problem is defined formally as:

Definition 5 (Basic Golden Mean) *Let $n \geq 2$ be a constant. Given a set of n agents, a set S of possible physical strategies, a set G_{soc} of socially acceptable goals, an efficiency parameter ϵ, and for each $g \in G_{soc}$ a payoff function $u_g : S^n \longrightarrow [0,1]$, find a set $\bar{S} \subseteq S$ of "socially acceptable" strategies such that for all $g \in G_{soc}$ there exists $s \in \bar{S}$ such that $u_g(s,\sigma) \geq \epsilon$ for all $\sigma \in \bar{S}^{n-1}$.*

The above definition illustrates the main issues involved in solving a golden mean problem in a game-theoretic sense. The designer of the system will have to disallow some of the physically possible strategies in order to allow certain goals to be efficiently achieved, while at the same time leaving enough strategies available to the agents so that they can achieve their goals in a reasonably efficient manner. This is done by finding a set of socially acceptable strategies such that for all socially acceptable goals, there exists a socially acceptable strategy which together with all combinations of socially acceptable strategies for all other agents in the system, returns a payoff greater than or equal to a given efficiency parameter. The computational complexity of such a problem is given by the following theorem:

Theorem 3 ([35]) *The decision problem corresponding to a basic golden mean problem is* NP*-complete (in the number of strategies and goals). If the number of goals is bounded by a constant, then the problem is polynomial.*

Again, as with Theorem 1, it is argued that this result can be interpreted as a positive one. This result indicates that an off-line trial and error procedure for determining the social laws should be adopted. The second part of the theorem, where we say the number of goals is bounded by a constant, is less important here, as we are mainly concerned with the design stage. However, instances of this problem need to be solved repeatedly in an online setting, to resolve conflicts between the intended actions of the agents.

Logical Reasoning about Social Systems

Now that we have defined exactly what an artificial social system is, in terms of the automata-theoretic approach outlined above, it is time to introduce the logic for reasoning about such systems. This will enable general semantics of artificial social systems to be given and will enable formal logical reasoning about the elements of social systems.

The first stage in presenting the semantics of an artificial social system is to define a general multi-agent system:

Definition 6 *A multi-agent system is a tuple*

$$S = \langle Ag, Q, \sim_1, ..., \sim_n, Ac, Able_1, ..., Able_n, \mathcal{I}, \tau \rangle$$

where:

- $Ag = \{1, ..., n\}$ *is a set of agents;*
- Q *is a set of possible worlds (states);*
- $\sim_i \subseteq Q \times Q$ *are accessibility relations (we assume that* $\sim_i$ *is an equivalence relation for all* $i \in Ag$*);*

- *Ac is a set of primitive individual actions;*
- *$Able_i : Q \rightarrow 2^{Ac}$ is a function that determines the possible physical actions for agent i (in any given world). $Able_i$ is required to satisfy the condition: $Able_i(q) \neq \emptyset$ for all i and q;*
- *$\mathcal{I}$ is a set of possible external inputs for the agents, and;*
- *$\tau : Q \times (Ac \times \mathcal{I})^n \rightarrow Q \cup \{\perp\}$ is a state transition function. This function determines what the next state of the world will be as a function of the actions performed and the inputs received in the current world. This function is not defined, $\tau(q, (\alpha, I)) = \perp$, iff there exists an action α_i in (α, I) such that $\alpha_i \notin Able_i(q)$*

The set of possible worlds, Q, represents all the possible configurations the system can be in. The $\sim_i$ relations are used to capture the agents' knowledge about the possible worlds, as described in Section 2.2. Thus, $(q, q') \in \sim_i$ indicates that the agent i cannot distinguish between the two possible worlds q and q' [16]. The external inputs $\mathcal{I}$ are used to represent messages the agent can receive; for example, an agent could receive goals from some master agent that it wishes the agent to pursue. An agent's goal g is identified with a sub-set $Q_g \subseteq Q$ of worlds. Intuitively, all the worlds in the set Q_g, are worlds where the goal, g, has been achieved. The $Able_i$ function specifies what the agent i is physically able to perform in a given world. The authors suggest that a good way of thinking of this function, is to think of it as the "physical law". Finally, notice that the transition function τ depends on the joint action of all the agents and the joint inputs. This makes it possible for several different worlds to result from one particular agent performing the same action.

Definition 7 *A plan for agent i is a function $Strategy_i : Q \rightarrow Ac$ that satisfies the following:*

1. *If $(q, q') \in \sim_i$, then $Strategy_i(q) = Strategy_i(q')$*
2. *$Strategy_i(q) \in Able_i(q)$ for all $q \in Q$.*

The *Strategy$_i$* function is used to represent the planning process of the agent. Given a world, the *Strategy$_i$* function will determine which action the agent should perform next in accordance with the two conditions. The first condition requires that the action chosen will depend only on agent i's knowledge. In two states that are indistinguishable to i, the same action will be chosen. The second condition requires that the agents are physically capable of performing the actions chosen by the *Strategy$_i$* function.

So far, the agents in the system are constrained only by the *Able$_i$* functions which specify which actions agent i is physically capable of performing in a given state. The authors suggest that this corresponds to a "physical law". The intended meaning of this is what the agents are physically capable of doing. For example, given the limitations on the strength of a human, it would not be "physically" possible for me to lift a car. The agents need to be restricted further by social laws. To understand the difference, consider an example where a car does not stop at a red light. The person driving the car is "physically" capable of driving through the red light, but however, the person is not "socially" *allowed* to drive through the red light. This brings us on to introduce a *normative system*, which restricts the agents based on social laws (or norms):

Definition 8 *A normative system extending a multi-agent system S is defined to be the pair $\mathcal{N} = \left\langle S, \{Legal_i\}_{i \leq n} \right\rangle$, where $Legal_i : Q \rightarrow 2^{Ac}$. Moreover, the functions $Legal_i$ are required to satisfy the following three conditions:*

1. *(epistemological adequacy): $Legal_i(q) = Legal_i(q')$ for all $(q, q') \in \sim_i$;*
2. *(physical adequacy): $Legal_i(q) \subseteq Able_i(q)$ for all i and q;*
3. *(non-triviality): $Legal_i(q) \neq \emptyset$ for all i and q.*

The function $Legal_i(q)$ returns a set of actions that agent i is *allowed* to perform in state q. The epistemological adequacy condition states that the agents will always know what actions they are allowed to perform. The physical adequacy condition requires that agents should be physically capable of performing all actions that are *allowed.* Finally, the non-triviality condition requires that the agents always have at least one action that they are allowed to perform at any one time. This does not mean that the agents are required to do *something* in each world, as there is assumed to be an explicit null action, corresponding to "doing nothing".

Example 5 *As an example, referring back to the level crossing scenario introduced earlier, if we assume the state q_1 to be where the train is on the crossing, the $Legal_i$ function would be as follows:*

$$Legal_t(q_1) = \{move_t\}$$

So, the only legal action the train can perform in state q_1, is the move action. This is to prevent the train from idling on the crossing causing delays for cars.

Given a normative system $\mathcal{N} = \left\langle S, \{Legal_i\}_{i \leq n} \right\rangle$, a strategy is said to be *legal* with respect to $\mathcal{N}$, if in addition to it being a valid strategy in S, all actions chosen by the $Strategy_i$ function will always be legal actions. So now, rather than a strategy having to choose actions which the agent is physically capable of performing in the current state, the actions have to be *legal* ones. This means that condition (2) from the definition of the $Strategy_i$ function is strengthened to: $Strategy_i(q) \in Legal_i(q)$ for all $q \in Q$.

A social system is a specific type of normative system. However, a normative system has nothing in-built to guarantee that nothing bad ever happens. Furthermore, there is nothing in-built to guarantee that something good will always happen. A social system is required to ensure that this is the case. This corresponds loosely to achieving liveness and safety goals. We define a set of "socially acceptable" worlds, Q_{soc}, and these are given by the system designer. We also define a set of "socially acceptable" goals, G_{soc}, which an agent should always be able to attain. The safety goal corresponds to never leaving a set of socially acceptable worlds, Q_{soc}, as long as all agents abide by the rules in place. The liveness goal corresponds to agents *always* being able to attain a socially acceptable goal from the set G_{soc}. Q_0 is the set of initial worlds that the agents can start from. If we assume $Q_0 \subseteq Q_{soc}$, *legally reachable* is defined to be: a world is reachable from a world in Q_0 by following a sequence of steps where all agents abide by the rules in place.

Definition 9 *A social system S consistent with Q_{soc} and G_{soc} will be a normative system extending S that satisfies:*

1. *A world $q \in Q$ is legally reachable only if $q \in Q_{soc}$;*

2. *For every legally reachable world q, if the goal of agent i in q is $g \in G_{soc}$, then there is a legal plan for i that, starting in q, will attain g so long as the other agents act according to the normative (social) system.*

Now that a semantic definition of artificial social systems has been given we are at a stage where we want to be able reason about such systems. In order to do this a propositional modal logic will be used. Basic formulae will be a set Φ of primitive propositions, including distinguished atoms *social* and *legal*, meaning the world is social and the world is legally reachable, respectively. There are also formulae relating to the ability of agents to perform actions in a given world: For every agent $i \in Ag$ and action $\alpha \in Ac$, $Pos_p(i, \alpha)$, $Nec_p(i, \alpha)$, $Pos_s(i, \alpha)$, and $Nec_s(i, \alpha)$ (read respectively as: α is *physically possible* for agent i, α is *physically necessary* for agent i, α is *socially possible* for agent i, and α is *socially necessary* for agent i). The knowledge operators K_i are used to model the agents knowledge about the state of the physical world. However, if the agents are assumed to be acting in accordance with the social laws, a new type of knowledge arises. This is knowledge under the assumption of correct normative behaviour. This is denoted by $B_i^s \varphi$, which says that agent i believes φ under the assumption that the current world is legally reachable.

Formally, the language formed with respect to a set of agents Ag, and a set of primitive propositions Φ, is given by the following grammar:

$$\begin{aligned}\varphi \quad ::= \quad & p \mid \neg\varphi \mid \varphi \vee \varphi \mid K_i\varphi \mid B_i^s\varphi \mid Pos_p(i,\alpha) \\ & Nec_p(i,\alpha) \mid Pos_s(i,\alpha) \mid Nec_s(i,\alpha)\end{aligned}$$

where $p \in \Phi$ is a propositional variable, $\alpha \in Ac$ is an action, and $i \in Ag$ is an agent.

A model for this language is a pair $\mathcal{M} = (S, \pi)$, where S is a social system, and $\pi : \Phi \longrightarrow 2^Q$ is a function associating with every primitive proposition the set of worlds in which it holds. The fact that a formula φ is satisfied in a world q of $\mathcal{M}$, is denoted by $\langle \mathcal{M}, q\rangle \models \varphi$. The definition of this is given by induction on the structure of φ:

(a) $\langle \mathcal{M}, q\rangle \models \varphi$ (for $\varphi \in \Phi$) iff $q \in \pi(\varphi)$.

(b) $\langle \mathcal{M}, q\rangle \models social$ iff $q \in Q_{soc}$.

(c) $\langle \mathcal{M}, q\rangle \models legal$ iff q is legally reachable.

(d) $\langle \mathcal{M}, q\rangle \models Pos_p(i,\alpha)$ iff $\alpha \in Able_i(q)$.

(e) $\langle \mathcal{M}, q\rangle \models Nec_p(i,\alpha)$ iff $Able_i(q) = \{\alpha\}$.

(f) $\langle \mathcal{M}, q\rangle \models Pos_s(i,\alpha)$ iff $\alpha \in Legal_i(q)$.

(g) $\langle \mathcal{M}, q\rangle \models Nes_s(i,\alpha)$ iff $Legal_i(q) = \{\alpha\}$.

(h) $\langle \mathcal{M}, q\rangle \models \neg\varphi$ iff $\langle \mathcal{M}, q\rangle \not\models \varphi$.

(i) $\langle \mathcal{M}, q\rangle \models \varphi \wedge \psi$ iff $\langle \mathcal{M}, q\rangle \models \varphi$ and $\langle \mathcal{M}, q\rangle \models \psi$.

(j) $\langle \mathcal{M}, q\rangle \models K_i\varphi$ iff $\langle \mathcal{M}, q'\rangle \models \varphi$ for every q' satisfying $(q,q') \in \sim_i$.

(k) $\langle \mathcal{M}, q\rangle \models B_i^s\varphi$ iff $\langle \mathcal{M}, q\rangle \models K_i(legal \Rightarrow \varphi) \wedge \neg K_i\neg legal$.

The clause (k) defines the social belief operator B_i^s as "belief as defeasible knowledge". That is to say that if a proposition φ is believed to be true, it can actually be the case that φ is false. $B_i^s\varphi$ can be true, when φ is actually false.

A formula φ is said to be *valid* in $\mathcal{M}$, denoted by $\mathcal{M} \models \varphi$, if $\langle \mathcal{M}, q\rangle \models \varphi$ for all worlds $q \in Q$. The formula φ is *valid*, denoted $\models \varphi$, if it is valid in $\mathcal{M}$ for all models $\mathcal{M}$. The following two key facts in this framework show the relationship between knowledge and social actions: $\models Nec_s(i,\alpha) \Rightarrow K_iNec_s(i,\alpha)$ and $\models Pos_s(i,\alpha) \Rightarrow K_iPos_s(i,\alpha)$. The authors give the following example formulae to demonstrate the power of their framework and claim that these are valid formulae in their language (they are validities):

1. $\models B_i^s(\varphi \vee Nec_s(i,\alpha)) \Rightarrow (B_i^s\varphi \vee B_i^s Nec_s(i,\alpha))$
2. $\models \neg B_i^s \neg Nec_s(i,\alpha) \Rightarrow B_i^s Nec_s(i,\alpha) \vee K_i(\neg legal)$
3. $\models B_i^s[(\varphi \Rightarrow Nec_s(i,\alpha)) \wedge (\neg\varphi \Rightarrow \neg Pos_s(i,\alpha))] \Rightarrow [B_i^s\varphi \vee B_i^s\neg\varphi]$

These are given to show the relationship between social necessity and social belief. The first formula says that if an agent believes that either φ holds or it must perform the action α, then the agent must explicitly believe one of these facts: It either believes φ or it believes that it must perform the action α. This formula is valid since any state where i believes φ or $Nec_s(i,\alpha)$, i must either believe φ or believe $Nec_s(i,\alpha)$, since if this is not the case, then i does not believe φ and i does not believe $Nec_s(i,\alpha)$, which contradicts the assumption we made about the state. The second formula says that if an agent believes that it *might* have to perform an action α, then the agent believes that either it must perform action α or it knows that the current state is not legal. To understand why this formula is valid, consider the state where $\neg B_i^s \neg Nec_s(i,\alpha)$ holds. It must either be the case that i believes $Nec_s(i,\alpha)$ or i knows the current state is not legal. These are the only two ways that $\neg B_i^s \neg Nec_s(i,\alpha)$ could hold. Finally, the third formula says that if a fact φ determines whether or not the agent is allowed to perform the action α in the current world, then the agent must either explicitly believe φ or it must explicitly believe its negation. Consider a state where $\models B_i^s[(\varphi \Rightarrow Nec_s(i,\alpha)) \wedge (\neg\varphi \Rightarrow \neg Pos_s(i,\alpha))]$ holds. For Item 3. to be false, it would mean i not believing φ and i not believing $\neg\varphi$. This means agent i considers φ and $\neg\varphi$ to be possible. However, this cannot be, as $Nec_s(i,\alpha)$ and $\neg Pos_s(i,\alpha)$ cannot be true at the same time.

So far, the logical reasoning has not covered any issues related to agents' goals. As goals are a very important aspect of an artificial social system (and indeed, any multi-agent system), the language is now extended to allow for reasoning about goals. It is assumed that in any given world an agent will only have one distinguished *current goal.* Propositional formulae of the form *current-goal*(i,g) are added to the language for every agent i and goal g. This formula will hold when agent i has the current goal of g.

Earlier on, a goal g of an agent was equated with a set Q_g of worlds where the goal g is satisfied. The authors suggest that a goal, in this sense, can be thought of as a proposition. When the proposition is satisfied, so is the goal. This enables us to reason about satisfaction of goals by talking about when a set T of agents can cause a fact φ to be satisfied. But, it is not enough just to reason about goals being satisfied, as it matters *how* the goals are achieved. The authors introduce an operator for reasoning about goals being achieved in a socially acceptable manner. They call this *social reachability* and this is defined to be: the agents in T have a joint plan consisting of solely socially acceptable actions that is guaranteed to attain φ, so long as all the other agents follow the rules of the social system. This is denoted by *s-reachable*(T,φ).

They also introduce *physical reachability*, where all physically possible actions and plans are considered. This is denoted by $\textit{p-reachable}(T, \varphi)$. It is also possible to reason about what will happen if a certain action will *actually* be executed. To do this, appropriate parameters have to be added to the reachability operators. $\textit{p-reachable}(T, \varphi, do_i(\alpha))$ will denote the fact that $\textit{p-reachable}(T, \varphi)$ holds in cases where agent i executes action α in the current world. This extra parameter to the *p-reachable* operator can be any element in the closure of $do_i(\alpha)$'s under conjunction and negation (e.g. $\textit{p-reachable}(T, \varphi, do_i(\alpha) \wedge do_j(\alpha'))$). Similar parameters can be used in the *s-reachable* operator.

The authors now formulate the two conditions in the definition of a social system in terms of *s-reachable*:

1. $\models \textit{legal} \Rightarrow \neg\textit{s-reachable}(T, \neg\textit{social})$ for all sets $T \subseteq \{1, ..., n\}$ of agents.

2. $\models \textit{legal} \wedge \textit{current-goal}(i, g) \Rightarrow \textit{s-reachable}(i, g)$ for every agent i and goal $g \in G_{soc}$.

The first condition states that in all worlds that are legally reachable, it is not socially reachable to achieve an asocial goal. The second condition states that if the world is legally reachable and the current goal of agent i is g, then g should be socially reachable to i, providing g is a socially acceptable goal. With these operators in place, the designer of the system and its users can reason about actions, goals and their achievement.

Finally, the authors introduce a high-level language for expressing social laws. So far, social laws have been given in terms of a restriction on the $Able_i$ functions, or on the actions that the agents can perform in a given world. Social laws can now be expressed as high-level rules in terms of the language defined above. Consider the following example of a high-level social law:

Example 6 *Imagine that we want to have a rule that states that whenever the circumstances satisfy a fact φ (say, i's house is on fire), then all members of set A must help i to attain ψ (say, put out the fire). We say that $\mathcal{M}$* enforces the rule $\textit{should-help}_{A,i}(\varphi, \psi)$ *iff* $\mathcal{M} \models \varphi \wedge \textit{s-reachable}_{A \cup \{i\}} \psi \wedge \neg\textit{s-reachable}_i \psi$.

This framework thus allows high-level rules to be expressed in a rigorous and concise manner.

3.2.2 Shoham & Tennenholtz

In [48], Shoham and Tennenholtz introduce a general model of social laws in a computational system. Firstly, they define an agent to be in one of a finite number of global states, and to engage in actions which change the state. A synchronous model is adopted, in which agents repeatedly and simultaneously take action, which leads them from their previous state to a new one. The agents have a repertoire of actions to choose from. A problem arises when trying to define the transition functions for agents. This is due to the fact that the change in an agent's state is not a direct result of the action the agent chose in its previous state. The change in an agent's state is also determined by the performance of actions by the *other* agents in the system. One approach would be to define the transition function as a mapping from the states and actions of *all* the agents in the system to the new states of all agents. Another approach would be to define the transition function for each of the agents, but make the function non-deterministic to account for the effects of the other agents in the system. In this framework, the idea of a *social law* is introduced as an intermediate approach between the two. In the former case, the transition function produces the most specific prediction, whilst in the latter case it produces the most general prediction. With social laws present, the transition function also takes the set of constraints as an argument and produces a prediction of the possible next states of the agent. The constraints specify which states a given action is prohibited in. Finally, an assumption of homogeneity is made. It is assumed that the set of states and actions are common to all agents. However, it is not assumed that the agents will necessarily be in the same state at the same time, nor that they will take the same action when in the same state. Also, it is assumed that the same constraints apply to all agents.

The Formal Model

First, we will formally define a social law:

Definition 10 *Given a set of states Q, a first order language $\mathcal{L}$ (with an entailment relation $\models$), and a set of actions Ac, a* constraint *is a pair (α, φ) where $\alpha \in Ac$ and $\varphi \in \mathcal{L}$ is a sentence. A* social law *is a set of constraints (α_i, φ_i), at most one for each $\alpha_i \in Ac$.*

The language $\mathcal{L}$ will be used to describe what is true and false in different states. Given a state $q \in Q$ and a sentence $\varphi \in \mathcal{L}$, q might satisfy or not satisfy φ. We denote the fact that q satisfies φ by $q \models \varphi$. Given a constraint pair (α, φ), intuitively φ will correspond to every situation in which the action α should be prohibited.

Given a pair of social laws, sl_1 and sl_2, we denote by $sl_2 < sl_1$ the fact that for every $(\alpha_i, \varphi_i) \in sl_2$ there exists $(\alpha_i, \varphi_j) \in sl_1$ such that $\varphi_i \models \varphi_j$. Intuitively, it will mean that sl_1 is more restrictive than sl_2. So everything that sl_2 restricts, sl_1 will restrict, and possibly more.

Next, we formally define a social agent:

Definition 11 *A social agent is a tuple $(Q, \mathcal{L}, Ac, SL, \tau)$ where Q, $\mathcal{L}$, Ac are as above, SL a set of social laws, and τ is a total transition function $\tau : Q \times Ac \times SL \to 2^Q$ such that:*

- *For every $q \in Q, \alpha \in Ac, sl \in SL$, if $q \models \varphi$ holds and $(\alpha, \varphi) \in sl$ then $\tau(q, \alpha, sl) = \emptyset$, the empty set.*

- *For every $q \in Q, \alpha \in Ac, sl_1 \in SL, sl_2 \in SL$, if $sl_2 < sl_1$ then $\tau(q, \alpha, sl_1) \subseteq \tau(q, \alpha, sl_2)$.*

So in this framework, the transition function takes the set of social laws as an argument and uses them along with the state and action of the agents to form the set of possible next states. There are two conditions on the transition function. The first says that if the situation described by φ is satisfied in the current state and the action, α, is forbidden in the current situation by one of the constraints in sl, then the transition function will return the empty set, thus it is not defined for forbidden actions. The second condition says that for two social laws, sl_1 and sl_2, if sl_1 is more restrictive than sl_2, then the states resulting from transitions with sl_1 will be a sub-set of the states resulting from transitions with sl_2, for all states and actions.

Finally, a social multi-agent system is defined as follows:

Definition 12 *A social multi-agent system is a collection of social agents which share the sets of states, the language for describing states, the set of potential actions, the set of potential social laws, and the transition function.*

In order to illustrate how social laws are formulated in this framework, we give the following example:

Example 7 *Recall the level crossing scenario introduced earlier. We wish to formulate that, if the gates are closed, the train will not cross the level crossing. In this framework, this could be expressed as follows:*

$\varphi_1 = waiting_t \wedge closed_g$

The above social law will be as follows:

$$sl_1 = \{(move_t, \varphi_1)\}$$

So this social law forbids the agents from performing the $move_t$ action when the train is waiting to use the crossing and the gates are closed. This is clearly to stop the train from crashing through the gates and onto the crossing, possibly colliding with cars using the crossing.

We also wish to ensure that the train will eventually enter the crossing. To ensure this liveness goal, we could implement the following social law:

$$sl_2 = \{(idle_g, \varphi_1)\}$$

This prevents the gates from idling when the train is waiting to use the crossing. So in the next state, the gates will open to allow the train to enter the crossing.

After looking at this example, we can see similarities between the Moses and Tennenholtz framework outlined in the previous section. In their social systems the restrictions are imposed via the $Legal_i(q)$ functions, where $Legal_i(q)$ returns the set of actions i is allowed to perform in state q. So they specify which actions are *allowed*, whereas here the social laws specify which actions are *forbidden*. However, the notion of a social law is the same in both instances.

The Computational Problem

Once we have chosen the social law to use and implemented it, the system reduces to a standard one with all transitions that are incompatible with the law removed. So, the computational problem is to select a social law that, given the social multi-agent system, will induce a 'good' standard system. To define a 'good' system, a subset of the set of states, called *focal states*, is introduced. The social law should ensure that for each agent, given two focal states, it is able to construct a plan guaranteed to move it from one state to the other - *no matter what the other agents do*. Intuitively, this corresponds to the agent being able to achieve its goals.

An agent's *legal plan* is a set of decisions about which action to perform in each state, such that the agent abides by the laws in place. It is defined as follows:

Definition 13 *Given a social agent $\langle Q, \mathcal{L}, Ac, SL, \tau \rangle$ and a social law $sl \in SL$, a* legal plan *is a total function $DO : Q \rightarrow Ac$ such that if $(\alpha, \varphi) \in sl$ and $q \models \varphi$ holds, then $DO(q) \neq \alpha$. An* execution *of the plan from a state q_0 is a sequence of states $q_0, q_1, q_2, \ldots$ such that $q_{i+1} \in \tau(q_i, DO(q_i), sl)$.*

The above definition of a plan requires the agent to perform an action at every step. However, there is an explicit null action, corresponding to doing nothing, thus leaving the states unchanged. In some circumstances this null action may be prohibited by the social law, meaning that "something" has to be done in this state. For example, if someone's house was on fire, doing nothing would be prohibited!

We now give the definition for a *useful social law*:

Definition 14 *Given a social multi-agent system and a subset F of the set of states (the focal states), a* useful law *is a law for which, given any $q_1, q_2 \in F$, there exists a legal plan such that* every *execution of that plan in q_1 reaches q_2.*

A precise computational problem can now be phrased:

Definition 15 (The Useful Social Law Problem (USLP)) *Given a social multi-agent system and a set of focal states, find a useful law if one exists, or, if no such law exists, announce that this is the case.*

The computational complexity of the above problem can now be analysed. First, it is necessary to define some precise details of the model it will be tested under. The number of states in the representation of an agent is assumed to be finite and is denoted by n, and we will measure the computational complexity as a function of n. The total size of an agent's representation is polynomial in n. It will also be assumed that each property of the form $q \models \varphi$, and of the form $sl_1 < sl_2$ can be efficiently verified.

The result is shown in the following theorem:

Theorem 4 *The USLP is* NP*-complete.*

The theorem shows that the USLP is intractable. However, as the USLP is computed offline, this result is not entirely negative, and a lower complexity could be achieved by introducing several restrictions on the structure of agents.

The first restriction made is on the number of actions that can be performed in each state. For each state q, the number of transitions which might change q is bounded by $O(log(n))$. This restriction says that the number of actions an agent might perform at any given state is small relative to the total number of states, while the quality of the information about constraints which might be relevant to the effects of a particular action in a particular state is still relatively high [48].

Definition 16 (The Bounded Useful Social Law Problem (BUSLP)) *Given a social multi-agent system where the number of transitions which might change a particular state is bounded by $O(log(n))$, and a set of focal states, find a useful law if one exists, or, if no such law exists, announce that this is the case.*

With this restriction alone, the BUSLP is NP-complete, as was the USLP. So there are further restrictions needed to reduce the complexity of the BUSLP. The conditions under which the BUSLP becomes tractable are roughly as follows:

1. The number of focal states is bounded by a constant.
2. Plans must be deterministic.
3. The plan must be short.

Theorem 5 *The BUSLP is polynomial if restrictions 1, 2, and 3 hold; if we drop any of these restrictions (and do not add additional ones) then the BUSLP is* NP*-complete.*

3.2.3 Minimality & Simplicity

In Fitoussi and Tennenholtz [17], two basic criteria for selecting among alternative useful social laws are presented. Here, the informal definition of a useful social law is given in much the same way as in the previous section. A useful social law is one which by following the social laws will lead to the agent's goals being achieved. The suggested criteria for selecting amongst useful social laws are minimality and simplicity.

The basic idea behind *minimality* is to reduce the set of constraints that the social laws impose on the system as much as possible so that following the social law still guarantees the goal state. In order to do this, we need to determine whether this is the *minimal* set of constraints needed to be obeyed by the agents in order to guarantee the goal specification. The idea behind this is that by constraining the agents *just enough* to reach their goal specification, we provide the agents with maximal individual flexibility.

Before giving any formal definitions, it is necessary to understand how two different laws will be compared. Given two useful social laws sl_1 and sl_2, we say that sl_2 is smaller than sl_1 if the set of behaviours induced by strategies consistent with sl_1 is included in the set of behaviours induced by strategies consistent with sl_2. So the set of behaviours induced by strategies consistent with sl_2 includes those induced by strategies consistent with sl_1, and more. This is due to sl_2 being less restrictive; sl_2 rules out less behaviours than sl_1. If we think of a social law sl as a set of constraints, as defined in [48], then sl_2 has fewer constraints than sl_1. Informally, the authors of [17] define a minimal social law as the following: a useful social law sl^* is minimal (and optimal) for some system specification, if and only if, for any other useful social law sl, sl is not smaller than sl^*. So this is basically saying that given a useful social law, this useful social law is minimal if no other useful social law exists that is smaller.

We will now give some formal definitions to formulate the notions of social laws and minimality in the framework of a general strategic model. The following definitions are presented for an environment with two agents.

Definition 17 *An environment E is a tuple $\langle Ag, \Sigma_1, \Sigma_2 \rangle$, where $Ag = \{1, 2\}$ is a set of agents and Σ_i is a set of strategies available to agent i.*

So, the above definition says that an environment consists of a set of agents and a set of strategies for each agent. The agents are assigned goals which they must try to attain using the strategies available to them. As well as these goals, there are also system-level goals that should always be guaranteed. These are defined as follows:

Definition 18 *In an environment $E = \langle Ag, \Sigma_1, \Sigma_2 \rangle$ a goal g is a subset of the Cartesian product over the agents' strategy spaces, i.e., $g \subseteq \Sigma_1 \times \Sigma_2$.*

The above definition presents the idea that a system-level goal is achieved as a result of the joint action of all the agents in the environment. As such, a goal in this sense is defined to be a sub-set of the Cartesian product over the agents' strategy spaces.

A formal definition of a *useful* social law is now given. But, first we need to introduce two different types of goals. These are the well known liveness and safety goals, used in many areas of Computer Science [32]. Liveness corresponds to ensuring that something "good" will eventually happen, while safety corresponds to ensuring that "nothing bad" will ever happen. Let G_i denote the set of liveness goals for agent i. These are the goals we wish agent i to attain. Let G_{safe} be a set of safety goals. These are the goals that should always be attained. Not attaining one or more goals in the set G_{safe} intuitively corresponds to a bad state of affairs. A useful social law is a social law that ensures safety and always enables the agents to achieve their liveness goals. This is given more formally as:

Definition 19 *Given an environment $\langle Ag, \Sigma_1, \Sigma_2 \rangle$, and given the sets of goals G_1, G_2, and G_{safe}, a social law is a set of forbidden strategies, $SL = \langle \overline{\Sigma_1}, \overline{\Sigma_2} \rangle$, such that $\overline{\Sigma_1} \subseteq \Sigma_1$ and $\overline{\Sigma_2} \subseteq \Sigma_2$. SL is useful if:*

1. *for every goal $g_{1_i} \in G_1$ there exists $\sigma_{1_i} \in \Sigma_1 \setminus \overline{\Sigma_1}$ such that for all $\sigma_2 \in \Sigma_2 \setminus \overline{\Sigma_2}$ we have $(\sigma_{1_i}, \sigma_2) \in g_{1_i}$.*

2. *for every goal $g_{2_i} \in G_2$ there exists $\sigma_{2_i} \in \Sigma_2 \setminus \overline{\Sigma_2}$ such that for all $\sigma_1 \in \Sigma_1 \setminus \overline{\Sigma_1}$ we have $(\sigma_1, \sigma_{2_i}) \in g_{2_i}$.*

3. *for every goal $g_j \in G_{safe}$ and for all $\sigma_1 \in \Sigma_1 \setminus \overline{\Sigma_1}$, $\sigma_2 \in \Sigma_2 \setminus \overline{\Sigma_2}$, we have that $(\sigma_1, \sigma_2) \in g_j$.*

So, a useful social law should allow the agents to achieve their liveness goals, while at the same time all safety goals should be maintained. This is defined formally in the above definition of a useful social law. A social law is a set of restrictions. These are given as a subset of the agent's strategies, that are disallowed. The first condition says that for every liveness goal of agent 1 there is a strategy from the set of restricted strategies, such that for all restricted strategies of agent 2, when these strategies are executed, this corresponds to one of agent 1's liveness goals being achieved. The second condition is as condition 1, but for liveness goals of agent 2. Finally, the third condition states that for all safety goals and for all the restricted strategies of agents 1 and 2 a safety goal will always be achieved.

Now we are at a stage where we can formally define a (useful) minimal social law:

Definition 20 *Consider an environment with a specification of liveness and safety goals. A useful social law $SL = \langle \overline{\Sigma_1}, \overline{\Sigma_2} \rangle$ is* minimal *if there is no other useful social law $SL' = \langle \Sigma'_1, \Sigma'_2 \rangle$ that satisfies $\Sigma'_i \subseteq \overline{\Sigma_i}$ for all i.*

It is now important to look at some of the computational aspects relating to the synthesis of minimal social laws. In order to do this, we first need to present the computational model in which the complexity issues can be investigated.

First a multi-agent system will be defined as follows:

Definition 21 *A (two-agent) system is a tuple*

$$S = \langle L_1, L_2, C_0, Ac, A_1, A_2, \tau \rangle$$

where

- *L_i is a finite set time-stamped states for agent i (i.e., a combination (s,t) of a state and a positive integer);*
- *$C_0 \subseteq L_1^{t=0} \times L_2^{t=0}$ is a set of initial configurations drawn from the agents' states with time-stamp $t = 0$. We refer to $L_1 \times L_2$ as the set of possible system configurations.*
- *Ac is a finite set of actions;*
- *A_i is a function from L_i to 2^{Ac} that determines the actions that are physically possible for agent i (as a function of its state);*
- *τ is a (partial) transition function $\tau : L_1 \times L_2 \times Ac \times Ac \rightarrow L_1 \times L_2$ such that if a state l in a configuration c is mapped to l' in configuration c' under the function τ then the time-stamp associated with l and the time-stamp associated with l' are consecutive integers (i.e., a joint action will lead an agent from a state with time-stamp t to a state with time-stamp $t+1$).*

Definition 22 *A plan for agent i is a total function from L_i to Ac, such that the action prescribed to agent i by the plan at any state $s \in L_i$ is in $A_i(s)$.*

The above definition says that a plan is a function from states to actions, but also, the actions chosen must be physically possible. An execution of a plan $\mathcal{P}$ by agent i is a sequence $s_0, s_1, \ldots, s_k$ of states in L_i.

Definition 23 *A liveness goal for agent i is associated with a subset of L_i.*

A liveness goal is said to be achieved, if one of the states in the set of liveness goals is reached.

Definition 24 *A safety goal is associated with a subset of $L_1 \times L_2$.*

A safety goal g_{safe} is said to be achieved if the system only reaches configurations in the set of safety goals, g_{safe}.

As we have seen throughout this chapter, a social law is a set of constraints that restrict the plans available to the agents. In this model, this is formally defined as follows:

Definition 25 *Given a system S, a social law sl in S consists of functions $\langle A'_1, A'_2 \rangle$, for agents 1 and 2 respectively, where A'_i is a function from L_i to 2^{Ac} that defines the subset of actions prohibited for agent i in each state ($A'_i(s) \subseteq A_i(s)$ for every agent i and state $s \in L_i$).*

This definition is very similar to the definition given earlier in [35]. In much the same way, with the social law sl in place, a new social system is formed S_{sl}, where the A_i functions are replaced with the restricted functions $A_i(s) \setminus A'_i(s)$. Thus $A'_i(s)$ functions correspond to the set of actions forbidden in state s.

Definition 26 *A social law sl in S is useful if the system guarantees each safety goal (regardless of the law-abiding strategies chosen by the agents), and if, for every liveness goal s_{goal} of agent i, there exists a plan $\mathcal{P}$ in S_{sl} that guarantees s_{goal}.*

SL_S is defined to be the set of useful social laws for the system S. A partial order $\prec$ is defined on SL_S: given two social laws $sl_1 = \left\langle A_1^{sl_1}, A_2^{sl_1} \right\rangle$ and $sl_2 = \left\langle A_1^{sl_2}, A_2^{sl_2} \right\rangle$ in SL_S, we say that $sl_1 \prec sl_2$ if $A_i^{sl_1}(s) \subseteq A_i^{sl_2}(s)$ for all i and all $s \in L_i$, with at least one strict inclusion for one s and i.

The notion of a minimal useful social law can now be formally defined:

Definition 27 *A minimal social law sl_i is a useful social law such that there is no useful social law sl_j in SL_S, $sl_j \neq sl_i$, that satisfies $sl_j \prec sl_i$.*

Consider the following example, in order to illustrate the idea of a minimal social law in the level crossing scenario defined earlier:

Example 8 *Here we have a system*

$$S = \langle L_T, L_G, C_0, Ac, A_t, A_g, \tau \rangle$$

where the agent t represents the train and the agent g represents the gate controller agent. We wish to ensure that the train never idles on the level crossing. If we assume s_1 to be the local state of the train where it is on the crossing, then the set of prohibited actions needed in order to achieve this is given by:

$$A'_t(s_1) = \{idle_t\}$$

This is a minimal social law, as there does not exist a less restrictive social law that achieves the goal of ensuring that the train does not idle on the crossing.

Finally, the authors consider the automatic synthesis of minimal social laws. This is achieved by implementing an algorithm which roughly starts from a useful social law and decrements the set of constraints. The question posed for each constraint is something like this: Given a system, an appropriate useful social law, and a pair (s, α) of a state s and an action α, where α is prohibited in state s for agent i, can we allow i to take the action α in s and still remain with a useful social law? The complexity of this question is given by the following theorem:

Theorem 6 *Given a system S, and a useful social law sl that prohibits action α in state s of an agent i, deciding whether by allowing α in s we get a useful social law, is* NP-*hard.*

This result led the authors to consider a special class of systems in which an agent's basic goal is to follow a predefined plan $\mathcal{P}_i$. The authors go on to show that an efficient incremental algorithm exists for this class of systems in which minimal social laws are computed in polynomial time.

Simplicity is an alternative approach to minimality for choosing between different useful social laws. Simplicity is a different concept, where the need for agents to rely heavily on their *sensoring capabilities* in order to comply with the laws is relaxed. The idea behind this approach is that some agents may only be able to follow simple laws due to their sensoring or other capabilities. Simplicity also reduces the sensitivity of the system to changes in the agent's capabilities. Also, it is likely to be faster for agents to learn simple laws, and the representation of simple laws is likely to be more succinct.

Now we formally introduce the notion of simplicity. The computational model is the same as defined for minimality. For convenience, the authors require that the set of actions prescribed by a law at state s_i of agent i includes at least one action (this could be a null action). The aim is to measure the complexity of useful social laws in order to compare them. First we give the definition for how a social law defines a partition over the state space:

Definition 28 *Consider a two-agent system $S = \langle L_1, L_2, Ac, A_1, A_2, \tau \rangle$ and a useful law sl for S such that $sl = \{A'_1, A'_2\}$. The partition P_i of the state space S_i of agent i is the set of equivalence classes over S_i under the following equivalence relation R: for two states s_1, s_2 in S_i, $R(s_1, s_2)$ iff $A_i(s_1) \setminus A'_i(s_1) = A_i(s_2) \setminus A'_i(s_2)$.*

The above definition partitions the state space into equivalence classes under an equivalence relation R. This relation will place two states s_1 and s_2 in the same element of the partition P if and only if the set of actions allowed by sl in s_1 is exactly the same as the set of actions allowed at s_2. Intuitively, this means that s_1 and s_2 are indistinguishable. Now we show how the social laws are compared:

Definition 29 *Given two useful social laws sl and sl′ for a (two-agent) system S, each one with corresponding partitions (for each agent respectively)* (P, Π) *and* (P', Π')*, sl′ is simpler than sl if for every element* $P_k \in P$ *and* $\Pi_r \in \Pi$*, there exist k′ and r′ such that* $P_k \subseteq P'_{k'}$ *(respectively* $\Pi_r \subseteq \Pi'_{r'}$*), with strict inclusion for at least one k (or r).*

The above definition shows how we can compare two social laws in terms of how simple they are. Now that the social laws are viewed as defining a partition over the state space, the measure of complexity of a social law is related to the complexity of the partition it defines.

Finally, we note that finding a simple social law is not a simple task, illustrated by the following theorem:

Theorem 7 *Given a (two-agent) system S with a single initial configuration* c_0 *and where each agent has a single (liveness) goal* g_i*, deciding whether there exists in S a useful social law with single-element partitions in the agents' state space is* NP*-hard.*

3.3 Summary

Existing research in the area of social laws has been presented in this chapter. As it stands, not a lot of research has been carried out in the online design of social laws. Research in this sub-field of social laws has mainly addressed the design of strategy update functions. As explained previously, the strategy update function is used to bring the society to a global agreement on which conventions to adopt. In this way, social laws *emerge* from within the society of agents. Such strategy update functions have been designed and tested by researchers such as Shoham, Tennenholtz, Wooldridge etc. Various experiments have been carried out in order to test which functions are the most efficient in making the society converge on a strategy. At this stage in the research, the only findings that have been made are experimental results, and these are arguably not very significant.

Our area of research is primarily concerned with the offline design of social laws. There has been far more research in this area, hence why it has received more coverage in this chapter. We have covered the two main frameworks of social laws and an approach for choosing between useful social laws. There is also a body of work related to the social laws paradigm, known as mechanism design [65]. It has similar aims to the work on social laws, in that it is a method to avoid costly coordination techniques such as planning or negotiation. As with the offline design of social laws, the need for run-time coordination will be reduced. Most similar is the way a set of constraints is designed such that a rational agent will comply with the norms. However, mechanism design is also concerned with giving agents incentives to comply with the norms, rather than the system designer having full control over the agents.

The concept of a social law is very similar in all of the frameworks, in that a social law imposes a restriction on the actions an agent can perform in a specific state. In the Moses and Tennenholtz framework this restriction is imposed upon the A_i functions. The A_i functions return a set of actions which an agent can perform in a given state. When a social law is implemented these functions are replaced with the restricted functions, A'_i, which satisfy $A'_i(s) \subseteq A_i(s)$ for every agent i and every state s. In the Shoham and Tennenholtz framework, this restriction is imposed on the transition function. The transition function τ is defined to be $\tau : Q \times Ac \times SL \longrightarrow 2^Q$, where Q is a set of global states, Ac is a set of actions and SL is a set of social laws. The function has a condition that if the current state matches the state described by the social law and the action specified matches that in the social law, then the action cannot be performed and the resulting set of states will be the empty set. So in this sense, both of the frameworks have the same basic underlying principles. However, each of the frameworks use different logics in order to reason about social laws. In the Moses and Tennenholtz framework a high-level language (modal logic) is constructed in order to reason about these laws, which includes knowledge and belief operators, and distinguished atoms and formulae for reasoning about various normative properties. The Shoham and Tennenholtz framework does not employ any high-level logic to reason about such laws, just first-order logic to describe a certain situation.

In terms of computational complexity, both of the frameworks are very similar. The computation problem that is addressed in both of the frameworks is (loosely), given a multi-agent system, find a social law such that, implementing this social law allows each of the agents to reach their goal state from their initial state, if such a law exists. This problem is NP-complete in both of the frameworks.

Finally, we have looked at minimality and simplicity; two alternative approaches for choosing between useful social laws. Minimality aims to give the agents the maximum freedom possible, by reducing the set of constraints imposed as much as possible, so that the social laws still guarantee the goals are attained. So, as such, a minimal social law is a law which constrains the agents *just enough* to still achieve their goals. Simplicity is a different concept, where the aim is to reduce the need for agents to rely heavily on their sensoring (or other) capabilities in order to follow the laws. Social laws for both minimality and simplicity are given in a very similar way to the Artificial Social Systems framework, whereby restrictions are placed on the A_i functions, which return the set of actions agent i is allowed to perform in a given state. The complexity of finding a minimal social law is NP-complete, as is the problem of finding a simple social law.

Part III

A Framework for Social Laws in MAS

Chapter 4

Action Based ATSs and ATL

In this chapter we introduce the semantic structures that our framework for social laws is based upon and define ATL over these structures. We extend these structures in Chapter 6 and then further extend them in Chapter 7.

4.1 Action Based ATSs

In Section 2.1 we introduced *Alternating Transition Systems* (ATSs), as these are the structures that underpin ATL. As the notions of *action* and *action pre-condition* play such a key role in our framework, we find it convenient to work with another version of ATSs, in which actions and pre-conditions are first-class citizens. We refer to these structures as *Action-based Alternating Transition Systems* (AATSs). We first assume that the systems of interest to us may be in any of a finite set Q of possible *states*, with some $q_0 \in Q$ designated as the *initial state*. Systems are populated by a set Ag of *agents*; a *coalition* of agents is simply a set $G \subseteq Ag$, and the set of all agents is known as the *grand coalition*. Now, each agent $i \in Ag$ is associated with a set Ac_i of possible actions, and we assume that these sets of actions are pairwise disjoint (i.e., actions are unique to agents). We denote the set of actions associated with a coalition $G \subseteq Ag$ by Ac_G, so $Ac_G = \bigcup_{i \in G} Ac_i$. A *joint action* for a coalition G is a tuple $\langle \alpha_1, \ldots, \alpha_k \rangle$, where $\alpha_i \in Ac_i$, for each $i \in G$. We denote the set of all joint actions for coalition G by J_G, so $J_G = \prod_{i \in G} Ac_i$. Given an element j of J_G and agent $i \in G$, we denote i's component of j by j_i.

An *Action-based Alternating Transition System* – hereafter referred to simply as an AATS – is an $(n+7)$-tuple

$$S = \langle Q, q_0, Ag, Ac_1, \ldots, Ac_n, \rho, \tau, \Phi, \pi \rangle$$

where Q, Ag, Φ and π are defined as before by Definition 1 in Section 2.1 and the following components are either new or modified:

- $q_0 \in Q$ is the designated *initial state* of the system;
- Ac_i is a finite, non-empty set of *actions*, for each $i \in Ag$, where $Ac_i \cap Ac_j = \emptyset$ for all $i \neq j \in Ag$;
- $\rho : Ac_{Ag} \longrightarrow 2^Q$ is an *action precondition function*, which for each action $\alpha \in Ac_{Ag}$ defines the set of states $\rho(\alpha)$ from which α may be executed;
- $\tau : Q \times J_{Ag} \rightarrow Q$ is a partial *system transition function*, which defines the state $\tau(q,j)$ that would result by the performance of j from state q – note that, as this function is partial, not all joint actions are possible in all states (cf. the pre-condition function above).

We require that AATSs satisfy the following two coherence constraints:

1. *Non-triviality [35].* Agents always have at least one action:

$$\forall q \in Q, \forall i \in Ag, \exists \alpha \in Ac_i \text{ s.t. } q \in \rho(\alpha)$$

2. *Consistency.* The ρ and τ functions agree on actions that may be performed:

$$\forall q \in Q, \forall j \in J_{Ag}, (q,j) \in \text{dom } \tau \text{ iff } \forall i \in Ag, q \in \rho(j_i)$$

We denote the set of sequences over Q by Q^*, the set of non-empty sequences over Q by Q^+, and the set of infinite sequences over Q by Q^ω. Note how the transition function $\tau(q,j)$ is defined differently here to how it was defined in ATSs in Section 2.1. As we have actions here, the transition function takes a state and a joint action of all the agents and returns the resultant state the system will reach after the joint action has been performed. The transition function in ATSs is defined as a choice function whereby given an agent and a state, the function returns the set of possible choices available.

Given an agent $i \in Ag$ and a state $q \in Q$, we denote the *options* available to i in q – the actions that i may perform in q – by $options(i,q)$:

$$options(i,q) = \{\alpha \mid \alpha \in Ac_i \text{ and } q \in \rho(\alpha)\}.$$

We define a strategy in a different way from how it was defined for ATSs in Section 2.1. Now that we have actions, we can explicitly refer to these and say that a strategy maps sequences of states to actions. Formally, a *strategy* for an agent $i \in Ag$ is a function:

$$\sigma_i : Q^+ \to Ac_i$$

which must satisfy the *legality* constraint that $\sigma_i(\overrightarrow{s}; q) \in options(i, q)$ for all $\overrightarrow{s} \in Q^*$ and $q \in Q$. We also have *memoryless strategies* here, defined in exactly the same way as in Section 2.1: If $\sigma_i(\overrightarrow{s}; q) = \sigma_i(\overrightarrow{s}'; q)$ for all $q \in Q$ and $\overrightarrow{s}, \overrightarrow{s}' \in Q^*$, we simply write $\sigma_i(q)$. For simplicity, we only refer to memoryless strategies in the remainder of this thesis.

A *strategy profile* for a coalition $G = \{1, \ldots, k\} \subseteq Ag$ is a tuple of strategies $\langle \sigma_1, \ldots, \sigma_k \rangle$, one for each agent $i \in G$. We denote by Σ_G the set of all strategy profiles for coalition $G \subseteq Ag$; if $\sigma_G \in \Sigma_G$ and $i \in G$, then we denote i's component of σ_G by σ_G^i. Given a strategy profile $\sigma_G \in \Sigma_G$ and state $q \in Q$, let $out(\sigma_G, q)$ denote the set of possible states that may result by the members of the coalition G acting as defined by their components of σ_G for one step from q. This function is similar to $out(\sigma_G, q)$ defined in Section 2.1, but defined differently, as now we refer to actions:

$$out(\sigma_G, q) = \{q' \mid \tau(q, j) = q' \text{ where } (q, j) \in \text{dom}\, \tau \text{ and } \sigma_G^i(q) = j_i \text{ for } i \in G\}$$

Notice that, for any grand coalition strategy profile σ_{Ag} and state q, the set of possible states $out(\sigma_{Ag}, q)$ will be singleton.

Given a strategy profile σ_G for some coalition G, and a state $q \in Q$, we define $comp(\sigma_G, q)$ to be the set of possible runs that may occur if every agent $i \in G$ follows the corresponding strategy σ_i, starting when the system is in state $q \in Q$. That is, the set $comp(\sigma_G, q)$ will contain all possible q-computations that the coalition G can "enforce" by cooperating and following the strategies in σ_G. This function is defined exactly as in Section 2.1, but here we refer to a different version of $out(\sigma_G, q)$:

$$comp(\sigma_G, q) = \{\lambda \mid \lambda[0] = q \text{ and } \forall u \in \mathbb{N} : \lambda[u+1] \in out(\sigma_G, \lambda[u])\}.$$

Again, note that for any state $q \in Q$ and any grand coalition strategy σ_{Ag}, the set $comp(\sigma_{Ag}, q)$ will be a singleton, consisting of exactly one infinite computation.

Now we revisit the train example, introduced earlier in Section 2.1. This example is identical to before, but now we refer explicitly to the actions available to the agents.

Example 9 *The scenario is exactly as described earlier. But now, each train $i \in \{E, W\}$ has two actions available: $Ac_i = \{move_i, idle_i\}$. The $idle_i$ action is the identity, which causes no change in the train's state (i.e., it stays where it is). If a train i executes a $move_i$ action while it is $away_i$, then it goes to a $waiting_i$ state; executing a $move_i$ while $waiting_i$ causes a transition*

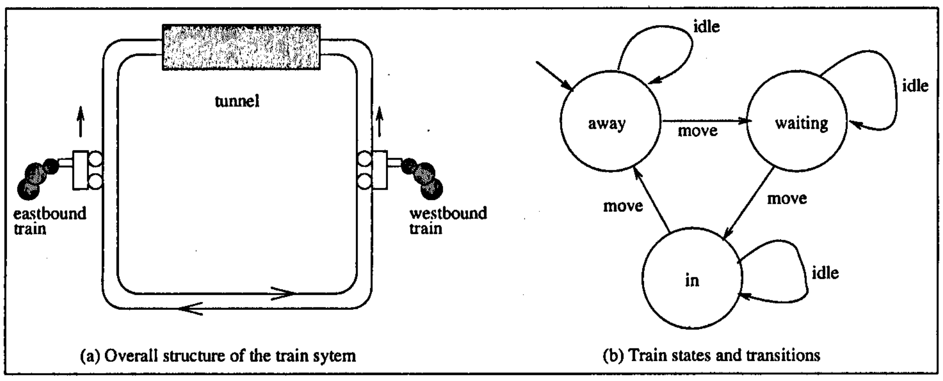

Figure 4.1: The train system.

to an in_i state; and finally, executing a $move_i$ while in_i causes a transition to $away_i$ as long as the other train was not in the tunnel, while if both trains are in the tunnel, then they have crashed, and are forced to idle indefinitely. Initially, both trains are away.

The overall structure of the train system, and the model of trains is illustrated in Figure 4.1, a formal definition of the train system AATS *is given in Figure 4.3, and Figure 4.2 shows in detail the states and transitions of the agents, where A, W and I stand for away, waiting and in, respectively, and i and m stand for idle and move, respectively. To prevent the diagram getting too cluttered, we omit the joint action $\langle idle_E, idle_W\rangle$, which would result in transitions represented by a reflexive arc at each state.*

Of course, not all combinations of the propositional variables correspond to reachable system states (i.e., states that the system could possibly enter). For example, an agent i cannot be both $waiting_i$ and in_i simultaneously. There are in fact just nine reachable states of the system; see Figure 4.3.

Using AATSs, we are able to refer to strategies in a much more intuitive manner. Now we can refer to the actions an agent will choose in a given situation. Consider the same strategies for E and W introduced in Section 2.1. Now, given $\sigma_E(\overrightarrow{s};q)$, if $length(\overrightarrow{s};q) \bmod 3 = 1$ then $\sigma_E(\overrightarrow{s};q) = move_E$. So rather than $\sigma_E(\overrightarrow{s};q)$ defining a choice for the eastbound train, it explicitly defines the action which E should perform if following that strategy.

4.2 ATL over AATSs

In this section we define the logical language that we use to reason about social laws over AATSs. The logical language is ATL as defined in Section 2.1. The grammar exactly as before, but now we give the truth definition of ATL formulae on an AATS S and a state q:

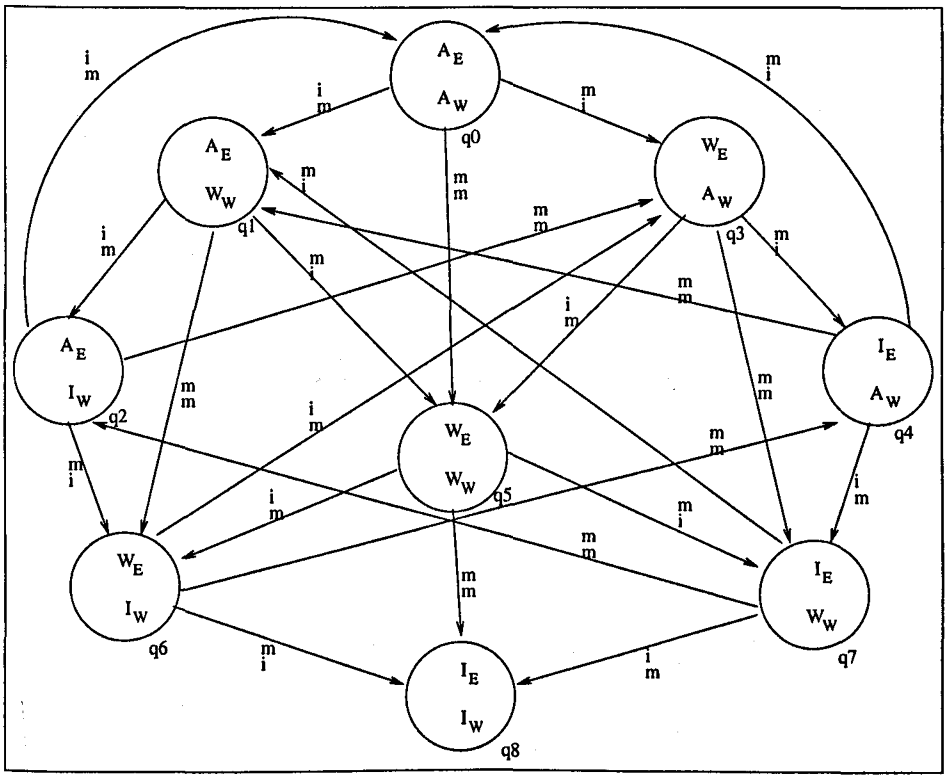

Figure 4.2: All states and transitions.

$S, q \models p$ iff $p \in \pi(q)$ (where $p \in \Phi$);

$S, q \models \neg\varphi$ iff $S, q \not\models \varphi$;

$S, q \models \varphi \vee \psi$ iff $S, q \models \varphi$ or $S, q \models \psi$;

$S, q \models \langle\langle G\rangle\rangle \bigcirc \varphi$ iff $\exists \sigma_G \in \Sigma_G$, such that $\forall \lambda \in comp(\sigma_G, q)$, we have $S, \lambda[1] \models \varphi$;

$S, q \models \langle\langle G\rangle\rangle \Box \varphi$ iff $\exists \sigma_G \in \Sigma_G$, such that $\forall \lambda \in comp(\sigma_G, q)$, we have $S, \lambda[u] \models \varphi$ for all $u \in \mathbb{N}$;

$S, q \models \langle\langle G\rangle\rangle \varphi \mathcal{U} \psi$ iff $\exists \sigma_G \in \Sigma_G$, such that $\forall \lambda \in comp(\sigma_G, q)$, there exists some $u \in \mathbb{N}$ such that $S, \lambda[u] \models \psi$, and for all $0 \leq v < u$, we have $S, \lambda[v] \models \varphi$.

Now that we have formally defined ATL over AATSs, it is interesting to note that AATSs and ATSs are equivalent:

$$\models_{AATS} \varphi \iff \models_{ATS} \varphi$$

For further details, the reader should consult [22].

<u>States and Initial States:</u>

$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8\}$ Initial state q_0

<u>Agents, Actions, and Joint Actions:</u>

$Ag = \{E, W\}$ $Ac_E = \{idle_E, move_E\}$ $Ac_W = \{idle_W, move_W\}$

$$J_{Ag} = \{\underbrace{\langle idle_E, idle_W\rangle}_{j_0}, \underbrace{\langle idle_E, move_W\rangle}_{j_1}, \underbrace{\langle move_E, idle_W\rangle}_{j_2}, \underbrace{\langle move_E, move_W\rangle}_{j_3}\}$$

<u>Transitions/Pre-conditions:</u>

$$\tau(q_0,j) = \begin{cases} q_0 & \text{if } j = j_0 \\ q_1 & \text{if } j = j_1 \\ q_3 & \text{if } j = j_2 \\ q_5 & \text{if } j = j_3 \end{cases} \qquad \tau(q_4,j) = \begin{cases} q_4 & \text{if } j = j_0 \\ q_7 & \text{if } j = j_1 \\ q_0 & \text{if } j = j_2 \\ q_1 & \text{if } j = j_3 \end{cases}$$

$$\tau(q_1,j) = \begin{cases} q_1 & \text{if } j = j_0 \\ q_2 & \text{if } j = j_1 \\ q_5 & \text{if } j = j_2 \\ q_6 & \text{if } j = j_3 \end{cases} \qquad \tau(q_5,j) = \begin{cases} q_5 & \text{if } j = j_0 \\ q_6 & \text{if } j = j_1 \\ q_7 & \text{if } j = j_2 \\ q_8 & \text{if } j = j_3 \end{cases}$$

$$\tau(q_2,j) = \begin{cases} q_2 & \text{if } j = j_0 \\ q_0 & \text{if } j = j_1 \\ q_6 & \text{if } j = j_2 \\ q_3 & \text{if } j = j_3 \end{cases} \qquad \tau(q_6,j) = \begin{cases} q_6 & \text{if } j = j_0 \\ q_3 & \text{if } j = j_1 \\ q_8 & \text{if } j = j_2 \\ q_4 & \text{if } j = j_3 \end{cases}$$

$$\tau(q_3,j) = \begin{cases} q_3 & \text{if } j = j_0 \\ q_5 & \text{if } j = j_1 \\ q_4 & \text{if } j = j_2 \\ q_7 & \text{if } j = j_3 \end{cases} \qquad \tau(q_7,j) = \begin{cases} q_7 & \text{if } j = j_0 \\ q_8 & \text{if } j = j_1 \\ q_1 & \text{if } j = j_2 \\ q_2 & \text{if } j = j_3 \end{cases}$$

$$\tau(q_8,j_0) = q_8$$

<u>Propositional Variables:</u>

$\Phi = \{away_E, away_W, waiting_E, waiting_W, in_E, in_W\}$

<u>Interpretation Function:</u>

$$\begin{aligned} \pi(q_0) &= \{away_E, away_W\} & \pi(q_4) &= \{in_E, away_W\} \\ \pi(q_1) &= \{away_E, waiting_W\} & \pi(q_5) &= \{waiting_E, waiting_W\} \\ \pi(q_2) &= \{away_E, in_W\} & \pi(q_6) &= \{waiting_E, in_W\} \\ \pi(q_3) &= \{waiting_E, away_W\} & \pi(q_7) &= \{in_E, waiting_W\} \\ & & \pi(q_8) &= \{in_E, in_W\} \end{aligned}$$

Figure 4.3: The AATS for the trains scenario.

4.2.1 Some Properties of ATL

Now we prove some properties of ATL that we use in subsequent proofs. For any AATS, $S = \langle Q, q_0, Ag, Ac_1, \ldots, Ac_n, \rho, \tau, \Phi, \pi \rangle$, let the relation R_{Ag} between states in Q be defined as: $q_1 R_{Ag} q_2$ iff for some $j \in J_{Ag}, \tau(q_1, j) = q_2$. In other words, $q_1 R_{Ag} q_2$ is true if the grand coalition can enforce a transition from q_1 to q_2. Now, let $S = \langle Q, q_0, Ag, Ac_1, \ldots, Ac_n, \rho, \tau, \Phi, \pi \rangle$ and $S' = \langle Q', q'_0, Ag, Ac'_1, \ldots, Ac'_n, \rho', \tau', \Phi, \pi' \rangle$ be two AATSs. We say that S' is a subsystem of S (notation $S' \sqsubseteq S$), or that S is a supersystem of S' (notation $S \sqsupseteq S'$), if there exists a relation $\Re \subseteq Q \times Q'$ such that:

1. $q_0 \Re q'_0$
2. $\forall q \in Q,\ q' \in Q' : q \Re q' \Rightarrow (\pi(q) = \pi'(q'))$
3. $\forall q_1 \in Q,\ q'_1, q'_2 \in Q' : ((q_1 \Re q'_1 \ \&\ q'_1 R'_{Ag} q'_2) \Rightarrow (\exists q_2 \in Q :\ q_1 R_{Ag} q_2 \ \&\ q_2 \Re q'_2))$

A special case is obtained when $S' = \langle Q, q_0, Ag, Ac_1, \ldots, Ac_n, \rho', \tau', \Phi, \pi \rangle$, in which case we write $S' \ll S$. The relation $\Re$ is essentially half of a *bisimulation* between two Kripke models [7, p.64]: the final clause represents the "backward clause" for such relations, with the "forward clause" not being present in the conditions on $\Re$. We are interested in which formulae are preserved when going from S to S', where $S' \sqsubseteq S$. To this end, we define a *universal* and an *existential* sublanguage of ATL, denoted $\mathcal{L}^u$ and $\mathcal{L}^e$, respectively. These languages are defined by the following grammars:

$$\begin{array}{ll} \mathcal{L}^u & \upsilon ::= p \mid \neg p \mid \upsilon \wedge \upsilon \mid \upsilon \vee \upsilon \mid \langle\langle\rangle\rangle \bigcirc \upsilon \mid \langle\langle\rangle\rangle \Diamond \upsilon \mid \langle\langle\rangle\rangle \Box \upsilon \mid \langle\langle\rangle\rangle \upsilon \mathcal{U} \upsilon \\ \mathcal{L}^e & \epsilon ::= p \mid \neg p \mid \epsilon \wedge \epsilon \mid \epsilon \vee \epsilon \mid \langle\langle Ag\rangle\rangle \bigcirc \epsilon \mid \langle\langle Ag\rangle\rangle \Diamond \epsilon \mid \langle\langle Ag\rangle\rangle \Box \epsilon \mid \langle\langle Ag\rangle\rangle \epsilon \mathcal{U} \epsilon \end{array}$$

where $p \in \Phi$.

Lemma 1 Suppose we have $S' \sqsubseteq S$ and $\upsilon \in \mathcal{L}^u$, $\epsilon \in \mathcal{L}^e$. Then:

1. $\forall q \in Q,\ q' \in Q'$ with $q \Re q'$: $S, q \models \upsilon \Rightarrow S', q' \models \upsilon$.
2. $\forall q \in Q,\ q' \in Q'$ with $q \Re q'$: $S', q' \models \epsilon \Rightarrow S, q \models \epsilon$.

Proof: We only prove the first item; the second follows from it and the observation that every $\epsilon \in \mathcal{L}^e$ is equivalent to the negation of some $\upsilon \in \mathcal{L}^u$. The proof is by structural induction. For the case υ is p or $\neg p$ where $p \in \Phi$, the claim follows immediately from the second condition in the definition of subsystems. The cases of conjunction and disjunction follow directly, so assume that υ equals $\langle\langle\rangle\rangle \Diamond \upsilon'$, and that the claim is proven for υ'. Assume that $S, q \models \langle\langle\rangle\rangle \Diamond \upsilon'$, and $q\Re q'$. The former implies that for all strategy profiles $\sigma \in \Sigma_{Ag}$, for the unique $\lambda_\sigma \in comp(\sigma, q)$, we have that for some i_σ, $S, \lambda[i_\sigma] \models \upsilon'$. In order to derive a contradiction, suppose that $S', q' \not\models \langle\langle\rangle\rangle\Diamond\upsilon'$. It would mean $S', q' \models \langle\langle Ag\rangle\rangle\Box\neg\upsilon'$, i.e., for some strategy profile $\sigma' \in \Sigma_{Ag}$, and the computation $\lambda' \in comp(\sigma', q')$ and all $i \in \mathbb{N}$, we have $S', \lambda'[i] \models \neg\upsilon'$. Consider the run $\lambda'[0]R_{Ag}\lambda'[1]\ldots\lambda'[i]\ldots$ that is induced by λ', with $\lambda'[0] = q$. By the backward condition on $\Re$, we hence also find $q_0 = q, q_1, \ldots q_i, \ldots$ with $q_i R_{Ag} q_{i+1}$ and $q_i \Re \lambda'[i]$. By the inductive assumption, we have $S, q_i \models \neg\upsilon'$ for all i. But then, the agents Ag have a strategy to always ensure $\neg\upsilon'$ from q: they just choose the actions that induce this path. This then means that $S, q \models \langle\langle Ag\rangle\rangle\Box\neg\upsilon'$, which contradicts the fact that $S, q \models \langle\langle\rangle\rangle\Diamond\upsilon'$. The remaining cases are similar. □

4.3 Summary

This chapter has introduced the semantic structures which are the basis for our framework of social laws for multi-agent systems. These structures are equivalent to ATSs introduced in Section 2.1. We extended ATSs into what we call Action-based ATSs (AATSs), as actions and action pre-conditions play an important role in our framework of social laws. We also revisited the train example, but gave explicit actions to the agents, and showed how strategies in these systems are more intuitive and correspond more closely to how humans would think of strategies. We make use of these systems, and more specifically, the train example, in the next chapter. We build-upon these structures in subsequent chapters. Finally, we defined ATL over these AATSs and went on to prove some properties of ATL that we use in subsequent proofs.

Chapter 5

Social Laws in Alternating Time

In this chapter, we make four key contributions to the literature on social laws. First, we demonstrate that Alternating-time Temporal Logic (ATL), the temporal logic of cooperation, introduced earlier in Section 2.1, provides a rich and natural technical framework within which to investigate social laws and their properties. Second, we show that the *effectiveness*, *feasibility*, and *synthesis* problems for social laws in the ATL framework can be posed as ATL *model checking* problems [9], and that existing model checkers for ATL can hence be directly applied to these problems. Third, we show that, despite its apparent expressive power, the complexity of the feasibility problem in our framework is no greater than the corresponding problem in the Shoham-Tennenholtz framework: it is NP-complete. We also identify cases where the problem becomes tractable. Finally, we show how our basic framework can easily be extended to permit social laws in which constraints on the legality or otherwise of some action may be explicitly required: for example, we show how the feasibility and synthesis problems for *dictatorship* social laws can be formulated.

We begin by introducing and formally defining our social laws framework with respect to AATSs and ATL. Next, we present the effectiveness, feasibility, and synthesis problems, and show how these can be understood as ATL model checking problems. We then show how we can extend the basic social laws framework to reason about laws which have explicit action constraints. Finally, a summary is presented in Section 5.2.

5.1 Social Laws

We now introduce the formal framework of social laws, which we build upon throughout the remainder of the chapter. Intuitively, a social law consists of two parts:

- An *objective*, which represents what the society aims to achieve by the introduction, or adoption of the law. In human societies, for example, the objective of a law might be to eliminate alcohol-related road accidents.

- A *behavioural constraint*, which corresponds to the requirements that a law places on the members of a society. A behavioural constraint corresponding to the objective of eliminating alcohol-related road accidents might be to forbid the action of driving a car after drinking alcohol.

We represent an objective for a social law as a formula of ATL, the intuition being that a social law is *effective* if it ensures that the objective is satisfied. (We give the formal definition shortly.) ATL provides a natural and powerful language for expressing the objectives of social laws, not least because such laws frequently refer to the *powers* or *rights* (or, conversely, the limits to powers) that agents have. The original social laws framework of Shoham and Tennenholtz illustrates this [47]. In this framework, each agent $i \in Ag$ is associated with a set $F_i \subseteq Q$ of *focal states*, the idea being that a successful social law would be one which ensured that if ever an agent $i \in Ag$ found the environment was in a state $q \in F_i$, then i should have the power to ensure that the environment eventually entered state $q' \in F_i$. This objective can naturally be expressed within our framework. Assume that, for each $q \in Q$, we have a proposition q that is satisfied iff the system is currently in state q. Then, given focal state sets $F_1, \ldots, F_n$, we can express the Shoham-Tennenholtz objective as follows:

$$\bigwedge_{i \in Ag} \left(\bigwedge_{\mathrm{q} \in F_i} \left[\mathrm{q} \rightarrow \bigwedge_{\mathrm{q}' \in F_i} \langle\langle i \rangle\rangle \Diamond \mathrm{q}' \right] \right) \tag{5.1}$$

The ATL framework allows us to express much richer objectives, however. For example, from CTL it inherits the ability to express liveness and safety properties, and moreover we can reason about what certain *coalitions* can bring about.

Example 10 *Recall the trains scenario introduced earlier in Chapter 4. The most obvious requirement for a social law is that the trains do not crash. The objective for this social law, O_1, is thus:*

$$O_1 = \neg(in_E \wedge in_W)$$

The basic system S_1 does not ensure that (O_1) is satisfied – there are initial computations of the basic system on which both trains enter the tunnel simultaneously:

$$S_1, q_0 \not\models \langle\langle\rangle\rangle \Box \neg(in_E \wedge in_W).$$

We model a behavioural constraint, β, as a function

$$\beta : Ac_{Ag} \rightarrow 2^{Q}$$

with the intended interpretation that if $q \in \beta(\alpha)$, then action α may *not* be performed when the system is in state q – that is, α is "forbidden" in state q. (As an aside, notice the duality between the pre-condition function ρ, and behavioural constraints β: if $q \in \rho(\alpha)$, then α is permitted in q, whereas if $q \in \beta(\alpha)$, then α is forbidden in q.) We will require that any behavioural constraint is "reasonable", in that it always permits an agent to have at least one action left that can be performed in any state:

$$\forall i \in Ag, \forall q \in Q, \exists \alpha \in options(i,q) \text{ s.t. } q \notin \beta(\alpha).$$

We can now see what it means for a behavioural constraint β to be *implemented* in an AATS S: it simply means eliminating from S all transitions that are forbidden by β. The operation of implementing a behavioural constraint is thus an *update* on AATSs, in the sense that it results in a new AATS, which potentially satisfies different formulae. We denote the AATS obtained from S by implementing β by $S \dagger \beta$. Formally, if $S = \langle Q, q_0, Ag, Ac_1, \ldots, Ac_n, \rho, \tau, \Phi, \pi \rangle$ is an AATS, and β is a behavioural constraint on S, then

$$S \dagger \beta = \langle Q, q_0, Ag, Ac_1, \ldots, Ac_n, \rho', \tau', \Phi, \pi \rangle,$$

where:

1. $\forall \alpha \in Ac$,

   $\rho'(\alpha) = \rho(\alpha) \setminus \beta(\alpha)$

2. $\forall q \in Q, \forall j \in J_{Ag}$,

   $$\tau'(q,j) = \begin{cases} \tau(q,j) & \text{if } (q,j) \in \text{dom } \tau \text{ and } \forall i \in Ag, q \notin \beta(j_i) \\ \text{undefined} & \text{otherwise} \end{cases}$$

3. All other components of $S \dagger \beta$ are as in S.

It is natural to ask what properties the implementation operator "$\dagger$" has. First, notice that AATSs are closed under the implementation of behavioural constraints. That is, if S is an AATS and β is a behavioural constraint over S, then $S \dagger \beta$ is an AATS – it satisfies the non-triviality and consistency coherence constraints given earlier. Second, notice that $(S \dagger \beta) \dagger \beta = S \dagger \beta$. Third, consider what properties of AATSs might be preserved through by the implementation of social laws. To answer this question, first recall the notion of a *subsystem* as defined in Section 4.2.1; we have the following result.

Lemma 2 Let S be an AATS, and let β be a behavioural constraint over S. Then $S \dagger \beta$ is a subsystem of S, i.e., $S \dagger \beta \sqsubseteq S$. (In fact, $S \dagger \beta \ll S$.)

From Lemma 1, we immediately obtain the following.

Lemma 3 Suppose we have an AATS S, a behavioural constraint β over S, a state q in S, and formulae $\upsilon \in \mathcal{L}^u$, $\epsilon \in \mathcal{L}^e$. Then:

1. If $S, q \models \upsilon$ then $S \dagger \beta, q \models \upsilon$.

2. If $S \dagger \beta, q \models \epsilon$ then $S, q \models \epsilon$.

The first of these two results tells us that implementing a behavioural constraint *guarantees to preserve the universal properties of a system.* However, it is easy to see that implementing a behavioural constraint does not guarantee to preserve existential properties. The second result, in contrast, tells us that if an existential property holds of a system in which some behavioural constraint has been implemented, then it must have held of the original system, before the constraint was imposed. Thus, implementing a behavioural constraint cannot *create* existential properties in a system.

Now, a *social law* over an AATS S is a pair:

$$(\varphi, \beta)$$

where:

- φ is an ATL formula called the *objective* of the law; and

- $\beta : Ac_{Ag} \to 2^Q$ is a behavioural constraint on S.

A social law (φ, β) is *effective* in AATS S if, after implementing β in S, we know that $\langle\langle\rangle\rangle \Box \varphi$ will be initially satisfied in S, i.e., if $S \dagger \beta, q_0 \models \langle\langle\rangle\rangle \Box \varphi$.

There are three key computational questions with respect to social laws, which we shall investigate throughout the remainder of this chapter:

1. *Effectiveness.* Given an AATS S and a social law (φ, β) over S, determine whether (φ, β) is effective in S.

2. *Feasibility.* Given an AATS S and a formula φ of ATL representing an objective, does there exist a behavioural constraint β such that (φ, β) is an effective social law in S.

3. *Synthesis.* Given an AATS S and a formula φ of ATL representing an objective, exhibit a behavioural constraint β such that (φ, β) is an effective social law in S if such a constraint exists, otherwise answer "no".

Our first result, with respect to the effectiveness problem, is now immediate.

Lemma 4 The effectiveness problem for social laws may be solved in time polynomial in the size of S and φ.

Proof: Generate $S' = S \dagger \beta$, and check that $S', q_0 \models \langle\langle\rangle\rangle \Box \varphi$. The first step can obviously be done in polynomial time: it simply requires eliminating every forbidden transition from τ, and modify ρ similarly. From Theorem 1, so can the second step. □

With respect to the trains example, is there a behavioural constraint β_1, such that (O_1, β_1) is an effective social law? Clearly there is. The constraint β_1 must ensure that the system never enters state q_8: from examination of the state transition function τ (see Figure 4.3), we can see that $\tau(q_5, j_3) = \tau(q_6, j_2) = \tau(q_7, j_1) = q_8$, and there are no other transitions leading to q_8 (apart from when the trains have already crashed, which we need not consider!) Consider the behavioural constraint β_1 as follows.

$$\beta_1(\alpha) = \begin{cases} \emptyset & \text{if } \alpha = idle_E \\ \emptyset & \text{if } \alpha = idle_W \\ \{q_5, q_6\} & \text{if } \alpha = move_E \\ \{q_7\} & \text{if } \alpha = move_W \end{cases}$$

The constraint ensures that:

- when both agents are waiting to enter the tunnel, the eastbound train is prevented from moving;
- when the westbound train is already in the tunnel and the eastbound train is waiting to enter the tunnel, then the eastbound train is prevented from moving; and
- when the eastbound train is already in the tunnel and the westbound train is waiting to enter the tunnel, then the westbound train is prevented from moving.

Notice that this constraint is, in a sense, asymmetric, as it constrains the eastbound train rather than the westbound train: we could equally well replace the first constraint with the requirement that if both trains are waiting to enter the tunnel, then the *westbound* train is prevented from moving, thus enabling the eastbound train to enter. Now, let $S_2 = S_1 \dagger \beta_1$. We claim that $S_2, q_0 \models \langle\langle\rangle\rangle \Box \neg(in_E \wedge in_W)$. Thus:

Proposition 1 *(O_1, β_1) is an effective social law in S_1.*

Proposition 1 can be proved in the following way:

Proof: We wish to prove Proposition 1: that (O_1, β_1) is an effective social law in S_1. This means showing that $S_2, q_0 \models \langle\langle\rangle\rangle \Box \neg(in_E \wedge in_W)$. To show that this holds, there should be no path in S_2 where $(in_E \wedge in_W)$ is true. In order to check whether this is the case, consult Figure 5.1. In this figure, the dotted lines represent transitions that have been removed by β_1. We can see that the only way to enter state q_8 $(in_E \wedge in_W)$ is from q_5, where both trains are waiting, q_6, where E is waiting and W is in, or q_7, where E is in and W is waiting. Assume the system is in state q_5. Referring to β_1 we can see that E is forbidden from moving in this state. So the only possible transitions are to q_5 or q_6. If we now assume the system is in state q_6, E is again prevented from moving. The only possible transitions are to q_6 or q_3. Finally, if the system is in state q_7, W is prevented from moving, thus the only possible transitions are to q_7 or to q_1. Therefore, we have shown there exists no path in S_2 where $(in_E \wedge in_W)$ holds, hence $S_2, q_0 \models \langle\langle\rangle\rangle \Box \neg(in_E \wedge in_W)$ is true, meaning Proposition 1 is also true. □

Of course, there are other, less "sensible" behavioural constraints that are effective in S_1 for O_1. Consider β_2:

$$\beta_2(\alpha) = \begin{cases} Q & \text{if } \alpha = move_E \text{ or } \alpha = move_W \\ \emptyset & \text{otherwise} \end{cases}$$

This behavioural constraint prevents both trains from moving, and yet:

Proposition 2 *(O_1, β_2) is an effective social law in S_1.*

Clearly, our original objective needs some refinement. Consider objective O_2:

$$O_2 = O_1 \wedge \bigwedge_{i \in \{E,W\}} (waiting_i \rightarrow \langle\langle i \rangle\rangle \Diamond in_i)$$

This objective ensures that, not only do the trains never crash, but that both trains can eventually safely enter the tunnel if they are waiting. Consider β_3, which works by forbidding trains from lingering in the tunnel and prevents the westbound train from idling when both trains are waiting to enter the tunnel, but is otherwise the same as β_1:

$$\beta_3(\alpha) = \begin{cases} \{q_4, q_7\} & \text{if } \alpha = idle_E \\ \{q_2, q_6, q_5\} & \text{if } \alpha = idle_W \\ \{q_5, q_6\} & \text{if } \alpha = move_E \\ \{q_7\} & \text{if } \alpha = move_W \end{cases}$$

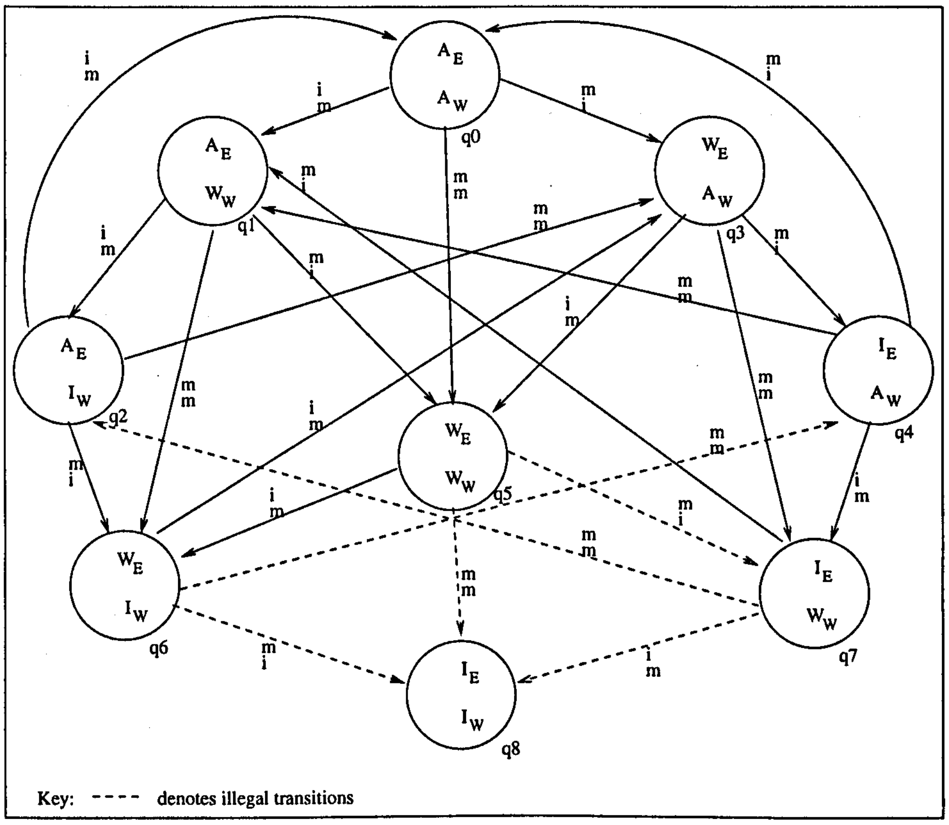

Figure 5.1: All states and transitions of S_2.

Proposition 3 *(O_2, β_3) is an effective social law in S_1.*

We wish to prove Proposition 3:

Proof: We wish to prove that (O_2, β_3) is an effective social law in S_1. Let $S_3 = S_1 \dagger \beta_3$. So, $S_3, q_0 \models \langle\langle\rangle\rangle \Box \left(O_1 \wedge \bigwedge_{i \in \{E,W\}} (waiting_i \rightarrow \langle\langle i \rangle\rangle \Diamond (in_i \wedge O_1))\right)$ is what we wish to prove. First, to show that O_1 holds, this means there exists no path in S_3 where $(in_E \wedge in_W)$ is true. The only way to enter state q_8 $(in_E \wedge in_W)$ is from q_5, q_6 or q_7. First assume the system is in state q_5. In this state, E is forbidden from moving and W is forbidden from idling, so the only transition is to q_6. Now assume the system is in q_6. In this state, E is forbidden from moving and W is forbidden from idling. The only possible transition is to q_3. Finally, in state q_7, E is forbidden from idling while W is forbidden from moving. Thus, the only transition is to state q_1. There is no transition leading to q_8, hence O_1 holds. Now, we must show that for all $waiting_i$ states that agent i has a strategy so that at some point in the future he is in the tunnel and O_1 still holds. Firstly we will look at states where $waiting_E$ is true. These are states q_3, q_5, and q_6. Starting with q_3, E could move here, causing a transition to q_7 or q_4, where in both states he is in the tunnel and O_1 holds. From q_5, E would have to idle, while W would have to move, causing a transition to q_6. Here, E would have to idle again, but W has to move, causing a transition to q_3, where finally E could move into the tunnel. Finally, we look at states where $waiting_W$ is true. These are states q_1, q_5 and q_7. Starting in state q_1, W could move, causing a transition to q_6 or q_2, where in both states he is in the tunnel and O_1 holds. From q_5, E is forced to idle, so W could move, causing a transition to q_6 again. Finally, from q_7, E is forbidden from idling here and W is forbidden from moving, so the only transition would be to q_1, where as before W could move causing a transition to q_6 or q_2. Therefore, we have proved that (O_2, β_3) is an effective social law in S_1.

□

However, objective O_2 may still need further refinement, as less sensible behavioural constraints can still be effective in S_1:

Proposition 4 *(O_2, β_2) is an effective social law in S_1.*

So, consider objective O_3:

$$O_3 = O_1 \wedge \bigwedge_{i \in \{E,W\}} \begin{pmatrix} (away_i \rightarrow \langle\langle i \rangle\rangle \Diamond waiting_i) & \wedge \\ (waiting_i \rightarrow \langle\langle i \rangle\rangle \Diamond in_i) & \wedge \\ (in_i \rightarrow \langle\langle i \rangle\rangle \bigcirc away_i) & \end{pmatrix}$$

This objective ensures that, not only do the trains never crash, but both trains may at some point enter the *waiting$_i$* state. The objective also ensures that, once in the *waiting$_i$* state, the train does eventually enter the state where it has safely entered the tunnel. Once the train is in the tunnel, it will not linger there, but its next state will be where it has safely left the tunnel and gone into the *away$_i$* state.

Proposition 5 *(O_3, β_2) is not an effective social law in S_1, but (O_3, β_3) is.*

We have also verified all of the above propositions by model checking. As introduced in Section 2.1, the model checker we used is called MOCHA. Model checking the above propositions involved programming the various systems in the REACTIVE MODULES language. We began by programming the system S_1, and then created updated versions of this system to represent the systems updated with the behavioural constraints given above. Then each objective was written as an ATL specification and checked with the appropriate system. To see the implementations, specifications and results from our model checking, visit `http://www.csc.liv.ac.uk/~mark/model_checking/`.

The first issue to which we address ourselves is the computational complexity of the feasibility problem. As it turns out, the feasibility problem in our framework is no harder, for objectives expressed as arbitrary ATL formulae, than the corresponding problem studied by Shoham and Tennenholtz, where objectives were expressed as sets of focal states (see above); and for objectives expressed in propositional logic, it is easier.

Theorem 8 *The feasibility problem for objectives expressed as arbitrary* ATL *formulae is* NP-*complete, and remains* NP-*complete for the* CTL *fragment of* ATL, *and also the universal ($\mathcal{L}^u$) fragment. However, for objectives expressed as propositional formulae, the feasibility problem is decidable in polynomial time.*

Proof: With respect to the NP-completeness results, membership in NP may be seen by the following non-deterministic algorithm:

1. guess a behavioural constraint β;
2. verify that β is effective.

Since dom $\beta = Ac_{Ag}$, step (1) can be done in (non-deterministic) polynomial time $O(|Ac_{Ag} \times Q|)$, while from Lemma 4, step (2) requires only polynomial time.

That the feasibility problem for ATL objectives is NP-hard follows immediately from the fact that the Useful Social Law problem of Shoham-Tennenholtz, (proven to be NP-complete in [49, p.612]), can be directly reduced to our feasibility problem, by encoding focal state sets as shown in equation (5.1): the reduction is clearly possible in polynomial time. However, to see that the effectiveness problem for the $\mathcal{L}^u$ fragment is NP-complete, we need to work a little harder. We reduce the directed Hamiltonian cycle problem (DHC) [38, p.209] to the effectiveness problem for $\mathcal{L}^u$ objectives.

An instance of DHC is given by a directed graph $H = \langle V, E \subseteq V \times V, v_0 \in V \rangle$, where v_0 is a distinguished "start" node. The aim is to determine whether or not H contains a directed Hamiltonian cycle starting from v_0, i.e., a cycle containing every vertex $v \in V$, in which no vertices are repeated[1]. The idea of the reduction is to encode the graph H directly in the state transformer function τ: actions correspond to edges of the graph. Formally, given a directed graph $H = \langle V, E \rangle$, we define a single-agent AATS $S^H = \langle Q^H, q_0^H, Ag^H, Ac_1^H, \rho^H, \tau^H, \Phi^H, \pi^H \rangle$ as follows:

- for each vertex $v \in V$, we create a state $q_v \in Q^H$, and in addition we create a "sink" state q_{sink};

[1]Of course, we strictly speaking do not need the start node, since if the graph contains a Hamiltonian cycle then we can start from *any* state – but it simplifi es the exposition.

- we set $q_0 = v_0$;
- we create a single agent, $Ag^H = \{a_1\}$;
- for each edge $(v, v') \in E$ where $v' \neq v_0$, we define an action $\alpha_{(v,v')}$, define the transition $\tau^H(q_v, \alpha_{(v,v')}) = q_{v'}$, and define ρ^H accordingly;
- for each edge $(v, v') \in E$ where $v' = v_0$, we define an action $\alpha_{(v,sink)}$, define the transition $\tau^H(q_v, \alpha_{(v,sink)}) = q_{sink}$, and define ρ^H accordingly;
- we create an action α_{loop}, which may be performed only in state q_{sink}, such that the transition $\tau^H(q_{sink}, \alpha_{loop}) = q_{sink}$, and define ρ^H accordingly;
- for each vertex $v \in V$, we create a proposition $p_v \in \Phi^H$; and finally,
- we define $\pi^H(q_v) = \{p_v\}$.

We then define the objective O^H by

$$O^H \hat{=} \varphi_1^H \wedge \varphi_2^H$$

where:

$$\begin{aligned} \varphi_1^H &\hat{=} \textstyle\bigwedge_{v \in V} \langle\langle\rangle\rangle \Diamond p_v \\ \varphi_2^H &\hat{=} \textstyle\bigwedge_{v \in V} p_v \rightarrow \langle\langle\rangle\rangle \bigcirc \langle\langle\rangle\rangle \Box \neg p_v \end{aligned}$$

We now claim that there exists a DHC in graph H starting from v_0 iff the objective O^H is feasible in AATS S^H. To see this, note that constraint φ_1^H ensures that every vertex is visited, while φ_2^H ensures that no vertex is visited more than once. Thus, the paths through any transition system resulting from the imposition of a behavioural constraint that is effective for O^H will be DHCs in H, with the agent ending up in v_{sink} and repeating the action α_{loop} infinitely often. Since O^H is a formula of $\mathcal{L}^u$, we are done.

The final result – that the feasibility problem for objectives expressed as propositional logic formulae may be decided in polynomial time – is an immediate corollary of the following result, which shows that, for objectives expressed as formulae of propositional logic, the feasibility problem reduces directly to model checking in ATL. □

Theorem 9 *Suppose φ is a propositional logic formula (representing an objective), and S is an* AATS. *Then $S, q_0 \models \langle\langle Ag\rangle\rangle \Box \varphi$ iff φ is feasible in S.*

Proof:

($\rightarrow$) Assume $S, q_0 \models \langle\langle Ag \rangle\rangle \Box \varphi$. By the semantics of ATL, we know that $\exists \sigma_{Ag} \in \Sigma_{Ag}$ such that $\forall \lambda \in comp(\sigma_{Ag}, q_0)$, we have $S, \lambda[u] \models \varphi$ for all $u \in \mathbb{N}$. In fact, since σ_{Ag} is a grand coalition strategy, $comp(\sigma_{Ag}, q_0)$ will be a singleton; let λ^* denote its member; so $\forall u \in \mathbb{N}$: $S, \lambda^*[u] \models \varphi$. We must show that this implies there exists a behavioural constraint β such that $S \dagger \beta, q_0 \models \langle\langle\rangle\rangle \Box \varphi$; we construct β as follows:

```
for each $i \in Ag$,
  for each $\alpha \in Ac_i$,
    set $\beta(\alpha) = \{q \mid \sigma_i(q) \neq \alpha\}$.
```

We claim that, for any q_0-computation λ of $S \dagger \beta$, we have $S \dagger \beta, \lambda[u] \models \varphi$ for all $u \in \mathbb{N}$. To see this, observe that there will in fact be a single q_0 computation of $S \dagger \beta$, namely λ^*, and we know that $\forall u \in \mathbb{N}$, $S, \lambda^*[u] \models \varphi$. We appeal to the fact that φ is a propositional formula, and that π is the same in S and $S \dagger \beta$, and conclude that for all $u \in \mathbb{N}$: $S \dagger \beta, \lambda^*[u] \models \varphi$.

($\leftarrow$) Assume φ is feasible in S. Then there exists a behavioural constraint β such that $S \dagger \beta, q_0 \models \langle\langle\rangle\rangle \Box \varphi$. Let Λ denote the set of q_0-computations of $S \dagger \beta$. Then by the semantics of ATL, for all $\lambda \in \Lambda$ and for all $u \in \mathbb{N}$, we have $S \dagger \beta, \lambda[u] \models \varphi$. We need to show that this implies $S, q_0 \models \langle\langle Ag \rangle\rangle \Box \varphi$. By the semantics of ATL, $S, q_0 \models \langle\langle Ag \rangle\rangle \Box \varphi$ iff $\exists \sigma_{Ag} \in \Sigma_{Ag}$ such that $\forall \lambda \in comp(q_0, \sigma_{Ag})$, and $\forall u \in \mathbb{N}$, we have $S, \lambda[u] \models \varphi$. We show how to construct such a σ_{Ag}. We start by constructing from β a *non-deterministic* strategy $\sigma_i^n : Q \rightarrow 2^{Ac_i}$ for each agent $i \in Ag$, as follows:

```
for each $i \in Ag$,
  for each $q \in Q$,
    set $\sigma_i^n(q) = \{\alpha \mid q \notin \beta(\alpha)\}$.
```

Now, say a (conventional deterministic) strategy σ_i is *consistent* with σ_i^n if $\sigma_i(q) \in \sigma_i^n(q)$, for all $q \in Q$. Now, let $\sigma_{Ag} = \langle \sigma_1, \ldots, \sigma_n \rangle$ be any grand coalition strategy profile such that each σ_i is consistent with the corresponding non-deterministic strategy σ_i^n. We claim that, thus defined, $\forall \lambda \in comp(\sigma_{Ag}, q_0)$ we have $S, \lambda[u] \models \varphi$ for all $u \in \mathbb{N}$. To see this, observe that by construction we have $comp(\sigma_{Ag}, q_0) \subseteq \Lambda$, as defined above. We know that $\forall \lambda \in \Lambda$, and for all $u \in \mathbb{N}$, we have $S \dagger \beta, \lambda[u] \models \varphi$. Since φ is propositional, its valuation depends only upon the state in which it is interpreted. Since π is unchanged in S and $S \dagger \beta$, it must be that $\forall \lambda \in \Lambda$, $\forall u \in \mathbb{N}$: $S, \lambda[u] \models \varphi$.

□

Notice that, as a direct corollary to Theorem 9, the synthesis problem for propositional logic objectives may also be solved in polynomial time: for if the answer to the model checking problem is "yes", then the witness to this will be the strategy for the grand coalition from which, as the proof of the theorem illustrates, we can extract a behavioural constraint implementing the objective.

To appreciate that Theorem 9 does not hold for arbitrary formulae, consider the following simple one agent ATS (one may also assume this agent to be a grand coalition Ag). Suppose we have two states, q_0 and q_1, in the first, the atom p is false, in the second it is true. Moreover, we have four actions, a_0, a_1, b_0 and b_1. Both a_i actions are possible in q_0, both b_i actions in q_1. Also, any action x_i ($x \in \{a, b\}$ takes the system to q_i). More formally: $\tau(q_0, a_0) = \tau(q_1, b_0) = q_0$, and $\tau(q_0, a_1) = \tau(q_1, b_1) = q_1$. Thus, our agent i can always take the system to a p-state, but also he can ensure the system's next state will be one in which $\neg p$ is true. Let $\varphi = \neg p \wedge \langle\langle i \rangle\rangle \Diamond p$. In this system S we have $S, q_0 \models \langle\langle Ag \rangle\rangle \Box \varphi$ which expresses that the agent has a strategy to ensure that always $\neg p$ is true, at the same time always having the opportunity to make p true. However, if we want any behavioural constraint β to be such that $S \dagger \beta, q_0 \models \langle\langle \rangle\rangle \Box \varphi$, we see that this is impossible: to ensure that φ is always true in all runs, p has to be always false, so β has to forbid any transition to the state q_1, in which case the second conjunct of φ, i.e., $\langle\langle i \rangle\rangle \Diamond p$ can never be made true anymore.

Theorem 9 illustrates a close relationship between the feasibility problem and model checking, which begs the question as to what extent the result can be extended for objectives beyond propositional logic. We have the following.

Theorem 10 *Suppose $\upsilon \in \mathcal{L}^u$ is a universal* ATL *formula (representing an objective), and S is an* AATS. *Then $S, q_0 \models \langle\langle Ag \rangle\rangle \Box \upsilon$ implies υ is feasible in S.*

Proof: (Sketch.) The basic structure of the proof is as Theorem 9, but as υ is no longer a propositional formula, but a universal ATL formula, we make use of Lemma 3. As we are only proving in one direction, and universal ATL properties are preserved after implementing a behavioural constraint, π is the same for S and $S \dagger \beta$. □

5.1.1 Objectives with Explicit Action Constraints

We now consider objectives for laws *that explicitly refer to the legality or otherwise of actions.* The following example illustrates the idea.

Example 11 *If α is an action, then let the proposition $\ell(\alpha)$ mean "action α is legal". Consider the following three objectives in the context of the trains system, as discussed earlier.*

$$\begin{aligned} O_5 &= \neg(in_E \wedge in_W) \wedge \ell(move_E) \wedge \ell(move_W) \\ O_6 &= \neg(in_E \wedge in_W) \wedge \ell(move_E) \\ O_7 &= \neg(in_E \wedge in_W) \wedge \ell(move_W) \end{aligned}$$

Objective O_5 requires that not only do the trains never crash, but that both trains are always able to move. *Any behavioural constraint for this objective must not prevent the trains from moving if they choose to do so. Objective O_6 is similar, but only requires that the Eastbound train is always able to move; and O_7 only requires that the Westbound train is always able to move.*

In our basic social laws framework, we have no way of expressing or reasoning about social laws with such objectives. We now show how the basic framework can be extended to include such constraints. In particular, we show how this can be done in the context of the MOCHA model checking system [3]. MOCHA takes as input a specification of an AATS, expressed in the REACTIVE MODULES language, and a formula of ATL, and is capable of either checking whether this formula is true in the AATS, or else giving a counter example. The actual syntax of the REACTIVE MODULES language used in MOCHA is rather complex, and so we adopt the following simplified syntax in the interests of easy comprehension. A REACTIVE MODULES agent (they are called "atoms" in the REACTIVE MODULES literature) has the following structure:

$$\begin{array}{l} \texttt{agent}\ \mathit{name}\ \texttt{reads}\ \mathit{in}\ \texttt{writes}\ \mathit{out} \\ \quad P_1 \mapsto \alpha_1; \\ \quad P_2 \mapsto \alpha_2; \\ \quad \ldots \\ \quad P_k \mapsto \alpha_k. \end{array} \tag{5.2}$$

where *name* is the name of the agent, $in \subseteq \Phi$ is a list of boolean variables that the agent observes, $out \subseteq \Phi$ is a list of boolean variables that it controls, and provides to the rest of the system, and the $P_j \mapsto \alpha_j$ structures are *guarded commands*, where P_j is a predicate over the variables the agent observes and controls (i.e., a boolean expression over variables $in \cup out$), and $\alpha_j \in Ac_{name}$ is an action. Actions can be understood as functions that take as input the variables visible to the agent ($in \cup out$), and produce as output an assignment for the variables controlled by the agent (*out*). The idea is that at each time step, each agent generates the set of rules whose pre-conditions are satisfied by the current state of the system. One of these rules is non-deterministically chosen for execution – this involves simply executing the corresponding action, which produces an assignment for the variables under its control (*out*), which is the value they take in the next state of the system. Of course, a guarded command may have $\top$ as a pre-condition, in which case it is always enabled for execution.

It should be clear how a collection of such agents maps to an AATS, as defined in Chapter 4. The most important point is that the pre-conditions to guarded commands correspond to the pre-condition function ρ: given a state of the system q and guarded command $P \mapsto \alpha$, then $q \in \rho(\alpha)$ iff $q \models P$.

We now show how an AATS, expressed in such a framework, can be extended to permit objectives referring to the legality or otherwise of actions. The idea is to define a transformation on AATSs:

$$\cdot^{\circ} : \text{AATS} \rightarrow \text{AATS}$$

such that the transformed system S° of S includes the $\ell(\alpha)$ propositions as before, but otherwise has the same properties.

Given an AATS $S = \langle Q, q_0, Ag, Ac_1, \ldots, Ac_n, \rho, \tau, \Phi, \pi \rangle$, defined using the REACTIVE MODULES language as above, the system S° is created as follows.

First, for each action $\alpha \in Ac$, we create:

- A new propositional variable $\ell(\alpha)$, with the intended interpretation that $\ell(\alpha)$ will be true in a state if the action α is legal according to the behavioural constraint that we are looking for.
- A new agent, which controls the variable $\ell(\alpha)$, as follows.

```
agent $\alpha$-controller reads $\Phi$ writes $\ell(\alpha)$
    $\top \mapsto \ell(\alpha) := \top$;
    $\top \mapsto \ell(\alpha) := \bot$.
```

  Thus, in every possible state, the agent *α-controller* has two actions available: make $\ell(\alpha)$ true, or make $\ell(\alpha)$ false. Note that because the condition on the guard of each of these actions is $\top$, the *α-controller* agent can *always* perform these actions. We will denote by *controllers* the set of all controller agents introduced in this way.

We then take each of the original agents i in the system, and transform them as follows.

- We replace each guarded command $P \mapsto \alpha$ for agent i with a rule as follows.

$$\ell(\alpha)' \wedge P \mapsto \alpha$$

  Notice that the first conjunct of the guard is "primed"; this notation in REACTIVE MODULES means the value that $\ell(\alpha)$ has been assigned *in the current round*, i.e., the value of this variable after it has been assigned by the *α-controller* agent. Thus agent i may perform the action α iff the original guard condition P is true in the current state, *and the new agent controlling $\ell(\alpha)$ has assigned this variable the value $\top$ in the current state.* In this way, the *α-controller* agent can determine whether or not α is performed.

- Next, if *in* is the set of variables that i `reads`, then we replace *in* by $in \cup \{\ell(\alpha) \mid \alpha \in Ac_i\}$. In other words, the agent i `reads` those ℓ-variables that determine whether its actions are legal.

  Notice that agent i is not dependent on the ℓ-variables of any other agent's actions, and hence any strategy computed for i by MOCHA will not be dependent on the ℓ-variables of other agents.

Now, we can prove:

Theorem 11 *Suppose φ is a propositional logic formula (representing an objective) and S is an* AATS. *Then*

$$S^{\circ}, q_0 \models \langle\langle controllers \rangle\rangle \Box \left(\varphi \wedge \left[\bigwedge_{i \in Ag} \bigvee_{\alpha \in Ac_i} \ell(\alpha) \right] \right)$$

iff φ is feasible in S.

Proof:

($\rightarrow$) Assume $S^{\circ}, q_0 \models \langle\langle controllers \rangle\rangle \Box \left(\varphi \wedge \left[\bigwedge_{i \in Ag} \bigvee_{\alpha \in Ac_i} \ell(\alpha) \right] \right)$. By the semantics of ATL, we know that $\exists \sigma_{controllers} \in \Sigma_{controllers}$ such that for all $\lambda \in comp(\sigma_{controllers}, q_0)$, we have $S^{\circ}, \lambda[u] \models \varphi \wedge \left(\bigwedge_{i \in Ag} \bigvee_{\alpha \in Ac_i} \ell(\alpha) \right)$ for all $u \in \mathbb{N}$. We must show that this implies there exists a behavioural constraint β such that $S \dagger \beta, q_0 \models \langle\langle\rangle\rangle \Box \varphi$. The condition, $\bigwedge_{i \in Ag} \bigvee_{\alpha \in Ac_i} \ell(\alpha)$, corresponds to the non-triviality property of the behavioural constraint: all agents should have at least one legal action at each state. We construct β as follows:

```
for each $i \in Ag$,
  for each $\alpha \in Ac_i$,
    set $\beta(\alpha) = \{q \mid \sigma_{\alpha-controller}(q) = \ell(\alpha) := \bot\}$.
```

We claim that, for any q_0-computation λ of $S \dagger \beta$, we have $S \dagger \beta, \lambda[u] \models \varphi$ for all $u \in \mathbb{N}$. We know that $\forall \lambda \in comp(\sigma_{controllers}, q_0)$, we have $S^{\circ}, \lambda[u] \models \varphi \wedge \left(\bigwedge_{i \in Ag} \bigvee_{\alpha \in Ac_i} \ell(\alpha) \right)$ for all $u \in \mathbb{N}$. We appeal to the fact that φ is a propositional formula, and that π is the same in S° and $S \dagger \beta$, and conclude that for all q_0-computations, λ, and for all $u \in \mathbb{N}$: $S \dagger \beta, \lambda[u] \models \varphi$.

($\leftarrow$) Assume φ is feasible in S. Then there exists a behavioural constraint β such that $S \dagger \beta, q_0 \models \langle\langle\rangle\rangle \Box \varphi$. Let Λ denote the set of q_0-computations of $S \dagger \beta$. Then by the semantics of ATL, for all $\lambda \in \Lambda$ and for all $u \in \mathbb{N}$, we have $S \dagger \beta, \lambda[u] \models \varphi$. We need to show that this implies $S^\circ, q_0 \models \langle\langle \mathit{controllers} \rangle\rangle \Box \left(\varphi \wedge \left[\bigwedge_{i \in Ag} \bigvee_{\alpha \in Ac_i} \ell(\alpha)\right]\right)$. By the semantics of ATL, we have $S^\circ, q_0 \models \langle\langle \mathit{controllers} \rangle\rangle \Box \left(\varphi \wedge \left[\bigwedge_{i \in Ag} \bigvee_{\alpha \in Ac_i} \ell(\alpha)\right]\right)$ iff $\exists \sigma_{controllers} \in \Sigma_{controllers}$ such that $\forall \lambda \in comp(\sigma_{controllers}, q_0)$, and $\forall u \in \mathbb{N}$, $S^\circ, \lambda[u] \models \varphi \wedge \left(\bigwedge_{i \in Ag} \bigvee_{\alpha \in Ac_i} \ell(\alpha)\right)$. We show how to construct such a $\sigma_{controllers}$. We construct from β a strategy $\sigma_{\alpha-controller} : Q \to Ac_i$ for each agent $i \in \mathit{controllers}$, as follows:

for each $i \in \mathit{controllers}$,

for each $q \in Q$,

$$\text{set } \sigma_{\alpha-controller}(q) = \left\{ \begin{array}{ll} \ell(\alpha) := \top \text{ if} & q \notin \beta(\alpha) \\ \ell(\alpha) := \bot \text{ if} & q \in \beta(\alpha) \end{array} \right\}.$$

Now, let $\sigma_{controllers}$ be the strategy profile for *controllers* such that the strategy for each $i \in \mathit{controllers}$, $\sigma_{\alpha-controller}$, is defined as above. We claim that, $\forall \lambda \in comp(\sigma_{controllers}, q_0)$ we have $S^\circ, \lambda[u] \models \varphi \wedge \left(\bigwedge_{i \in Ag} \bigvee_{\alpha \in Ac_i} \ell(\alpha)\right)$ for all $u \in \mathbb{N}$. The extra condition, $\bigwedge_{i \in Ag} \bigvee_{\alpha \in Ac_i} \ell(\alpha)$, comes from the definition of $\beta(\alpha)$, which requires it to be reasonable, in that it should always permit an agent to have at least one action left that can be performed in any state. Observe that by construction we have $comp(\sigma_{controllers}, q_0) \subseteq \Lambda$, as defined above. We know that $\forall \lambda \in \Lambda$, and for all $u \in \mathbb{N}$, we have $S \dagger \beta, \lambda[u] \models \varphi$. Since φ is propositional, its valuation depends only upon the state in which it is interpreted. Since π is unchanged in S° and $S \dagger \beta$, it must be that $\forall \lambda \in \Lambda$, $\forall u \in \mathbb{N}$: $S^\circ, \lambda[u] \models \varphi \wedge \left(\bigwedge_{i \in Ag} \bigvee_{\alpha \in Ac_i} \ell(\alpha)\right)$.

□

Notice that the obvious analogue of Theorem 10 also holds. To show this, we formulate the following:

Theorem 12 *Suppose $\upsilon \in \mathcal{L}^u$ is a universal* ATL *formula (representing an objective), and S is an* AATS. *Then $S^\circ, q_0 \models \langle\langle \mathit{controllers} \rangle\rangle \Box \left(\upsilon \wedge \left[\bigwedge_{i \in Ag} \bigvee_{\alpha \in Ac_i} \ell(\alpha)\right]\right)$ implies υ is feasible in S.*

Proof: (Sketch.) The basic structure of the proof is as Theorem 11, but as υ is no longer a propositional formula, but a universal ATL formula, we make use of Lemma 3. Again, as we are only proving in one direction, and universal ATL properties are preserved after implementing a behavioural constraint, π is the same for S and $S \dagger \beta$. □

Proposition 6 *Objective O_5 is not feasible in S_1, while both objectives O_6 and O_7 are feasible.*

We wish to prove that objective O_6 is feasible:

Proof: To prove that objective O_6 is feasible in system S_1 means showing that $S^\circ, q_0 \models \langle\langle controllers \rangle\rangle \Box \left((\neg(in_E \wedge in_W) \wedge \ell(move_E)) \wedge \left[\bigwedge_{i \in Ag} \bigvee_{\alpha \in Ac_i} \ell(\alpha) \right] \right)$ holds. By the semantics of ATL, this says that there exists a strategy for *controllers* such that it is always the case that $(\neg(in_E \wedge in_W) \wedge \ell(move_E)) \wedge \left[\bigwedge_{i \in Ag} \bigvee_{\alpha \in Ac_i} \ell(\alpha) \right]$ is true. It is not difficult to see that such a strategy exists. If the *controllers* always allow E to move and always allow W to idle, $\neg(in_E \wedge in_W) \wedge \ell(move_E)$ will always be true and the extra condition $\bigwedge_{i \in Ag} \bigvee_{\alpha \in Ac_i} \ell(\alpha)$, which says all agents in Ag will always have at least one legal action, will be true, as E can always move and W can always idle. □

Using this framework, we can also investigate more general properties of social laws. Consider behavioural constraints that act as *dictatorships* (cf. [39, p.17]). A behavioural constraint is a dictatorship if, once it is implemented, it never presents an agent with more than one possible action at any given time; that is, if an agent has no choice about what action it may perform. Let Λ denote the set of q_0-computations of $S \dagger \beta$. Formally, a behavioural constraint β with respect to an AATS S is a dictatorship if

$$\forall \lambda \in \Lambda, \forall i \in Ag, \forall u \in \mathbb{N}, |options(i, \lambda[u])| = 1$$

Example 12 *The following is a dictatorship behavioural constraint for the trains scenario:*

$$\beta_4(\alpha) = \begin{cases} \{q_0, q_4, q_6\} & \text{if } \alpha = idle_E \\ \{q_0, q_5, q_6\} & \text{if } \alpha = idle_W \\ \{q_5\} & \text{if } \alpha = move_E \\ \{q_4\} & \text{if } \alpha = move_W \end{cases}$$

Theorem 13 *Suppose φ is a propositional logic formula (representing an objective), and S is an AATS. Then φ is feasible in S by a dictatorship behavioural constraint iff*

$$S^\circ, q_0 \models \langle\langle controllers \rangle\rangle \Box \left(\varphi \wedge \bigwedge_{i \in Ag} \left[\bigvee_{\alpha \in Ac_i} \ell(\alpha) \right] \wedge \left[\bigwedge_{\alpha_1 \in Ac_i} \bigwedge_{\substack{\alpha_2 \in Ac_i, \\ \alpha_1 \neq \alpha_2}} \neg(\ell(\alpha_1) \wedge \ell(\alpha_2)) \right] \right).$$

Proof: (Sketch.) As before, except that we have an additional constraint, which ensures that no more than one action from an agent's action set is enabled at any given time. □

Proposition 7 *(O_2, β_4) and (O_3, β_4) are effective social laws in S_1.*

5.2 Summary

In this chapter, we have demonstrated how the Alternating-time Temporal Logic of Alur, Henzinger, and Kupferman provides a natural and powerful framework within which to express and reason about social laws. Following a formulation of social laws within ATL, we demonstrated how the effectiveness, feasibility, and synthesis problems could be understood as model checking problems for ATL, and also demonstrated how our basic framework could naturally be extended to include social laws that require explicit constraints on actions.

The next chapter builds upon the framework of social laws introduced here. We incorporate knowledge into the framework and investigate various epistemic properties in an example scenario.

Chapter 6

Knowledge and Social Laws

We have just shown that ATL provides a rich and natural technical framework within which to investigate social laws and their properties. However, the framework is based on quite a strong and unrealistic assumption that agents know everything about the state of the system. This is modelled with a global state space which all agents have access to. In this chapter we do away with the need for this assumption by extending the framework further by incorporating the notion of knowledge. Knowledge is important in any non-trivial multi-agent system, and as knowledge will come to play in almost any coordination scenario, we can clearly see a need to extend our framework. Not only will the agents' knowledge of the system be used in order to follow certain laws, but we can have laws in which certain information must become known (or remain private) to an agent or a coalition of agents. So, by imposing certain social laws, we can force desirable knowledge properties to hold.

Furthermore, we can have laws which are only known to a certain agent or group of agents, and only these agents need follow the law. In this chapter, we demonstrate how to 'transform' the objective of a social law into several, individual objectives, which have the following format: only if an agent knows certain information, can we require him to achieve some situation that is under his control. We say such an objective is *feasible* for the agent.

This chapter is structured as follows: We begin by introducing Action-based Alternating Epistemic Transition Systems (AAETSs), based on an essentially equivalent version of AATSs introduced in Chapter 4, but extended in order to represent the knowledge of the agents, in much the same way as Alternating Epistemic Transition Systems (AETS) introduced in Section 2.3. Next, we give some properties of ATEL in Section 6.2, and in Section 6.3 we formally define this new extended framework of social laws with respect to AAETS and ATEL. Then, in Section 6.4 we present a case study to illustrate the power of our framework and to show how, in some cases, model checking can be performed using standard ATL model checkers. Finally, we summarise in Section 6.5.

6.1 Semantic Structures

In Chapter 4 we introduced semantic structures known as AATSs. In this section we extend AATSs into what we call *Action-based Alternating Epistemic Transition Systems* (AAETSs). The main difference is that we introduce an *epistemic accessibility relation* for each agent in the system, which is used to capture each agent's knowledge. The approach is standard in the literature of epistemic logic, as seen in Section 2.2.

These structures are identical to AATSs introduced in Chapter 4, but now, for each agent $i \in Ag$, we also have an epistemic accessibility relation $\sim_i$, which represents indistinguishable states to agent i, used to capture i's knowledge. Thus, an AAETS is an $(2n+7)$-tuple

$$\langle Q, q_0, Ag, Ac_1, \ldots, Ac_n, \sim_1, \ldots, \sim_n, \rho, \tau, \Phi, \pi \rangle$$

where $\langle Q, q_0, Ag, Ac_1, \ldots, Ac_n, \rho, \tau, \Phi, \pi \rangle$ is an AATS, as defined in Chapter 4, and we have the following additional component for each agent:

- $\sim_i \subseteq Q \times Q$ is an epistemic accessibility relation for each agent $i \in Ag$. Each $\sim_i$ must be an equivalence relation.

The truth definition is defined in exactly the same way as for ATL in Section 2.1 for the non-epistemic ATEL formulae. The epistemic formulae, $K_i\varphi$, $E_G\varphi$ and $C_G\varphi$ are defined in the same way as in Section 2.2. We also define the operator $M_i\varphi$, which means i considers φ possible and is defined as an abbreviation of $\neg K_i \neg\varphi$.

6.2 Some Properties of ATEL

The same properties for ATL presented in Section 4.2.1 also hold for ATEL, the cooperation logic introduced in Section 2.3, where now the *universal* and *existential* sublanguages of ATEL, denoted $\mathcal{L}_k^u$ and $\mathcal{L}_k^e$, respectively are defined by the following grammars ($p \in \Phi$):

$$\begin{array}{lrl} \mathcal{L}_k^u & v ::= & p \mid \neg p \mid v \wedge v \mid v \vee v \mid K_i v \mid E_G v \mid C_G v \mid \\ & & \langle\langle\rangle\rangle \bigcirc v \mid \langle\langle\rangle\rangle \Diamond v \mid \langle\langle\rangle\rangle \Box v \mid \langle\langle\rangle\rangle v \mathcal{U} v \\ \mathcal{L}_k^e & \epsilon ::= & p \mid \neg p \mid \epsilon \wedge \epsilon \mid \epsilon \vee \epsilon \mid M_i \epsilon \mid \langle\langle Ag\rangle\rangle \bigcirc \epsilon \mid \\ & & \langle\langle Ag\rangle\rangle \Diamond \epsilon \mid \langle\langle Ag\rangle\rangle \Box \epsilon \mid \langle\langle Ag\rangle\rangle \epsilon \mathcal{U} \epsilon \end{array}$$

We now get the following result, analogous to Lemma 1 for ATL, which basically says that existential formulae are preserved when moving to a larger structure, whereas universal formulae are preserved when restricting the structure.

Lemma 5 *Let* $S' \sqsubseteq S, v \in \mathcal{L}_k^u, \epsilon \in \mathcal{L}_k^e$. *Then:*

$$S', q \models \epsilon \Rightarrow S, q \models \epsilon \text{ and } S, q \models v \Rightarrow S', q \models v$$

6.3 Knowledge and Social Laws

We now present our formal framework of social laws and demonstrate how knowledge is incorporated into the framework by investigating various knowledge properties. We now represent an objective for a social law as a formula of ATEL, with the same intuition that a social law is *effective* if it ensures that the objective is satisfied. ATEL inherits all the properties of ATL, allowing us to reason about what certain coalitions can bring about, as well as the ability to express liveness and safety properties from CTL, but now we can also express knowledge properties in order to reason about what agents and coalitions of agents should and shouldn't know.

Implementing a behavioural constraint on an AAETS is exactly the same as implementing a behavioural constraint on an AATS, as outlined in Section 5.1. This is because implementing a behavioural constraint causes no update on the accessibility relations, $\sim_i$, for each agent.

A *social law* over an AAETS is defined exactly as in Section 5.1 over an AATS, but now, the objective φ is an ATEL formula.

In Chapter 5, we proved that, for objectives expressed in ATL, the feasibility problem is NP-complete. We now show that, for ATEL objectives, the problem is no worse.

Theorem 14 *The feasibility problem for objectives expressed as arbitrary* ATEL *formulae is* NP-*complete.*

Proof: Since feasibility for ATEL objectives subsumes feasibility for ATL objectives, we only need to prove the upper bound. This follows from the fact that we can guess a behavioural constraint β, which will be polynomial in the size of $S = \langle Q, q_0, Ag, Ac_1, \ldots, Ac_n, \sim_1, \ldots, \sim_n, \rho, \tau, \Phi, \pi \rangle$, and then verify that $S \dagger \beta, q_0 \models \langle\langle\rangle\rangle \Box \varphi$. The process of generating $S \dagger \beta$ can easily be done in time polynomial in the size of S (we simply delete from S all transitions forbidden by β), and since model checking for ATEL is in P [55], the result follows. □

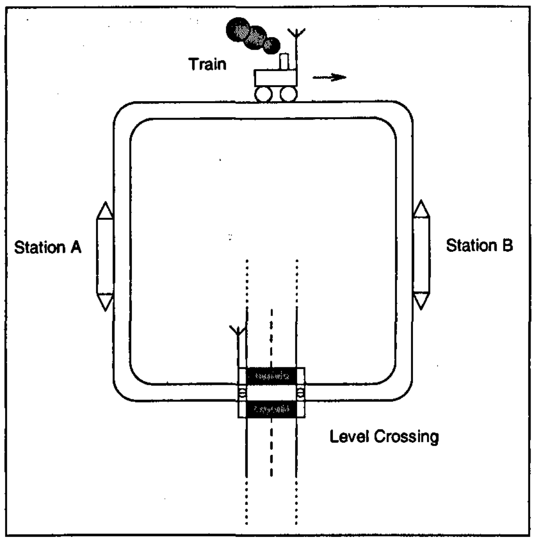

Figure 6.1: The Level Crossing system.

6.4 A Case Study

In this scenario[1], there is a train on a circular track, which at one point crosses a road (see Figure 6.1). The place where the track crosses the road (level crossing) is controlled by gates operated by a gate controller agent. If the train moves on to the level crossing while the gates are closed, the train will be in a crash situation, in which it has crashed into a car on the crossing. Also, if the gates close while the train is on the crossing, a crash will occur. We are interested in social laws that can prevent such situations from arising.

This system consists of two agents called t and g, where t is the agent representing the train and g is the gate controller agent. The train can be in one of four states, "staA" (train is at station A, which is the initial state of the train), "staB" (train stopped at station B), "wait" (waiting to use the crossing) and "oncr" (the train is on the crossing). The gates can only be in two states, either "open" (gates are open to the train) or "closed" (gates are closed to the train).

There is a communication medium by which the train and gate controller agent can exchange messages about their local states. The communication works by signals sent and received via an aerial on each of the agents. Signals received from the gates controller agent are stored in a local variable of the train called $Flag_g$. Similarly, signals received from the train are stored in a local variable of the gate controller agent called $Flag_t$.

[1]This case study is based on the example introduced earlier in Section 3.2, but now we introduce a communication medium through which the agents can exchange information.

The train has six actions available to it: $move_t$, $moveann_t$, $tell\text{-}away_t$, $tell\text{-}wait_t$, $tell\text{-}oncr_t$, and $idle_t$. The $idle_t$ action is the identity which causes no change in the train's state. If the train executes a $move_t$ action while it is at *staA*, then it goes to $staB_t$; executing $move_t$ while at $staB_t$ causes a transition to the $wait_t$ state; and, executing $move_t$ while $wait_t$ causes a transition to the $oncr_t$ state. Finally, executing $move_t$ while $oncr_t$ causes a transition to $staA_t$, as long as the gates were not in the $closed_g$ state at the same time as the train was in the $oncr_t$ state, as, if this is the case, the train is said to have crashed and is forced to $idle_t$ indefinitely.

The $moveann_t$ action causes the same state transitions as the $move_t$ action, but additionally the train sends its new location information to g. The $moveann_t$ action is always truthful about the new location of the train. The actions prefixed with *tell* have the effect of telling the gates that the location of the train is away, waiting, and on the crossing respectively. Note that, without any social law in place, it is not guaranteed that the train tells his location *truthfully* (e.g., performing $tell\text{-}wait_t$ when actually in state $oncr_t$)!

The gates controller has five actions available to it: $gates_g$, $gatesann_g$, $tell\text{-}open_g$, $tell\text{-}closed_g$, and $idle_g$. As with the train, this $idle_g$ action causes no change in the state of the gates. The $gates_g$ action causes the position of the gates to be toggled, i.e., performing $gates_g$ when the gates are $closed_g$ will result in the gates being $open_g$ and vice versa. The $gatesann_g$ action causes the same transitions as the $gates_g$ action, but the gate controller agent sends its new status to the train agent. The $gatesann_g$ is always truthful about the new status of the gates. The actions prefixed with *tell* have the effect of telling the train that the new status of the gates is $open_g$ and $closed_g$ respectively.

A state in our system is defined to be a tuple

$$\langle Loc_t, Flag_g, Status_g, Flag_t \rangle$$

where:

- $Loc_t \in \{staA_t, staB_t, wait_t, oncr_t\}$
- $Flag_g \in \{true, false\}$
- $Status_g \in \{open_g, closed_g\}$
- $Flag_t \in \{a, w, o\}$

We assume that we have atoms in the object language (like $Flag_t = w$) corresponding to these states. We will refer to this tuple generally as $\langle W, X, Y, Z \rangle$. We let q_i denote the ith component of q. Now, $\langle q_1, q_2, q_4 \rangle$ is the local state of agent t and $\langle q_2, q_3, q_4 \rangle$ is the local state of agent g. The initial state (or root) of our system is $\rho = \langle staA_t, false, closed_g, a \rangle$.

Given two states $q, r \in Q$, then the epistemic accessibility relation for agent t is given as: $q \sim_t r$, iff $q_1 = r_1$, $q_2 = r_2$ and $q_4 = r_4$. The epistemic accessibility relation for agent g is given as: $q \sim_g r$, iff $q_2 = r_2$, $q_3 = r_3$ and $q_4 = r_4$. Let us call this overall system S. The transitions are given in Figure 6.2, where the first line for instance represents that a $move_t$ in any state $\langle staA_t, X, Y, Z\rangle$, when done jointly with any action ac_g of the gates, results in any of $\langle staB_t, X', Y', Z\rangle$ states, where the precise X' and Y' values (denoting the state of the gates), depend on X, Y, and ac_g, the location of the train changes from station A to station B, and the value of $Flag_t$, stored in the variable Z, remains the same.

Current State	Actions	Resulting States
$staA_t, X, Y, Z$	$move_t, ac_g$	$staB_t, X', Y', Z$
$staB_t, X, Y, Z$	$move_t, ac_g$	$wait_t, X', Y', Z$
$wait_t, X, Y, Z$	$move_t, ac_g$	$oncr_t, X', Y', Z$
$oncr_t, X, open_g, Z$	$move_t, ac_g$	$staA_t, X', Y', Z$
$oncr_t, X, closed_g, Z$	$move_t, ac_g$	–
$W, X, closed_g, Z$	$ac_t, gates_g$	$W', X, open_g, Z'$
$W, X, open_g, Z$	$ac_t, gates_g$	$W', X, closed_g, Z'$
W, X, Y, Z	$idle_t, ac_g$	W, X', Y', Z
W, X, Y, Z	$ac_t, idle_g$	W', X, Y, Z'
$staA_t, X, Y, Z$	$moveann_t, ac_g$	$staB_t, X', Y', a$
$staB_t, X, Y, Z$	$moveann_t, ac_g$	$wait_t, X', Y', w$
$wait_t, X, Y, Z$	$moveann_t, ac_g$	$oncr_t, X', Y', o$
$oncr_t, X, open_g, Z$	$moveann_t, ac_g$	$staA_t, X', Y', a$
$oncr_t, X, closed_g, Z$	$moveann_t, ac_g$	–
$W, X, closed_g, Z$	$ac_t, gatesann_g$	$W', true, open_g, Z'$
$W, X, open_g, Z$	$ac_t, gatesann_g$	$W', false, closed_g, Z'$
W, X, Y, Z	$tell\text{-}away_t, ac_g$	W, X', Y', a
W, X, Y, Z	$tell\text{-}wait_t, ac_g$	W, X', Y', w
W, X, Y, Z	$tell\text{-}oncr_t, ac_g$	W, X', Y', o
W, X, Y, Z	$ac_t, tell\text{-}closed_g$	$W', false, Y, Z'$
W, X, Y, Z	$ac_t, tell\text{-}open_g$	$W', true, Y, Z'$

Figure 6.2: State Transitions

6.4.1 Knowledge Properties

There are certain knowledge properties that are of interest to us. The first property we look at is called a *knowledge pre-condition*. In [57], a knowledge pre-condition for a particular plan is said to be: *the information an agent must have in order to be able to successfully carry* *this plan out.* Their first attempt to formulate a knowledge pre-condition in ATEL is of the form $\langle\langle a\rangle\rangle\Diamond\varphi \rightarrow K_a\psi$, which specifies the requirement that, in order for agent a to be able to eventually bring about state of affairs φ, it must know ψ. This requirement is too strong. It says that knowing ψ is a *necessary* requirement to bring about φ. But bringing about φ in the future does not mean I have to know ψ right now. We use a slightly different definition for knowledge pre-conditions. We say that a knowledge pre-condition for an action or plan is the information that is *sufficient* for an agent to be able to successfully carry out the action or plan. Knowledge pre-condition formulae take the form $K_i\varphi \rightarrow \psi$. This formula states that agent i knowing the fact φ implies that the fact (state of affairs) ψ should hold. These knowledge pre-condition formulae can be used as ATEL objectives in our social laws. Let us define $[\![i]\!]\varphi$ as the dual of $\langle\langle i\rangle\rangle\varphi$. Remember that $\langle\langle i\rangle\rangle\varphi$ means that there exists a strategy for agent i, such that he can achieve φ, no matter what the other agents do. So, $[\![i]\!]\varphi$ means that no matter what i does, there exists a strategy for the other agents in the system to make φ true (whichever strategy i choses, the others can complement it in such a way that φ).

Then, a property like

$$K_i\varphi \rightarrow [\![i]\!](\bigcirc\psi \vee \bigcirc\bigcirc\psi \vee \bigcirc\bigcirc\bigcirc\psi) \tag{6.1}$$

expresses in ATEL* (cf. ATL* [2]) that, if the gate-controller (i) on a crossing knows that the train is waiting to cross (φ), then, whatever the gates do, the train should be able to safely cross (ψ) in the 'near future' (i.e., within the next three time steps). We can extend this example immediately to cases where ψ is itself epistemic: if the train (j) knows that the constraint (6.1) applies, and he wants to safely cross, he should be able to notify the the gate-controller of this:

$$K_j\varphi \rightarrow \langle\langle j\rangle\rangle\bigcirc K_i\varphi \tag{6.2}$$

Implied knowledge is knowledge that comes about as a result of the performance of actions, which lead to a change in the current state of affairs: this type of knowledge property takes the form $\varphi \rightarrow K_i\psi$. This formula states that the state of affairs φ should bring about the knowledge of the fact ψ to agent i.

This type of formula can also be used in the objectives to our social laws. For example, the following could express that everytime that train i is able to pass a crossing, the gate at that crossing should know it.

$$\langle\langle i\rangle\rangle\bigcirc\varphi \rightarrow K_j\varphi \tag{6.3}$$

Finally, nested knowledge is information that the agents have about what other agents know. For instance, $[\![i]\!]\Diamond\psi \rightarrow K_iE_G\varphi$ expresses that if i cannot ensure that ψ will ever become true, he should make sure (know) that everybody in G knows this. Nested knowledge can also play the role of a pre-condition (I should know that you know we have a meeting at the station, before I go there, for instance).

6.4.2 Epistemic Social Laws

Now we can formulate some objectives for social laws. We will begin by constructing objectives which have the general format $K_i\varphi \rightarrow \psi$, where ψ is some property under control of agent i. This format makes perfect sense: (only) if an agent knows certain preconditions, can we require him to take appropriate action. Let us hence say that such an objective is *feasible for agent i.*

$$O_1 = K_t wait_t \rightarrow (Flag_t = w) \tag{6.4}$$

The objective O_1 is equivalent to $O'_1 = wait_t \rightarrow (Flag_t = w)$ (if this is not immediately clear, it should become clear once we have discussed the notion of a *local proposition*). Property (6.4) denotes that the train is *accurate* with respect to recording his waiting state. We give a 'weakest' constraint β_1 that implements this social law, which can be verified by checking $S \dagger \beta_1, \rho \models \langle\langle\rangle\rangle \Box O'_1$. The constraint β_1 works as follows: in $\langle staB_t, X, Y, Z\rangle$, if $Z \neq w$, the action for t forbidden by β_1 is *move* (since this would inaccurately leave the $Flag_t$ as a or o); moreover, in any state $\langle wait_t, X, Y, Z\rangle$, constraint β_1 forbids t to perform any *tell* action, except when it is *tell-wait*. Loosely speaking: the train is accurate about his waiting, if he announces it when he starts to wait, and, once waiting, never tells anything otherwise. We claim that moreover β_1 is a weakest constraint for its aims: any constraint β that forbids less than β_1 has the property that $S \dagger \beta, \rho \models \langle\langle\{t,g\}\rangle\rangle \Diamond \neg O_1$.

The converse of (6.4) would mean that the train is *truthful* with respect to the waiting state:

$$O_2 = K_t(Flag_t = w) \rightarrow wait_t \tag{6.5}$$

Again, this property is equal to $O'_2 = (Flag_t = w) \rightarrow wait_t$. The behavioural constraint, β_2, that makes this social law effective, forbids t to falsely suggest that it is waiting: t is only allowed to perform *tell-wait*$_t$ in any $\langle wait_t, X, Y, Z\rangle$, moreover, β_2 forbids t to perform a $move_t$ in $\langle wait_t, Y, Z, w\rangle$, since otherwise it would falsely suggest that it is still waiting (of course, the train can still leave this state truthfully, by performing a $moveann_t$-action). The fact that indeed β_2 implements O'_2 is verified by showing $S \dagger \beta_2, \rho \models \langle\langle\rangle\rangle \Box O'_2$. As an aside, note that the constraint β_3 which *never* allows the train to do a $move_t$ action and only allows t to perform *tell-wait* when $wait_t$ is true, is a way to implement the objective $O'_2 = ((Flag_t = w) \rightarrow wait_t)$.

Now we observe the following: $S \dagger \beta_1 \dagger \beta_2, q \models wait_t \rightarrow K_g wait_t$. This is interesting, since to make the gates know that the train is waiting, has become a train-feasible objective in $S \dagger \beta_1 \dagger \beta_2$: he just has to accurately and truthfully set $Flag_t$ to w!

Now consider the following objective:

$$O = \langle\langle\rangle\rangle \Box (\neg(oncr_t \wedge closed_g) \wedge \langle\langle g, t\rangle\rangle \Diamond onct_t) \tag{6.6}$$

This objective combines a safety property (the train is never on the crossing while the gates are closed) with a liveness property (the train can eventually pass the crossing). The question is whether we can 'break' this objective 'down' into a number of constraints that are feasible for t or g. Here is a high level description: Let O' be the conjunction of:

$$\begin{array}{ll} (K_t wait_t \rightarrow K_g wait_t) & (K_g wait_t \rightarrow [\![g]\!] \Diamond open_g) \\ (K_g open_g \rightarrow K_t open_g) & (K_t open_g \rightarrow \langle\langle t\rangle\rangle \Diamond oncr_t) \\ (K_g closed_g \rightarrow K_t closed_g) & (K_t closed_g \rightarrow [\![t]\!] \Box \neg oncr_t) \end{array}$$

Objective O' states that t should inform g about waiting, and g should inform t about the states of the gates. Recall that $[\![i]\!] \Diamond \varphi$ means that the other agents can ensure $\Diamond \varphi$, and, if φ is 'under the control' of i, then $[\![i]\!] \Diamond \varphi$ means that i cannot avoid that φ will eventually be true. Keeping this in mind, O' then also requires that if g knows that t is waiting, it cannot but avoid that eventually the gates will be open; if t knows the gates to be open, it will eventually pass the crossing, and, finally, if t knows the gates are closed, it will never attempt to cross. We claim that O' can be indeed turned into a set of feasible laws for g and t, and also that the implemented law for O' guarantees O.

We demonstrated how to make the first implication feasible for t by imposing suitable constraints; the same can be done for the other implications. We will now demonstrate how to actually model-check such knowledge properties.

6.4.3 Model Checking Some Properties

There are certain knowledge properties that need to be satisfied in the level crossing example. We obtain them using social laws. In order to test knowledge properties, we used the model checker, MOCHA, as introduced earlier in Section 2.1. We programmed the system in the REACTIVE MODULES language and also have modified versions of the system, which incorporate social laws. It is important to note that we are not adding to the theory of model checking, simply making use of it.

The first knowledge property that we wish to investigate is that the gates always know when the train is waiting, in the system updated with the behavourial constraint β_2, described earlier. More formally, we want to verify that, given a state q in which $q \models wait_t$, we also have

$q \models K_g wait_t$. The intuition behind this knowledge property is that if the train is waiting at the gates for them to be opened, this can only happen if the gate controller agent knows that the train is waiting.

Now we wish to verify properties involving knowledge by using a standard model checker that does not deal with epistemic operators, in our case MOCHA ([1]). To do this, we employ the machinery of *local propositions*, as introduced in [15] and applied to model checking epistemic properties in the context of linear temporal logic in [54]. We give an informal explanation; for details the reader should consult [54].

A proposition φ is i-local in a system S if

$$\forall q, q' \in Q : ((q \sim_i q') \Rightarrow (S, q \models \varphi \Leftrightarrow S, q' \models \varphi))$$

that is, an i-local proposition never changes truth value *within* an i-equivalence class of states. This formalises the idea that such propositions depend on what the agent can observe. We immediately see that in the train and gates example, every proposition about the location of the train and the flag of the gates is t-local, likewise are assertions about the status of the gates and the flag of the train local for the gates. Now, generalising the linear temporal logic analysis from [54] to the branching time context of this chapter, we have the following:

Theorem 15 *Suppose that φ is i-local. Then $S, q \models \varphi$ and $S, q_0 \models \langle\langle\rangle\rangle\Box(\varphi \rightarrow \psi)$ are sufficient to prove both $S, q \models K_i\psi$ and $S \models \varphi \rightarrow K_i\psi$. In such a case, φ is called i-local for ψ.*

The above definition of a proposition being i-local shows that the laws O_2 and O'_2 are equivalent. To show that $q \models K_g wait_t$ in any waiting state q, we must first find a g-local proposition for $wait_t$. We can take $(Flag_t = w)$: To show that $(Flag_t = w)$ is g-local in $S \dagger \beta_2$, we need to verify that $\forall q, r((q \sim_g r) \Rightarrow (q \models Flag_t = w \Leftrightarrow r \models Flag_t = w))$, which is obvious, since $Flag_t$ is part of g's local state. Now, to apply Theorem 15, we must show that $S \dagger \beta_2, \rho \models \langle\langle\rangle\rangle \Box(Flag_t = w) \rightarrow wait_t$ holds. We do this by model checking the following MOCHA ATL specification in $S \dagger \beta_2$:

```
<<>> G ((tFlag = w) => (tState = wait))
```

Since the answer to this is positive, we have verified the desired property in $S \dagger \beta_2$. Note that this is relative to a state q. We now show how we can check this property across *all* states of the system. The only thing we required in q is that $q \models wait_t$. So now we can check the following property across all states, in $S \dagger \beta_2$:

$$wait_t \rightarrow K_g wait_t \tag{6.7}$$

Now we model check the following:

```
<<>> G ((tState = wait) => (tFlag = w))
```

When this formula is model checked in the original system S it fails: This is as expected, as there is nothing in-built in S to guarantee that the train is truthful. However, model checking the above formula in $S \dagger \beta_2$ passes, which shows that this social law is effective.

Now we look at a nested knowledge formula:

$$wait_t \rightarrow K_t K_g wait_t \tag{6.8}$$

We briefly sketch how to do this in the system that is constrained with β_3. By Theorem 15, it is sufficient to show that $wait_t$ is t-local for $K_g wait_t$. That is, $wait_t$ is t-local (which is obviously the case) and $S \dagger \beta_3, \rho \models \langle\langle\rangle\rangle\Box(wait_t \rightarrow K_g wait_t)$. This can be either model checked or established from (6.7), the fact that $S\dagger\beta_3 \sqsubseteq S\dagger\beta_2$ and Lemma 5. We can now apply Theorem 15 (take $\varphi = wait_t$ and $\psi = K_g wait_t$), to conclude that $S \dagger \beta_3 \models wait_t \rightarrow K_t K_g wait_t$.

The social laws imposed on our system are quite restrictive in that agent t knows everything about the state of agent g, and agent g knows a lot about the state of agent t. The knowledge that each of the agents have is *necessary* to ensure the system runs efficiently and that no crash situations occur. However, when the train is at $staA_t$ or $staB_t$, it is not beneficial to the system that agent g should know which of the two states the train is at, only that the train is away, in order to close the gates. We show this by investigating the following property in S:

$$staA_t \rightarrow M_g \neg staA_t \tag{6.9}$$

This property states that if the train is at station A, agent g *considers it possible* that in fact, the train is *not* at station A. To check whether (6.9) holds across all states of our system S, we model check the following formula:

$$\bigwedge_{x,y} (\langle\langle\rangle\rangle \Box [(staA_t \wedge x \wedge y \wedge z) \rightarrow \langle\langle t, g\rangle\rangle \bigcirc (\neg staA_t \wedge x \wedge y \wedge z)]) \tag{6.10}$$

where the conjunction is taken over all

$$x \in \{open_g, closed_g\},$$

$$y \in \{Flag_t = a, Flag_t = w, Flag_t = o\},$$

and z is either $Flag_g$ or $\neg Flag_g$.

This requires performing twelve different model checking problems, for each of the different values of x, y and z.

To understand that (6.10) is indeed sufficient to show (6.9), we first strengthen (6.9) to:

$$S, \rho \models \langle\langle\rangle\rangle \Box (staA_t \rightarrow M_g \neg staA_t) \tag{6.11}$$

Now, suppose (6.11) is false. This means

$$S, \rho \models \langle\langle t, g\rangle\rangle \Diamond (staA_t \wedge K_g staA_t)$$

So for some q, $S, q \models staA_t \wedge K_g staA_t$. Suppose q is $\langle W, true, open_g, a\rangle$, so $S, q \models x \wedge y \wedge z$. Applying (6.10) we also find a q' with $S, q' \models \neg staA_t \wedge x \wedge y \wedge z$. For g, q and q' are similar: $q \sim_g q'$. Since $S, q \models K_g staA_t$, we have $S, q' \models K_g staA_t$, and $S, q' \models staA_t$, which is absurd. Hence, (6.11) holds.

After performing this model checking, as the formula passed each time, we have shown that (6.9) holds. Finally notice that even if the constraints concerning accuracy and truthfulness of both agents are implemented, (6.9) holds, since the most specific information that t will give when being at station A or B will be that he is 'away'.

6.5 Summary

In this chapter we have shown how our social laws framework presented in Chapter 5 can naturally be extended to incorporate the notion of knowledge. In order to express such social laws we use the language of ATEL, essentially ATL with epistemic extensions. We demonstrated the power of such a framework with the use of a case study, in which many interesting knowledge properties were investigated. In particular, we showed how overall objectives of the system can be broken down into feasible properties for the agents, involving laws for each agent i of type $K_i\varphi \rightarrow \psi$, where ψ is under the control of i.

We also demonstrated how several different types of knowledge, including knowledge preconditions, implied knowledge, and nested knowledge can be verified using an ATL model checker (MOCHA). We showed how ATEL formulae can be reduced to ATL formulae with the use of local propositions substituted for knowledge. This allowed us to verify such properties with a standard model checker that does not deal with epistemic operators.

Chapter 7

Social Laws, Social Belief, and Anti-Social Behaviour

In this chapter we extend our social laws framework further. So far, in our framework, it is assumed that once social laws are imposed on the system, all the agents will abide by these laws. However, this does not seem to be the most realistic way of modelling social laws. Certainly in human societies, just because laws are imposed does not mean that all members of the society will follow these laws. In this chapter we do away with the need for this assumption and give agents the choice of whether to follow these laws or not. We make a distinction between agents acting *physically* and agents acting *socially*. Acting physically —we use this in lack of a better term we know— corresponds to performing any action that is physically possible to be performed, in the sense of possible by the system description. Acting socially then corresponds to performing any action from a subset of these physical actions, known as the social actions. Social actions are those that are consistent with the social laws imposed on the system. An example scenario is in the case of traffic laws. Traffic lights are used to coordinate the flow of traffic on our roads to ensure no collisions occur when cars cross each other's paths. Cars are only *allowed* to move from the traffic lights when the light is green. Acting socially (and physically) would correspond to only moving when the lights are green. However, acting physically (but not socially) could correspond to moving when the lights are red. This would be an illegal action, but still a physically possible action. It is important to note that all social actions are also physically possible actions.

Agents now have both physical and social strategies, where the set of social strategies is always a subset of the physical strategies. So after giving the agents these extra possibilities, we need to extend the logic to be able to reason about whether an agent or coalition of agents is able to act socially or physically (of course if an agent is able to act physically, this agent can choose to only perform social strategies if it desires). We introduce a new logical language, *Social* ATEL (SATEL), which allows us to express properties like the following: "Even if the coalition abides by the laws, and all the other agents neglect them, our coalition can achieve a particular temporal property", or "the server knows that, where all the clients are acting socially, then eventually each granted write-permission will terminate".

We make another extension to the framework by adding the notion of social belief. Now that agents have the possibility of not following these social laws, a new type of belief arises. This is belief based on the assumption that all the agents in the system are acting according to the social laws imposed. We call this type of belief "*social belief*". Using social belief and knowledge together seems more realistic, but may also have other advantages. If an agent has to act in a timely fashion, it may be preferable for the agent to act sooner on its social beliefs rather than waiting longer to acquire the knowledge that is important for the agent to act. Of course, in situations with drastic consequences the agent may be required to act on knowledge rather than social belief. We incorporate the notion of social belief and show how it can play an important role in coordinating the activities of agents.

This chapter is structured as follows: We first introduce the semantic structures that our model is based on. We call these Social Action-based Alternating Epistemic Transition Systems (SAAETS). These are similar to AAETSs introduced in the previous chapter (Chapter 6) with a few extensions. We then introduce our logical language, Social ATEL (SATEL), which is an extension of ATEL (introduced in Section 2.3), and give the semantics of this language. In Section 7.3 we introduce a case study known as the Alternating Bit Protocol and go on to investigate various interesting properties of this model. Finally, we conclude in Section 7.4.

7.1 Semantic Structures (SAAETS)

In this section we introduce the semantic structures our model is based upon. The structures we use are known as Social Action-Based Alternating Epistemic Transition Systems (SAAETSs). Our structures are most similar to the AAETSs we introduced in the previous chapter. However, our structures differ from those in several ways. Instead of having one action pre-condition function, ρ, we now have both a physical action precondition function, ρ, and a legal action precondition function, ℓ. Also, where the emphasis in Chapter 5 is on implementing a social law on a system, we do not consider such updates: rather, we assume the ℓ function is given, constraining the set of possible transitions, and we are not concerned about which social goal such constraints are supposed to achieve. This could prove to be a limitation of our framework. In making decisions about whether to follow the law or not, agents will need to know what the law is in place to achieve. As there is no notion of an objective for a social law in this extension to the framework, such reasoning is not possible. However, this framework is used to investigate social laws at the system level. Hence the system designer is able to see which properties hold depending on who follows the social laws. Future work could incorporate objectives of social laws into this framework.

Formally, an SAAETS is a $(2n+8)$-tuple

$$\langle Q, q_0, Ag, Ac_1, \ldots, Ac_n, \sim_1, \ldots, \sim_n, \rho, \ell, \tau, \Phi, \pi \rangle$$

where $\langle Q, q_0, Ag, Ac_1, \ldots, Ac_n, \sim_1, \ldots, \sim_n, \rho, \tau, \Phi, \pi \rangle$ is an AAETS, as defined in Chapter 6, but now the following components are either new or altered:

- $\rho : Ac_{Ag} \to 2^Q$ is a *physical action precondition function*, which for each action $\alpha \in Ac_{Ag}$ defines the set of states $\rho(\alpha)$ from which α may be *physically* performed; and,
- $\ell : Ac_{Ag} \to 2^Q$ is a *legal action precondition function*, which for each action $\alpha \in Ac_{Ag}$ defines the set of states $\ell(\alpha)$ from which α may be *physically* and *legally* performed (for all $\alpha \in Ac_{Ag}$, $\ell(\alpha) \subseteq \rho(\alpha)$).

As with AATSs, SAAETSs must satisfy two coherence constraints. These constraints differ slightly from those outlined in Chapter 4: Firstly, *non-triviality [35]*: Agents always have at least one *legal* action: $\forall q \in Q, \forall i \in Ag, \exists \alpha \in Ac_i$ s.t. $q \in \ell(\alpha)$. Secondly, *consistency*: The ρ and τ functions agree on actions that may be performed: $\forall q, \forall j \in J_{Ag}, (q, j) \in \text{dom}\ \tau$ iff $\forall i \in Ag, q \in \rho(j_i)$.

Given an agent $i \in Ag$ and a state $q \in Q$, we denote the *physical options* available to i in q by

$$\rho\text{-}options(i, q) = \{\alpha \mid \alpha \in Ac_i \text{ and } q \in \rho(\alpha)\}$$

and we denote the *legal options* available to i in q by

$$\ell\text{-}options(i, q) = \{\alpha \mid \alpha \in Ac_i \text{ and } q \in \ell(\alpha)\}$$

Now we can define a *physical strategy* and a *legal strategy* for an agent. A *physical strategy* for an agent $i \in Ag$ is a function: $\gamma_i : Q \to Ac_i$ which must satisfy the constraint that $\gamma_i(q) \in \rho$-*options*(i, q) for all $q \in Q$. A *legal strategy* for an agent $i \in Ag$ is a function: $\delta_i : Q \to Ac_i$ which must satisfy the *legality* constraint that $\delta_i(q) \in \ell$-*options*(i, q) for all $q \in Q$.

A *physical strategy profile* for a coalition $G = \{1, \ldots, k\} \subseteq Ag$ is a tuple of physical strategies $\langle \gamma_1, \ldots, \gamma_k \rangle$, one for each agent $i \in G$. Similarly, a *legal strategy profile* for a coalition $G = \{1, \ldots, k\} \subseteq Ag$ is a tuple of legal strategies $\langle \delta_1, \ldots, \delta_k \rangle$, one for each agent $i \in G$. By Δ_G we denote the set of all *legal* strategy profiles for coalition G and by Γ_G, we denote the set of all *physical* strategy profiles for coalition G. By σ_G, we denote a strategy profile for coalition G where we are not concerned about whether the strategy profile is legal or physical; if $\sigma_G \in \Gamma_{Ag}$ and $i \in Ag$, then we denote i's component of σ_G by σ^i. A grand coalition strategy profile is defined as a tuple $\langle \sigma_G, \sigma'_{\bar{G}} \rangle$, $\forall \sigma_G, \sigma'_{\bar{G}}$, where σ_G represents the choices made by agents in G and $\sigma'_{\bar{G}}$ represents the choices made by all other agents in the system.

Given a strategy profile σ_{Ag} and state $q \in Q$, let $out(\sigma_{Ag}, q)$ denote the next state that will result by the members of Ag acting as defined by their components of σ_{Ag} for one step from q. Formally, $out(\sigma_{Ag}, q) = \tau(q, j) = q'$, where $\sigma^i_{Ag}(q) = j_i$ for $i \in Ag$. This function is different from $out(\sigma_G, q)$ defined in Chapter 4, as now we are only concerned with grand coalition strategy profiles, hence the outcome is a state rather than a set of states.

Given a strategy profile σ_{Ag} for the grand coalition Ag, and a state $q \in Q$, we define $comp(\sigma_{Ag}, q)$ to be the run that will occur if every agent $i \in Ag$ follows the corresponding strategy σ_i, starting when the system is in state $q \in Q$. Formally, $comp(\sigma_{Ag}, q) = \lambda$ where $\lambda[0] = q$ and $\forall u \in \mathbb{N} : \lambda[u+1] = out(\sigma_{Ag}, \lambda[u])$. This function also differs from the $comp(\sigma_G, q)$ function defined in Chapter 4 for the same reasons outlined above.

Given an SAAETS, S, and the initial state of the system, q_0, we define a set of *socially reachable states* as follows:

$$sreach(S) = \{q \mid \exists \delta_{Ag} \in \Delta_{Ag} \text{ and } \exists u \in \mathbb{N} \text{ s.t. } q = comp(\delta_{Ag}, q_0)[u]\}$$

This function, $sreach(S)$, defines the set of socially reachable states in a system S, starting from the initial state, q_0. The socially reachable states are all the states which are reachable when every agent in the system performs only social strategies.

The following is immediate by construction.

Lemma 6 *For any coalition of agents, G, it is always the case that $\Delta_G \subseteq \Gamma_G$.*

7.2 Social ATEL

In this section we define the logical language used to express social laws in our framework. Our language is essentially an extension of ATEL, introduced in Section 2.3. In SATEL, we reason about physical and about social strategies. Our modalities are of the form $\langle\langle G \rangle\rangle^y_x$, where x and y denote which kind of strategies G and $Ag \setminus G$, respectively, are allowed to use: only social strategies (denoted by s), or all their available ones (denoted by p). For example, the formula $\langle\langle G \rangle\rangle^s_p \Diamond goal$ expresses that there exists a strategy for the coalition G, such that, no matter what the other agents do, providing they only follow social strategies, at some point in the future G can achieve some *goal* state. We can also require all the agents to act socially, e.g.,

$\langle\langle G\rangle\rangle_s^s \Box\neg \mathit{fail}$, which expresses that G has a social strategy, such that, no matter what the other agents do, providing they act socially, the system will never enter a *fail* state. Finally, consider the nested formula, $\langle\langle G\rangle\rangle_s^s \bigcirc \langle\langle G\rangle\rangle_s^p \Box\varphi$, which reads: "$G$ can ensure, by using a social strategy, and assuming that all the others also act socially, that in the next state, G can ensure, again by acting socially, that even if the others from now on act non-socially, φ will always hold". Or, a bit more informally: "if we require G to act socially, and the others socially for at least one step, but unconstrained thereafter, then G can guarantee that always φ".

ATEL adds knowledge operators on top of ATL. However, on top of that we add operators to express more enhanced informational attitudes. First of all, now that the possibility of violating social laws exists, we define a notion of *social belief*, i.e., belief under the assumption that *all* agents in the system are acting as they should do according to the social laws imposed. We introduce a belief operator SB_i, where $SB_i\varphi$ expresses that, if i assumes that everybody acts socially, he believes φ to be the case. For example, if we use φ to denote the fact that a car is stopped at a red traffic light, $SB_i\varphi$ means that agent i believes that a car is stopped at a red traffic light, under the assumption that all agents in the system are acting socially. So in all indistinguishable *social* states to agent i, a car is stopped at a red traffic light. An obvious difference with the standard notion of knowledge is that social belief is not veridical: it is perfectly well possible that agent i socially believes φ ($SB_i\varphi$) and $\neg\varphi$ at the same time.

Formally, the set of formulae, formed with respect to a set of agents Ag, and a set of primitive propositions Φ, is given by the following grammar:

$$\begin{aligned}\varphi \quad ::= \quad & p \mid \neg\varphi \mid \varphi \vee \varphi \mid K_i\varphi \mid SB_i\varphi \mid \langle\langle G\rangle\rangle_x^y \bigcirc \varphi \\ & \langle\langle G\rangle\rangle_x^y \Box\varphi \mid \langle\langle G\rangle\rangle_x^y \varphi \,\mathcal{U}\, \varphi\end{aligned}$$

where $p \in \Phi$ is a propositional variable, $G \subseteq Ag$ is a set of agents, $i \in Ag$ is an agent, and x and y can be either p or s.

We now give the truth definition of Social ATEL formulae on an SAAETS S and a state q:

$S, q \models p$ iff $p \in \pi(q)$ (where $p \in \Phi$);

$S, q \models \neg\varphi$ iff $S, q \not\models \varphi$;

$S, q \models \varphi \vee \psi$ iff $S, q \models \varphi$ or $S, q \models \psi$;

$S, q \models \langle\langle G\rangle\rangle_p^p \bigcirc \varphi$ iff $\exists \sigma_G \in \Gamma_G$ s.t. $\forall \sigma_{\bar{G}} \in \Gamma_{\bar{G}}$, if $\lambda = comp(\langle \sigma_G, \sigma_{\bar{G}} \rangle, q)$, we have $S, \lambda[1] \models \varphi$;

$S, q \models \langle\langle G\rangle\rangle_s^p \bigcirc \varphi$ iff $\exists \sigma_G \in \Delta_G$ s.t. $\forall \sigma_{\bar{G}} \in \Gamma_{\bar{G}}$, if $\lambda = comp(\langle \sigma_G, \sigma_{\bar{G}} \rangle, q)$, we have $S, \lambda[1] \models \varphi$;

$S, q \models \langle\langle G \rangle\rangle_p^s \bigcirc \varphi$ iff $\exists \sigma_G \in \Gamma_G$ s.t. $\forall \sigma_{\bar{G}} \in \Delta_{\bar{G}}$, if $\lambda = comp(\langle \sigma_G, \sigma_{\bar{G}} \rangle, q)$, we have $S, \lambda[1] \models \varphi$;

$S, q \models \langle\langle G \rangle\rangle_s^s \bigcirc \varphi$ iff $\exists \sigma_G \in \Delta_G$ s.t. $\forall \sigma_{\bar{G}} \in \Delta_{\bar{G}}$, if $\lambda = comp(\langle \sigma_G, \sigma_{\bar{G}} \rangle, q)$, we have $S, \lambda[1] \models \varphi$;

$S, q \models K_i \varphi$ iff for all q' such that $q \sim_i q' : S, q' \models \varphi$;

$S, q \models SB_i \varphi$ iff for all $q' \in sreach(S)$ such that $q \sim_i q'$, we have $S, q' \models \varphi$.

We omit the definitions of $\langle\langle G \rangle\rangle_x^y \Box$ and $\langle\langle G \rangle\rangle_x^y \varphi \mathcal{U} \psi$, as once you understand the pattern between p and s and the type of strategy used, these definitions easily follow from the truth definition of ATL, given before in Section 2.1. The other connectives ("$\wedge$", "$\rightarrow$", "$\leftrightarrow$") are assumed to be defined as abbreviations in terms of $\neg, \vee$. $\langle\langle G \rangle\rangle_x^y \Diamond \varphi$ is shorthand for $\langle\langle G \rangle\rangle_x^y \top \mathcal{U} \varphi$. We write $\langle\langle i \rangle\rangle$ rather than $\langle\langle \{i\} \rangle\rangle$.

In ATL, the cooperation modality $\langle\langle\rangle\rangle T$ denotes a special case: it means that the empty set of agents has a strategy, such that, no matter what the other (i.e., *all*) agents do, T holds. In other words, no matter what the agents in Ag do, T. This resembles the CTL operator AT: on every future path, T. Similarly, $\langle\langle Ag \rangle\rangle T$ means that the grand coalition has a strategy such that, no matter what the empty coalition does, T. In CTL terminology: ET, or, for some path, T. For SATEL this gives us the following. Since in $\langle\langle\rangle\rangle_x^y$, the constraint to play an x-type of strategy is a constraint for nobody, it does not really matter whether x is s or p. Similarly, in $\langle\langle Ag \rangle\rangle_x^y T$ the constraint to play a y-type of strategy is a void restriction for $Ag \setminus Ag$, i.e., the empty set, so it does not matter whether x equals s or equals p. Summarising, we have

$$\langle\langle\rangle\rangle_s^y T \equiv \langle\langle\rangle\rangle_p^y T \text{ and } \langle\langle Ag \rangle\rangle_x^s T \equiv \langle\langle Ag \rangle\rangle_x^p T$$

As a convention, when having an empty coalition, we will only write $\langle\langle\rangle\rangle_s^s$ and $\langle\langle\rangle\rangle_p^p$, which is no restriction, given the equivalence above. Similarly, for the full coalition, we will only write $\langle\langle Ag \rangle\rangle_s^s$ and $\langle\langle Ag \rangle\rangle_p^p$.

Multiple $\langle\langle G \rangle\rangle_x^y \bigcirc$ operators are used as an abbreviation in the following way:

$$(\langle\langle G \rangle\rangle_x^y \bigcirc)^n \varphi \triangleq \begin{cases} \langle\langle G \rangle\rangle_x^y \bigcirc \varphi \text{ if } n = 1 \\ (\langle\langle G \rangle\rangle_x^y \bigcirc)(\langle\langle G \rangle\rangle_x^y \bigcirc)^{n-1} \varphi \text{ otherwise} \end{cases}$$

From Lemma 6 we immediately derive:

$$\langle\langle G \rangle\rangle_s^y T \rightarrow \langle\langle G \rangle\rangle_p^y T \text{ and } \langle\langle G \rangle\rangle_x^p T \rightarrow \langle\langle G \rangle\rangle_x^s T \tag{7.1}$$

where T here is an arbitrary temporal formula and x and y are variables over p and s. These properties express the following. The first, $\langle\langle G\rangle\rangle_s^y T \rightarrow \langle\langle G\rangle\rangle_p^y T$ says that if a coalition G are able to enforce a property φ by adopting social strategies, then they can also enforce this same property when adopting physical strategies ('if you can enforce it nicely, you can enforce it anyhow'); and $\langle\langle G\rangle\rangle_x^p T \rightarrow \langle\langle G\rangle\rangle_x^s T$ can be interpreted as saying that if a coalition G are able to enforce a property φ when playing against an adversary who is able to use physical strategies, then they can also enforce this property when playing against the same adversary when this adversary is constrained to use only *social* strategies ('if you can beat an opponent when he can cheat, you can beat him when he plays by the rules').

7.3 Case Study

We now present a case study in order to demonstrate Social ATEL. The case study is known as "The Alternating Bit Protocol" and is adapted from [56].

In this scenario there are two agents, a sender S and a receiver R, who wish to communicate through a communication environment. S wants to send the value of a bit to R, but the communication environment is not reliable. The environment can delete messages, but messages that are received are always correct. Although the environment is unreliable, it satisfies the fairness property that messages sent arbitrarily often will eventually arrive. The sender has a sequence of bits $X = \langle x_0, x_1 \ldots, \rangle$ that it wishes to communicate to the receiver. When the receiver receives the bits, it prints them to a tape Y, which after receiving k bits is $Y = \langle y_0, y_1 \ldots, y_k \rangle$. The alphabet for the tape, X, over which symbols may be chosen is $Alph = \{0, 1\}$. We wish to design a protocol that satisfies the safety requirement that the receiver never prints incorrect bits, and the liveness requirement that every bit will eventually be printed by the receiver.

The obvious solution to this problem is to use acknowledgements to let the sender know a bit has been received. So the sender would repeatedly send the bit until eventually it receives an acknowledgement back from the receiver, at which point it would go on to send the next bit. The problem arises when the value of the next bit is the same as the previous bit. The receiver does not know whether its acknowledgement has been received, so the receiver does not know whether this bit is the same bit, or if this bit is supposed to be the next bit on the tape. To overcome this problem, the sender can put more information on which bit he sends by adding a "colour" to it: a 0 to every x_i for even i, and a 1 for every odd i. So now the alphabet is updated as follows: $Alph'$ becomes $\{x.0 \mid x \in Alph\} \cup \{x.1 \mid x \in Alph\}$. We now also need two acknowledgements: *ack*0 and *ack*1. So the protocol works as follows: S first sends $x_0.0$. When it receives *ack*0, it goes on to the next state on the tape and sends $x_1.1$. R can see that this is a new message (since it has a different colour), and sends *ack*1 to acknowledge the receipt of it. S can also see that this is a different acknowledgement and starts to send $x_2.0$, and so on. In the interests of clarity, we are going to abstract away some of the unnecessary details of this system. This will also reduce the state space and make it easier to understand. We are not actually concerned with the value of the bits so we abstract from the value of x. Also, we do not want to reason about the behaviour of the environment. We only wish to reason about the behaviour of the sender and the receiver. So we assume that the environment always behaves as it should do, but the agents do not know this, hence the reason for the protocol. This also has the effect of reducing the number of state transitions.

We model the scenario with an SAAETS called *AltBit*. We introduce atoms with the following interpretation: *ssx* means that the last bit S sent was of type $x.1$; *sra* indicates the last message received by S was *ack*1; *rrx* means the last bit R received was of type $x.1$; and *rsa* indicates the last message R sent was *ack*1. Rather than having two atoms *ssx*.0 and *ssx*.1, we use the fact that $ssx.0 \leftrightarrow \neg ssx.1$ to reduce the number of states. The same applies to the other atoms.

A state in our system is defined to be a tuple

$$q_i = \langle ssx, sra \mid rrx, rsa \rangle$$

where:

- $ssx, sra, rrx, rsa \in \{0, 1\} = \mathbb{B}$; and
- q_i, where $1 \leq i \leq 16$, is the name of the state. This is just a decimal representation of the binary number the state corresponds to (e.g. $q_3 = \langle 0, 0 \mid 1, 1 \rangle$).

In every state of the system, the sender has two physical actions available to it: *send*.0 and *send*.1, corresponding to sending a bit with colour 0 and sending a bit with colour 1, respectively. The receiver also has two physical actions available to it in every state of the system: *sendack*.0 and *sendack*.1, corresponding to sending an acknowledgement of a bit with colour 0 and colour 1, respectively. Figure 7.1 shows physical runs of the alternating bit protocol. As the system is unconstrained by any social laws, the agents can perform any action in every state. Note also that in fact we should have drawn a reflexive arrow from every state to itself, denoting the possibility that the system can decide not to deliver a message that was sent. However, both in the diagram and in the verification of properties we assume that the system does not show such adverse behaviour. In the properties that we will discuss, this means that often when we write a $\bigcirc$ (under the idealisation that no messages are lost) they should in general be replaced with occurrences of $\Diamond$ (resembling the assumptions that messages eventually will be delivered).

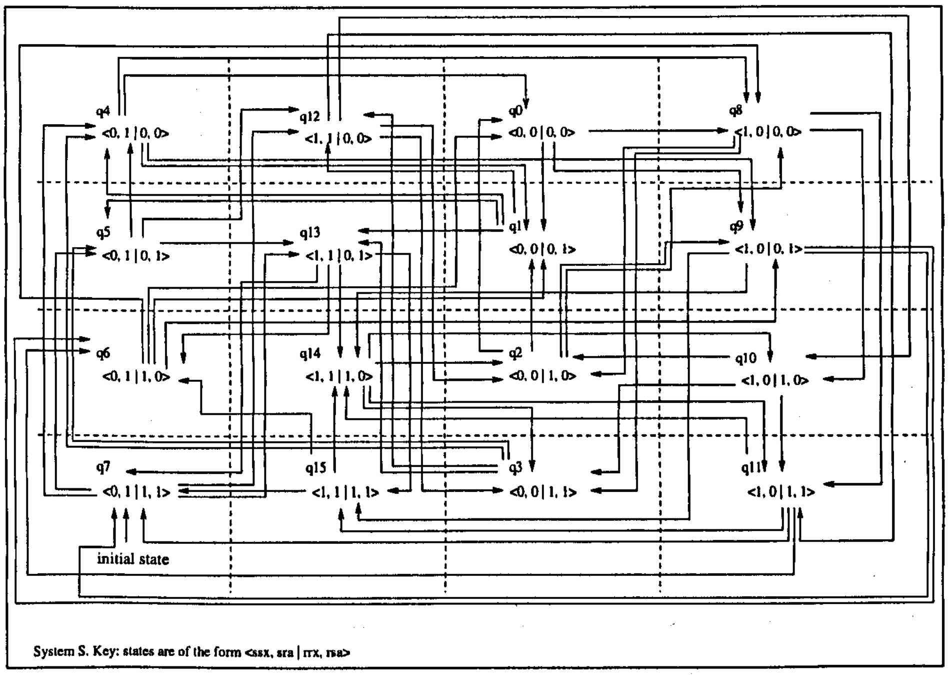

Figure 7.1: State transitions resulting from *physical* strategies.

Of course, the idea is that S should alternate $x.0$'s, once known to be received, with $x.1$'s, and R is expected not to acknowledge receipt of a bit he did not receive. This is the idea of the social actions, which are given in Figure 7.2 below. The columns show the states where the action can be performed and the rows show which agent is carrying out the specified action. To understand the first column in the figure, note that in, for instance, $q_8 = \langle 1,0 \mid 0,0 \rangle$, in which $ssx \wedge \neg sra \wedge \neg rrx \wedge \neg rsa$ is true, the Sender should re-send his bit $x.1$, since he has not received an acknowledgement for it ($\neg sra$). At the same time, since R has not received this $x.1$ yet (we have $\neg rrx$), R cannot acknowledge with $ack1$, but should repeat sending $ack0$. Clearly, when S and R are acting in a social manner they have much fewer choices available to them than when they are able to act physically.

$i\backslash q$	$\langle 0,0 \mid 0,0\rangle$	$\langle 0,1 \mid 1,0\rangle$	$\langle 0,1 \mid 0,0\rangle$	$\langle 0,0 \mid 1,0\rangle$
	$\langle 0,0 \mid 0,1\rangle$	$\langle 0,1 \mid 1,1\rangle$	$\langle 0,1 \mid 0,1\rangle$	$\langle 0,0 \mid 1,1\rangle$
	$\langle 1,0 \mid 0,0\rangle$	$\langle 1,1 \mid 1,0\rangle$	$\langle 1,1 \mid 0,0\rangle$	$\langle 1,0 \mid 1,0\rangle$
	$\langle 1,0 \mid 0,1\rangle$	$\langle 1,1 \mid 1,1\rangle$	$\langle 1,1 \mid 0,1\rangle$	$\langle 1,0 \mid 1,1\rangle$
S	$send.1$	$send.0$	$send.0$	$send.1$
R	$sendack.0$	$sendack.1$	$sendack.0$	$sendack.1$

Figure 7.2: Social strategies for S and R.

Where Figure 7.1 takes into account the worst possible behaviour by the agents, Figure 7.3 shows the other extreme: here, not only do we assume that all agents act socially, but also that this is common knowledge: nobody finds it epistemically possible that somebody finds it possible that ... somebody acts anti-socially. In this model, we can interpret formulae of type $SB_i\varphi$, but not if φ contains an occurrence of $K_j\psi$: since SB_i only takes into account the *social states* (defined by $sreach(AltBit)$) as epistemic alternatives of agent i, due to the assumption that all agents act socially, while K_j takes into account *all states* as epistemic alternatives of j. In this chapter, we will not look at such nested occurrences. Summarising: the runs in Figure 7.3 are runs that are generated by only performing social strategies. Arrows indicate successor states, and some states are not reachable when only performing such social strategies. The states in this diagram are exactly those defined by $sreach(AltBit)$. The Sender cannot distinguish states that are placed on top of each other and the Receiver cannot distinguish those placed horizontally. So, columns represent equivalence classes of S-indistinguishable states and rows represent equivalence classes of R-indistinguishable states. These equivalence classes represent *social* belief for the agents, due to the fact that we are only looking at states in $sreach(AltBit)$ (i.e., social states).

One can introduce the notion of a *socially necessary fact* (SNF). A socially necessary fact is a fact which should be true no matter what, providing all the agents in the system act in a

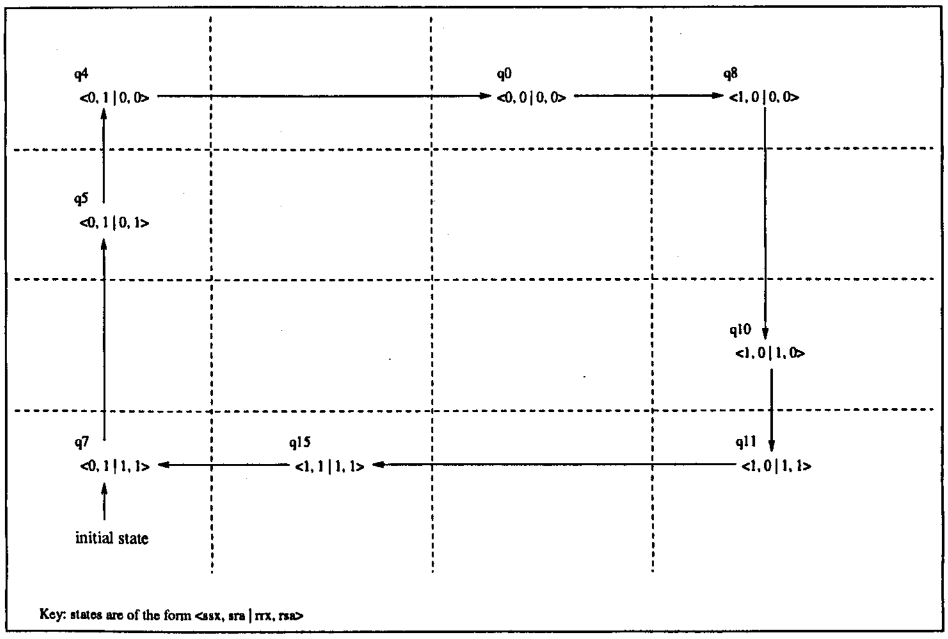

Figure 7.3: The model if it is common knowledge that only social strategies are played.

social manner. A SNF φ is defined as follows:

$$SNF(\varphi) \equiv \langle\langle\rangle\rangle_s^s \Box \varphi$$

If an SNF does not involve epistemic properties, one can verify them on Figure 7.3. A socially necessary fact is different from a social belief. For example, in state q_0 the social belief $SB_S(\neg rrx \wedge \neg rsa)$ holds. However, $SNF(\neg rrx \wedge \neg rsa)$ does not, as it is not something that is true across all socially reachable states.

7.3.1 Properties of the Model

Now we investigate some interesting properties in the Alternating Bit Protocol, in order to show the effects of different agents following social strategies while others follow physical ones, etc. The properties we have investigated are listed in Figure 7.4 along with a $\surd$ (showing that the property holds) or a $\times$ (showing that the property does not hold).

Many of the properties in the table are of type $\langle\langle\rangle\rangle_p^p \Box \varphi$. Such a formula expresses that φ is an invariant that is true in every reachable state. Since in the model of the physical system, all states are reachable from each other, this means that φ must be true everywhere. We will begin

No.	Property	Result
1	$\langle\langle\rangle\rangle_p^p \Box(rrx \rightarrow (\langle\langle Ag\rangle\rangle_p^p \bigcirc)^2 sra)$	√
2	$\langle\langle\rangle\rangle_p^p \Box(rrx \rightarrow (\langle\langle S\rangle\rangle_s^p \bigcirc)^2 sra)$	×
3	$\langle\langle\rangle\rangle_p^p \Box(rrx \rightarrow (\langle\langle S\rangle\rangle_p^s \bigcirc)^2 sra)$	√
4	$\langle\langle\rangle\rangle_p^p \Box(rrx \rightarrow (\langle\langle\rangle\rangle_s^s \bigcirc)^2 sra)$	√
5	$\langle\langle\rangle\rangle_p^p \Box(ssx \rightarrow (\langle\langle Ag\rangle\rangle_p^p \bigcirc)^3 sra)$	√
6	$\langle\langle\rangle\rangle_p^p \Box(ssx \rightarrow (\langle\langle S\rangle\rangle_s^p \bigcirc)^3 sra)$	×
7	$\langle\langle\rangle\rangle_p^p \Box(ssx \rightarrow (\langle\langle R\rangle\rangle_s^p \bigcirc)^3 sra)$	√
8	$\langle\langle\rangle\rangle_p^p \Box(ssx \rightarrow (\langle\langle\rangle\rangle_s^s \bigcirc)^3 sra)$	√
9	$\langle\langle\rangle\rangle_p^p \Box(ssx \rightarrow \langle\langle Ag\rangle\rangle_p^p \Diamond rrx)$	√
10	$\langle\langle\rangle\rangle_p^p \Box(ssx \rightarrow \langle\langle\rangle\rangle_p^p \Diamond rrx)$	√
11	$\langle\langle\rangle\rangle_p^p \Box((ssx \wedge sra) \rightarrow rrx)$	×
12	$\langle\langle R\rangle\rangle_s^p \Box((ssx \wedge sra) \rightarrow rrx)$	×
13	$\langle\langle S\rangle\rangle_s^p \Box((ssx \wedge sra) \rightarrow rrx)$	×
14	$\langle\langle\rangle\rangle_s^s \Box((ssx \wedge sra) \rightarrow rrx)$	√
15	$\langle\langle\rangle\rangle_p^p \Box((rrx \wedge rsa \wedge sra) \rightarrow \langle\langle\rangle\rangle_p^p \bigcirc \neg ssx)$	×
16	$\langle\langle\rangle\rangle_p^p \Box((rrx \wedge rsa \wedge sra) \rightarrow \langle\langle R\rangle\rangle_s^p \bigcirc \neg ssx)$	×
17	$\langle\langle\rangle\rangle_p^p \Box((rrx \wedge rsa \wedge sra) \rightarrow \langle\langle R\rangle\rangle_p^s \bigcirc \neg ssx)$	√
18	$\langle\langle\rangle\rangle_p^p \Box((rrx \wedge rsa \wedge sra) \rightarrow \langle\langle\rangle\rangle_s^s \bigcirc \neg ssx)$	√

Figure 7.4: General properties and results

by looking at properties 1 - 4. Let us first look at property 1. Here, the invariant φ says that if the receiver has received a bit $x.1$, then there should exist a physical strategy for the grand coalition of agents so that in two time steps, the sender has received an acknowledgment $ack.1$. If we look at Figure 7.1, this corresponds to finding every state where *rrx* is true and checking that the agents can choose a physical strategy there such that in two steps, *sra* is true. This holds (as expected, since if the grand coalition can't make it happen, it won't be possible for any subsets of agents). In fact, it is not difficult to see that we even have $(\langle\langle Ag\rangle\rangle_p^p \bigcirc)^2 sra$: no matter where we are in the system *AltBit*, the receiver can always (if no constraints are imposed) send an $ack.1$ acknowledgement, which leads us to a state in which *sra* holds. Property 2 is similar, but expresses that if the receiver has received a bit $x.1$, then the sender S should have a social strategy while all other agents can act physically, so that in two time steps, the sender has received an acknowledgement $ack.1$. This fails as the receiver has the possibility of acting physically, thus sending an erroneous acknowledgment. Of course, it is only R who can guarantee, using a social strategy, that S receives the acknowledgement that it should (i.e., $ack.1$). It even allows the Sender to behave in an arbitrary physical way. Property 4, expresses that if the last bit received was of a form $x.1$, then all socially allowed computations will guarantee that in two steps, S has the corresponding acknowledgement (recall that in all these examples, we assume that the environment behaves optimal in the sense that messages don't get lost; otherwise we should weaken the consequent in property 4 to $\langle\langle\rangle\rangle_s^s \Diamond sra$).

We now discuss properties 11 - 14. Property 11 expresses that, globally, if S recently sent $x.1$ and received its acknowledgment $ack.1$, then R should have received the $x.1$ (the 'matching pair' of $x.1$ and $ack.1$ should not have come about 'by coincidence', so to speak). This fails, as there is nothing in place to ensure that the receiver will send the correct acknowledgment back (it might have sent $ack.1$ to fool S), or to ensure that the sender will alternate the colour of bits it sends. What if we demand R to act socially (property 12)?, The desired property still fails: Since S is not required to act socially, he could enforce a transition from $\langle 0,1 \mid 0,1\rangle$ (where he is sending $x.0$ but still receiving $ack.1$, and R received $x.0$) to $\langle 1,1 \mid 0,1\rangle$ (S changes the colour of his bit, without having received an acknowledgment). When only S is required to act socially (property 13), our property still fails as the receiver may send an erroneous acknowledgement to the sender. When all agents act socially (14), however, this formula passes. This is as expected as the social laws guarantee the agents adhere to the protocol.

Finally we look at properties 15 - 18. Property 15 looks at whether the sender is alternating his bits correctly. Specifically, this property says, if the receiver has received a bit (colour 1), the receiver has sent an acknowledgment (colour 1), and the sender has received an acknowledgment (colour 1), if everyone acts physically, on all paths in the next state the sender will send a bit with colour 0. This fails as expected as there is nothing to ensure that the agents follow the protocol if they are allowed to act physically. Here we require the sender to alternate the colour of the bit it sends. The next variation (property 16) is when R is acting socially and everyone else is capable of acting physically. This fails as it is only up to the sender whether to send a colour 0 or 1 bit. Then we look at property 17 where R is capable of acting physically and everyone else is acting socially. This passes as expected, as it only requires the sender to act socially to ensure the colour of bits is alternated. Finally, when all agents act socially (property 18) this passes, as the sender is still acting socially. This property might be over constraining, as the objective is still achieved with property 17.

7.3.2 Informational Properties

Now we look at some informational properties in the alternating bit protocol. Some of such properties are given in Figure 7.5. We are going to investigate knowledge and social belief. For the standard notion of knowledge, we refer to Figure 7.1. Here columns represent indistinguishable states to S and rows represent indistinguishable states to R. For the *social belief*, we refer to Figure 7.3, where rows and columns have the same interpretation of indistinguishable states. We notice in Figure 7.1, where the agents are totally unconstrained, the agents actually have *no* knowledge whatsoever about the other agent's state. In other words, if agents can act arbitrarily, then very little knowledge ensues. For instance, in q_{15}, S knows of course its own state, but has no clue of R's: $q_{15} \models \neg K_S rrx \wedge \neg K_S \neg rrx \wedge \neg K_S rsa \wedge \neg K_S \neg rsa$.

However, this is not the case in Figure 7.3, where the agents are constrained by the social laws. In this diagram, there are states where agents have social belief about the state of the other agent. For example, if we find a state where both *sxx* and *sra* are true (e.g., q_{15}), we can see that in all S-indistinguishable worlds, *rrx* and *rsa* both hold. This means the sender knows these variables in state q_{15}: he in fact has perfect information of what the global state is. This shows us the importance of social laws, in that they can totally change the epistemic properties of the system. We now investigate some specific knowledge properties in order to illustrate our point further.

First we look at property 19. This property is an invariant property and it says, if the sender has sent a bit and the sender has received an acknowledgement back, then the sender should know that the receiver has received the bit. So the first thing we need to do is look at all states in Figure 7.1 where both *sra* and *ssx* are true. Now we must check that in all these S-indistinguishable states, *rrx* is also true. This, however, is not the case. This property does not hold. This is due to the fact that the receiver may send an erroneous acknowledgement to the sender, and so the sender won't know that the bit has been received. Now we consider a variation where the sender is acting socially but everyone else is able to act physically (property 20). This formula does not hold, as the receiver is still able to send erroneous acknowledge-

No.	Property	Result
19	$\langle\!\langle\rangle\!\rangle_p^p \Box((sra \wedge ssx) \rightarrow K_S rrx)$	×
20	$\langle\!\langle S\rangle\!\rangle_s^p \Box(sra \wedge ssx) \rightarrow K_S rrx$	×
21	$\langle\!\langle S\rangle\!\rangle_p^s \Box(sra \wedge ssx) \rightarrow K_S rrx$	×
22	$\langle\!\langle\rangle\!\rangle_s^s \Box(sra \wedge ssx) \rightarrow B_S^s rrx$	√
23	$\langle\!\langle\rangle\!\rangle_p^p \Box(ssx \wedge rrx) \rightarrow \langle\!\langle\rangle\!\rangle_p^p \bigcirc^2 K_S rrx$	×
24	$\langle\!\langle\rangle\!\rangle_p^p \Box(ssx \wedge rrx) \rightarrow \langle\!\langle S\rangle\!\rangle_s^p \bigcirc^2 K_S rrx$	×
25	$\langle\!\langle\rangle\!\rangle_p^p \Box((ssx \wedge rrx) \rightarrow \langle\!\langle R\rangle\!\rangle_s^p \bigcirc^2 K_S rrx)$	×
26	$\langle\!\langle\rangle\!\rangle_s^s \Box((ssx \wedge rrx) \rightarrow \langle\!\langle\rangle\!\rangle_s^s \bigcirc^2 B_S^s rrx)$	√
27	$\langle\!\langle\rangle\!\rangle_p^p \Box(ssx \rightarrow \langle\!\langle\rangle\!\rangle_p^p \Diamond K_S rrx)$	×
28	$\langle\!\langle\rangle\!\rangle_p^p \Box(ssx \rightarrow \langle\!\langle R\rangle\!\rangle_s^p \Diamond K_S rrx)$	×
29	$\langle\!\langle\rangle\!\rangle_s^s \Box(ssx \rightarrow \langle\!\langle\rangle\!\rangle_s^s \Diamond B_S^s rrx)$	√

Figure 7.5: Knowledge properties and results

ments due to the fact that it is capable of acting physically. Now we look at property 21, where now the sender is following physical strategies, but everyone else (i.e., the receiver) is following social strategies. Intuitively one might expect this property to hold, as now the receiver is behaving as it should do. However, this property does not hold. Even though the receiver acts socially and sends the sender the correct acknowledgement, the sender does not *know* that the receiver is acting socially, so it does not hold. Finally we consider a variation where all agents are assumed to act socially (property 22). Now we must look at Figure 7.3 and again find states where both *sra* and *ssx* are true. As one can see, this is only state q_{15}. Now in all S-indistinguishable states, we need to see whether *rrx* is also true. This is the case as q_{15} is the only state. When all agents are assumed acting socially, this formula holds as expected. So now the sender assumes the receiver is acting socially, and under this assumption, he believes *rrx*.

The next property we look at is property 23. This property is again an invariant property. This says if the sender has sent a bit and the receiver has received a bit, no matter what happens if the agents follow physical strategies, then in two time steps, the sender should know that the receiver has received the bit. Now we need to find all the states in Figure 7.1 where both *ssx* and *rrx* are true and see if it is the case that all paths from these states lead to states in two time steps where the sender knows *rrx* (i.e., in all S-indistinguishable worlds where *sra* and *ssx* is true, *rrx* is also true). This property fails when all the agents are capable of acting physically. This can only be expected for the same reasons as above. Now we look at a variation where the sender acts socially but the other agents can act physically (property 24). Naturally, this property does not hold as the receiver can choose to follow strategies where he does not send the correct acknowledgement. Now, looking at property 25, again we might expect this property to hold as it is only the receiver who is responsible for sending the correct acknowledgements to the sender. However, the sender does not know that the receiver is acting in a social manner, thus making this property false. The final variation we look at is property 26, where everyone is acting socially. To see if this holds we need to look at Figure 7.3. We find a state where both *ssx* and *rrx* are true. Here, q_{10} is the only such state. Now we need to look at all social state transitions and see if they lead to a state where the sender knows *rrx*. The only transition is to state q_{15}, where indeed the sender does know *rrx*. So this property holds under the assumption that the receiver is acting socially.

Finally we look at property 27, which says if the sender has sent a bit then no matter what the agents do, if they are capable of acting physically then at some point in the future, the sender knows that the receiver has received the bit. This is similar to the previous property (23), but more general. Again, this property should only require the receiver to act socially and send the acknowledgement to the sender. This property does not hold when all the agents are capable of acting physically. This is again due to the receiver not sending the correct acknowledgement. So if we just consider a variation where the receiver is acting socially (property 28), this should hold. However, as previously explained, the sender won't know *rrx* as he doesn't know if the receiver is acting socially or not. Finally, when we consider the case where all agents are *assumed* to be acting socially (property 29), this does indeed hold.

7.4 Summary

In this chapter we have extended our social laws framework further. In the previous chapter, the basic framework was extended to incorporate epistemic notions, but here, we have extended this by removing the assumption that all agents in the system will adhere to the social laws. Firstly, we extended the semantic structures to allow us to model physical action pre-conditions and legal action pre-conditions, in turn, allowing us to construct both physical and legal strategies for the agents. With the semantic constructs in place, we introduced our logical language for reasoning about such systems – Social ATEL – based on ATEL but extended to allow us to refer to the type of strategies being followed by the coalition of agents and the other agents in the system. With the possibility of violating social laws came a new type of epistemic property, known as Social Belief, which is belief under the assumption that all agents in the system are adhering to the social laws. We also introduced the notion of a Socially Necessary Fact, which is a fact which should be true no matter what, providing all agents in the system act in a social manner.

We presented a case study, known as the Alternating Bit Protocol, in order to demonstrate Social ATEL. We were able to investigate various interesting properties in the case study, to see the affects of agents following social laws while other agents did not. We also investigated informational properties, to see the affects social laws have on the knowledge of the agents. After investigating informational properties, we have shown the important roles social laws play in our systems. *When the agents are following strategies that are not restricted in any way, we see that the only thing they do know is their local states.* Also, it is not enough for simply one agent to act socially, even if this is the only agent who can affect the outcome of a knowledge property, as if the other agent does not know he is acting socially, it can not gain knowledge from him. This is where the notion of *social belief* comes into play. When all the agents in the system act socially, we only need consider those states defined by $sreach(AltBit)$. Comparing Figure 7.1 to Figure 7.3, one can immediately see that much more *social belief* exists than simply standard knowledge. The crucial point about this social belief is that it is belief *under the assumption* that all the agents in the system are conforming to the norms (social laws).

Chapter 8

Reducing Social ATEL to ATL*

In this chapter we introduce an alternative approach for expressing properties of systems that refer to whether the agents are acting socially or physically. Rather than using our logical language, Social ATEL, introduced in the previous chapter, we see to what extent we can capture the same notions using only ATL and ATL*. To this end, we introduce the notion of "good" and "bad" states, similar to the "Red" and "Green" states of Lomuscio and Sergot in [30]. We modify our Social Action-Based Alternating Epistemic Transition Systems by introducing an atomic proposition for each agent, which is only true in the current state if the agent arrived here from the previous state using a legal action. So essentially, we are labeling the states based on how the agents arrived at each state. This gives rise to a larger state space in the modified systems, as now we have copies of each state, representing all the combinations of the agents acting socially or physically to reach it. Ideally we would like to be able to reduce properties expressed in Social ATEL into ATL, expressed using these "good" atomic propositions, in order to automatically verify Social ATEL properties using existing ATL model checkers, such as MOCHA. However, due to the fact that Social ATEL appears to be more expressive than ATL, it is not feasible to find direct equivalences between the two. We can, however, express interesting properties using ATL and ATL*.

Using the approach outlined above, we look at several types of Social ATEL properties and see how closely we can express these in ATL*. We investigate the relationship between the two using three special cases of G, where G is the coalition of agents cooperating to achieve some objective. We look at the case where G is the grand coalition (Ag), the empty coalition ($\emptyset$), and finally, an arbitrary coalition (G). We show that there is a general pattern between Social ATEL properties and properties expressed in this approach, which holds regardless of the coalition type and the temporal operators being used. Finally, we prove equivalences between general formulae expressed in Social ATEL and formulae expressed using this approach, for each combination of the coalition acting socially or physically, while the other agents act socially or physically.

This chapter is structured as follows: We start by introducing the atomic propositions and their interpretation and then show how we modify SAAETSs to take into account these new atomic propositions. We then go on to see how closely we can express formulae of Social ATEL in this approach. In Section 8.3, we prove some reductions between formulae expressed in the two approaches and finally in Section 8.4 we summarise.

8.1 Modifying SAAETSs

In this section we introduce the atomic propositions used to give a labeling to each state based on how each agent arrived there. This can either be by performing a *legal* action or by simply performing any *physically possible* action. We introduce an atomic proposition, g_i, one for each agent $i \in Ag$, with the interpretation that g_i is true in the current state if agent i's last action was a legal one. This corresponds to agent i acting in a social manner (for one time step). $GP = \{g_1, \ldots, g_n\}$ is a set of *good* propositions where $GP \subseteq \Phi$. In order to reason about coalitions of agents, we introduce a proposition $good(G)$ which holds if all the agents in G acted socially to reach the current state. $good(G)$ is defined as follows:

$$good(G) = \bigwedge_{i \in G} g_i$$

We do not define any explicit propositions to express that an agent i got to the current state via an illegal action, as obviously $\neg g_i$ would simply hold.

Now we must modify SAAETSs with these g_i propositions. It is important to note that the definition of the g_i propositions comes from the $\ell(\alpha)$ function in the given SAAETS. We are not concerned with implementation of a social law on a system, as defined in Section 7.1, the $\ell(\alpha)$ function is assumed to be given. Now, given a Social Action-Based Alternating Epistemic Transition System (SAAETS),

$$Sys = \langle Q, q_0, Ag, Ac_1, \ldots, Ac_n, \sim_1, \ldots, \sim_n, \rho, \ell, \tau, \Phi, \pi \rangle$$

in order to express properties of systems in this way, we need to convert the system into a modified system as below:

$$Sys^\circ = \langle Q^\circ, q_0^\circ, Ag, Ac_1, \ldots, Ac_n, \rho^\circ, \tau^\circ, \Phi^\circ, \pi^\circ \rangle$$

where the modified components have the following interpretation:

- Q°: For each state $q \in Q$, we now have at worst $2^{|Ag|}$ copies of q to represent all the combinations of agents being "good" and "bad". We encode this extra information as $q_{x_1,\ldots,x_n}$, where $x_i \in \{0,1\}$. x_i being 1 means agent i's last action was a social one, whereas a 0 means it was anti-social. The new set of states formed, Q°, in the worst case, is of size $|Q \times 2^{Ag}|$. See Figures 8.1 and 8.2 for an example of how the train system is transformed;

- q_0°: q_0 becomes q_0°, which is an abbreviation of $q^\circ_{0_{x_1,\ldots,x_n}}$, where $\forall i \in Ag, x_i = 1$. This is the initial state of the system, which is the same as before, except g_i is true for all agents;

- ρ°: $\forall i \in Ag, \forall \alpha \in Ac_{Ag}, \forall q \in Q, \forall x_i \in \{0,1\} : q \in \rho(\alpha) \iff q_{x_1,\ldots,x_n} \in \rho^\circ(\alpha)$;

- Φ°: Φ is updated with the new g_i propositions: $\Phi^\circ = \Phi \cup GP$;

- π°:

$$\forall i \in Ag, \forall p \in \Phi, \forall q \in Q, \forall x_i \in \{0,1\} : p \in \pi(q) \iff p \in \pi^\circ(q_{x_1,\ldots,x_n})$$
$$\forall i \in Ag, \forall g_i \in GP : g_i \in \pi^\circ(q_{x_1,\ldots,x_n}) \iff x_i = 1.$$

- τ°: $\forall i \in Ag, \forall q, q' \in Q, \forall \alpha_i \in Ac_i, \forall x_i \in \{0,1\} : [\tau(q, \langle \alpha_1, \ldots, \alpha_k \rangle) = q' \iff \tau^\circ(q_{x_1,\ldots,x_n}, \langle \alpha_1, \ldots, \alpha_k \rangle) = q'_{f_1(x_1),\ldots,f_n(x_n)}$, where $f_i(x_i) = 1 \iff q \in \ell(\alpha_i)]$.

So now we have modified SAAETSs to work with these g_i propositions. Firstly, the set of states has been modified. We have a new copy of each state for all combinations of g_i, for all agents. The initial state is the same as before, but g_i is true for all agents, hence the system starts in a good state. The new action precondition function, ρ°, is directly equivalent to ρ for all states and actions, regardless of the g_i propositions. In other words, if an agent can perform α in q, then the agent can perform α in $q_{x_1,\ldots,x_n}$, no matter what the values of x_i are. The set of atomic propositions, Φ, is updated with the set of good propositions, GP. The truth definition function, π, is the same as before for atomic propositions in Φ. It is updated for the propositions in GP, where a g_i proposition is true in a state where $x_i = 1$. Finally, the transition function, τ, is updated in the following way: transitions are the same as before, but now, if agent i performs a legal action, in the resultant state, g_i will be true and the x_i subscript of the state will be 1. Performing an illegal action results in the g_i proposition being false and the x_i subscript of the state being 0.

Remark In this chapter we do not look at properties that refer to knowledge. We briefly consider how we would modify the epistemic accessibility relations from SAAETSs to account for these good propositions. We propose the following modification to the epistemic accessibility relations:

$$\sim_i^\circ: \forall i \in Ag, \forall q, q' \in Q, \forall x_i \in \{0,1\} : q \sim_i q' \iff q_{x_1,\ldots,x_n} \sim_i^\circ q'_{x_1,\ldots,x_n}$$

So if two states, q and q' are indistinguishable in Sys, $q_{x_1,\ldots,x_n}$ and $q'_{x_1,\ldots,x_n}$ will be indistinguishable in Sys°. This will now mean that agents will know if they have acted socially, and also if other agents have acted socially. This would allow us to formulate informational properties about systems where agents can reason about the behaviour of the other agents in the system. An alternative way to modify the epistemic accessibility relations would be as follows:

$$\sim_i^\circ: \forall i \in Ag, \forall q, q' \in Q, \forall x_i, x_i' \in \{0,1\} : q \sim_i q' \iff q_{x_1,\ldots,x_n} \sim_i^\circ q'_{x_1',\ldots,x_n'}$$

where $x_i = x_i'$.

This is the same as above, but now agents only know if they have acted socially and nothing about how other agents have acted. This would mean agents would only be able to reason about their own behaviour.

8.2 Reducing Some Formulae

There are various interesting types of properties we can form with these $good(G)$ propositions. We will now look at several types of Social ATEL properties and see how closely we can express such properties with these $good(G)$ propositions. We will investigate this using special cases of G, namely the grand coalition (Ag), the empty coalition ($\emptyset$), and an arbitrary coalition (G). We will construct example properties using the train scenario that we introduced in Chapter 4.

We introduce the systems *Train* and *Train*°. $Train = S_1 \dagger \beta_1$, where $S_1 \dagger \beta_1$ is the system constrained with β_1 introduced in Chapter 5 and *Train*° is the AATS after modifying *Train*. We can see the affect modifying the systems has by comparing Figure 8.1, showing the states and transitions of the standard train system implemented with β_1, with Figure 8.2, which shows the same system after being modified. States which do not have an edge labelled 'i, i' leaving them are assumed to have a reflexive arc labelled 'i, i', which we omit for clarity.

We start by looking at the following Social ATEL formula:

$$Sys, q \models \langle\langle G \rangle\rangle_s^s \Box \varphi \tag{8.1}$$

where φ is assumed to be a propositional logic formula. This Social ATEL formula says that G has a social strategy, such that, no matter what the other agents do, providing they follow social strategies, φ will always be true. We will now try to capture the above using these $good(G)$

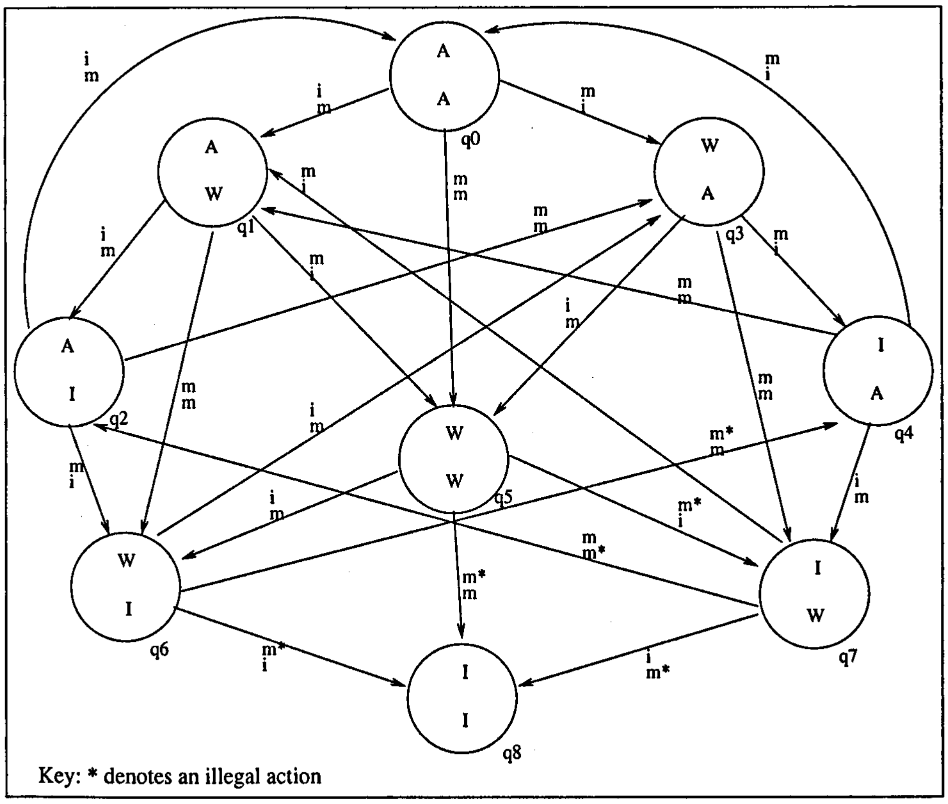

Figure 8.1: State transitions in *Train*.

propositions where we take G to be the grand coalition, Ag:

$$Sys^{\circ}, q_{x_1,\ldots,x_n} \models \langle\langle Ag \rangle\rangle \Box (good(Ag) \wedge \varphi) \tag{8.2}$$

where $\forall i \in Ag : x_i = 1$ and φ is assumed to be a propositional logic formula. This formula states that the grand coalition of agents has a strategy such that it will always be the case that $good(Ag)$ is true and φ is true at the same time. This appears to express the same as (8.1) above. If we refer to the train scenario we can formulate the following example property:

$$Sys^{\circ}, q_{0_{11}} \models \langle\langle Ag \rangle\rangle \Box (good(Ag) \wedge \neg (in_E \wedge in_W)) \tag{8.3}$$

So this property states that the grand coalition should have a strategy so that it is always the case that the agents follow only social strategies and that both of the trains are not in the tunnel at the same time. Since the primary goal of the trains scenario is to avoid a crash situation, it

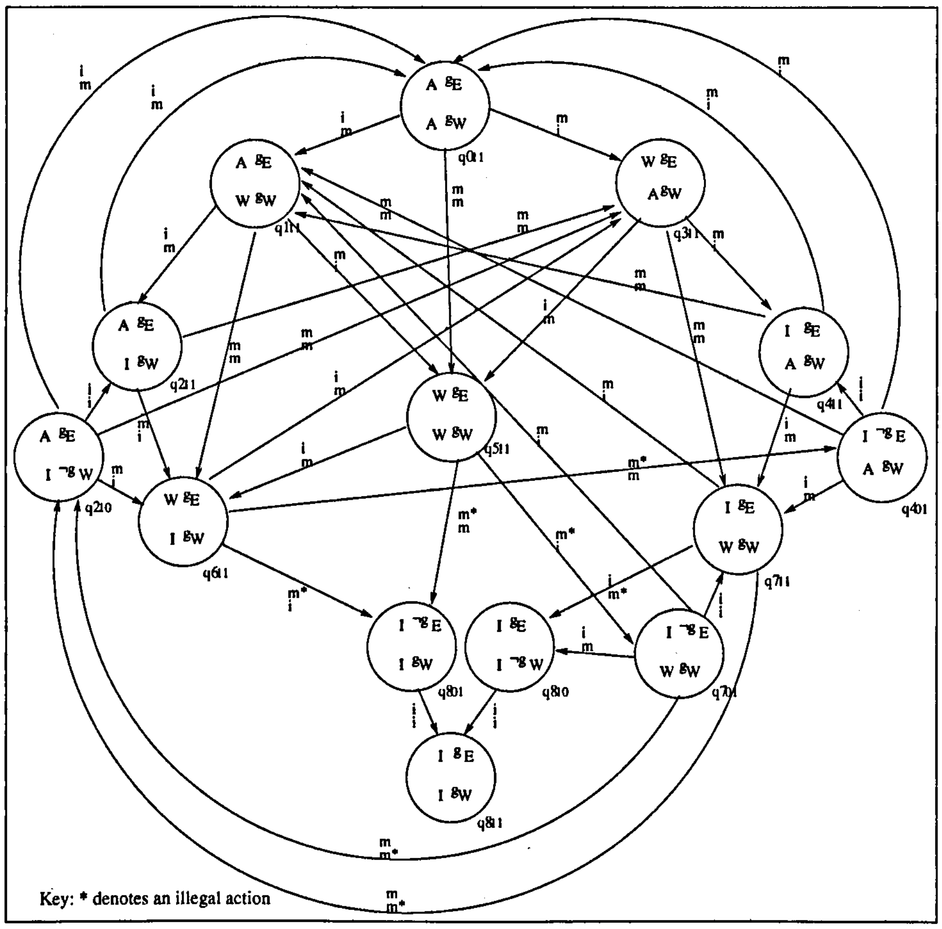

Figure 8.2: State transitions in *Train*°.

seems obvious that the grand coalition could act in such a way, to only follow social strategies and still achieve the goal of both agents not entering the tunnel at the same time. This seems obvious, as always acting in a *social* manner should lead to the goal being achieved. Property (8.3) passes when model checking in the system *Train*°.

Following on from what we said about acting socially leading to the goal, we can formulate the following, where now we take G to be the empty set, $\emptyset$:

$$Sys^{\circ}, q_{x_1,\dots,x_n} \models \langle\langle\rangle\rangle\, (\Box good(Ag) \rightarrow \Box\varphi) \tag{8.4}$$

where $\forall i \in Ag : x_i = 1$ and φ is assumed to be a propositional logic formula. The above

ATL* formula reads that on all paths, no matter what any of the agents do, *good*(Ag) always being true implies that φ will always be true. In other words, if the agents *always* follow social strategies, the goal, φ, is achieved. Referring back to the train scenario, we look at this example property:

$$Sys^{\circ}, q_{0_{11}} \models \langle\langle\rangle\rangle\, (\Box good(Ag) \rightarrow \Box\neg\, (in_E \wedge in_W)) \tag{8.5}$$

So this reads, no matter what the agents do, on all paths, if the agents always follow social strategies, this implies that the trains will never enter the tunnel at the same time. This sounds intuitively correct based on what we said earlier about social strategies leading to the objective of the social law being satisfied. Looking at Figure 8.2, we can verify by inspection that this property (8.5) is satisfied. In order to do this, we start in state $q_{0_{11}}$ and look at all paths of the system. On paths where *good*(Ag) is always true we need to check that $\neg(in_E \wedge in_W)$ is also true. This corresponds to choosing a path where g_E and g_W are *both* true in each state and then checking that $\neg(in_E \wedge in_W)$ also holds in all states across this path. Paths where *good*(Ag) is not always true require us to do nothing, as once the left hand side of the implication is made false, the whole implication becomes true regardless of the right-hand side.

We can see similarities between (8.4) and (8.1). However, they are not directly equivalent. When verifying (8.4), we look at *all* possible paths through the system, whether they are social or not. Therefore, we are going to encounter instances of *good*(G) which are false, while (8.4) will still hold. However, (8.1) only considers social paths in the system. So therefore, they are not directly equivalent, they just have a *similar* interpretation. Another thing that separates the two formulae is that (8.4) is referring to what the agents have done in the past, i.e. if they have always acted socially then φ will always hold. However, (8.1) is saying that on all paths, no matter what the agents do providing they act socially, φ will always hold. Hence (8.1) is talking about the future.

This leads us to consider this formula:

$$Sys^{\circ}, q_{x_1,\ldots,x_n} \models \langle\langle\rangle\rangle\, (\Box good(Ag) \rightarrow \Box\varphi) \wedge \langle\langle Ag\rangle\rangle \Box good(Ag) \tag{8.6}$$

where $\forall i \in Ag : x_i = 1$ and φ is assumed to be a propositional logic formula. So, as before in (8.4), this formula says that the agents always acting socially implies that the goal will always be achieved, but now also the grand coalition should have a strategy such that they will always act in accordance with the social laws. However, if we refer back to the definition of SAAETSs we see that there is a condition which states that agents should always have at least one legal action available to them in every state of the system. As Sys° is a modified version of *Sys*, this condition still holds in Sys°, thus guaranteeing that there is at least one social path, hence Ag will *always* have a strategy to ensure *good*(Ag) is true. This makes this property redundant, as it is directly equivalent to (8.4).

Finally we consider the case where we have an arbitrary coalition, G:

$$Sys^{\circ}, q_{x_1,\ldots,x_n} \models \langle\langle G \rangle\rangle \left(\Box good(G) \wedge \left(\Box good(\bar{G}) \rightarrow \Box\varphi \right) \right) \tag{8.7}$$

where $\forall i \in Ag : x_i = 1$ and φ is assumed to be a propositional logic formula. So this formula reads that coalition G has a strategy so that agents in G are always good and that if the other agents in the system are always good then φ will always hold. We needed to separate out the $good(Ag)$ proposition into $good(G)$ and $good(\bar{G})$, as G has no control over what the agents in $\bar{G}$ do. Therefore, we can precisely capture (8.1) in this way. If we refer to the trains example, we can formulate a property such as the following:

$$Sys^{\circ}, q_{0_{11}} \models \langle\langle E \rangle\rangle \left(\Box good(E) \wedge \left(\Box good(W) \rightarrow \Box\neg \left(in_E \wedge in_W \right) \right) \right) \tag{8.8}$$

This property states that the Eastbound train has a strategy so that it is always social, and if the Westbound train is always social, then the two trains will never be in the tunnel at the same time. This property holds and can be verified by inspection of Figure 8.2.

Now we consider another type of Social ATEL formula:

$$Sys, q \models \langle\langle G \rangle\rangle_s^s \Diamond\varphi \tag{8.9}$$

where φ is assumed to be a propositional logic formula. This formula says that G has a social strategy, such that, no matter what the other agents do, as long as they all only follow social strategies, then φ will be true either now or at some point in the future. We will now try to express a similar property using $good(G)$ propositions, where we take G to be the empty set, $\emptyset$:

$$Sys^{\circ}, q_{x_1,\ldots,x_n} \models \langle\langle\rangle\rangle \left(\Box good(Ag) \rightarrow \Diamond\varphi \right) \tag{8.10}$$

where $\forall i \in Ag : x_i = 1$ and φ is assumed to be a propositional logic formula. This formula says that no matter what the agents do, on all paths, $good(Ag)$ always being true means that φ will either be true now or at some point in the future. That is to say that if all the agents always act socially, the goal φ will eventually be achieved. Here we can see similarities between the above and (8.9). They are similar, but not directly equivalent, for similar reasons that we discussed earlier regarding (8.4).

We can also consider situations in which the *other* agents are constrained to social strategies and the coalition of interest can act in an unconstrained manner:

$$Sys, q \models \langle\langle G \rangle\rangle_p^s \Box \varphi \tag{8.11}$$

where φ is assumed to be a propositional logic formula. This formula states that G has a strategy to ensure φ is always true, providing the other agents always act in a social manner. We can express something similar to this in the good states approach in the following way:

$$Sys^{\circ}, q_{x_1,\dots,x_n} \models \langle\langle G \rangle\rangle \left(\Box good(\bar{G}) \rightarrow \Box \varphi\right) \tag{8.12}$$

where $\forall i \in Ag : x_i = 1$ and φ is assumed to be a propositional logic formula. So, here we are saying that G has a strategy such that no matter what the other agents in the system do, providing the other agents follow only social strategies, G can always achieve φ. We can look at an example property of the same type as (8.12):

$$Sys^{\circ}, q_{0_{11}} \models \langle\langle W \rangle\rangle \left(\Box good(E) \rightarrow \Box \neg (in_E \wedge in_W)\right) \tag{8.13}$$

This property states that the westbound train has a strategy so that if the eastbound train always acts socially, then the trains will never enter the tunnel at the same time. This property holds and can be verified by inspection of Figure 8.2.

8.3 Reducing Social ATEL to ATL*

After investigating the above formulae, we have noticed a general pattern between formulae of Social ATEL and formulae expressed in this good states approach, which seems to follow regardless of the coalition type (grand, empty or arbitrary coalition) and the temporal operator being used.

8.3.1 Bisimulations between computations

Thinking in terms of strategies, there are in general more strategies σ_i° for an agent i in S° than there are strategies σ_i for him in S. To see this consider the following example.

Example 13 *Suppose we have two agents, called* 1 *and* 2*. Suppose agent* 1 *can perform three actions in q: actions a and b (which we assume are legal) and action c (an illegal action). Suppose agent* 2 *has two actions to his disposal in q: action d (legal) and e (illegal), and no other action to choose there. Suppose furthermore that the transition function τ is such that $\tau(q, \langle a, d \rangle) = \tau(q, \langle c, d \rangle) = q$. In S°, this gives rise to transitions $\tau(q_{11}\langle a, d \rangle) = q_{11}$, while*

$\tau(q_{11}, \langle c, d\rangle) = q_{01}$. This enables agent 2 in S° to allow for strategies that depend on how lawful agent 1 behaved in the past in S°. For instance, agent 2 might play a kind of 'tit for tat' by using strategy σ with $\sigma(q_{11}q_{11}) = d$, but at the same time $\sigma(q_{11}q_{01}) = e$: a legal action of agent 1 is replied to by a legal action of agent 2, but after an illegal move of agent 1, agent 2 responds by an illegal action. Notice that such a strategy cannot be implemented in S, since both sequences $q_{11}q_{01}$ and $q_{11}q_{01}$ only correspond to the single sequence qq in S.

However, as we will see, there is a way to connect *computations* in both systems directly to each other.

Definition 30 *We say that a computation $\lambda = q_0q_1q_2\ldots$ is* compliant *with strategy profile σ_{Ag}, if $comp(\sigma_{Ag}, q_0) = \{\lambda\}$. That is, the computation can be seen as the effect of applying the strategy profile σ_{Ag} to q_0. If such a strategy profile exists for λ we also say that λ is an* enforceable *computation in the given system.*

For any state $q_{x_1..x_n} \in Q^{\circ}$, let $proj_Q(q_{x_1..x_n})$ be its corresponding state $q \in Q$. Similarly, for a sequence of states $\vec{s}$ in Q°, the sequence $proj_{Q^+}(\vec{s})$ denotes the point-wise projection of every state in the sequence to the corresponding state in Q.

Given a system S and its associated system with good states S°, let $\lambda = q_0q_1\ldots$ be a computation in S, and $\lambda^{\circ} = s_0s_1\ldots$ a computation in S°. We say that λ and λ° bisimulate *if there are two strategy profiles, σ_{Ag} in S and σ°_{Ag} in S° such that*

1. *λ is compliant with σ_{Ag} and λ° is compliant with σ°_{Ag}*
2. *for every $u \in \mathbb{N}$, and every $i \in Ag$, $\sigma_i(q_0\ldots\lambda[u]) = \sigma^{\circ}_i(s_0,\ldots\lambda^{\circ}[u])$*
3. *for every $u \in \mathbb{N}$, $\lambda[u] = proj_Q(\lambda^{\circ}[u])$*

We say in such a case also that λ and λ° bisimulate with strategy profiles *σ_{Ag} and σ°_{Ag}. Notation: $\langle\lambda, \sigma_{Ag}\rangle \simeq \langle\lambda^{\circ}, \sigma^{\circ}_{Ag}\rangle$.*

Note that a computation need not be compliant with any strategy, and, moreover, if it is compliant with one, it will in general be compliant with several others as well. The latter is so because of two reasons: first of all, a computation only specifies what the transitions are *within* a particular sequence of states, and says nothing about choices on states that do not occur in that computation. Secondly, even within a computation, it is well possible that a transition from q_i to q_{i+1} can be the effect of different choices by the grand coalition Ag. For two computations to be bisimilar, Definition 30 demands however that there are two strategies, one for each computation, in which exactly the same actions are taken, at every state in the computation. Moreover, item 3 guarantees that the computations also only visit corresponding states.

Let an *objective temporal formula* ψ be defined as follows:

$$\psi := p \mid \neg\psi \mid \psi \wedge \psi \mid \bigcirc\psi \mid \Box\psi \mid \psi \mathcal{U} \psi$$

Such formulae can be interpreted on infinite paths of states $\lambda = q_0 q_1 \ldots$ in a straightforward way:

$S, q_0 q_1 \ldots \models p$	iff	$S, q_0 \models p$
$S, q_0 q_1 \ldots \models \psi_1 \wedge \psi_2$	iff	$S, q_0 q_1 \ldots \models \psi_1$ and $S, q_0 q_1 \ldots \models \psi_2$
$S, q_0 q_1 \ldots \models \bigcirc\psi$	iff	$S, q_1 \ldots \models \psi$
$S, q_0 q_1 \ldots \models \Box\psi$	iff	$\forall i, S, q_i q_{i+1} \ldots \models \psi$
$S, q_0 q_1 \ldots \models \psi_1 \mathcal{U} \psi_2$	iff	$\exists i$ s.t. $S, q_i q_{i+1} \ldots \models \psi_2$ and $\forall 0 \leq j < i, S, q_j q_{j+1} \ldots \models \psi_1$

Note that, in particular, since the propositions g_i are atomic propositions in S°, we can interpret them on an infinite path in S°.

Lemma 7 *Let $\lambda = q_0 q_1 q_2 \ldots$ and $\lambda^\circ = s_0 s_1 s_2 \ldots$. Suppose furthermore that $\langle \lambda, \sigma_{Ag} \rangle \simeq \langle \lambda^\circ, \sigma^\circ_{Ag} \rangle$. Then:*

1. *for every $u \in \mathbb{N}$ and every objective temporal formula ψ:*

$$S, \lambda[u] \models \psi \text{ iff } S^\circ, \lambda^\circ[u] \models \psi$$

2. *for every $u \in \mathbb{N}, i \in Ag$,*

$$\lambda[u] \in \ell(\sigma_i(q_0 \ldots \lambda[u])) \text{ iff } S^\circ, \lambda^\circ[u+1] \models g_i$$

Proof:

1. Let λ and λ° be as specified. Recall that, by item 3 of Definition 30, for all $u \in \mathbb{N}$, $\lambda[u] = proj_Q(\lambda^\circ[u])$ $(*)$. We now prove by induction ψ that for all $u \in \mathbb{N}$, $S, \lambda[u] \models \psi$ iff $S^\circ, \lambda^\circ[u] \models \psi$. For atomic propositions p, this follows immediately from the equivalence $(*)$ and the definition of π°. Now suppose the property is proven for ψ: we only do the $\bigcirc$-case. Let $u \in \mathbb{N}$ be arbitrary.

$S, \lambda[u] \models \bigcirc\psi$	iff	(definition of $\bigcirc$)
$S, \lambda[u+1] \models \psi$	iff	(induction hypothesis)
$S^\circ, \lambda^\circ[u+1] \models \psi$	iff	(definition of$\bigcirc$)
$S^\circ, \lambda^\circ[u] \models \bigcirc\psi$		

2. Let $\lambda[u+1] = q$, for some $q \in Q$. Note that, by item 3 of Definition 30 $proj_Q(\lambda^\circ[u+1]) = q$. So, $\lambda^\circ[u+1]$ is $q_{x_1 \ldots x_{i-1}, x_i, x_{i+1}, \ldots x_n}$ for some sequence $x_1 \ldots x_{i-1}, x_i, x_{i+1}, \ldots x_n \in$

$\{0,1\}^n$. By definition of τ°, we have

$$x_i = 1 \text{ iff } \lambda[u] \in \ell(\sigma_i(q_0 \dots \lambda[u])) \tag{8.14}$$

Moreover, by the truth definition of g_i, we have

$$x_i = 1 \text{ iff } S^\circ, \lambda^\circ \models g_i \tag{8.15}$$

The result now immediately follows from (8.14) and (8.15) together.

□

So, if we have two computations that bisimulate, then according to item 1 of Lemma 7, they verify the same objective temporal formulae, and, by item 2, a choice for an agent i in the original system is legal, if and only if in the associated system, in the state that results the proposition g_i is true.

Definition 31 *Let a computation λ be compatible with strategy profile σ_{Ag}. We say that coalition G behaves social according to this profile along the computation λ, if $\forall u \in \mathbb{N}, \forall i \in G$, $(\lambda[u] \in \ell(\sigma_i(q_0 \dots \lambda[u])))$.*

So, G behaves social along λ, if it has a contribution to generate the computation λ that consists only of social actions.

Corollary 1 *Suppose that λ and λ° are bisimilar computations, with strategy profiles σ_{Ag} and σ°_{Ag}, respectively. Let G be an arbitrary coalition. Then*

1. *G behaves social according to σ_{Ag} along λ iff $S^\circ, \lambda^\circ \models \Box good(G)$*
2. *If for all $i \in G$, $\sigma_i \in \Delta_i$, then $S^\circ, \lambda^\circ \models \Box good(G)$.*

Proof:

1. Note that item 2 of Lemma 7 implies

$$\forall u \in \mathbb{N}, \forall i \in G \quad (\lambda[u] \in \ell(\sigma_i(q_0 \dots \lambda[u])) \text{ iff } S^\circ, \lambda^\circ[u+1] \models g_i)$$

This, in turn, implies

$$(\forall u \in \mathbb{N}, \forall i \in G \;\; \lambda[u] \in \ell(\sigma_i(q_0 \dots \lambda[u]))) \;\text{ iff } \forall u \in \mathbb{N}, \forall i \in G(S^\circ, \lambda^\circ[u+1] \models g_i)$$

The left-hand side of the above 'iff' states that G behaves social according to σ_{Ag} along λ, and the right-hand side is equivalent to $S^\circ, \lambda^\circ \models \Box good(G)$.

2. This follows immediately from the above: note that if for all $i \in G$, $\sigma_i \in \Delta_i$, then $\forall u \in \mathbb{N}, \forall i \in G\ \lambda[u] \in \ell(\sigma_i(q_0 \ldots \lambda[u]))$.

□

The converse of item 2 of Corollary 1 is in general not true: if $S^\circ, \lambda^\circ \models good(G)$, then we only know that along the computation λ, all members of G behave well, but outside λ, they may not and hence their strategy need not be social.

Definition 32 *Let λ be a computation. We say that strategies σ_i and σ'_i for i* coincide along *λ, written, $\sigma_i \equiv_\lambda \sigma'_i$, if for all $u \in \mathbb{N}$, $\sigma_i(\lambda[0] \ldots \lambda[u]) = \sigma'_i(\lambda[0] \ldots \lambda[u])$. For a coalition, $\sigma_G \equiv \sigma'_G$, if for every $i \in G$, $\sigma_i \equiv_\lambda \sigma'_i$. Moreover, for any strategy profile σ_{Ag}, and τ_i a strategy for i, we write $\sigma[\tau_i/\sigma_i]$ for the strategy profile that is like σ_{Ag}, except that for agent i, the component σ_i is replaced by τ_i. Similarly for $\sigma[\tau_G/\sigma_G]$.*

It is easy to see that if σ_{Ag} is compliant with λ, and $\sigma_i \equiv_\lambda \sigma'_i$, then $\sigma[\sigma'_i/\sigma_i]_{Ag}$ is also compliant with λ.

Lemma 8 *Suppose λ is a computation, and strategy profile σ is compliant with it. Suppose furthermore that G behaves social according to this profile along λ. Then there exist a set of strategies τ_G, such that $\tau_G \in \Delta_G$ and $\sigma_G \equiv_\lambda \tau_G$.*

Proof: For every finite prefix $q_0q_1 \ldots q_n$ of λ, and every $i \in G$, we take $\tau_i(q_0q_1 \ldots q_n) = \sigma_i(q_0q_1 \ldots q_n)$. For this choice, we obviously have $\sigma_i \equiv_\lambda \tau_i$. Also, note that every action $\tau_i(q_0q_1 \ldots q_n)$ is a legal choice, because i behaves social according to the profile σ along λ. Since by definition of a system S every agent can always perform a social action, we can extend τ_i for any other sequence $\vec{s}$ of states in such a way that the choice $\tau_i(\vec{s})$ is a legal action. It is clear that τ_G satisfies the conditions of the lemma. □

We noted above that in general there are more strategies for a coalition in S° than there are in S. Our main result connecting a system S with S° now says that all enforceable computations in one of the systems have a computation that is bisimilar in the other.

Theorem 16 *Let S and S° be as defined before. Suppose λ is compliant with profile σ_{Ag}. Then: there is a computation λ° in S° and a strategy profile σ°_{Ag}, such that $\langle \lambda, \sigma_{Ag} \rangle \simeq \langle \lambda^\circ, \sigma^\circ_{Ag} \rangle$. The converse is also true: for every computation λ° in S° that is compliant with a strategy λ°_{Ag} we can find a strategy profile σ_{Ag} and computation λ in S such that $\langle \lambda, \sigma_{Ag} \rangle \simeq \langle \lambda^\circ, \sigma^\circ_{Ag} \rangle$.*

Proof: From S to S°: let $\lambda = q_0q_1 \ldots$ be an enforceable computation in S and let it be compliant with σ_{Ag}. Let σ° be a strategy in S° for agent i satisfying:

$$\sigma^\circ_i(\vec{s}) = \sigma_i(proj_{Q^+}(\vec{s}))$$

The strategy profile σ°_{Ag} generates a computation $\lambda^{\circ} = s_0 s_1 \ldots$ for any s_0 with $proj_Q(s_0) = q_0$. Hence, by definition, σ°_{Ag} is compliant with λ°. This shows item 1 of Definition 30. By definition of this λ° and σ°_i, also item 2 of Definition 30 is satisfied. We now demonstrate item 3 using induction on the length of the computations, u. If $u = 0$, $\lambda[0] = q_0 = proj_Q(s_0) = \lambda^{\circ}[0]$ (note that $q_0 = proj_Q(s_0)$ by our choice of s_0). Now suppose the following induction hypothesis (*ih*) holds: $\forall n \leq u : \lambda[n] = \lambda^{\circ}[n]$. Then

$$\begin{array}{lll} \lambda[u+1] & = & \text{by definition of computation} \\ \tau(\lambda[u], \sigma_{Ag}(q_0 \ldots \lambda[u])) & = & \text{definition of } \tau^{\circ} \text{ and } ih \\ \tau^{\circ}(\lambda^{\circ}[u], \sigma^{\circ}_{Ag}(s_0 \ldots \lambda^{\circ}[u])) & = & \text{by definition of computation} \\ \lambda^{\circ}[u+1] & & \end{array}$$

From S° to S: Let $\lambda^{\circ} = s_0 s_1 \ldots$. Since λ° is enforceable, we know that λ° is generated by some strategy profile σ°_{Ag}. Let $\lambda = proj_{Q^+}(\lambda^{\circ}) = proj_Q(s_0) proj_Q(s_1) \ldots$. It is easy to see that λ is generated by any strategy profile σ_{Ag} that satisfies, for any $i \in Ag$:

$$\sigma_i(proj_Q(s_0) proj_Q(s_1) \ldots proj_Q(s_u)) = \sigma^{\circ}_i(s_0 s_1 \ldots s_u)$$

Hence, σ_{Ag} is compliant with λ. Item 2 of Definition 30 follows directly, as does item 3 of that definition. □

Going briefly back to Example 13, although the strategy τ in S° for agent 2 that simulates 'tit for tat' has not immediately a counterpart in S, in a *computation* λ°, agent 1 must have made up his mind between behaving 'good' and 'bad', and hence we either have the computation $q_{11} q_{11} \ldots$ or $q_{11} q_{01} \ldots$. And *those do have* a counterpart $qq \ldots$ in S, and they are generated by different strategy profiles: in the first, 1 plays initially a, in the second, it would be c.

8.3.2 Proving some Reductions

We propose that the following reduction is true where G acts socially and the other agents also act socially:

$$Sys, q \models \langle\langle G \rangle\rangle^s_s T\varphi \iff Sys^{\circ}, q_{x_1,\ldots,x_n} \models \langle\langle G \rangle\rangle \left(\Box good(G) \wedge \left(\Box good(\bar{G}) \rightarrow T\varphi \right) \right)$$

where T is an arbitrary temporal operator, φ is a propositional logic formula and $\forall x : x = 1$.

To illustrate this general pattern, consider the following variations of (8.1):

1. G is an arbitrary coalition:

$$Sys^{\circ}, q_{x_1,\ldots,x_n} \models \langle\langle G \rangle\rangle \left(\Box good(G) \wedge \left(\Box good(\bar{G}) \rightarrow \Box\varphi \right) \right)$$

2. $G = \emptyset$:
$$Sys^\circ, q_{x_1,\ldots,x_n} \models \langle\langle\rangle\rangle\,(\Box\top \wedge (\Box good(Ag) \rightarrow \Box\varphi))$$

3. $G = Ag$:
$$Sys^\circ, q_{x_1,\ldots,x_n} \models \langle\langle Ag\rangle\rangle\,(\Box good(Ag) \wedge (\Box\top \rightarrow \Box\varphi))$$

Notice how 2. reduces down to be similar to (8.10) and 3. reduces to be (8.2). Now we will prove the following:

Proposition 8 *Let ψ be an objective path formula, and T a temporal operator. Let q° be such that $proj_Q(q^\circ) = q$.*

$$Sys, q \models \langle\langle G\rangle\rangle_s^s T\psi \iff Sys^\circ, q^\circ \models \langle\langle G\rangle\rangle\,\big(\Box good(G) \wedge \big(\Box good(\bar{G}) \rightarrow T\psi\big)\big)$$

Proof: Let ψ be an objective path formula. Suppose $Sys, q \models \langle\langle G\rangle\rangle_s^s T\psi$. This means that there is a social strategy $\sigma_G \in \Delta_G$ for G, such that for any social strategy $\sigma_{\bar{G}} \in \Delta_{\bar{G}}$ for $\bar{G}$, if $\lambda = comp(\langle\sigma_G, \sigma_{\bar{G}}\rangle, q)$, then $Sys, \lambda \models T\psi$. $(*)$.

Now consider an arbitrary state q° in Sys° for which $proj_Q(q^\circ) = q$. Let, for every $i \in G$, the strategy σ_i° for agent i be *fixed:* it is exactly like σ_i on every projected sequence, that is, let $\sigma_i^\circ(\vec{s}; s)$ be $\sigma_i(proj_{Q^+}(\vec{s}); proj_Q(s))$. Now consider an *arbitrary strategy* $\sigma_{\bar{G}}^\circ$ for $\bar{G}$, and let $\lambda^\circ = comp(\langle\sigma_G^\circ, \sigma_{\bar{G}}^\circ\rangle, q^\circ)$. If we can show that for any such λ°, we have $S^\circ, \lambda^\circ \models \big(\Box good(G) \wedge \big(\Box good(\bar{G}) \rightarrow \Box\psi\big)\big)$, we are done.

To show that $Sys^\circ, \lambda^\circ \models \Box good(G)$, recall that σ_G is a social strategy for G, and then apply Corollary 1, item 2. To show that also $Sys^\circ, \lambda^\circ \models \Box good(\bar{G}) \rightarrow \Box\psi$, assume that $Sys^\circ, \lambda^\circ \models \Box good(\bar{G})$. Recall that λ° is a computation $\lambda^\circ = comp(\langle\sigma_G^\circ, \sigma_{\bar{G}}^\circ\rangle, q^\circ)$, where σ_G° is fixed. To stress that the computation depends on the strategy σ_G°, we will also write $\lambda^\circ(\sigma_G^\circ)$ for λ°. Obviously, each such $\lambda^\circ(\sigma_{\bar{G}}^\circ)$ is compliant with $\langle\sigma_G^\circ, \sigma_{\bar{G}}^\circ\rangle$. For each such $\sigma_{\bar{G}}^\circ$, let the strategy $\sigma_{\bar{G}}$ in Sys be obtained from $\sigma_{\bar{G}}^\circ$ in the standard way: it has to satisfy, for every agent $\bar{g}$ in the coalition $\bar{G}$, that $\sigma_{\bar{g}}(proj_Q(\lambda^\circ[0]) \ldots proj_Q(\lambda^\circ[n])) = \sigma_{\bar{g}}^\circ(\lambda^\circ[0], \ldots \lambda^\circ[n])$. Let $\lambda(\sigma_{\bar{G}}) = comp(\langle\sigma_G, \sigma_{\bar{G}}\rangle, q)$, one computation for each $\sigma_{\bar{G}}$. It is clear that, for each $\sigma_{\bar{G}}^\circ$, $\lambda(\sigma_{\bar{G}})$ and $\lambda^\circ(\sigma_{\bar{G}}^\circ)$ are bisimilar computations, with strategy profiles $\langle\sigma_G, \sigma_{\bar{G}}\rangle$ and $\langle\sigma_G^\circ, \sigma_{\bar{G}}^\circ\rangle$, respectively. Since we assumed $Sys^\circ, \lambda^\circ(\sigma_{\bar{G}}^\circ) \models \Box good(\bar{G})$, by Corollary 1, item 1, we have that $\bar{G}$ behaves social according to $\langle\sigma_G, \sigma_{\bar{G}}\rangle$ along λ. By Lemma 8, there is a strategy τ_G for G such that $\tau_G \equiv_\lambda \sigma_G$, and $\tau_G \in \Delta_G$. That is, τ_G is a social strategy. By $(*)$, we then have for all $u \in \mathbb{N}$, that $Sys, \lambda[u] \models \psi$, i.e., $Sys, \lambda \models \Box\psi$. Since λ and λ° are bisimulating computations, we also have $Sys^\circ, \lambda^\circ \models T\psi$, which had to be proven.

For the converse, suppose $Sys^\circ, q^\circ \models \langle\langle G\rangle\rangle\,\big(\Box good(G) \wedge \big(\Box good(\bar{G}) \rightarrow T\psi\big)\big)$. By the semantics of ATL, this means that there is a strategy σ_G° for G, such that for all strategies $\sigma_{\bar{G}}^\circ$, if $\lambda^\circ = comp(\langle\sigma^\circ_G, \sigma^\circ_{\bar{G}}\rangle, q^\circ)$, then $Sys^\circ, \lambda^\circ \models (\Box good(G) \wedge (\Box good(\bar{G}) \rightarrow T\psi))$. Note that $\sigma^\circ_{\bar{G}}$ is fixed. For any $q \in Q$, let $q_{\bar{1}}$ be the a corresponding state in S° with all indices being a 1. Now we define a strategy σ_i, for each $i \in G$, as follows:

$$\sigma_i(q_0 q_1 \dots q_n) = \sigma^\circ_i(q_{0_{\bar{1}}} q_{1_{\bar{1}}} \dots q_{n_{\bar{1}}})$$

That is, σ_i 'copies' the behaviour as prescribed in σ°_i on 'good paths'. We are going to show that σ_G has the property that, if we combine it with any $\sigma_{\bar{G}} \in \Delta_{\bar{G}}$, every computation $\lambda = comp(\langle\sigma_G, \sigma_{\bar{G}}\rangle, q)$ has the property that $S, \lambda \models T\psi$. This would complete the proof that $S, q \models \langle\langle G\rangle\rangle^s_s T\psi$.

So, take any social strategy $\sigma_{\bar{G}}$ for $\bar{G}$. Take the strategy profile $\sigma_{Ag} = \langle\sigma_G, \sigma_{\bar{G}}\rangle$. Obviously, this is in Δ_{Ag}, since both coalitions act socially. Let λ be the computation $comp(\langle\sigma_G, \sigma_{\bar{G}}\rangle, q)$. We know from Theorem 16 that there is a computation $\lambda^\bullet$ and a strategy profile $\sigma^\bullet_{Ag}$ such that $\langle\lambda, \sigma_{Ag}\rangle \simeq \langle\lambda^\bullet, \sigma^\bullet_{Ag}\rangle$. Note that we use fresh symbols for this compuation and strategy profile, since λ° and σ° already have a meaning. Now we argue that $\lambda^\bullet$ can be conceived as a computation $comp(\langle\sigma^\circ_G, \sigma^\circ_{\bar{G}}\rangle, q^\circ)$. First of all, recall that both G and $\bar{G}$ are acting socially in S. Hence, the computation $\lambda^\bullet$ is of the form $q^\circ q_{1_{\bar{1}}} q_{2_{\bar{1}}} \dots$. In other words, apart from possibly the first state, all states have indices x_i that are all equal to 1! But then, on this computation, we can just assume that the strategy of G is the earlier σ°_G: on sequences with only 1's, we have copied the choices of σ°_G to σ_G and now back to $\lambda^\bullet$. Formally: for all $u \in \mathbb{N}$ and $i \in G$: $\sigma^\bullet_i(q^\circ q_{1_{\bar{1}}} q_{2_{\bar{1}}} \dots q_{u_{\bar{1}}}) = \sigma_i(q q_1 q_2 \dots q_u) = \sigma^\circ_i(q^\circ q_{1_{\bar{1}}} q_{2_{\bar{1}}} \dots q_{u_{\bar{1}}})$. Moreover, since $\sigma_{Ag} \in \Delta_{Ag}$, we have $S^\circ, \lambda^\bullet \models \Box(good(G) \wedge good(\bar{G}))$. But we know that on such paths, when they are generated by σ_G, that $T\psi$ holds. Now we use Lemma 7, to conclude that $S, \lambda \models T\psi$, as required. □

So far we have only looked at the reduction for cases where G is acting socially and the other agents are also acting socially. When we alter the type of strategies that the agents must follow, we conjecture the following reductions:

Conjecture 1 *Let ψ be an objective path formula, and T a temporal operator. Let q° be such that $proj_Q(q^\circ) = q$.*

1. $Sys, q \models \langle\langle G\rangle\rangle^p_s T\psi \iff Sys^\circ, q^\circ \models \langle\langle G\rangle\rangle (\Box good(G) \wedge T\psi)$
2. $Sys, q \models \langle\langle G\rangle\rangle^s_p T\psi \iff Sys^\circ, q^\circ \models \langle\langle G\rangle\rangle (\Box good(\bar{G}) \rightarrow T\psi)$
3. $Sys, q \models \langle\langle G\rangle\rangle^p_p T\psi \iff Sys^\circ, q^\circ \models \langle\langle G\rangle\rangle T\psi$

Now we consider a more complex Social ATEL formula and see whether we can express a similar property in the good states approach:

$$Sys, q \models \langle\langle G \rangle\rangle_p^s \bigcirc \langle\langle \bar{G} \rangle\rangle_s^p \bigcirc \varphi \tag{8.16}$$

where φ is assumed to be a propositional logic formula. This formula states that G has a strategy, providing all the other agents only follow social strategies, such that in the next state $\bar{G}$ will have a social strategy to achieve φ in the next state, no matter how the other agents act. We can express a similar formula in the good states approach:

$$Sys^\circ, q_{x_1,\dots,x_n} \models \langle\langle G \rangle\rangle \left((\Box good(\bar{G})) \rightarrow \bigcirc \left(\langle\langle \bar{G} \rangle\rangle \left(\Box good(\bar{G}) \right) \rightarrow \bigcirc \varphi \right) \right) \tag{8.17}$$

where $\forall i \in Ag : x_i = 1$ and φ is assumed to be a propositional logic formula.

There are also properties we can express with the good states approach that we can't express with Social ATEL. For example, the following formula:

$$Sys^\circ, q_{x_1,\dots,x_n} \models \langle\langle \rangle\rangle \left(good(Ag)\, \mathcal{U}\, \varphi \right) \tag{8.18}$$

Now this formula is indeed similar to (8.9), however there is a difference. Both are generally saying that if Ag act in a social manner then φ will eventually be true, however, (8.18) only requires Ag to follow social strategies *until* φ is satisfied whereas (8.9) requires Ag to *always* act in a social manner.

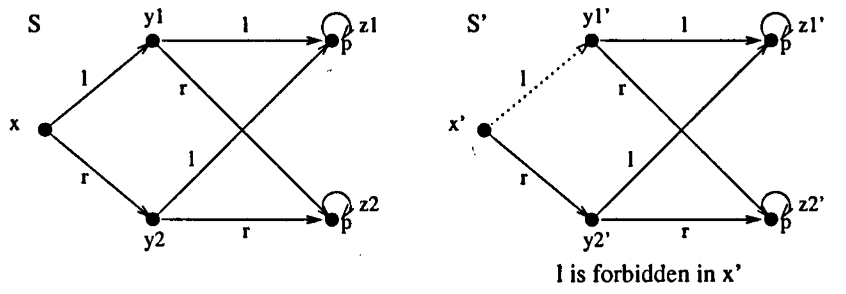

Figure 8.3: Systems S and S'

We now prove that (8.18) can not be expressed in Social ATEL. Firstly, we introduce two systems, S and S', illustrated in Figure 8.3. Both of these systems consist of only one agent, which in each state is faced with the choice of two actions: l or r (for left or right, respectively). There is only one atomic proposition, p, which holds in states z_1 and z_2 (z_1' and z_2' in S'). Both the systems are the same apart from in S' the action l is forbidden in x'. It does not matter if the agent chooses l or r as both of these actions lead to a state where the same propositions hold.

This is given formally by the following lemma:

Lemma 9 *For the two systems S and S', and the set of all Social* ATEL *formulae, Φ, the following holds:*

a. $\forall \varphi \in \Phi : S, y_1 \models \varphi \iff S, y_2 \models \varphi$ *and* $\forall \varphi \in \Phi : S, z_1 \models \varphi \iff S, z_2 \models \varphi$;

b. $\forall \varphi \in \Phi : S', y_1' \models \varphi \iff S', y_2' \models \varphi$ *and* $\forall \varphi \in \Phi : S', z_1' \models \varphi \iff S', z_2' \models \varphi$.

Proof: We only prove part a., as the proof of part b. follows exactly the same reasoning. Firstly, we want to prove:

$$\forall \varphi \in \Phi : S, z_1 \models \varphi \iff S, z_2 \models \varphi$$

by induction on φ.

1. For atomic propositions p, this can easily be verified by inspection.

2. For the $\neg\varphi$ case suppose the following induction hypothesis holds: $S, z_1 \models \varphi \iff S, z_2 \models \varphi$.

$S, z_1 \models \neg\varphi$	iff	(definition of $\neg\varphi$)
not $S, z_1 \models \varphi$	iff	(induction hypothesis)
not $S, z_2 \models \varphi$	iff	(definition of $\neg\varphi$)
$S, z_2 \models \neg\varphi$		

3. For the case of $\varphi_1 \wedge \varphi_2$, suppose the following induction hypothesis holds: $S, z_1 \models \varphi_1 \iff S, z_2 \models \varphi_1$ and $S, z_1 \models \varphi_2 \iff S, z_2 \models \varphi_2$.

$S, z_1 \models \varphi_1 \wedge \varphi_2$	iff	(definition of $\varphi_1 \wedge \varphi_2$)
$S, z_1 \models \varphi_1$ and $S, z_1 \models \varphi_2$	iff	(induction hypothesis)
$S, z_2 \models \varphi_1$ and $S, z_2 \models \varphi_2$	iff	(definition of $\varphi_1 \wedge \varphi_2$)
$S, z_2 \models \varphi_1 \wedge \varphi_2$		

4. For the $\langle\langle G \rangle\rangle_p^P T\varphi$ case, where T is an arbitrary temporal operator, suppose the following induction hypothesis holds: $S, z_1 \models \varphi \iff S, z_2 \models \varphi$. $S, z_1 \models \langle\langle G \rangle\rangle_p^P T\varphi$ gives rise to computations which only visit z_1. $S, z_2 \models \langle\langle G \rangle\rangle_p^P T\varphi$ gives rise to computations which only visit z_2. If $S, z_1 \models \varphi$, then by the induction hypothesis $S, z_2 \models \varphi$, hence $S, z_1 \models \langle\langle G \rangle\rangle_p^P T\varphi \iff S, z_2 \models \langle\langle G \rangle\rangle_p^P T\varphi$. To illustrate this, consider the example case of $\langle\langle G \rangle\rangle_p^P \Box\varphi$. $S, z_1 \models \langle\langle G \rangle\rangle_p^P \Box\varphi$ iff $\exists \sigma_G \in \Gamma_G$ s.t. $\forall \sigma_{\bar{G}} \in \Gamma_{\bar{G}}$ if $\lambda = comp(\langle \sigma_G, \sigma_{\bar{G}} \rangle, z_1)$ then $\forall u \in \mathbb{N} : S, \lambda[u] \models \varphi$. Notice that $\forall u \in \mathbb{N} : \lambda[u] = z_1$. $S, z_2 \models \langle\langle G \rangle\rangle_p^P \Box\varphi$ iff $\exists \sigma_G' \in \Gamma_G$ s.t. $\forall \sigma_{\bar{G}}' \in \Gamma_{\bar{G}}$ if $\lambda' = comp(\langle \sigma_G', \sigma_{\bar{G}}' \rangle, z_2)$ then $\forall u \in \mathbb{N} : S, \lambda'[u] \models \varphi$.

Also, notice that $\forall u \in \mathbb{N} : \lambda'[u] = z_2$. If $S, z_1 \models \varphi$, then by the induction hypothesis, $S, z_2 \models \varphi$, hence, $S, z_1 \models \langle\langle G\rangle\rangle_p^p \Box\varphi \iff S, z_2 \models \langle\langle G\rangle\rangle_p^p \Box\varphi$. The $\langle\langle G\rangle\rangle_s^s T\varphi$ case follows the same reasoning.

Finally, we want to prove:

$$\forall\varphi \in \Phi : S, y_1 \models \varphi \iff S, y_2 \models \varphi$$

by induction on φ.

1. The cases of p, $\neg\varphi$ and $\varphi_1 \wedge \varphi_2$ are trivial and follow the same reasoning as above.

2. For the $\langle\langle G\rangle\rangle_p^p T\varphi$ case, notice how $\tau(y_1, l) = z_1$ and $\tau(y_2, l) = z_1$, and also $\tau(y_1, r) = z_2$ and $\tau(y_2, r) = z_2$. So l and r lead to the same states no matter whether they are performed in y_1 or y_2. As we have already proven $\forall\varphi \in \Phi : S, z_1 \models \varphi \iff S, z_2 \models \varphi$, it follows that $S, y_1 \models \langle\langle G\rangle\rangle_p^p T\varphi \iff S, y_2 \models \langle\langle G\rangle\rangle_p^p T\varphi$. To illustrate this, consider the example case of $\langle\langle G\rangle\rangle_p^p \bigcirc\varphi$. $S, y_1 \models \langle\langle G\rangle\rangle_p^p \bigcirc\varphi$ iff $\exists\sigma_G \in \Gamma_G$ s.t. $\forall\sigma_{\bar{G}} \in \Gamma_{\bar{G}}$ if $\lambda = comp(\langle\sigma_G, \sigma_{\bar{G}}\rangle, y_1)$ then $S, \lambda[1] \models \varphi$. As $\lambda[1] = z_1$ or $\lambda[1] = z_2$ and we have already proven $\forall\varphi \in \Phi : S, z_1 \models \varphi \iff S, z_2 \models \varphi$, it follows that $S, y_1 \models \langle\langle G\rangle\rangle_p^p \bigcirc\varphi \iff S, y_2 \models \langle\langle G\rangle\rangle_p^p \bigcirc\varphi$. The $\langle\langle G\rangle\rangle_s^s T\varphi$ case follows the same reasoning.

□

The following theorem illustrates the fact that (8.18) can not be expressed in Social ATEL:

Theorem 17 *Given the two systems, S and S', shown in Figure 8.3, we claim that $\forall\varphi \in \Phi$, where Φ is the set of all Social* ATEL *formulae: $S, x \models \varphi \iff S', x' \models \varphi$, but in the system with good states, $S, x \models \langle\langle\rangle\rangle(good(Ag)\,\mathcal{U}\,p)$ while $S', x' \not\models \langle\langle\rangle\rangle(good(Ag)\,\mathcal{U}\,p)$.*

Proof: We prove

$$\forall q \in Q, \forall\varphi \in \Phi : S_1, q \models \varphi \iff S_2, q' \models \varphi$$

by induction on φ.

1. For atomic propositions p, this can easily be verified by inspection.

2. For the $\neg\varphi$ case, let $q \in Q$ be arbitrary and suppose the following induction hypothesis holds: $\forall q \in Q : S, q \models \varphi \iff S', q' \models \varphi$.

$S, q \models \neg\varphi$	iff	(definition of $\neg\varphi$)
not $S, q \models \varphi$	iff	(induction hypothesis)
not $S', q' \models \varphi$	iff	(definition of$\neg\varphi$)
$S', q' \models \neg\varphi$		

3. For the case of $\varphi_1 \wedge \varphi_2$, let $q \in Q$ be arbitrary and suppose the following induction hypothesis holds: $\forall q \in Q : S, q \models \varphi_1 \iff S', q' \models \varphi_1$ and $\forall q \in Q : S, q \models \varphi_2 \iff S', q' \models \varphi_2$.

$S, q \models \varphi_1 \wedge \varphi_2$	iff	(definition of $\varphi_1 \wedge \varphi_2$)
$S, q \models \varphi_1$ and $S, q \models \varphi_2$	iff	(induction hypothesis)
$S', q' \models \varphi_1$ and $S', q' \models \varphi_2$	iff	(definition of $\varphi_1 \wedge \varphi_2$)
$S', q' \models \varphi_1 \wedge \varphi_2$		

4. For the $\langle\langle\rangle\rangle_s^s \bigcirc \varphi$ case, we suppose the following induction hypothesis holds: $\forall q \in Q : S, q \models \varphi \iff S', q' \models \varphi$. We begin by taking the state $x \in Q$:

   ($\Rightarrow$) $S, x \models \langle\langle\rangle\rangle_s^s \bigcirc \varphi$ iff $\forall \sigma_{Ag} \in \Delta_{Ag}$ s.t. if $\lambda = comp(\sigma_{Ag}, x)$ we have $S, \lambda[1] \models \varphi$. $\lambda[1]$ is either y_1 or y_2, but by lemma 9a, we can take y_2. So, $S, y_2 \models \varphi$ and by the induction hypothesis $S', y_2' \models \varphi$. Then, using lemma 9b, we see that $\forall \sigma'_{Ag} \in \Delta'_{Ag}$ if $\lambda' = comp(\sigma'_{Ag}, x')$ we have $S', \lambda'[1] \models \varphi$. Hence, $S', x' \models \langle\langle\rangle\rangle_s^s \bigcirc \varphi$.

   ($\Leftarrow$) $S', x' \models \langle\langle\rangle\rangle_s^s \bigcirc \varphi$ iff $\forall \sigma'_{Ag} \in \Delta'_{Ag}$ s.t. if $\lambda' = comp(\sigma'_{Ag}, x')$ we have $S', \lambda'[1] \models \varphi$. $\lambda'[1]$ is y_2', as $\sigma_{Ag}(x') = r$ because l is forbidden in x'. So, $S', y_2' \models \varphi$ and by the induction hypothesis $S, y_2 \models \varphi$. Then, using lemma 9a, we see that $\forall \sigma_{Ag} \in \Delta_{Ag}$ if $\lambda = comp(\sigma_{Ag}, x)$ we have $S, \lambda[1] \models \varphi$. Hence, $S, x \models \langle\langle\rangle\rangle_s^s \bigcirc \varphi$.

   Now we take the state $y_1 \in Q$:

   ($\Rightarrow$) $S, y_1 \models \langle\langle\rangle\rangle_s^s \bigcirc \varphi$ iff $\forall \sigma_{Ag} \in \Delta_{Ag}$ s.t. if $\lambda = comp(\sigma_{Ag}, y_1)$ we have $S, \lambda[1] \models \varphi$. $\lambda[1]$ is either z_1 or z_2. In the case where $\lambda[1] = z_1$: $S, z_1 \models \varphi$ and by the induction hypothesis $S', z_1' \models \varphi$. In the case where $\lambda[1] = z_2$: $S, z_2 \models \varphi$ and by the induction hypothesis $S', z_2' \models \varphi$. Then $\forall \sigma'_{Ag} \in \Delta'_{Ag}$ if $\lambda' = comp(\sigma'_{Ag}, y_1')$ we have $S', \lambda'[1] \models \varphi$. Hence, $S', y_1' \models \langle\langle\rangle\rangle_s^s \bigcirc \varphi$.

   ($\Leftarrow$) $S', y_1' \models \langle\langle\rangle\rangle_s^s \bigcirc \varphi$ iff $\forall \sigma'_{Ag} \in \Delta'_{Ag}$ s.t. if $\lambda' = comp(\sigma'_{Ag}, y_1')$ we have $S', \lambda'[1] \models \varphi$. $\lambda'[1]$ is either z_1' or z_2'. In the case where $\lambda'[1] = z_1'$: $S', z_1' \models \varphi$ and by the induction hypothesis $S, z_1 \models \varphi$. In the case where $\lambda'[1] = z_2'$: $S', z_2' \models \varphi$ and by the induction hypothesis $S, z_2 \models \varphi$. Then $\forall \sigma_{Ag} \in \Delta_{Ag}$ if $\lambda = comp(\sigma_{Ag}, y_1)$ we have $S, \lambda[1] \models \varphi$. Hence, $S, y_1 \models \langle\langle\rangle\rangle_s^s \bigcirc \varphi$.

   The cases for the remaining states all follow the same reasoning as from y_1. The $\langle\langle i\rangle\rangle_s^s \bigcirc \varphi$ case is very similar and the $\langle\langle Ag\rangle\rangle_s^s \bigcirc \varphi$ is identical to $\langle\langle i\rangle\rangle_s^s \bigcirc \varphi$ as we only have one agent.

5. For the $\langle\langle\rangle\rangle_p^p \bigcirc \varphi$ case, we suppose the following induction hypothesis holds: $\forall q \in Q : S, q \models \varphi \iff S', q' \models \varphi$. We begin by taking the state $x \in Q$:

($\Rightarrow$) $S,x \models \langle\langle\rangle\rangle_p^p \bigcirc \varphi$ iff $\forall \sigma_{Ag} \in \Gamma_{Ag}$ s.t. if $\lambda = comp(\sigma_{Ag},x)$ we have $S,\lambda[1] \models \varphi$. $\lambda[1]$ is either y_1 or y_2. In the case where $\lambda[1] = y_1$: $S,y_1 \models \varphi$ and by the induction hypothesis $S',y'_1 \models \varphi$. In the case where $\lambda[1] = y_2$: $S,y_2 \models \varphi$ and by the induction hypothesis $S',y'_2 \models \varphi$. Then $\forall \sigma'_{Ag} \in \Gamma'_{Ag}$ if $\lambda' = comp(\sigma'_{Ag},x')$ we have $S',\lambda'[1] \models \varphi$. Hence, $S',x' \models \langle\langle\rangle\rangle_p^p \bigcirc \varphi$.

($\Leftarrow$) $S',x' \models \langle\langle\rangle\rangle_p^p \bigcirc \varphi$ iff $\forall \sigma'_{Ag} \in \Gamma'_{Ag}$ s.t. if $\lambda' = comp(\sigma'_{Ag},x')$ we have $S',\lambda'[1] \models \varphi$. $\lambda'[1]$ is either y'_1 or y'_2. In the case where $\lambda'[1] = y'_1$: $S',y'_1 \models \varphi$ and by the induction hypothesis $S,y_1 \models \varphi$. In the case where $\lambda'[1] = y'_2$: $S',y'_2 \models \varphi$ and by the induction hypothesis $S,y_2 \models \varphi$. Then $\forall \sigma_{Ag} \in \Gamma_{Ag}$ if $\lambda = comp(\sigma_{Ag},x)$ we have $S,\lambda[1] \models \varphi$. Hence, $S,x \models \langle\langle\rangle\rangle_p^p \bigcirc \varphi$.

The cases for the remaining states all follow the same reasoning as from x. The $\langle\langle i\rangle\rangle_p^p \bigcirc \varphi$ case is very similar and the $\langle\langle Ag\rangle\rangle_p^p \bigcirc \varphi$ is identical to $\langle\langle i\rangle\rangle_p^p \bigcirc \varphi$ as we only have one agent.

6. The cases for the remaining temporal operators follow the same reasoning.

□

8.4 Summary

In this chapter we have illustrated an alternative approach for expressing properties of systems which refer to whether agents are forced to abide by the social laws or not. We are able to express properties in this approach which are similar to those that we are able to express in Social ATEL. We discovered a general reduction which holds between properties expressed in both approaches, which holds for all extremes of coalitions (empty, arbitrary, and grand) and all temporal operators, but this reduction changes when we alter the requirement of the coalition of agents to follow the social laws and the requirement of the other agents. As a result, we have four reductions between formulae in the two approaches, for each combination of G acting socially or physically while $\bar{G}$ acts socially or physically. We were unable to find reductions between properties expressed in Social ATEL and properties expressed in ATL using this good states approach. This is due to the fact that Social ATEL appears to be more expressive than ATL, and as a result, we had to use ATL*. After investigating this alternative approach, it can be seen that Social ATEL allows us to express these properties in a much more succinct and elegant manner, but also, we have proved that certain properties can be expressed in the good states approach that cannot be expressed using Social ATEL.

Part IV

Conclusions

Chapter 9

Conclusions

This chapter concludes the thesis by first giving a review of the work, then by giving an evaluation, and finally providing some avenues for possible future work.

9.1 Review

The first part of the thesis presented the necessary background information. Firstly, in Chapter 2, we gave a survey of some popular modal logics, which were either used in, or are somehow related to the rest of the thesis. The first logic we presented, Alternating-time Temporal Logic (ATL), is very important for the thesis, as it provides a basis for our formal framework of social laws. ATL is a temporal logic of cooperation, which allows us to express properties of multi-agent systems that agents and coalitions of agents can achieve over time. We then surveyed Epistemic Logic, the logic of knowledge, and explained the possible worlds semantics. Next, we presented Alternating-time Temporal Epistemic Logic (ATEL), an extended version of ATL enriched with epistemic operators. ATEL was also used in our framework of social laws in Chapter 6. Next, we presented BDI logic, used for expressing the beliefs, desires and intentions of agents, and finally presented deontic logic, the logic used for expressing normative behaviour.

In Chapter 3, we gave a survey of social laws for multi-agent systems. We began by introducing social laws in detail and defined a social law as a restriction on the behaviour of the agents to ensure they can work individually in a mutually compatible manner in order to fulfil their individual goals. We then explained the effects that social laws have on such systems and outlined the two approaches by which social laws can come to exist in a multi-agent system: they can emerge from within the system itself or they can be designed offline and hardwired into the system. We then surveyed some approaches to the emergence of social laws at run-time. As more research has been carried out into the offline design of social laws, the main body of this chapter focussed on such approaches. We introduced the Artificial Social Systems approach of Moses and Tennenholtz, in which they model an artificial social system as a set of dependent automata in order to address some of the computational problems, and go on to construct a rich modal language for expressing social laws. We also surveyed the Shoham and Tennenholtz framework of social laws, in which a computational model of social laws is constructed to address issues such as the "Useful Social Law problem". Finally, we looked at two different ways of choosing between social laws: minimality and simplicity.

The main contribution of the thesis was presented in Chapters 4 through 8. In Chapter 4, we introduced the semantic structures that our framework of social laws is based upon. These structures are known as Action-Based Alternating Transition Systems (AATS), and are very similar to the Alternating Transition Systems that underpin ATL. The main difference is that in AATSs we have an explicit set of actions for each agent, and action pre-condition function which determines for each action, where this action may be executed from. We outlined an example scenario, known as "The Train Scenario", which we refer to in the subsequent chapters. Finally, we defined ATL over AATSs and proved some properties of ATL that we used in subsequent proofs.

In Chapter 5, we presented our framework of social laws in ATL. We introduced a social law as consisting of two parts, an objective and a behavioural constraint. The former specifies what the social law is in place to achieve, while the latter corresponds to the requirements the law places on the agents. We outlined the three problems associated with social laws, namely, the effectiveness, feasibility and synthesis problems and showed how, in many cases, these can be framed directly as ATL model checking problems. The feasibility problem was proved to be NP-complete, which is no harder than the corresponding problem in the Shoham and Tennenholtz framework. Finally, we showed how we could extend our framework to allow us to express objectives of social laws that explicitly refer to the legality of actions.

In Chapter 6, we extended our framework by incorporating the notion of knowledge. We introduced various types of knowledge properties along with the general format taken by each of them. We looked at some epistemic social laws in the context of a case study and showed how an objective can be broken down into several individual objectives which have a certain format that we call feasibility. Finally, we showed how we can perform model checking of ATEL using only standard ATL model checkers. To do this, we gave an informal description of local propositions and gave a theorem allowing us to substitute these local propositions for epistemic properties.

In Chapter 7, we extended our framework further. We removed the assumption that the agents will follow the social laws just because they are in place. With the possibility of agents not following the social laws, we made a distinction between agents acting socially (where they abide by the social laws) and the agents acting physically (where they can act in any way possible). We constructed a language called Social ATEL, which is an extension of ATEL, to allow us to express how the agents in the system are acting. We investigated various properties in a case study and found in the absence of social laws, very little knowledge ensues.

Finally, in Chapter 8, we investigated an alternative approach for expressing properties of systems that refer to whether the agents are acting socially or physically. We tried to find equivalences between properties expressed in Social ATEL and properties expressed in ATL using an approach which labels states based on whether the agent reached the state through social action. As Social ATEL appears to be more expressive than ATL, we were only able to reduce Social ATEL down to ATL*.

9.2 Evaluation

The work in this thesis has contributed to the area of Social Laws in Multi-Agent Systems in numerous ways. We have shown how ATL provides a natural and somewhat elegant framework for expressing social laws. Unique to our framework is the way we *explicitly* define a social law as consisting of an objective and a behavioural constraint. No previous frameworks have this explicit logically expressed objective, the societal goal which the behavioural constraint is in place to achieve. In the Shoham and Tennenholtz framework, the notion of an objective is captured through the set of focal states, which given any two focal states, the social law should guarantee that there exists a legal plan, which starting in one of the states, reaches the other state, no matter what the other agents do. In our framework, expressing the objective logically in ATL, not only allows us to refer to the powers of agents and what they can achieve over time, but we can talk about social laws being effective if the objective is satisfied.

In our framework of social laws, we have identified three computational problems, namely, the effectiveness, feasibility and synthesis problems. Expressing social laws in ATL allows us to reduce these problems directly to ATL model checking problems, which can be checked using existing ATL model checkers, such as MOCHA. Despite the apparent expressive power of our framework, the feasibility problem is no more complex than the corresponding problem in the Shoham and Tennenholtz framework; it is NP-complete. Although NP-completeness is normally interpreted as a negative result, as we have a more expressive and arguably more elegant framework for expressing social laws, but have not added to the complexity, we consider this to be a positive result.

Not only can we express objectives of social laws in ATL, we have extended our framework to allow us to express objectives of social laws in ATEL and properties of systems in SATEL.

The former allows us to express objectives of social laws that refer to the knowledge of individual agents and of coalitions of agents, while the latter adds on to this the ability to express properties that refer to whether or not the cooperating coalition are following social strategies and whether or not the other agents in the system are following social strategies. In the Moses and Tennenholtz framework of Artificial Social Systems, introduced in Chapter 3, the logical language that they construct has the ability to refer to *similar* notions to Social ATEL. Firstly, they can refer to knowledge, in much the same way as in our framework. They also have a notion of social belief, defined in a different way. Secondly, operators such as *s-reachable*(T, φ), state that the agents in T have a joint plan consisting of only socially acceptable actions in order to attain φ, as long as all the other agents follow the rules of the social system. This could be deemed similar to the Social ATEL formula, $\langle\langle G \rangle\rangle_s^s \Diamond \varphi$. However, we can express much more in our framework. We can refer to the powers of agents and the type of strategies they are using and specify what they can achieve over time. Moreover, we can do this in a concise and elegant manner.

Overall, we have constructed a very expressive, natural and concise language for reasoning about social laws. Social laws consist of both an objective and a behavioural constraint, a very natural way of modeling social laws, and the three associated problems, in many cases, can be posed as ATL model checking problems. Taking all of this into consideration and the fact that we have not added to the complexity, clearly shows how this thesis has contributed to the area of Social Laws.

9.3 Future Work

There are several possible avenues for future work. One possible avenue would be to develop a tool for the automatic verification of Social ATEL. The major paradigm in verification of Multi-Agent Systems is model-checking, at this moment. Although there exists a model checker MOCHA for standard ATL, it can not directly deal with notions like knowledge, belief or discriminate between legal and illegal transitions. The paper [54] uses a standard model checker (SPIN) to verify knowledge properties in a linear time model, using the notion of *local propositions*. Such an approach seems also promising for epistemic extensions of ATL. However, as our framework at the same time assumes that certain transitions can be blocked in certain states, our challenge is greater than this. In the long run, what is needed is a tool that can take social laws as an additional parameter, and preferably hypothetically reason about the effects of them.

Another possible route could be to incorporate the notion of minimality into the framework. Minimality, as introduced in Chapter 3, is essentially a social law that constrains the agents just enough to make the social law effective, thus giving the agents maximal individual flexibility. This would involve asking the question: Is the system not over-constrained, i.e., do the agents still have reasonable choices, and could the desired behaviour have been achieved with fewer constraints?

Finally, an interesting avenue would be to extend the framework further by giving a priority ordering to social laws. When designing a multi-agent system, the designer is faced with the task of deciding what the design objectives of the system should be. This includes deciding on social law objectives that should hold in the system. There may be many objectives that would ideally be achieved, but in reality, some of these objectives may conflict with one another. Naturally, some objectives of social laws are more important than others. For example, making sure that two trains do not collide in a tunnel could be deemed more important than making sure none of the trains linger in the tunnel, thus making the system as efficient as possible. We would like to extend our framework to make it possible to reason about the importance of laws. We could classify laws as safety laws, which always guarantee to satisfy the safety properties of the system, and liveness laws which guarantee liveness properties. Safety laws would be in place to prevent drastic consequences from occurring and should *always* be achieved no matter what. Liveness laws themselves could be given a priority ordering, so one liveness law might be more important than another liveness law, and thus would take priority.

Part V

Bibliography and Index

Bibliography

[1] R. Alur, L. de Alfaro, T. A. Henzinger, S. C. Krishnan, F. Y. C. Mang, S. Qadeer, S. K. Rajamani, and S. Taşiran. MOCHA user manual. University of Berkeley Report, 2000.

[2] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49(5):672–713, September 2002.

[3] R. Alur, T. A. Henzinger, F. Y. C. Mang, S. Qadeer, S. K. Rajamani, and S. Taşiran. Mocha: Modularity in model checking. In *CAV 1998: Tenth International Conference on Computer-aided Verification, (LNCS Volume 1427)*, pages 521–525. Springer-Verlag: Berlin, Germany, 1998.

[4] A.R. Anderson. A reduction of deontic logic to alethic modal logic. *Mind*, 67(265):100–103, 1958.

[5] K. Binmore. *Essays on the Foundations of Game Theory*. Basil Blackwell, Cambridge, Massachusetts, 1990.

[6] L. Birnbaum. Rigor mortis. In D. Kirsh, editor, *Foundations of Artificial Intelligence*, pages 57–78. The MIT Press: Cambridge, MA, 1992.

[7] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge University Press: Cambridge, England, 2001.

[8] A. H. Bond and L. Gasser, editors. *Readings in Distributed Artificial Intelligence*. Morgan Kaufmann Publishers: San Mateo, CA, 1988.

[9] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press: Cambridge, MA, 2000.

[10] D. C. Dennett. *The Intentional Stance*. The MIT Press: Cambridge, MA, 1987.

[11] M. Deutch and H. B. Gerard. A study of normative and informational social influence upon judgment. *Journal of Abnormal and Social Psychology*, 51:629–636, 1955.

[12] G. van Drimmelen. Satisfiability in alternating-time temporal logic. In *Eighteenth Annual IEEE Symposium on Logic in Computer Science (LICS 2003)*, pages 208–217, Ottawa, Canada, 2003.

[13] E. H. Durfee. *Coordination of Distributed Problem Solvers*. Kluwer Academic Publishers: Dordrecht, The Netherlands, 1988.

[14] E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science Volume B: Formal Models and Semantics*, pages 996–1072. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 1990.

[15] K. Engelhardt, R. van der Meyden, and Y. Moses. Knowledge and the logic of local propositions. In *Proceedings of the 1998 Conference on Theoretical Aspects of Reasoning about Knowledge (TARK98)*, pages 29–41, Evanston, IL, July 1998.

[16] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning About Knowledge*. The MIT Press: Cambridge, MA, 1995.

[17] D. Fitoussi and M. Tennenholtz. Choosing social laws for multi-agent systems: Minimality and simplicity. *Artificial Intelligence*, 119(1-2):61–101, 2000.

[18] L. Gasser and M. Huhns, editors. *Distributed Artificial Intelligence Volume II*. Pitman/Morgan Kaufman, 1989.

[19] M. R. Genesereth, M. Ginsberg, and J. S. Rosenschein. Cooperation without communication. In *Proceedings of the Fifth National Conference on Artificial Intelligence (AAAI-86)*, pages 51–57, Philadelphia, PA, 1986.

[20] M. R. Genesereth and N. Nilsson. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann Publishers: San Mateo, CA, 1987.

[21] M. Ginsberg. *Essentials of Artificial Intelligence*. Morgan Kaufmann Publishers: San Mateo, CA, 1993.

[22] V. Goranko and W. Jamroga. Comparing semantics for logics of multi-agent systems. *Synthese*, 139(2):241–280, 2004.

[23] P. G. Hansen. Towards a theory of convention. *ΦNEWS*, 9:30–62, 2006.

[24] J. Hintikka. *Knowledge and Belief*. Cornell University Press: Ithaca, NY, 1962.

[25] J. Hintikka. Reasoning about knowledge in philosophy. In J. Y. Halpern, editor, *Proceedings of the 1986 Conference on Theoretical Aspects of Reasoning About Knowledge*, pages 63–80. Morgan Kaufmann Publishers: San Mateo, CA, 1986.

[26] N. R. Jennings. *Joint Intentions as a Model of Multi-Agent Cooperation.* PhD thesis, Department of Electronic Engineering, Queen Mary & Westfield College, 1992.

[27] J. E. Kittock. Emergent conventions and the structure of multi-agent systems. In *Proceedings of the 1993 Santa Fe Institute Complex Systems Summer School*, 1993.

[28] S. Kripke. Semantical analysis of modal logic. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 9:67–96, 1963.

[29] D. Lewis. *Convention — A Philosophical Study.* Harvard University Press: Cambridge, MA, 1969.

[30] A. Lomuscio and M. Sergot. Deontic interpreted systems. *Studia Logica*, 75(1):63–92, 2003.

[31] E. Mally. *Grundgesetze des Sollens, Elemente der Logik des Willens.* Leuschner and Lubensky, Graz, 1926.

[32] Z. Manna and A. Pnueli. *Temporal Verification of Reactive Systems — Safety.* Springer-Verlag: Berlin, Germany, 1995.

[33] J.-J. Ch. 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, 1988.

[34] J.-J. Ch. Meyer and R. J. Wieringa, editors. *Deontic Logic in Computer Science — Normative System Specification.* John Wiley & Sons, 1993.

[35] Y. Moses and M. Tennenholtz. Artificial social systems. *Computers and AI*, 14(6):533–562, 1995.

[36] G. M. P O'Hare and N. R. Jennings, editors. *Foundations of Distributed Artificial Intelligence.* John Wiley & Sons, 1996.

[37] L. Padgham and P. Lambrix. Agent capabilities: Extending BDI theory. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence (AAAI/IAAI-00)*, pages 68–73, Austin, TX, 2000.

[38] C. H. Papadimitriou. *Computational Complexity.* Addison-Wesley: Reading, MA, 1994.

[39] M. Pauly. *Logic for Social Software.* PhD thesis, University of Amsterdam, 2001. ILLC Dissertation Series 2001-10.

[40] M. Pauly. A modal logic for coalitional power in games. *Journal of Logic and Computation*, 12(1):149–166, 2002.

[41] A. Pretschner, M. Hilty, and D. Basin. Distributed usage control. *Commun. ACM*, 49(9):39–44, 2006.

[42] A. S. Rao and M. P. Georgeff. Modeling rational agents within a BDI-architecture. In R. Fikes and E. Sandewall, editors, *Proceedings of Knowledge Representation and Reasoning (KR&R-91)*, pages 473–484. Morgan Kaufmann Publishers: San Mateo, CA, April 1991.

[43] A. Ross. Imperatives and logic. *Theoria*, 7:53–71, 1941.

[44] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach.* Prentice-Hall, 1995.

[45] M. Sergot. Normative positions. In Henry Prakken and Paul McNamara, editors, *Norms, Logics and Information Systems. New Studies in Deontic Logic and Computer Science*, pages 289–310. IOS Press, Amsterdam, 1998.

[46] Y. Shoham and M. Tennenholtz. Emergent conventions in multi-agent systems. In C. Rich, W. Swartout, and B. Nebel, editors, *Proceedings of Knowledge Representation and Reasoning (KR&R-92)*, pages 225–231, 1992.

[47] Y. Shoham and M. Tennenholtz. On the synthesis of useful social laws for artificial agent societies. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, San Diego, CA, 1992.

[48] Y. Shoham and M. Tennenholtz. On social laws for artificial agent societies: Off-line design. *Artificial Intelligence*, 73(1-2):231–252, 1995.

[49] Y. Shoham and M. Tennenholtz. On social laws for artificial agent societies: Off-line design. In P. E. Agre and S. J. Rosenschein, editors, *Computational Theories of Interaction and Agency*, pages 597–618. The MIT Press: Cambridge, MA, 1996.

[50] Y. Shoham and M. Tennenholtz. On the emergence of social conventions: Modelling, analysis, and simulations. *Artificial Intelligence*, 94(1-2):139–166, July 1997.

[51] W. van der Hoek, M. Roberts, and M. Wooldridge. Knowledge and social laws. In *Proceedings of the Fourth International Conference on Autonomous Agents and Multiagent Systems (AAMAS-05)*, Utrecht, The Netherlands, July 2005.

[52] W. van der Hoek, M. Roberts, and M. Wooldridge. Social laws in alternating time: Effectiveness, feasibility, and synthesis. *Synthese*, 156(1):1–19, 2007.

[53] W. van der Hoek and R. Verbrugge. Epistemic logic: A survey. *In L. Petrosjan and V. Mazalov, editors, Game Theory and Applications*, 8:53–94, 2002.

[54] W. van der Hoek and M. Wooldridge. Model checking knowledge and time. In D. Bošnački and S. Leue, editors, *Model Checking Software, Proceedings of SPIN 2002 (LNCS Volume 2318)*, pages 95–111. Springer-Verlag: Berlin, Germany, 2002.

[55] W. van der Hoek and M. Wooldridge. Tractable multiagent planning for epistemic goals. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2002)*, pages 1167–1174, Bologna, Italy, 2002.

[56] W. van der Hoek and M. Wooldridge. Model checking cooperation, knowledge, and time — a case study. *Research in Economics*, 57(3), September 2003.

[57] W. van der Hoek and M. Wooldridge. Time, knowledge, and cooperation: Alternating-time temporal epistemic logic and its applications. *Studia Logica*, 75(1):125–157, 2003.

[58] W. van der Hoek and W. Wooldridge. Towards a logic of rational agency, 2003.

[59] J.A. van Eck. A system of temporally relative modal and deontic predicate logic and its philosophical applications. *Logique et Analyse*, 100:249–381, 1982.

[60] G.H. von Wright. Deontic logic. *Mind*, 60:1–15, 1951.

[61] G.H. von Wright. A new system of deontic logic. *Danish Yearbook of Philosophy*, 1:173–182, 1964.

[62] A. Walker and M. Wooldridge. Understanding the emergence of conventions in multi-agent systems. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pages 384–390, San Francisco, CA, June 1995.

[63] M. Wooldridge. *An Introduction to Multiagent Systems*. John Wiley & Sons, 2002.

[64] E. N. Zalta. Stanford encyclopedia of philosophy. See `http://plato.stanford.edu/`.

[65] G. Zlotkin and J. S. Rosenschein. Mechanism design for automated negotiation, and its application to task oriented domains. *Artificial Intelligence*, 86(2):195–244, 1996.

Index