Modal Logics of Partial Observability

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

Christos Moyzes

May 2019

To my dear parents, Aristeidis and Foteini.

To my favourite aunt, Matina.

In memory of my loving uncle, Georgios.

Contents

Illustrations

List of Figures

List of Tables

Abstract

This thesis examines the notion of "vision" and how it can be integrated into the established formalism of dynamic and/or epistemic propositional logics.

First a logic is developed for reasoning about semi-public environments, i.e. environments in which a process is executing, and where agents in the environment have partial and potentially different views of the process. Previous work on this problem illustrated that it was problematic to obtain both an adequate semantic model and a language for reasoning about semi-public environments. Here, the use of Program Models is introduced for representing the changes that occur during the execution of a program. These models serve both as syntactic objects and as semantic models, and are a modification of action models of Dynamic Epistemic Logic, in the sense that they allow for ontic change (i.e., change in the world or state). We show how Program Models can elegantly capture a notion of observation of the environment. The use of these models resolves several difficulties identified in earlier work, and admit a much simpler treatment than what was possible in previous work on semi-public environments.

It soon becomes apparent that while non-deterministic choice is handled intuitively in program models, it is nonetheless treated differently than in traditional formalisms. To shed light on these concepts we start by making a comparison between notions of non-deterministic choice, and between notions of sequential composition, of settings with dynamic and/or epistemic character; namely Propositional Dynamic Logic (PDL), Dynamic Epistemic Logic (DEL), and the aforementioned logic of Semi-Public Environments (SPE). These logics indeed represent two different approaches for defining the aforementioned actions, and in order to provide unified frameworks that encompass both, we define the logics DELVO (DEL+Vision+Ontic change) and PDLVE (PDL+Vision+Epistemic operators). DELVO is then given a sound and complete axiomatisation.

Furthermore, we fill an important gap in the literature by providing complexity results for the satisfiability and model checking problems of DEL, especially for the case of single-agent DEL. Considering DELVO to be a more contemporary version of DEL, we move on to expand these results for DELVO as well. For both logics and for any number of agents, the satisfiability is proven to be in NEXPTIME, while model checking is proven to be in PSPACE.

Acknowledgements

I have seen in other theses and I tend to agree, that people may think this section a bit generic or dull. I would add that perhaps this observation itself might be generic. And this could go on forever. Perhaps there is some comfort to be found in the relative uniqueness of observing the recursive quality of the above. Regardless, it is still the case that this section is especially important to the respective author.

First and foremost I would like to thank my two supervisors: my main supervisor Wiebe van der Hoek, and my second supervisor Davide Grossi. They were both very supportive throughout my doctoral studies, they were always more than happy to discuss any ideas I had, gave me thoughtful and honest opinions, encouraged me to see my projects through. They meticulously checked any written results I produced and gave advice on everything, from the directions the research could take and technical details, to aesthetic details and latex formatting. Wiebe was supportive from the first moment I applied to study for my PhD in Liverpool and has done his best to see me through my studies. Davide, despite being the second supervisor, never missed any of the supervisory meetings and his office door was always open for further discussions. I was especially lucky to have them as supervisors and I owe them a debt of gratitude. I do hope there are many more opportunities for us in the future to continue our collaboration.

I would also like to wholeheartedly thank Costas Koutras for his continuous support over many years. Costas supervised my Msc. thesis; our collaboration started then and still continues to this day. But Costas other than an excellent teacher and colleague is also the kind of person that one would be proud to call his friend, and I am truly honoured that I can do so.

I would like to address special thanks to Jane Gallagher who is, simply put, a remarkable woman with extraordinary characteristics. Her support was crucial during the final stages of my studies. For the same reason many thanks should go to Alison Goodyear.

Many thanks as well to my friends and fellow PhD students, both in Greece and in Liverpool, especially Ioannis, Matoula, and Thanos.

Last but not least, words cannot describe the gratitude I owe to my loving parents, for their infinite support and for enduring having their only-child live abroad.

Chapter 1

Introduction

1.1 Program Models and Semi-Public Environments

We are interested in settings where agents are involved in a (possibly non-deterministic) computational activity in which the *computation* itself is performed publicly, but the *data* is distributed privately. Specifically, agents are assumed to have only partial knowledge of the values of the variables used in the computation. In this setting, the basic question to which we address ourselves in this work is: *What can we say about what the agents know and learn as the computation proceeds?*

We begin with a simple example, to illustrate the general setting of semi-public environments:

> Suppose we have three variables, say x, y, and z, and we have an agent i who can see *only* the variable z. We aim to have a program that swaps the values of x and y, using only assignments of the form $a := b$, where a and b are program variables. Now, *can we do this in our setting without i learning anything about the values of the variables x and y*? Although i cannot directly see the value of x or y, the "standard" solution to this problem (using z as a temporary variable) will result in i knowing the value of one of the variables.

Van der Hoek *et al* [39] give more examples: among others, they show how the *dining cryptographers* problem can be modelled as a semi-public environment. Again, this is an example of a setting where there is a commonly agreed computation, but the data (the outcomes of the toss of a coin) is not.

Although there is much related work in the literature, to the best of our knowledge this issue was first explicitly considered in [39]. This work illustrates that it is problematic both to develop an appropriate language for reasoning about semi-public environments, and at the same time develop an appropriate semantic model for such a language. The approach presented in Chapter 2 resolves several difficulties and limitations identified by Van der Hoek *et al* [39].

Our work in particular, and semi-public environments in general, are in the intersection of at least three closely related research areas:

- First, they are closely related to *interpreted systems* [28], where what an agent knows derives from what agents can see about their environments (typically, their internal state).
- Second, the way in which we model programs in our work is close to that of Propositional Dynamic Logic [37], which uses basic programs as the building blocks, and combines them to create more complex ones. Note that PDL is a language for reasoning about *ontic change*, i.e., change in the real world or the real state.
- Third, our work is a close relative of Dynamic Epistemic Logic (DEL, [19]), which is intended for reasoning about *information change*, i.e., changes in the knowledge or beliefs of agents.

One approach to modelling informational change in DEL is with using *action models* – see [5, 6] and [19, Chapter 6]. Their domain is that of abstract actions together with an accessibility relation capturing which actions 'look the same' for which agents. The first papers on action models already suggested that it would be 'obvious' to include mechanisms to deal with ontic change in such models. Work in DEL that tried to study ontic change and informational change in one and the same framework were subsequently undertaken in [21, 22], in [8] and in [20], among others.

We utilize such models, where actions can have both an ontic (change the world) and an epistemic (change the information) effect to build *program models* where the notion of visibility of the environment plays a central role.

A key design principle in this here approach is that of simplicity: we make a number of assumptions that keep it technically lightweight. For example, programs handle propositional variables, and there are only finitely many such variables. In addition, we assume that which agent sees which variables is common knowledge. Finally, we allow public announcements only on objective formulas, rather than on epistemic/dynamic ones. Obviously, lifting (some of) these assumptions offers interesting questions for further research.

The thesis makes the following contributions: we give a realistic interpretation of the program construct $\cup$ in semi-public environments, different from that of [39]; we employ program models to give a clear interpretation to such programs; we show that the logic for semi-public environments is in fact equivalent to epistemic logic enriched with a notion of *vision*; our use of program models is the first to employ action models that give an account on such a notion; using program models, we obtain a relatively simple and natural axiomatisation for semi-public environments, without the need to use a universal modality, as in [39] and, finally, in contrast with the mentioned paper, we are able to shed light on sequential composition and iteration within the context of semi-public environments.

1.2 DELVO and PDLVE

The chapters focus on the dynamics of knowledge in environments where agents are aware of (possibly non-deterministic) actions that are executed, but data is distributed privately. By data being distributed privately we mean that the agents may or may not have constant, real-time knowledge, of the value of the variables involved. We set out to investigate how non-deterministic choice in this type of environments affects the knowledge of an agent after an action is executed.

Non-deterministic choice is a central action composition operator for most logics of action. For our approach the starting points will be Propositional Dynamic Logic (PDL, [37]), Dynamic Epistemic Logic with Action Models (DEL, [19], [5, 6]) and the aforementioned logic for Semi-Public Environments (SPE, [34]).

In DEL and PDL, for any actions a and b of the respective language, we have the validity

$$([a]\varphi \wedge [b]\varphi) \leftrightarrow [a \cup b]\varphi. \tag{1.1}$$

If φ expresses knowledge, e.g. is $K_i\psi$, the scheme expresses a strong relation between knowledge and non-determinism: an agent knows ψ after non-deterministically executing a or b, if and only if he knows ψ after executing just a and after executing just b. In DEL, an instance of (1.1) is

$$[x!](K_i x \vee K_i y) \wedge [y!](K_i x \vee K_i y) \leftrightarrow [x! \cup y!](K_i x \vee K_i y).$$

This property makes perfect sense; after being announced that x, or being announced that y, the agent at least knows one of the two to be true. Another interesting instance of (1.1), however, should arguably fail. That is what happens in SPE, where the following formula is *not* valid (let $Kw_i x$ be shorthand for $K_i x \vee K_i \neg x$, and $x{:=}\bot$ (respectively, $\top$) stands for the action of setting variable x to false (respectively, true):

$$[x{:=}\bot]Kw_i x \wedge [x{:=}\top]Kw_i x \leftrightarrow [x{:=}\top \cup x{:=}\bot]Kw_i x.$$

The failure of (1.1) in SPE also has an intuitive appeal: some part of a non-deterministic action is executed, but which part is actually performed is not necessarily revealed to the agent.

So, in contrast to what happens in DEL (or PDL extended with epistemic operators), it is *not* the case that if the agent knows ψ after executing a and after executing b, that he necessarily knows ψ after the execution of the non-deterministic choice between them.

A similar analysis applies to sequential composition. In the same fashion as before, we have the validity that essentially defines sequential composition in PDL:

$$[a; b]\varphi \leftrightarrow [a][b]\varphi. \tag{1.2}$$

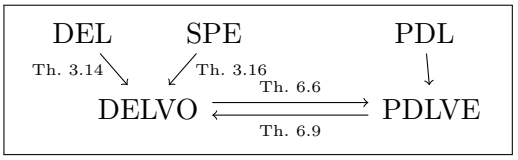
And in DEL, we have that formula $[x!; y!]K_i(x \wedge y) \leftrightarrow [x!][y!]K_i(x \wedge y)$ is a validity, for x, y being atoms. This is again intuitive: announcing $x \wedge y$ ought to be the same as

announcing x and then announcing y. But also this schema is not valid in SPE; we very briefly refer to an example from [34] below. In its perspective the failure of (1.2) should not come as a surprise.

Example 1.1. The agent can see only variable x, and we execute the action where we non-deterministically toggle the values of x and y, or just y. After this action, we also non-deterministically toggle the value of x, or of y.

In both steps the agent can distinguish between the two possible actions; one changes x, the other does not. Consider that the agent has the opportunity to observe the impact of the first action, before the second starts, and then the impact of the second. As a result, he can distinguish between any of the four total alternatives (where both x and y, or just x, or just y, or nothing, was changed; for later use let us denote the four individual actions with $\mathsf{v}_1, \mathsf{v}_2, \mathsf{v}_3, \mathsf{v}_4$ respectively). But what if the agent has no opportunity to observe what happened in-between? Then he can only observe what happened at the very end, and in our example, he will not be able to distinguish between all possibilities, particularly those with same truth value for x, but different for y.

The resulting contributions are that we introduce and study a logic that integrates both the above approaches to choice and composition, defining two novel action operators which we call *opaque* choice and composition. The proposed logic is a generalisation of both SPE and DEL and is called DELVO: DEL plus *vision* plus *ontic change*, which are features of SPE. Logic DELVO is then compared with an extension of PDL with vision and epistemic modalities we also introduce (called PDLVE). This logic can be considered the natural 'abstraction' of DELVO in PDL style, in the sense that the actions involved will be arbitrary connections between possible worlds (as opposed to the other dynamic logic, DL-PA [4], that uses assignments). The relationship between DELVO and PDLVE is studied in detail. The drawing below recapitulates the relations among the investigated logics.



Chapter 2

Program Models and Semi-Public Environments

The chapter is structured as follows:

- In Section 2.1 we provide the context, language, and models of our logic along with truth definitions for its formulas.
- Section 2.2 contains the definition for the class of program models, and the specific program models corresponding to the several meta-logical programs. We then show that these two classes of models coincide. Section 2.2 also contains a finite axiomatisation and a completeness result – we point out the validities that will serve as axioms along the way.
- Section 2.3 refers to notions of equality between our programs.
- Section 2.4 discusses related work and concludes.

2.1 Preliminaries

Throughout the chapter, we will be dealing with a set $Ag = \{1, \ldots, m\}$ of agents, and a set $\mathcal{V}ar = \{x_1, \ldots, x_n\}$ of propositional variables. The variables $\mathcal{V}ar$ will be those manipulated by the programs. For each agent $i \in Ag$, we let $V(i) \subseteq \mathcal{V}ar$ be the set of variables visible to i. Thus the set $V(i)$ represents i's view of the environment. The set of propositional formulas is denoted by $\mathcal{L}_0$. A valuation $\theta : \mathcal{V}ar \to \{true, false\}$ assigns a truth value to each variable $x_j \in \mathcal{V}ar$. The set of all valuations is denoted by Θ. For two valuations θ_1, θ_2, we write $\theta_1 \sim_i \theta_2$ if for all $x \in V(i)$, $\theta_1(x) = \theta_2(x)$, i.e., θ_1 and θ_2 coincide on the values of the variables observed by agent i. We extend this definition in a natural way; for $I \subseteq Ag$ we write $\theta_1 \sim_I \theta_2$ if for all $i \in I$, $\theta_1 \sim_i \theta_2$.

Definition 2.1 (Toggling values)**.** Let $\theta \in \Theta$ and $S \subseteq \mathcal{V}ar$. The toggling of values for the variables in S in θ, notation $\updownarrow(S)(\theta)$, is defined as follows:

$$\updownarrow(S)(\theta)(x) = \begin{cases} \textit{not } \theta(x) & \text{if } x \in S \\ \theta(x) & \text{otherwise} \end{cases}$$

Also, slightly abusing notation, for $\varphi_0 \in \mathcal{L}_0$ we define $\updownarrow(S)(\varphi_0)$, as:

$$\updownarrow(S)(\varphi_0) = \begin{cases} \varphi_0 & \text{if } S = \emptyset \\ \updownarrow(S \setminus \{x\})\varphi_0[\neg x/x] & \text{for any } x \in S \end{cases}$$

where $\varphi_0[\neg x/x]$ denotes the formula φ_0 with every occurrence of x replaced by $\neg x$.

Given two sets of variables S_1 and S_2, we denote $S_1 \triangle S_2$ to be their symmetric difference, i.e., $S_1 \triangle S_2 = (S_1 \cup S_2) \setminus (S_1 \cap S_2)$.

So $\updownarrow(V)(\theta)$ is like θ, but with the assignment of values to variables in V toggled. Obviously, $\updownarrow(\emptyset)$ is the identity function: $\updownarrow(\emptyset)(\theta) = \theta$. An examples of applying a toggle to a formula is obtained by $\updownarrow(\{x, y\})((x \vee \neg y) \leftrightarrow z)$ which yields $(\neg x \vee \neg\neg y) \leftrightarrow z$.

2.1.1 Language and Models

We now define our object language for reasoning about semi-public environments. In our setting, programs have both a syntactic and semantic flavour: our basic construct is a *pointed program model*, denoted (M, w) or M, w, where w is called a *program point*. The set of pointed program models will be formally defined in Section 2.2; we denote this set by PM. The fact that a program has several program points is to cater for the uncertainty by the agents in the environment as to what the exact changes are, enforced by the program. The language is an extension of the well-known multi-agent epistemic language S5$_n$ [28]. We have modal epistemic operators K_i, where $K_i\varphi$ expresses that agent i knows φ, an operator V_i, where $V_i x$ indicates that variable x is visible to agent i, and a dynamic "box" operator $[\mathsf{M}, \mathsf{w}]$, where $[\mathsf{M}, \mathsf{w}]\varphi$ means that the execution of M, w leads to states where φ holds. Formally, the syntax of formulas φ of the language $\mathcal{L}$ is defined by the following grammar:

$$\varphi \ ::= \ \top \mid x_j \mid V_i x_j \mid \neg\varphi \mid \varphi \wedge \varphi \mid [\mathsf{M}, \mathsf{w}]\varphi \mid K_i\varphi$$

where $i \in Ag$, $x_j \in \mathcal{V}ar$, and $\mathsf{M}, \mathsf{w} \in \mathsf{PM}$. We will use the standard classical logic abbreviations for $\bot, \vee, \rightarrow$ and $\leftrightarrow$, and we write $M_i\varphi \stackrel{\circ}{=} \neg K_i \neg\varphi$ and $\langle \mathsf{M}, \mathsf{w}\rangle\varphi \stackrel{\circ}{=} \neg[\mathsf{M}, \mathsf{w}]\neg\varphi$. We also introduce $[\mathsf{M}]\varphi$ for $\bigwedge_{\mathsf{w} \in \mathsf{M}} [\mathsf{M}, \mathsf{w}]\varphi$. A formula $\psi \in \mathcal{L}$ is called *program-free* if it has no occurrences of $[\mathsf{M}, \mathsf{w}]$. Recall that the propositional fragment of $\mathcal{L}$ is denoted by $\mathcal{L}_0$.

Definition 2.2 (Epistemic Models)**.** An *epistemic model* M for $\mathcal{L}$ is a tuple

$$M = \langle W, R, V, f \rangle, \text{ where}$$

1. W is a finite (possibly empty) set of states;
2. $f : W \to \Theta$ assigns a valuation θ to each state in W;
3. $V : Ag \to 2^{\mathcal{V}ar}$ keeps track of the variables that agent i 'can see';
4. $R : Ag \to 2^{(W \times W)}$ assigns an accessibility relation to each agent $i \in Ag$. We write uR_iv or $R_i(u, v)$ rather than $(u, v) \in R(i)$. Each $R(i)$ is an equivalence relation, and moreover, we assume that uR_iv implies $f(u) \sim_i f(v)$.

If $M = \langle W, R, V, f\rangle$, we write $w \in M$ for $w \in W$. For $u, v \in M$, we write $u \sim_i v$ if $f(u) \sim_i f(v)$. A pair (M, v) (with $v \in M$) is called a pointed model. Let $\mathcal{EM}_m$ denote the class of pointed epistemic models for m agents. Finally, the set of all vision functions V is denoted by $\mathcal{V}$is.

Note that R_i determines what agent i knows, while $\sim_i$ is the indistinguishability based on what i sees. We have $R_i \subseteq \sim_i$: i.e., an agent can know more than based on what he sees. Take for instance an agent i who sees x, and consider a program that first assigns x the value of y, and then assigns x the value of z. Although i does not see y, he does know its value after execution of this program, since this value was assigned to x and is not changed since (the formal definitions of these programs are given in Section 2.2).

We also ought to explain the definition of the set of states W. Given that the sets Ag and $\mathcal{V}ar$ are finite, and the set of possible values for the variables is finite ($true$ or $false$), we can assume that $\mathrm{S5}_n$ is characterised by the class of finite epistemic models (cf. [49], Section 2.1.5, finite model property). Furthermore, as we will see later, program models are also finite, and thus there is no way an infinite model will occur after the execution of a program. Regarding the possibility of an empty model: it has no impact on our results (in terms of validities, for instance) but it caters for the possibility of a program to 'fail', in which case there are no resulting states.

Example 2.1. We use the following example of an epistemic model M (see Figure 2.1). Consider $Ag = \{1, 2, 3, 4, 5, 6\}$ and $\mathcal{V}ar = \{x, y\}$. Moreover, $V(1) = \{x\}$, $V(2) = \{y\}$, $V(3) = V(4) = V(5) = \{\}$ and $V(6) = \{x, y\}$. In the figure, we have given each state a name and the valuation it represents. For instance, $f(z)$ is the valuation that assigns $true$ to x and $false$ to y. Reflexive or transitive arrows are not drawn, for instance every agent considers s possible given s, and we also have $R_3(w, u)$ and $R_3(v, z)$, for example. Agent 4 knows the value of x, and he knows the value of y if x is $true$, and agent 5 knows the value of y, and also that of x if y is $false$. Agent 6 sees, and hence knows, the value of both variables.

Definition 2.3 (Truth)**.** Let $M = \langle W, R, V, f\rangle$ with $W \neq \emptyset$ be an epistemic model, and $\mathsf{M}, \mathsf{w} \in \mathsf{PM}$ a pointed program model. The latter induces a relation between pointed epistemic models, where $(M, w)[\![\mathsf{M}, \mathsf{w}]\!](M', w')$ means that execution of the pointed program model M, w in (M, w) leads to (M', w'): the relation $(M, w)[\![\mathsf{M}, \mathsf{w}]\!](M', w')$ will be

defined in Section 2.2. We define the notion "formula φ of the language $\mathcal{L}$ is true at (M, w)", denoted $(M, w) \models \varphi$, recursively as follows:

$(M, w) \models \top$ is always the case;

$(M, w) \models x$ iff $f(w)(x) = true$;

$(M, w) \models V_i x$ iff $x \in V(i)$;

$(M, w) \models \neg\varphi$ iff not $(M, w) \models \varphi$;

$(M, w) \models \varphi \wedge \psi$ iff $(M, w) \models \varphi$ and $(M, w) \models \psi$;

$(M, w) \models K_i\varphi$ iff wR_iu implies $(M, u) \models \varphi$;

$(M, w) \models [\mathsf{M}, \mathsf{w}]\varphi$ iff $(M, w)[\![\mathsf{M}, \mathsf{w}]\!](M', w')$ implies $(M', w') \models \varphi$.

We say that a formula φ in $\mathcal{L}$ is *valid in* M if it is true at every (M, w) with $w \in M$, and φ is *valid* if it is valid in every model M. Of course these notions of validity are relative to the set PM that we will shortly specify.

Definition 2.4 (Model Equivalence)**.** Let $M = \langle W, R, V, f\rangle$ and $M' = \langle W', R', V', f'\rangle$ be two epistemic models, with $w \in M$ and $w' \in M'$. Then the pointed epistemic models (M, w) and (M', w') are *equivalent* (denoted $M, w \equiv M', w'$) if for all $\varphi \in \mathcal{L}$: $M, w \models \varphi$ iff $M', w' \models \varphi$. Moreover, M and M' are *equivalent* (denoted $M \equiv M'$) if for all $\varphi \in \mathcal{L}$: $M \models \varphi$ iff $M' \models \varphi$.

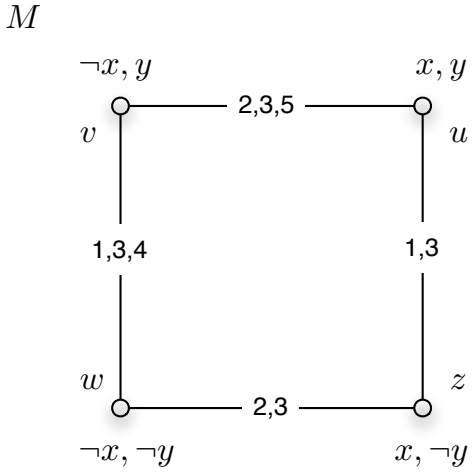


FIGURE 2.1: $M = \langle W, R, V, f\rangle$.

2.2 Program Models

In Dynamic Epistemic Logic, one way to represent epistemic actions is by using so-called *action models*: see, e.g., [6] and [19, Chapter 6]. Analogously (but nevertheless differently), we will define *program models* M, where the pair (M, w) represents a program point w in program model M. One basic difference is that we define program models as finite sets of pairs $\mathsf{w} = (\varphi_0, X)$, with φ_0 being a consistent propositional formula and $X \subseteq \mathcal{V}ar$ a set of variables. Similarly to actions in action models, the idea behind such a pair (φ_0, X) is that φ_0 represents the precondition for the program point to be executed, and X is the set of variables affected by the point, so they represent in some sense the postcondition of w. We identify points (φ_0, X) and (ψ_0, Y) iff φ_0 and ψ_0 are equivalent, and $X = Y$. More formally, consider the Lindenbaum-Tarski algebra of $\mathcal{L}_0$. Let $\mathcal{A}$ be its carrier set, i.e., the set of equivalence classes $[\varphi]$ based on tautological equivalence. When clear from context we will often write φ instead of $[\varphi]$. We define $\mathcal{A}^- = \mathcal{A} \setminus \{\bot\}$. In words: $\mathcal{A}^-$ represents the set of consistent propositional formulas that are based on $\mathcal{V}ar$, with the constraint that all members are mutually non-equivalent.

Definition 2.5 (Program Points and Models)**.** Let $\mathcal{A}^-$ and $\mathcal{V}ar$ be as before.

- An element $\mathsf{w} \in \mathcal{A}^- \times \mathcal{P}(\mathcal{V}ar)$ is called a *program point*.
  A program point $\mathsf{w} = (\varphi, X)$ is called a basic program, if $|X| \leq 1$. A basic program is atomic if either $\varphi = \top$ or $X = \{\}$.
  For any $\mathsf{w} = (\varphi, X)$ we define the projections $\mathsf{pre}(\mathsf{w}) = \varphi$, called the precondition for w, and $tgl(\mathsf{w}) = X$, where $\mathsf{tgl}(\mathsf{w})$ are the variables that are toggled by w.

- Any finite set of program points is called a *program model*.
  Additionally, given a vision function V, the relation $\approx_i^V \subseteq \mathsf{M} \times \mathsf{M}$ is defined as follows: for all $\mathsf{u}, \mathsf{w} \in \mathsf{M}$:

$$\mathsf{w} \approx_i^V \mathsf{u} \text{ iff } (\mathsf{tgl}(\mathsf{w}) \triangle \mathsf{tgl}(\mathsf{u})) \cap V(i) = \emptyset$$

  The set of all pointed program models M, w, given $\mathcal{L}_0$ and $\mathcal{V}ar$, is denoted by $\mathsf{PM}(\mathcal{L}_0, \mathcal{V}ar)$ or PM, if the parameters are clear from context.

So, given the visible variables $V(i)$ of agent i, they cannot distinguish two program points w and u if the changes they bring about are the same for the variables in $V(i)$. That is, agent i cannot distinguish w and u if for every variable $x \in V(i)$, either x gets changed by both w and u, or else both w and u leave x untouched.

Unlike actions models, program models do not have an indistinguishability relation as part of their definition. This is because our program models provide a so-called *concrete semantics*, in the sense that the accessibility relations $\approx_i^V$ are not abstract, given equivalence relations, but rather, they are exclusively derived from our notion of visibility of the variables – the function V – and the toggle sets tgl. In this sense, our program models are related to arbitrary action models (in the sense of [19]) in the same

way as interpreted systems ([28]) are related to general models for knowledge ([49]). Moreover, we want *at most one* program point of the same type in a program model. This is translated as considering propositionally equivalent preconditions to be 'the same' and names for the program points not being part of the definition (again in contrast with action models or even Kripke frames). The two differences above are essentially what allow us to represent program models as a subset of a Cartesian product. Finally, we insist on the preconditions being (propositionally) *consistent* formulas for technical reasons, in relation to action emulation (Definition 2.27). This does not affect our results on a conceptual level; if a program model had a program point with an inconsistent precondition, nothing would change if we removed it, as the specific program point could never be applied.

As a final comment on our program models, note that they can be conceived of as pure syntactic objects. This implies that, despite their name, our use of program models M, w in the syntax of our language $\mathcal{L}$ allows us to sidestep (philosophical) discussions relating to the use of semantic objects in the object language: a discussion which action models are frequently prone to (cf. [19, Section 6.1]).

Definition 2.6 (Model product)**.** Let $M = \langle W, R, V, f\rangle$ be an epistemic model, and M be a program model. $(M \times \mathsf{M})$ is the epistemic model $M' = \langle W', R', V', f'\rangle$ defined as follows:

- $W' = \{(w, \mathsf{w}) \mid w \in W, \mathsf{w} \in \mathsf{M} \;\&\; (M, w) \models \mathsf{pre}(\mathsf{w})\}$;
- $(w, \mathsf{w})R'_i(u, \mathsf{u})$ iff wR_iu and $\mathsf{w} \approx_i^V \mathsf{u}$;
- $V' = V$;
- $f'((w, \mathsf{w})) = {\updownarrow}\big(\mathsf{tgl}(\mathsf{w})\big)\big(f(w)\big)$.

This definition says that the points of the new epistemic model $M \times \mathsf{M}$ are pairs $(w, \mathsf{w}) \in W \times \mathsf{M}$, but only those pairs for which $\mathsf{pre}(\mathsf{w})$ holds in w. Two such points (w, w) and (u, u) look the same for agent i if the epistemic points w and u look the same for i, and the program points w and u look the same as well, *using the visibility function V from the epistemic model M*. The visibility set stays the same, and the valuation in (w, w) is the one from w, but with the variables in $\mathsf{tgl}(\mathsf{w})$ toggled.

Pending from Truth Definition 2.3 in Section 2.1 is the definition of the relation $[\![\mathsf{M}, \mathsf{w}]\!]$. The relation that (M, w) induces on epistemic pointed models says that $(M, w)[\![\mathsf{M}, \mathsf{w}]\!](M', w')$ iff

$$(M, w) \models \mathsf{pre}(\mathsf{w}) \text{ and } (M', w') = (M \times \mathsf{M}, (w, \mathsf{w}))$$

In words: executing the pointed program (M, w) in the pointed epistemic model (M, w) leads to a new pointed epistemic model (M', w') iff the program (M, w) is executable in (M, w) and the pointed model (M', w') can be obtained as a way of updating (M, w) with the program (M, w).

From the second bullet of Definition 2.6 it becomes apparent that the vision function V found in the epistemic model is 'borrowed' by the program model being applied. We will need this fact to be reflected in our axiomatisation, and so we will need to describe V with a formula (and not only in the meta-language). Such a characteristic formula can be achieved by a conjunction of literals of the special atoms $V_i x$. Specifically,

$$\chi_V \stackrel{\circ}{=} \bigwedge_{\substack{i\in\mathcal{AG},\\ x\in V(i)}} V_i x \;\wedge \bigwedge_{\substack{i\in\mathcal{AG},\\ x\notin V(i)}} \neg V_i x.$$

We list the axioms regarding programs in Table 2.5 under "Dynamic Component". Of those, ontic change indicates that to calculate the effect of M,w for a propositional formula φ_0, we apply the toggling of the variables collected in $\mathsf{tgl}(\mathsf{w})$ to φ_0. Soundness of all properties in the dynamic component in this table follow from the definition of $[\![\mathsf{M},\mathsf{w}]\!]$ and that of $M \times \mathsf{M}$.

One way to think of pointed program models in the language, is that they are in some sense 'place holders' that signal what kind of changes are happening. To appreciate this statement, it may help to formulate a main result of this paper at this stage, the proof of which is distributed over this section.

Theorem 2.7. *Every formula ψ is equivalent to a program-free formula ψ'.*

This theorem is proven by 'pushing in' occurrences of $[\mathsf{M},\mathsf{w}]$ in formulas of the form $[\mathsf{M},\mathsf{w}]\varphi$, and then showing that those occurrences can be abandoned when φ is either of the form φ_0 or $V_i x$. The axioms under 'Dynamic Component' of Table 2.5 indicate how we will achieve this. For instance, we have $[\mathsf{M},\mathsf{w}](\psi_1 \wedge \psi_2) \leftrightarrow ([\mathsf{M},\mathsf{w}]\psi_1 \wedge [\mathsf{M},\mathsf{w}]\psi_2)$. How knowledge operators are dealt with is through the so-called knowledge-reduction property 'program and knowledge' as given in Table 2.5, and which we display in (kn_red) below. This property is standard in DEL, but because of its importance and the fact that we have to modify it for our setting, we give a proof of its soundness in Lemma 2.8.

$$[\mathsf{M},\mathsf{w}]K_i\varphi \leftrightarrow \big(\mathsf{pre}(\mathsf{w}) \to \bigwedge_{V\in\mathcal{V}is} (\chi_V \to \bigwedge_{\mathsf{w}\approx_i^V \mathsf{u}} K_i[\mathsf{M},\mathsf{u}]\varphi)\big) \qquad \text{(kn\_red)}$$

Lemma 2.8. *Property* (kn_red) *is valid.*

Proof. Let M be an epistemic model with vision function V, $w \in M$ and let M be a program model, with $\mathsf{w} \in \mathsf{M}$. Let $M' = M \times \mathsf{M}$ as defined in Definition 2.6.

Suppose $M, w \models [\mathsf{M},\mathsf{w}]K_i\varphi \wedge \mathsf{pre}(\mathsf{w})$. For $V' \neq V$ the implications are trivially true, we need only concern ourselves with the single implication involving V. By $M, w \models [\mathsf{M},\mathsf{w}]K_i\varphi$ we have $M', (w,\mathsf{w}) \models K_i\varphi$. So for all (w',w') with $R_i'(w,\mathsf{w})(w',\mathsf{w}')$, we have $M', (w',\mathsf{w}') \models \varphi$. Now let $\mathsf{u} \in \mathsf{M}$ and $v \in M$ be such that $\mathsf{w} \approx_i^V \mathsf{u}$ and $R_i wv$. We need to show that $M, v \models [\mathsf{M},\mathsf{u}]\varphi$. If v does not make $\mathsf{pre}(\mathsf{u})$ true, we are done. So suppose $M, v \models \mathsf{pre}(\mathsf{u})$. Then $(v,\mathsf{u}) \in M'$. Since $R_i wv$ and $\mathsf{w} \approx_i^V \mathsf{v}$, we have $R_i'(w,\mathsf{w})(v,\mathsf{v})$ and hence $M', (v,\mathsf{v}) \models \varphi$, which achieves our aim.

For the converse, suppose $M, w \models \mathsf{pre}(\mathsf{w}) \to \bigwedge_{V \in \mathcal{V}is}(\chi_V \to \bigwedge_{\mathsf{w} \approx_i^V \mathsf{u}} K_i[\mathsf{M}, \mathsf{u}]\varphi)$. If $M, w \models \neg\mathsf{pre}(\mathsf{w})$, the goal $M, w \models [\mathsf{M}, \mathsf{w}]K_i\varphi$ trivially holds. So now suppose $M, w \models \mathsf{pre}(\mathsf{w})$. Given also that $M, w \models \chi_V$, we have $M, w \models \bigwedge_{\mathsf{w} \approx_i^V \mathsf{u}} K_i[\mathsf{M}, \mathsf{u}]\varphi$. We want to show that $M', (w, \mathsf{w}) \models K_i\varphi$, so assume $R_i'(w, \mathsf{w})(w', \mathsf{w}')$. By definition of R_i', this means wR_iw' and $\mathsf{w} \approx_i^V \mathsf{w}'$. The first of those statements implies $M, w' \models [\mathsf{M}, \mathsf{u}]\varphi$ for all u for which $\mathsf{w} \approx_i^V \mathsf{u}$. This in turn implies $M, w' \models [\mathsf{M}, \mathsf{w}']\varphi$, i.e., $M', (w', \mathsf{w}') \models \varphi$. □

Note how the dynamic operator gets 'pushed in' when reading the equivalence from left to right. It explains how agent i's knowledge that φ is obtained after the performance of a program M, w: if the program is executable, then, for every program M, u that looks the same for the agent, the agent knows that it will bring about φ.

Another thing to note is that, when 'filtering' for the correct vision function through the use of χ_V we could, instead of going through all vision functions, restrict ourselves only to functions that map agent i to a subset of the variables that appear in the toggle sets of M. For the sake of brevity we chose not to do so.

In epistemic logic with dynamic operators, the principle $[\alpha]K_i\varphi \to K_i[\alpha]\varphi$ is called *no learning*, or *no surprise* (or *no miracles* in [9]): everything an agent will know after the execution of an action α, was already known in advance to be a result of α.

Reversing the direction of *no learning* gives $K_i[\alpha]\varphi \to [\alpha]K_i\varphi$, a property known as *perfect recall*: the agent remembers the effect he anticipates of the action α. We will later in this paper discuss the extent to which our programs satisfy *no learning* and *perfect recall*.

In the next section we identify the place of several meta-logical "programs" in our setting, in fashion similar to dynamic logic.

Definition 2.9. The set of programs $\mathcal{P}rog$, is defined (inductively) as follows:

- If $\varphi_0 \in \mathcal{L}_0$ is a propositionally consistent formula, then $!\varphi_0 \in \mathcal{P}rog$, called the *public announcement* of φ_0;
- If $x \in \mathcal{V}ar$, then $\updownarrow x \in \mathcal{P}rog$, called the *toggle* of the value of x;
- If $x \in \mathcal{V}ar$, and $\varphi_0 \in \mathcal{L}_0$, then $x := \varphi_0 \in \mathcal{P}rog$, called the assignment of x to the truth value of φ_0;
- If $\pi_1, \pi_2 \in \mathcal{P}rog$, then $\pi_1 \cup \pi_2 \in \mathcal{P}rog$, called the *non-deterministic choice* between π_1 and π_2.
- If $\pi_1, \pi_2 \in \mathcal{P}rog$, then $\pi_1; \pi_2 \in \mathcal{P}rog$, called the *sequential composition* of π_1 and π_2.

In addition to the usual sequential composition (";") and non-deterministic choice ("∪"), we preoccupy ourselves with public announcement ("!"), as well as an operator $\updownarrow x$ which toggles the value of x (from $false$ to $true$ or inversely). Public announcement and Toggle are considered the basic constructs for this set of programs as we will shortly

see; assignment operations $x := \varphi_0$ can be defined in terms of the other constructs. It is also the case that $\updownarrow x$ could be defined if we instead adopted assignment as our basic construct, so this exchange does not have a serious impact.

For each of the programs in the set $\mathcal{P}rog$ we identify a corresponding program model. Strictly speaking, if $\mathcal{M}$ is the set of all program models, consider a function $I : \mathcal{P}rog \mapsto \mathcal{M}$. $I(\pi) = \mathsf{M}_\pi$ will be the program model for program π. The exact definition for this function is spread over the next subsections. We also introduce the notation $[\pi]\varphi$ for $[\mathsf{M}_\pi]\varphi$. As a reminder from Section 2.1.1, let us in turn point out that if M is a program model, $[\mathsf{M}]\varphi$ is shorthand for $\bigwedge_{\mathsf{w}\in\mathsf{M}}[\mathsf{M},\mathsf{w}]\varphi$.

2.2.1 Atomic Programs

Recall that atomic programs M,w come in two flavours: we start with those for which $\mathsf{tgl}(\mathsf{w}) = \emptyset$.

Definition 2.10 (Public Announcement)**.** The program model for public announcement $!\varphi_0$ is the model $\mathsf{M}_{!\varphi_0} = \{\mathsf{e}\} = \{(\varphi_0, \emptyset)\}$.

By Definition 2.5, on $\mathsf{M}_{!\varphi_0}$ we have $\approx_i^V = \{(\mathsf{e},\mathsf{e})\}$ for all $i \in Ag$ and $V \in \mathcal{V}is$.

Example 2.2 (Public Announcement)**.** See Figure 2.2 (middle) for the program model for the announcement $!x$. On the right of the figure, we have $M \times \mathsf{M}_{!x}$. This model is as expected: everybody knows that x is *true*, but agents 1 and 3 still do not know the value of y. Also agents 4 and 5 (who do not see any variable) 'remember' the value of both x and y, and agent 6 (who sees all variables) can of course distinguish the two results.

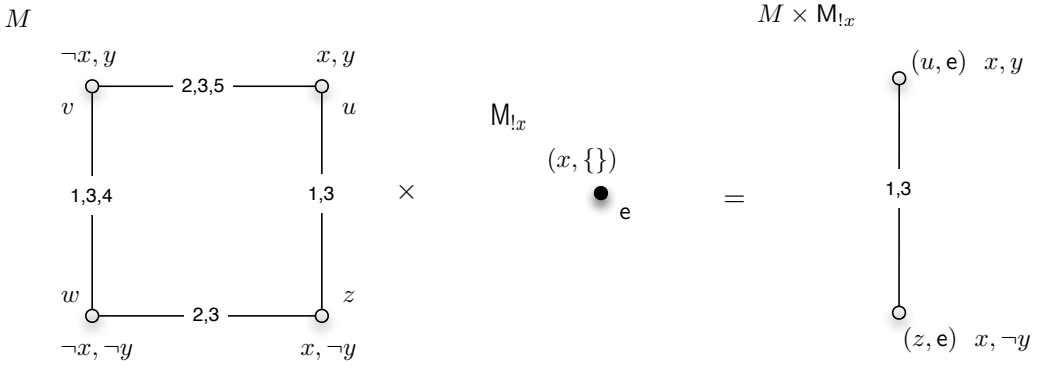


FIGURE 2.2: The product $M \times \mathsf{M}_{!x}$.

The knowledge reduction property for announcements is relatively simple:

$$[\mathsf{M}_{!\varphi_0},\mathsf{e}]K_i\varphi \leftrightarrow (\varphi_0 \to K_i[\mathsf{M}_{!\varphi_0},\mathsf{e}]\varphi) \qquad \text{(kn\_red(announce))}$$

Valid properties for announcement ($\varphi_0, \psi_0 \in \mathcal{L}_0,\ \varphi \in \mathcal{L}$)	
$[\mathsf{M}_{!\psi_0}, \mathsf{e}]\varphi \leftrightarrow [!\psi_0]\varphi$	one program point
$[!\psi_0]\varphi_0 \leftrightarrow (\psi_0 \to \varphi_0)$	truthfulness
$[!\psi_0]K_i\varphi \leftrightarrow (\psi_0 \to K_i[!\psi_0]\varphi)$	kn_red(announce)

TABLE 2.1: Properties of public announcement.

This property can be derived from (kn_red) by observing that $\mathsf{M}_{!\varphi_0}$ has only one action point e with $\mathsf{e} \approx_i^V \mathsf{e}$ and, moreover, that $\mathsf{pre}(\mathsf{e}) = \varphi_0$. Some derived properties of announcements are given in Table 2.1.

We have already seen that *no surprise* is not valid for arbitrary programs. However, for pointed program models with a single program point, we immediately derive from (kn_red) that we have something that is very close, i.e., $[\mathsf{M}, \mathsf{w}]K_i\varphi \to (\mathsf{pre}(\mathsf{w}) \to K_i[\mathsf{M}, \mathsf{w}]\varphi)$. This follows since $\mathsf{w} \approx_i^V \mathsf{w}$, for all i,V and w. Conditioning on $\mathsf{pre}(\mathsf{w})$ is necessary, as we can now show using announcements: suppose x is $false$ but i does not know this. Then $[\mathsf{M}_{!x}, \mathsf{e}]K_i\bot$ (since the announcement 'fails'), but obviously we do not have $K_i[\mathsf{M}_{!x}, \mathsf{e}]\bot$.

The reader can verify that *perfect recall* does hold for announcements: from $K_i[!\psi_0]\varphi$ infer $\psi_0 \to K_i[!\psi_0]\varphi$ from which, using kr_red(announce), one concludes $[!\psi_0]K_i\varphi$. More generally, if the program model M has only one point w, then *perfect recall* holds for M, w.

We continue by defining a program model for the other atomic action.

Definition 2.11 (Simple Toggle)**.** The program model for the program $\updownarrow x$ that toggles the value of the variable x, is the model $\mathsf{M}_{\updownarrow x} = \{\mathsf{a}\} = \{(\top, \{x\})\}$.

By Definition 2.5, we have $\approx_i^V = \{(\mathsf{a}, \mathsf{a})\}$ for all $i \in Ag$ and $V \in \mathcal{V}is$.

Example 2.3 (Simple Toggle)**.** See Figure 2.3 (middle) for the program model for $\updownarrow x$. On the right of the figure, we have $M \times \mathsf{M}_{\updownarrow x}$. The properties are straightforward: we have for instance $(M, w) \models [\updownarrow x](K_1x \wedge K_2\neg y \wedge \neg(K_3x \vee K_3y) \wedge K_4x \wedge K_5\neg y \wedge K_6(x \wedge \neg y))$.

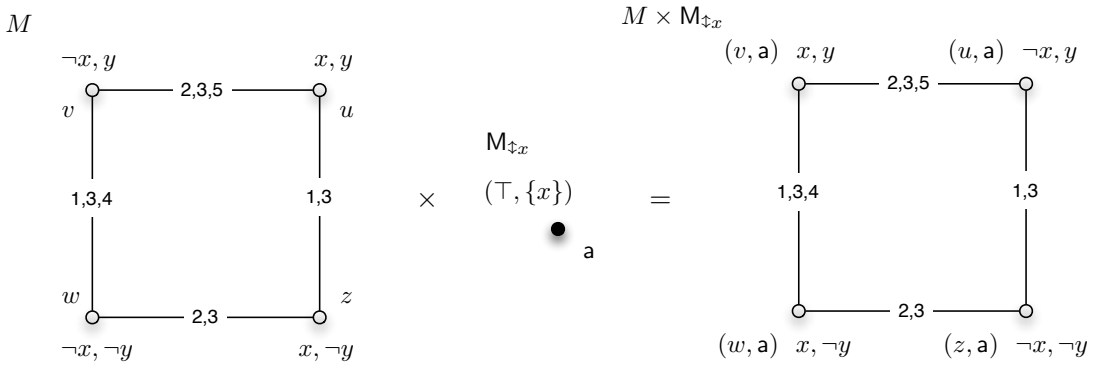


FIGURE 2.3: The product $M \times \mathsf{M}_{\updownarrow x}$.

The knowledge reduction property for simple toggle, (kn_red(toggle), see Table 2.2) is even simpler than that for public announcements. Note that the property does not depend on who sees the toggled variable: both *perfect recall* and *no surprise* hold for simple toggle. An instance of this property (combined with the 'toggle and objective') is $[\updownarrow \mathsf{x}]K_i x \leftrightarrow K_i \neg x$: agent i knows after toggling of x that x is *true*, if and only if i knows now that x is $false$.

Valid properties for toggle ($\varphi_0 \in \mathcal{L}_0,\ \varphi \in \mathcal{L}$)	
$[\mathsf{M}_{\updownarrow \mathsf{x}}, \mathsf{a}]\varphi \leftrightarrow [\updownarrow \mathsf{x}]\varphi$	one program point
$[\updownarrow \mathsf{x}]\varphi_0 \leftrightarrow \updownarrow(\{x\})(\varphi_0))$	toggle and objective
$[\updownarrow \mathsf{x}]K_i\varphi \leftrightarrow K_i[\updownarrow x]\varphi)$	kn_red(toggle)

TABLE 2.2: Properties of simple toggle.

In some sense we are now done: we could proceed by defining program models for non-deterministic choice ($\cup$) and for sequential composition ($;$) and, once those definitions are in place, we could use the equality

$$x := \varphi \stackrel{\circ}{=} \big(!(x \leftrightarrow \varphi) \cup (!(x \leftrightarrow \neg\varphi); \updownarrow x)\big) \tag{2.1}$$

to define assignments. However, since the definition of program models for choice and sequential composition are a little involved, we first define a program model for assignment, and will then later state that this definition indeed satisfies (2.1).

2.2.2 Assignment

We now define the program model for the assignment $x := \varphi$. In fact, as we will later see, $x := \neg x$ and $\updownarrow x$ 'are the same'.

Suppose we chose to model the assignment $x := \varphi$ by a single pointed model (M, a). Let (M, w) and (M, u) be two states in an epistemic model. Now suppose that the truth value of φ is different in (M, w) from the value in (M, u). An agent who knows that value will not confuse (M, u) with (M, w), and hence he will also distinguish (w, a) from (u, a). In particular, an agent who knows the truth value of φ before the assignment $x := \varphi$ will know the value of x after it. But now suppose we have an agent i who can see x, but who does not know the value of φ. In particular, suppose for this agent, the states (M, u) and (M, w) look the same. For this agent, the states (w, a) and (u, a) should look different (since the agent sees x, which has a different value in each of them), but, by the definition of product and the fact that wR_iu, we get $(w, \mathsf{a}) \approx_i^V (u, \mathsf{a})$, i.e., they look the same. Therefore, we will model the assignment with a program model with two states: one program point in which the value of x stays the same, and one point in which it gets toggled.

Definition 2.12 (Assignment)**.** Let $x \in \mathcal{V}ar$ and $\varphi \in \mathcal{L}_0$. The program model for the program $x := \varphi$ is defined (see also Figure 2.4, left) by stipulating $\mathsf{M}_{x:=\varphi} = \{\mathsf{s} = (x \leftrightarrow \varphi, \{\}), \mathsf{t} = (x \leftrightarrow \neg\varphi, \{x\})\}$.

It is easy to see that $\mathsf{s} \approx_i^V \mathsf{t}$ iff $x \notin V(i)$. In sum, s denotes that the value of x will stay the same, while t denotes that the value of x as a result of the assignment, will toggle. Note that the program model for assignment consists of a basic program (toggle) and an atomic one (announcement).

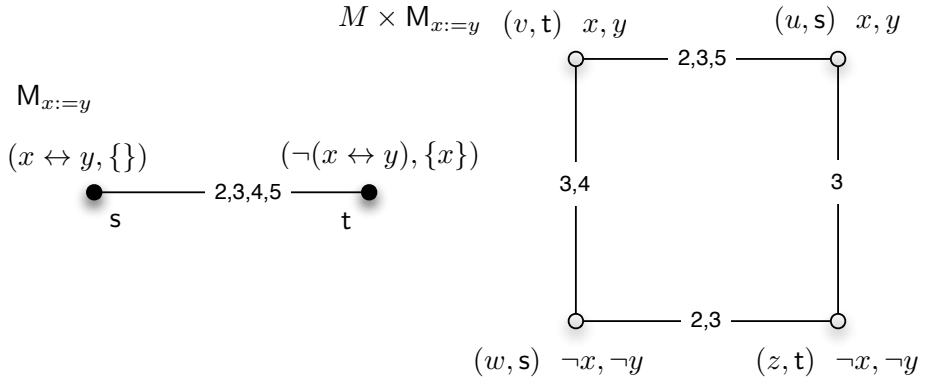


FIGURE 2.4: The assignment $\mathsf{M}_{x:=y}$ (left) and the model $M \times \mathsf{M}_{x:=y}$.

Example 2.4 (Assignment)**.** Let us consider the assignment $x := y$. Its program model is given in Figure 2.4 (left). Also, the product $M \times \mathsf{M}_{x:=y}$ of our example epistemic model M with this program model $\mathsf{M}_{x:=y}$ is given at the right of that figure. Note that, as expected, agent 1 (who sees x) comes to learn the value of y. Symmetrically, agent 2 (who sees y) comes to know the value of x.

Another perspective from which we can look at the process of defining assignment $x := \varphi$ in our logic, is that we eliminate any appearance of φ in the 'postconditions' (in our case, the toggle sets) of the program points, and we 'push' any required changes to the preconditions. In our case this is of course inevitable; we don't employ postconditions and assignment as the mechanism of program models and we could not have defined assignment per se. Regardless, this push to the preconditions is something that also happens in [20] with which we can make a meaningful comparison. In [20, Def. 3.5, Prop. 3.6] a way is given to transform any action model with assignments of the form $x := \varphi$ into an action model with assignments only of the form $x := \top$ and $x := \bot$. Specifically the base case – a model with a single point with postcondition the assignment $x := \varphi$ – is modelled as an action model with two points e and f. Point e has $\mathsf{pre}(\mathsf{e}) = \varphi$, $\mathsf{post}(\mathsf{e}) = \{x := \top\}$, while point f has $\mathsf{pre}(\mathsf{f}) = \neg\varphi$, $\mathsf{post}(\mathsf{f}) = \{x := \bot\}$. This re-affirms the correctness of our choice for the program model of assignment.

We display some derived properties of assignment in Table 2.3. Note that an assignment is basically a choice between two options: either x stays the same or it changes (third line of the table). For agents that see the variable, those options are distinguishable (line 4 and 5), for agents that don't see x, they are not (line 6).

We noted earlier that *perfect recall* holds if the program model only holds one point. We can now use assignments (with models having two points) as an example of failure

Valid properties for assignment ($\varphi_0, \psi_0 \in \mathcal{L}_0,\ \varphi \in \mathcal{L}$)	
$[\mathsf{M}_{\mathsf{x}:=\psi_0}, \mathsf{s}]\varphi_0 \leftrightarrow ((x \leftrightarrow \psi_0) \rightarrow \varphi_0))$	no change on s
$[\mathsf{M}_{\mathsf{x}:=\psi_0}, \mathsf{t}]\varphi_0 \leftrightarrow ((x \leftrightarrow \neg\psi_0) \rightarrow \updownarrow(\{x\})(\varphi_0))$	toggle on t
$[x := \psi_0]\varphi \leftrightarrow (x \leftrightarrow \psi_0 \wedge [\mathsf{M}_{x:=\psi_0}, \mathsf{s}]\varphi) \vee (x \leftrightarrow \neg\psi_0 \wedge [\mathsf{M}_{x:=\psi_0}, \mathsf{t}]\varphi)$	two choices
$V_i x \rightarrow ([x := \psi_0]K_i\varphi \leftrightarrow ((x \leftrightarrow \psi_0) \wedge K_i[\mathsf{M}_{x:=\psi_0}, \mathsf{s}]\varphi) \vee ((x \leftrightarrow \neg\psi_0) \wedge K_i[\mathsf{M}_{x:=\psi_0}, \mathsf{t}]\varphi))$	knowl. & ass. (1)
$\neg V_i x \rightarrow ([x := \psi_0]K_i\varphi \leftrightarrow (K_i[\mathsf{M}_{x:=\psi_0}, \mathsf{t}]\varphi \wedge K_i[\mathsf{M}_{x:=\psi_0}, \mathsf{s}]\varphi))$	knowl. & ass. (2)

TABLE 2.3: Properties of assignment.

of *perfect recall*. Note that in Example 2.4, we have $M, w \models K_4[\mathsf{M}_{x:=y}, \mathsf{s}]\neg x$ (agent 4 knows in w that x is $false$, so he knows that if a program point is executed that keeps the value of x untouched, then afterwards x will still be $false$). However, we do *not* have $M, w \models [\mathsf{M}_{x:=y}, \mathsf{s}]K_4\neg x$ (if an assignment $x := y$ is executed, and it happens to be in a state where both x and y are false, and hence leaves the value of x unchanged, then 4 does not need to know this: for him, y might have the value $true$, in which case the value of x was to be toggled).

Assignments also do not in general satisfy *no learning*. For an agent i who sees x for instance, when $\theta(y) = false$, we have $[\mathsf{x} := \mathsf{y}]K_i\neg y$, but typically not $K_i[\mathsf{x} := \mathsf{y}]\neg y$. Indeed, as an effect of the assignment, the agent has learned the value of y.

2.2.3 Non-deterministic choice

The following validity holds both for PDL (ontic change) and DEL (information change):

$$\models ([\pi_1]\varphi_1 \wedge [\pi_2]\varphi_2) \rightarrow [\pi_1 \cup \pi_2](\varphi_1 \vee \varphi_2)$$

However, this validity is not appropriate for our framework. Although both $[!x]K_i x$ and $[!\neg x]K_i\neg x$ are validities (regardless of whether $x \in V(i)$), we would not expect $[!x \cup !\neg x](K_i x \vee K_i\neg x)$ to be valid, since the agent simply does not need to know which of the two announcements was in fact executed. So, writing $Kw_i x$ for $K_i x \vee K_i\neg x$ (agent i knows *what* the value of x is, or, equivalently, *whether* x) we have

$$\not\models ([!x]Kw_i x \wedge [!\neg x]Kw_i x) \rightarrow [!x \cup !\neg x]Kw_i x$$

Definition 2.13 (Choice)**.** Let $\mathsf{M}_1, \mathsf{M}_2 \in \mathbb{PM}$. The non-deterministic choice between them is defined as the program model $\mathsf{M}_1 \cup \mathsf{M}_2$. For programs π_1 and π_2, we define $\mathsf{M}_{\pi_1 \cup \pi_2} = \mathsf{M}_{\pi_1} \cup \mathsf{M}_{\pi_2}$.

Example 2.5 (Choice)**.** Consider first the program $\pi \stackrel{\circ}{=} !x \cup !\neg x$ and the epistemic model M of Example 2.1. The program model M_π has two actions, e_0 (with precondition $\neg x$) and e_1 (with precondition x). No agent can distinguish e_0 from e_1 (since no variable is affected). It is easy to check that $M \times \mathsf{M}_\pi$ is equivalent to M: indeed, no agent learns anything from the program π.

Valid properties for choice ($\varphi_0 \in \mathcal{L}_0,\ \varphi \in \mathcal{L}$)	
$[\pi_1 \cup \pi_2]\varphi_0 \leftrightarrow ([\pi_1]\varphi_0 \wedge [\pi_2]\varphi_0)$	choice and objective
$[\mathsf{M}_1 \cup \mathsf{M}_2, \mathsf{w}_1]\varphi_0 \leftrightarrow [\mathsf{M}_1, \mathsf{w}_1]\varphi_0$	if $\mathsf{w}_1 \in \mathsf{M}_1$
$[\mathsf{M}_1 \cup \mathsf{M}_2, \mathsf{w}_2]\varphi_0 \leftrightarrow [\mathsf{M}_2, \mathsf{w}_2]\varphi_0$	if $\mathsf{w}_2 \in \mathsf{M}_2$
$[\mathsf{M}_1 \cup \mathsf{M}_2, \mathsf{w}]K_i\varphi \leftrightarrow$ $(\mathsf{pre}(\mathsf{w}) \to \bigwedge_{V \in \mathcal{V}is}(\chi_V \to \bigwedge_{\mathsf{w} \approx_i^V \mathsf{u}} K_i[\mathsf{M}_1 \cup \mathsf{M}_2, \mathsf{u}]\varphi))$	kn_red(choice)

TABLE 2.4: Properties of choice.

Secondly, consider the choice between two programs we have considered before: $!x$ and $x := y$ (see the model on the left in Figure 2.5). Note that we have $M, w \models [!x \cup x := y]K_1(\neg x \wedge \neg y)$. This is as desired: in w, agent 1 knows $\neg x$, so he knows that the announcement $!x$ will not succeed, so the program $x := y$ will be executed. Since in the resulting state, 1 can see that x is $false$, he knows that y is $false$. We also have $M, z \models [!x \cup x := y]K_1(\neg x \wedge \neg y)$. This is for similar reasons as the previous case. So we have $M \models (\neg x \wedge \neg y) \to [!x \cup x := y]K_1(\neg x \wedge \neg y)$. For agent 4, note that in M we had $\neg x \leftrightarrow K_4 \neg x$. This is no longer true after execution of the program: 4 cannot distinguish (w, s) (which is the result of executing $x := y$ when both x and y are false), from (v, t) (the result of executing $x := y$ when x is $false$ and y is $true$). Note that even agent 6, who sees all variables, is not able to distinguish (u, s) from (u, e): in u, when $x \leftrightarrow y \leftrightarrow \top$, the assignment $x := y$ and the announcement that x are indistinguishable.

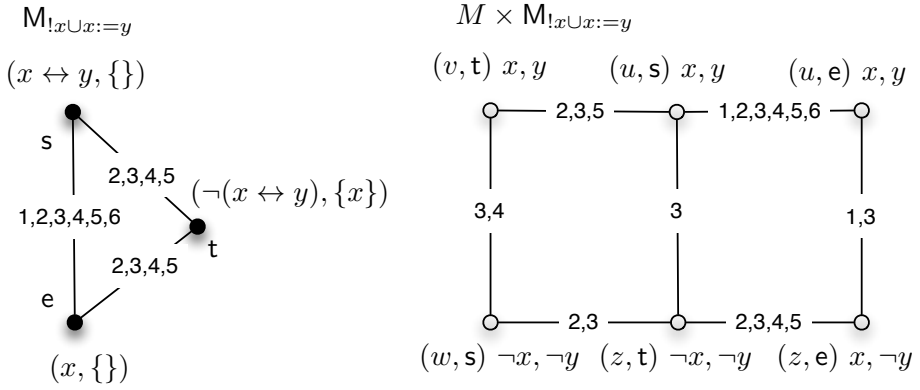


FIGURE 2.5: Program model M_π with $\pi \stackrel{\circ}{=} !x \cup x := y$ (left) and $M \times \mathsf{M}_\pi$.

When φ is not propositional there is not necessarily a decomposition of a formula of type $[\mathsf{M}_1 \cup \mathsf{M}_2, \mathsf{w}]\varphi$ in terms of $[\mathsf{M}_1]\varphi$ and $[\mathsf{M}_2]\varphi$. However, the knowledge reduction principle still applies (see Table 2.4). Note that the actions u for which $\mathsf{w} \approx_i^V \mathsf{u}$, are taken from the full model $\mathsf{M}_1 \cup \mathsf{M}_2$, not only from the model where w is from.

2.2.4 Sequential composition

We now define program composition of two program models $\mathsf{M}_1, \mathsf{M}_2$.

Definition 2.14 (Product)**.** Let two program models M_1 and M_2 be given. The product of $\mathsf{M}_1, \mathsf{M}_2$ is the program model

$$\begin{aligned}\mathsf{M}_1 \otimes \mathsf{M}_2 = \{\mathsf{w} \in \mathcal{A}^- \times \mathcal{P}(\mathcal{V}ar) \mid \exists \mathsf{w}_1 \in \mathsf{M}_1, \mathsf{w}_2 \in \mathsf{M}_2 \text{ such that} \\ \mathsf{pre}(\mathsf{w}) = \mathsf{pre}(\mathsf{w}_1) \wedge \updownarrow(\mathsf{tgl}(\mathsf{w}_1))(\mathsf{pre}(\mathsf{w}_2)) \;\&\; \mathsf{tgl}(\mathsf{w}) = \mathsf{tgl}(\mathsf{w}_1) \triangle \mathsf{tgl}(\mathsf{w}_2)\}\end{aligned} \tag{2.2}$$

By denoting a program point by $(\mathsf{w}_1, \mathsf{w}_2)$ we imply that it exists as a result of the above definition, with $\mathsf{w}_1, \mathsf{w}_2$ as the witnesses. For programs π_1 and π_2, we define $\mathsf{M}_{\pi_1;\pi_2} = \mathsf{M}_{\pi_1} \otimes \mathsf{M}_{\pi_2}$

This definition is best explained using an example. For this, we use a slightly more involved starting model than before, see Example 2.6. Informally, the new precondition in $(\mathsf{w}_1, \mathsf{w}_2)$ guarantees that, for a program $\mathsf{M}_1 \otimes \mathsf{M}_2$, $(\mathsf{w}_1, \mathsf{w}_2)$ to be executable in a given state, the point w_1 needs to be executable now, and w_2 needs to be executable after w_1 is finished. For the program $x := y; z := x$ for instance (so w_1 implements $x := y$ and w_2 implements $z := x$), suppose that we are in a state w where $(x \leftrightarrow \neg y) \wedge (x \leftrightarrow z)$ holds. In order for w_1 to be executable in w, we have $\mathsf{pre}(\mathsf{w}_1) = (x \leftrightarrow \neg y)$, and $\mathsf{tgl}(\mathsf{w}_1) = \{x\}$, and, for w_2 to be executable in w we require $\mathsf{pre}(\mathsf{w}_2) = (x \leftrightarrow z)$ and $\mathsf{tgl}(\mathsf{w}_2) = \{\}$. However, in order for the sequential composition $(\mathsf{w}_1, \mathsf{w}_2)$ to be executable in w, we have to guarantee that $x \leftrightarrow \neg y$ holds, together with $\neg x \leftrightarrow z$: since w_2 expects x and z to have the same truth value, and w_1 will toggle the value of x, before the composition is executed, x and z need to have a different value. In this example, $\mathsf{tgl}(\mathsf{w}_1, \mathsf{w}_2) = \{x, z\}$, which happens to coincide with the union of $\mathsf{tgl}(\mathsf{w}_1)$ and $\mathsf{tgl}(\mathsf{w}_2)$. However, when executing for instance $x := y; x := \neg y$ using two program points, say, w_1 and w_3, the set of variables to be toggled would be empty, which is indeed $\mathsf{tgl}(\mathsf{w}_1) \triangle \mathsf{tgl}(\mathsf{w}_3)$.

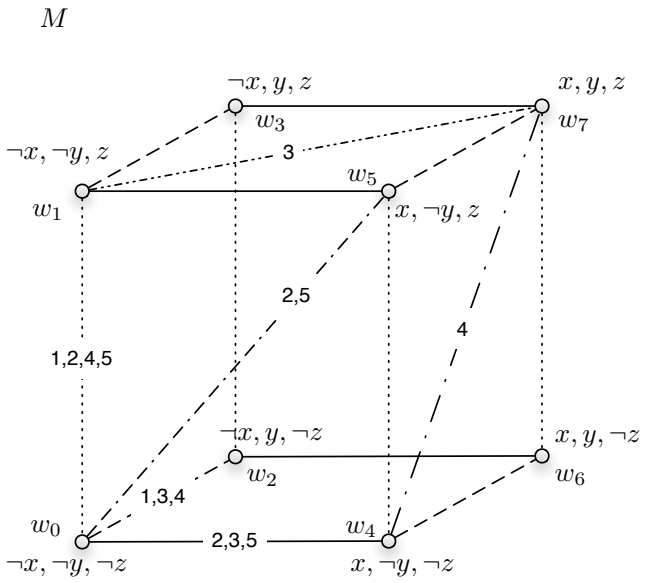


FIGURE 2.6: The epistemic model for Example 2.6. Each pair of parallel lines is labeled with the same agents. Not all lines are drawn in the Figure.

Example 2.6 (Product)**.** Let us now assume $\mathcal{V}ar = \{x, y, z\}$ and consider the program $\pi \stackrel{\circ}{=} (z := x; x := y; y := z)$, which swaps the values of x and y using a variable z. Let us assume we have five agents: agent 1 sees x, 2 sees y and 3 sees z. Agents 4 and 5, whose relevance will become clear later, do not see any variables. However, agent 4 'happens to know' the value of x, while agent 5 happens to know the value of y. Let $\mathsf{M}_{z:=x}$ have two points s_z ('z stays the same') and t_z ('z toggles'). Similarly for $\mathsf{M}_{x:=y}$ with points s_x and t_x, and model $\mathsf{M}_{y:=z}$ with points s_y and t_y. The individual program models for the three assignments are given in Figure 2.7.

The states of those models, taken together, give rise to 4 program states (and not 8 as it would be if we did not remove program points with inconsistent preconditions) in the program model for the composition π. The program $((\mathsf{t}_z, \mathsf{s}_x), \mathsf{t}_y)$ for instance indicates that the value of z will change, that of x will stay the same, and that of y will change (note the order of assignments in π). Note however that its precondition is unsatisfiable: indeed, no program that swaps the values of x and y can change the value of y but keep that of x the same, and therefore this program point is not included in the model (we will shortly give an example of how the pre-condition of a product program is computed).

The program model for π is depicted in Figure 2.8. Let us consider the program $((\mathsf{t}_z, \mathsf{t}_x), \mathsf{t}_y)$, which says that all three values of the variables should change. This would for instance be the case when x is $false$, y and z are $true$. At first sight, one might think that the last assignment $y := z$ would not change the value of y under this valuation, but when performed after $z := x; x := y$ the variable z has of course become false so indeed the assignment $y := z$ will change y's value. This is exactly how the new preconditions in the context of sequential composition are computed.

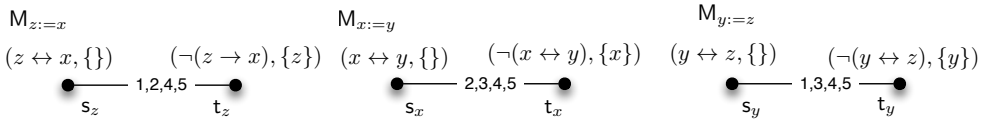


FIGURE 2.7: The three assignments $z := x$, $x := y$ and $y := z$.

To stay with the example $((\mathsf{t}_z, \mathsf{t}_x), \mathsf{t}_y)$, according to Definition 2.14 the program $(\mathsf{t}_z, \mathsf{t}_x)$ has the precondition $\mathsf{pre}(\mathsf{t}_z, \mathsf{t}_x) \stackrel{\circ}{=} (z \leftrightarrow \neg x) \wedge (x \leftrightarrow \neg y)$ and $\mathsf{tgl}(\mathsf{t}_z, \mathsf{t}_x) = \{x, z\}$. Now, to determine the precondition $\mathsf{pre}((\mathsf{t}_z, \mathsf{t}_x), \mathsf{t}_y)$ we obtain $\mathsf{pre}(\mathsf{t}_z, \mathsf{t}_x) \wedge \updownarrow(\mathsf{tgl}(\mathsf{t}_z, \mathsf{t}_x))(\mathsf{pre}(\mathsf{t}_y))$ which (since $\updownarrow(\mathsf{tgl}(\mathsf{t}_z, \mathsf{t}_x))(\mathsf{pre}(\mathsf{t}_y))$ is $\updownarrow(\{x, z\})(y \leftrightarrow \neg z)$ which in turn is $y \leftrightarrow z$) equals $(z \leftrightarrow \neg x) \wedge (x \leftrightarrow \neg y) \wedge (y \leftrightarrow z)$. Indeed, if y initially equals z, and z changes its value, then y will also change its value, during the program π.

The outcome of performing the assignments π to M is given in Figure 2.9. Where agent 4 knew the value of x in model M, as the result of π he forgets the value of x, but learns the values of y and z. Likewise, while agent 5 knew the value of y in M, in the resulting model he does not remember y, but he has learned the value of x. The agents 1, 2 and 3 all learn the value of z. In fact, 1 and 2 learn the values of all variables, while

3 learns the new values of y and z. Everybody knows in the new model that $y \leftrightarrow z$. In particular, we have that 3 learns what the old value of x was.

Let us finally mention that there is a program π_{swap} that swaps the values of x and y in such a way that agent 3 does not learn any of their values:

$$!(\neg x \wedge y); x\top :=; y\bot :=) \cup (!(x \wedge \neg y); x\bot :=; y\top :=) \cup !x \leftrightarrow y$$

That is, π_{swap} makes a distinction between three mutually exclusive but exhaustive cases:

- x is equivalent to y, in which case, when we want to swap their values, nothing needs to be done;
- currently we have $\neg x \wedge y$, in which case π_{swap} assigns the value $true$ to x and $false$ to y;
- currently $x \wedge \neg y$ holds, in which case π_{swap} assigns $false$ to x and $true$ to y.

Note that, e.g., it now holds that $([\pi_{swap}]Kw_i x \leftrightarrow Kw_i y)$: an agent will know the value of x after the swap, only if he knew the value of y before it (likewise, i will know that value of y after the swap only if the value of x is initially known).

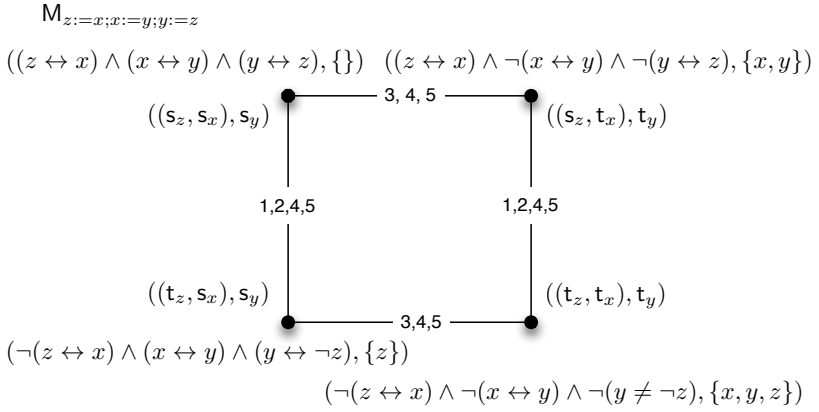


FIGURE 2.8: The program model for $z := x; x := y; y := z$. Diagonal connections for the relations $\approx_i^V$ are not shown. For instance, we have $((\mathsf{t}_x, \mathsf{s}_x), \mathsf{s}_y) \approx_4^V ((\mathsf{s}_z, \mathsf{t}_x), \mathsf{t}_y)$.

It is imperative to note that the indistinguishability relation of the composition follows the general definition of program models; it is derived only from the tgl sets of the resulting model (given V). This is in contrast to DEL, where the indistinguishability relations of the action models that are composed come into play. As a result, sequential composition in our setting does not follow the standard behaviour, in the sense that applying two models in turn and applying their composition does *not* always produce equivalent epistemic models (Example 2.7):

$$\big((M, w) \times (\mathsf{M}_1, \mathsf{w}_1)\big) \times (\mathsf{M}_2, \mathsf{w}_2) \;\not\equiv\; (M, w) \times \big((\mathsf{M}_1, \mathsf{w}_1) \otimes (\mathsf{M}_2, \mathsf{w}_2)\big)$$

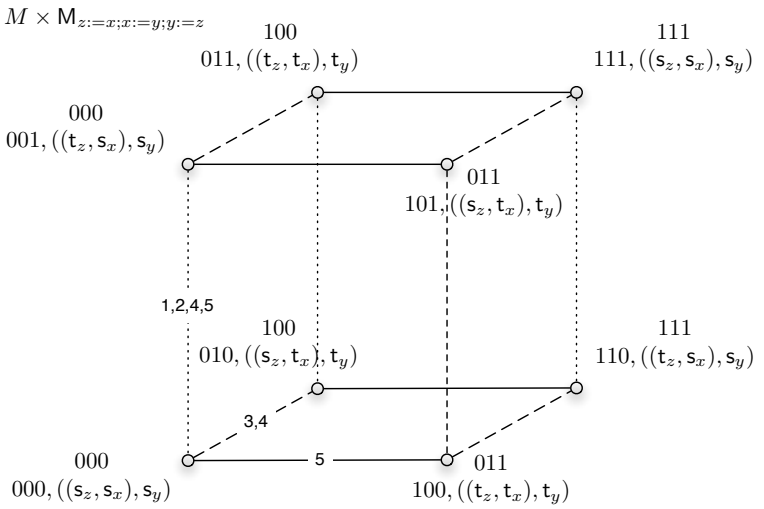


FIGURE 2.9: The product $M \times \mathsf{M}_{z:=x;x:=y;y:=z}$.

This is perfectly normal on an intuitive level: imagine agent 1 observing an object at all times, while agent 2 leaves and returns some time later, and thus observes only in the end. Agent 1 should be a able to distinguish between the case where the object is moved and then put back, and the case where it was not moved at all, while agent 2 should not.

Example 2.7. Take an epistemic model with one point w. Suppose $V(1) = \{x\}$. Let $\mathsf{M}_1 = \{\mathsf{u}, \mathsf{v}\} = \{(\top, \{x, y\}), (\top, \{y\})\}$ and $\mathsf{M}_2 = \{\mathsf{w}, \mathsf{z}\} = \{(\top, \{x\}), (\top, \{y\})\}$. Observe that the agent can distinguish between all different states (Def. 2.5). In $(M \times \mathsf{M}_1) \times \mathsf{M}_2$ the agent can also distinguish between all states (Def. 2.6), and therefore Kw_1y is valid. On the other hand, $\mathsf{M}_1 \otimes \mathsf{M}_2 = \{(\top, \{x, y\}), (\top, \{y\}), (\top, \{x\}), (\top, \{\})\}$ and the agent cannot distinguish between $(\mathsf{v}, \mathsf{w}) = (\top, \{x, y\})$ and $(\mathsf{u}, \mathsf{z}) = (\top, \{x\})$, therefore $(M \times (\mathsf{M}_1 \otimes \mathsf{M}_2), (w, \mathsf{v}, \mathsf{w})) \approx_1^V (M \times (\mathsf{M}_1 \otimes \mathsf{M}_2), (w, \mathsf{u}, \mathsf{z}))$. In these two states y has different values so $\neg Kw_1y$ holds.

2.2.5 Iteration

In dynamic logic, the operation * (Kleene Star) denotes unlimited repetition: $[\pi^*]\varphi$ is true if φ holds after any arbitrary number of executions of π. It can be used to define the so-called `while` program ('while ψ holds, do π'), as follows: $((?\psi; \pi)^*; ?\neg\psi)$. From a technical point of view, the Kleene star often leads to complications (cf. [33]). In our setting we can deal with the iteration that denotes repetition of the operator ";" in a straightforward way. We remind the reader that this kind of composition is not equivalent to executing actions serially, as mentioned in the previous subsection. Some additional needed definitions follow.

Definition 2.15. Let $\varphi_0, \varphi_1 \in \mathcal{L}_0$. Define $var(\varphi_0) = \{x\}$ if φ_0 is either x or $\neg x$. Define $var(\varphi_0 \wedge \varphi_1) = var(\varphi_0) \cup var(\varphi_1)$. Let w be a program point. Let $var(\mathsf{w}) = var(\mathsf{pre}(\mathsf{w}) \cup \mathsf{tgl}(\mathsf{w}))$. Finally, for a program model M, let $var(\mathsf{M}) = \cup_{\mathsf{w}\in\mathsf{M}} var(\mathsf{w})$.

Now suppose M is such that $var(\mathsf{M}) = X = \{x_1, \ldots, x_n\}$. Then there are only finitely many program models N with $var(\mathsf{N}) \subseteq X$. This is seen as follows. Let a valuation description over X be a conjunction of the form $l_1 \wedge \cdots \wedge l_n$, with $l_i \in \{x_i, \neg x_i\}$. Then any φ_0 with $var(\varphi_0) = X$ is equivalent to a disjunction of such valuation descriptions. There are 2^n valuation descriptions over X, so there are 2^{2^n} logically different term formulas φ_0 with $var(\varphi_0) = X$. Since a program point is characterised by $\mathsf{pre}(\mathsf{w})$ and $\mathsf{tgl}(\mathsf{w})$, there are at most $2^{2^n} \times 2^n$ different program points over X. Hence there are at most $2^{2^{2^n} \times 2^n}$ different program models M with $var(\mathsf{M}) = X$. Call this number $\#(\mathsf{M})$. Let $\mathsf{M}^1 = \mathsf{M}$ and $\mathsf{M}^k = \mathsf{M}^{k-1}; \mathsf{M}$. We then have:

$$[\mathsf{M}]^*\varphi = [\mathsf{M}]\varphi \wedge [\mathsf{M}^2]\varphi \wedge \ldots [\mathsf{M}^{\#(\mathsf{M})}]\varphi \tag{2.3}$$

In words: φ is true after an arbitrary number of toggles induced by M iff φ stays true after applying M as often as it is possible to make changes to a model. This shows that iteration is definable in our framework. For this reason, we don't need to deal with iteration any further, and we omit reference to it in our subsequent analysis.

Remark 2.16. Note that the above definition is not to say that we semantically define M^* as $\mathsf{M}^* = \mathsf{M}^1 \cup \cdots \cup \mathsf{M}^{\#(\mathsf{M})}$. This would indeed not be correct, as the following example shows. Suppose we have an epistemic model M consisting of only one state w, at which x is *true*. In particular, at this state, we have that Kw_ix, i.e., i knows what the value is of x. Let us also assume that $x \notin V(i)$. Now let $\mathsf{M} = \mathsf{M}_{\updownarrow x}$ from Definition 2.11, the model that simply toggles the value of x. Then $M, w \models [\mathsf{M}]Kw_ix$ and also $M, w \models [\mathsf{M}^2]Kw_ix$: since i knew the value of x, as long as he knows how often this values has been toggled, he will stay aware of that value. However, we do *not* have $M, w \models [\mathsf{M} \cup \mathsf{M}^2]Kw_ix$, since in the program $\mathsf{M} \cup \mathsf{M}^2$, agent i does not know which program point is executed, i.e., he does not know how often x gets toggled. Note that we *do* have $M, w \models [\mathsf{M}]^*Kw_ix$, as desired.

2.2.6 Inductively Defined Pointed Program Models

We make it explicit that the set of all program models and the set of the program models that can be constructed using the models corresponding to the set of program $\mathcal{P}rog$, coincide.

Definition 2.17. We give the set of inductively defined pointed program models, or $\mathsf{IPM}(\mathcal{L}_0, \mathcal{V}ar)$, as follows.

1. if $\varphi_0 \in \mathcal{A}^-$, then $\mathsf{M}_{!\varphi_0}, \mathsf{e} \in \mathsf{IPM}(\mathcal{L}_0, \mathcal{V}ar)$
2. If $x \in \mathcal{V}ar$ then $\mathsf{M}_{\updownarrow x}, \mathsf{a} \in \mathsf{IPM}(\mathcal{L}_0, \mathcal{V}ar)$

3. If $\mathsf{M}, \mathsf{w}, \mathsf{N}, \mathsf{v} \in \mathsf{IPM}(\mathcal{L}_0, \mathcal{V}ar)$ then
   $(\mathsf{M} \cup \mathsf{N}, \mathsf{w}), (\mathsf{M} \cup \mathsf{N}, \mathsf{v}) \in \mathsf{IPM}(\mathcal{L}_0, \mathcal{V}ar)$

4. If $\mathsf{M}, \mathsf{w}, \mathsf{N}, \mathsf{v} \in \mathsf{IPM}$ and $\mathsf{pre}(\mathsf{w}) \wedge \Updownarrow(\mathsf{tgl}(\mathsf{w}))(\mathsf{pre}(\mathsf{v}))$ is consistent, then
   $(\mathsf{M}, \mathsf{w} \otimes \mathsf{N}, \mathsf{v}) \in \mathsf{IPM}(\mathcal{L}_0, \mathcal{V}ar)$.

Theorem 2.18. $\mathsf{PM}(\mathcal{L}_0, \mathcal{V}ar) = \mathsf{IPM}(\mathcal{L}_0, \mathcal{V}ar)$.

Proof. It is clear that $\mathsf{PM}(\mathcal{L}_0, \mathcal{V}ar)$ contains $\mathsf{M}_{!\varphi_0}, \mathsf{e}$ and $\mathsf{M}_{\Updownarrow x}, \mathsf{a}$, and is closed under non-deterministic choice and 'consistent' sequential composition. This implies that $\mathsf{PM}(\mathcal{L}_0, \mathcal{V}ar) \supseteq \mathsf{IPM}(\mathcal{L}_0, \mathcal{V}ar)$. For $\subseteq$, let $\mathsf{M}, \mathsf{w} \in \mathsf{PM}(\mathcal{L}_0, \mathcal{V}ar)$. Take a program point $\mathsf{u} \in \mathsf{M}$ and let $\mathsf{tgl}(\mathsf{u}) = \{x_1, \ldots, x_n\}$. We have that u is equal to the program $!\mathsf{pre}(\mathsf{u}); \Updownarrow x_1; \ldots; \Updownarrow x_n$ and so it can obviously be inductively defined. $\mathsf{M} = \bigcup_{\mathsf{u} \in \mathsf{M}} \{\mathsf{u}\}$ thus M itself (and therefore M, w) can be inductively defined. □

After this theorem it should be clear that the existence of more than one program points in a model can be attributed to the program's non-deterministic character; the only way to introduce new points is through non-deterministic choice $\cup$. "Vision" then provides a way to resolve uncertainty in case it provides proof of what actually occurred during the computation. On that note we can also elaborate on our statement that non-deterministic choice is given a realistic interpretation. One can alternatively think of semi-public environments that the execution is not being performed publicly, but each agent is given a range of *deterministic* programs (the program points that are of the form $!\mathsf{pre}(\mathsf{u}); \Updownarrow x_1; \ldots; \Updownarrow x_n$) from which one will be executed. Thus effectively 'hiding' non-deterministic operators behind the agent's ignorance.

2.2.7 Completeness

Using the validities for our logic of semi-public environments, it is possible to eliminate programs from formulas, thereby proving Theorem 2.7 (for details on a similar translation – for the logic of action models – we refer the reader to [19]). In other words, we can reduce the logic of semi-public environments to a multi-agent epistemic logic enriched with a notion of vision expressed by the atoms $V_i x$. Starting with the standard $\mathrm{S5}_n$ validities for knowledge modalities K_i, and then by adding two additional axioms for the V_i operators we have a sound and complete axiomatisation for the logic "S5+V". Putting everything together we obtain the following result:

Theorem 2.19. *The formulas in Table 2.5 constitute a sound and complete axiomatisation for the logic of semi-public environments.*

Observe that we do not include an axiom to immediately translate formulas of the form $[\mathsf{M}_1, \mathsf{w}_1][\mathsf{M}_2, \mathsf{w}_2]\varphi$ but this can be overcome by starting the translation from the innermost dynamic operator. This is facilitated by the rule of substitution of equivalents (which says that provably equivalent formulas can be substituted for each other within

Propositional Component	
φ	if φ is a prop. tautology
Epistemic Component	
$V_i x \to (K_i x \vee K_i \neg x)$	seeing implies knowing
$V_i x \to K_j V_i x$	vision is common knowledge
$K_i(\varphi \to \psi) \to (K_i\varphi \to K_i\psi)$	K-axiom
$K_i\varphi \to \varphi$	veridicality (truth axiom)
$K_i\varphi \to K_iK_i\varphi$	positive introspection
$\neg K_i\varphi \to K_i\neg K_i\varphi$	negative introspection
Dynamic Component ($\varphi_0 \in \mathcal{L}_0,\ \varphi, \psi \in \mathcal{L}$)	
$[\mathsf{M},\mathsf{w}]\varphi_0 \leftrightarrow (\mathsf{pre}(\mathsf{w}) \to \updownarrow(\mathsf{tgl}(\mathsf{w}))(\varphi_0))$	ontic change
$[\mathsf{M},\mathsf{w}]V_i x \leftrightarrow (\mathsf{pre}(\mathsf{w}) \to V_i x)$	vision permanence
$[\mathsf{M},\mathsf{w}]\neg\varphi \leftrightarrow (\mathsf{pre}(\mathsf{w}) \to \neg[\mathsf{M},\mathsf{w}]\varphi)$	program and negation
$[\mathsf{M},\mathsf{w}](\varphi \wedge \psi) \leftrightarrow ([\mathsf{M},\mathsf{w}]\varphi \wedge [\mathsf{M},\mathsf{w}]\psi)$	program and conjunction
$[\mathsf{M},\mathsf{w}]K_i\varphi \leftrightarrow \big(\mathsf{pre}(\mathsf{w}) \to \bigwedge_{V\in\mathcal{V}is}(\chi_V \to \bigwedge_{\mathsf{w}\approx_i^V \mathsf{u}} K_i[\mathsf{M},\mathsf{u}]\varphi)\big)$	program and knowledge
Rules of Inference	
if $\vdash \varphi$ and $\vdash (\varphi \to \psi)$ then $\vdash \psi$	modus ponens
if $\vdash \varphi$ then $\vdash K_i\varphi$	knowledge-necessitation
if $\vdash \varphi$ then $\vdash [\mathsf{M},\mathsf{w}]\varphi$	program-necessitation
if $\vdash \psi_1 \leftrightarrow \psi_2$ then $\vdash \varphi[\psi_1/\psi] \leftrightarrow [\psi_2/\psi]$	substitution of equivalents

TABLE 2.5: Axiomatisation of SPE

a formula ψ without effecting the truth of ψ —- see for instance [50, Section 5] for more on this rule).

We refer to axiom $V_i x \to K_j V_i x$ as "vision is common knowledge". This terminology may appear to be misleading, given the fact that we do not have operators for common knowledge in our object language. However, the axiom, together with knowledge-necessitation and the K-axiom, guarantees that common knowledge of vision can be obtained in the following sense: any formula of the form $V_i x \to K_{j_1} \dots K_{j_n} V_i x$ is a theorem. (To see a derivation of $V_i x \to K_1K_2K_3V_i x$, note that by the axiom under consideration, we have $V_i x \to K_3V_i x$, and, by knowledge necessitation $K_2(V_i x \to K_3V_i x)$ from which, using the K axiom, we obtain $K_2V_i x \to K_2K_3V_i x$. Since we also have $V_i x \to K_2V_i x$, we then get $V_i x \to K_2K_3V_i x$. Applying once more necessitation and the K-axiom for agent 1, we get the desired result).

Finally, note that the models of $S5 + V$ are what we intended from the start. Vision of the value of a variable works as a 'lower bound' for the knowledge of its value, and $V_i x$ is either true in all of the possible worlds, or false (in all possible worlds).

2.3 When are two programs the same?

In modal logic (a good reference is [12]), the notion of *bisimulation* helps to give an answer to the question: when are two pointed models the same? Being bisimilar (written $(M, w) \leftrightarroweq (M', w')$) guarantees that (M, w) and (M', w') satisfy the same static formulas,

and on finite models, the converse holds as well. We assume the reader to be familiar with the notion of bisimulation between epistemic models: we state the key clauses without further explanation.

Formally, if $M = \langle W, R, V, f\rangle$ and $M' = \langle W', R', V', f'\rangle$ are two epistemic models, in order for a bisimulation $\mathfrak{R}$ to have $\mathfrak{B}(w, w')$, we require that

Atom $f(w) = f'(w')$ and $\forall i\ V(i) = V'(i)$;

Forth-Bisim If for some $v \in W$ we have wR_iv then for some $v' \in W'$ we have $w'R'_iv'$ and $\mathfrak{B}(v, v')$;

Back-Bisim If for some $v' \in W'$ we have $w'R'_iv'$ then for some $v \in W$ we have $w'R_iv$ and $\mathfrak{B}(v, v')$;

We use $\mathfrak{B} : (M, s) \leftrightarrows (M', s')$ for the claim that $\mathfrak{B}$ is a bisimulation between M and M' such that $\mathfrak{R}(s, s')$.

Proposition 2.20. *Suppose* $(M, s) \leftrightarrows (M', s')$. *Also suppose* M, s *is such that* $M, s \models \mathsf{pre(s)}$. *Then* $(M \times \mathsf{M}, (s, \mathsf{s})) \leftrightarrows (M' \times \mathsf{M}, (s', \mathsf{s}))$.

Proof. The proof of Proposition 2.20 follows that of [19, Proposition 6.21], and is constructive: assume $\mathfrak{B} : (M, s) \leftrightarrows (M', s')$ is the witness the bisimulation of the epistemic models. Then consider $\mathfrak{B}'$, defined by $\mathfrak{B}'((u, \mathsf{u}), (u', \mathsf{u}')$ iff both $\mathfrak{B}(u, u')$ and $\mathsf{u} = \mathsf{u}'$. We leave it to the reader to show that $\mathfrak{B}' : (M \times \mathsf{M}, (s, \mathsf{s})) \leftrightarrows (M' \times \mathsf{M}, (s', \mathsf{s}))$. □

Following work by van Eijck *et. al* ([25, 55]) we propose a notion of 'equivalence' for program models. Our presentation is similar to the exposition of [19, Chapter 6, Section 6.1], here, we built on results presented there, and will focus in our proofs on the differences. Note that where [19] talks about action bisimulation and action emulation, in our terminology this will be program bisimulation and program emulation. The idea is that we want to achieve that if two pointed program models emulate, then for every pointed epistemic model (M, w), we have $((M, w) \times (\mathsf{M}, \mathsf{w})) \leftrightarrows ((M, w) \times (\mathsf{M}', \mathsf{w}'))$, and hence, the two programs (M, w) and $(\mathsf{M}', \mathsf{w}')$ have the same effect (see Proposition 2.25). Program emulation is weaker than program bisimulation: here, the idea is that having two programs emulate is a sufficient (Theorem 2.28) and necessary (Theorem 2.29) condition to make them have the 'same effect' on epistemic models.

Definition 2.21 (Bisimulation of program models)**.** Given two pointed program models (M, w), $(\mathsf{M}', \mathsf{w}')$, and a vision function V, a *program bisimulation* between them is a relation $\mathfrak{B}_\mathsf{p} \subseteq (\mathsf{M} \times \mathsf{M}')$ such that $\mathfrak{B}_\mathsf{p}(\mathsf{w}, \mathsf{w}')$ and the following three conditions are met for each agent i (for arbitrary program points):

- **Here-Bisim** $_\mathsf{p}$ If $\mathfrak{B}_\mathsf{p}(\mathsf{u}, \mathsf{u}')$, then $\mathsf{pre(u)} \leftrightarrow \mathsf{pre(u')}$ and $\mathsf{tgl(u)} = \mathsf{tgl(u')}$.
- **Forth-Bisim** $_\mathsf{p}$ If $\mathfrak{B}_\mathsf{p}(\mathsf{u}, \mathsf{u}')$ and $\mathsf{u} \approx_i^V \mathsf{v}$, then there is $\mathsf{v}' \in \mathsf{M}'$ such that $\mathfrak{B}_\mathsf{p}(\mathsf{v}, \mathsf{v}')$;
- **Back-Bisim** $_\mathsf{p}$ If $\mathfrak{B}_\mathsf{p}(\mathsf{u}, \mathsf{u}')$ and $\mathsf{u}' \approx_i^V \mathsf{v}'$, then there is $\mathsf{v} \in \mathsf{M}$ such that $\mathfrak{B}_\mathsf{p}(\mathsf{v}, \mathsf{v}')$;

We write $\mathsf{M}, \mathsf{w} \leftrightarrows^{V}_{\mathsf{p}} \mathsf{M}', \mathsf{w}'$ if there is a program bisimulation $\mathfrak{B}_{\mathsf{p}} \subseteq (\mathsf{M} \times \mathsf{M}')$ based on V for which $\mathfrak{B}_{\mathsf{p}}(\mathsf{w}, \mathsf{w}')$.

The reader will notice that bisimulation and emulation have a vision function V as a parameter. This is to be expected since we need a vision function V to produce an indistinguishability relation for the program models we are comparing. Also, as we have already stated, we are interested in the effect these programs would have on the same epistemic model, therefore it also makes sense to compare program models using the same V.

Comparing our **Forth-Bisim**$_{\mathsf{p}}$ clause with that of **Forth-Bisim** of bisimulation between epistemic models, and with the Forth-clause for action models [19], the reader would expect the additional clause $\mathsf{u}' \approx^{V}_{i} \mathsf{v}'$:

- **Forth-Bisim**$'_{\mathsf{p}}$ If $\mathfrak{B}_{\mathsf{p}}(\mathsf{u}, \mathsf{u}')$ and $\mathsf{u} \approx^{V}_{i} \mathsf{v}$, then there is $\mathsf{v}' \in \mathsf{M}'$ such that $\mathsf{u}' \approx^{V}_{i} \mathsf{v}'$ and $\mathfrak{B}_{\mathsf{p}}(\mathsf{v}, \mathsf{v}')$;

This addition would in fact be redundant, due to the fact that relation $\approx^{V}_{i}$ is not arbitrary and can be derived exclusively from the relevant tgl sets and vision function V. This is made explicit with the following Lemma:

Lemma 2.22. *Let two program models* M, M', *a vision function* V, *and a relation* $\mathfrak{B}_{\mathsf{p}} \subseteq (\mathsf{M} \times \mathsf{M}')$ *be given. If* **Here-Bisim**$_{\mathsf{p}}$ *and* **Forth-Bisim**$_{\mathsf{p}}$ *are true for* $\mathfrak{B}_{\mathsf{p}}$, *then so is* **Forth-Bisim**$'_{\mathsf{p}}$.

Proof. So let $\mathsf{u}, \mathsf{v} \in \mathsf{M}$ and $\mathsf{u}' \in \mathsf{M}'$. Also let $\mathfrak{B}_{\mathsf{p}}(\mathsf{u}, \mathsf{u}')$ and $\mathsf{u} \approx^{V}_{i} \mathsf{v}$. It suffices to show that $\mathsf{u}' \approx^{V}_{i} \mathsf{v}'$.

Because of **Forth-Bisim**$_{\mathsf{p}}$, there exists $\mathsf{v}' \in \mathsf{M}'$ such that $\mathfrak{B}_{\mathsf{p}}(\mathsf{v}, \mathsf{v}')$. By **Here-Bisim**$_{\mathsf{p}}$ and because $\mathfrak{B}_{\mathsf{p}}(\mathsf{u}, \mathsf{u}')$ we have that $\mathsf{tgl}(\mathsf{u}) = \mathsf{tgl}(\mathsf{u}')$. Similarly, By **Here-Bisim**$_{\mathsf{p}}$ and $\mathfrak{B}_{\mathsf{p}}(\mathsf{v}, \mathsf{v}')$ we have that $\mathsf{tgl}(\mathsf{v}) = \mathsf{tgl}(\mathsf{v}')$. Because $\mathsf{u} \approx^{V}_{i} \mathsf{v}$ we have (Def. 2.5) that $(\mathsf{tgl}(\mathsf{u}) \triangle \mathsf{tgl}(\mathsf{v})) \cap V(i) = \emptyset$. Substituting $\mathsf{tgl}(\mathsf{u})$ with $\mathsf{tgl}(\mathsf{u}')$ and $\mathsf{tgl}(\mathsf{v})$ with $\mathsf{tgl}(\mathsf{v}')$ we get $(\mathsf{tgl}(\mathsf{u}') \triangle \mathsf{tgl}(\mathsf{v}')) \cap V(i) = \emptyset$. Therefore $\mathsf{u}' \approx^{V}_{i} \mathsf{v}'$. □

A similar remark applies to **Back-Bisim**$_{\mathsf{p}}$. To emphasise this further we provide the following propositions.

For a program model M, a program point w, vision function V and agent i, define $\mathsf{R}^{\mathsf{M},V}_{i}(\mathsf{w}) = \{\mathsf{u} \in \mathsf{M} \mid \mathsf{w} \approx^{V}_{i} \mathsf{u}\}$.

Proposition 2.23. *Let* $(\mathsf{M}, \mathsf{w}), (\mathsf{M}', \mathsf{w}')$ *be two pointed program models. Then:* $\mathsf{M}, \mathsf{w} \leftrightarrows^{V}_{\mathsf{p}} \mathsf{M}', \mathsf{w}'$ *iff for all* $i \in Ag$, $\mathsf{R}^{\mathsf{M},V}_{i}(\mathsf{w}) = \mathsf{R}^{\mathsf{M}',V}_{i}(\mathsf{w}')$.

Proposition 2.24. *Let* (M, w), $(\mathsf{M}', \mathsf{w}')$ *be two pointed program models and* V, Y *two vision functions. If* $\mathsf{M}, \mathsf{w} \leftrightarrows^{V}_{\mathsf{p}} \mathsf{M}', \mathsf{w}'$ *and* $V \subseteq Y$, *then* $\mathsf{M}, \mathsf{w} \leftrightarrows^{Y}_{\mathsf{p}} \mathsf{M}', \mathsf{w}'$

The following proposition captures our main intention for bisimulation.

Proposition 2.25. *Suppose we have an epistemic pointed model M, s with $M, s \models \mathsf{pre(s)}$ and vision function V, and two pointed program models $\mathsf{M, s}$ and $\mathsf{M', s'}$ such that $\mathsf{M, s} \leftrightarrows^V_{\mathsf{p}} \mathsf{M', s'}$. Then*

$$(M \times \mathsf{M}, (s, \mathsf{s})) \leftrightarrows (M \times \mathsf{M'}, (s, \mathsf{s'}))$$

Proof. The proof is similar to that of [19, Proposition 6.23]. Let $\mathfrak{B}_{\mathsf{p}}$ be the program bisimulation between $\mathsf{M, s}$ and $\mathsf{M', s'}$. Then $\mathfrak{B}'$ is a witness for the claim we are after:

$$\mathfrak{B}'((u, \mathsf{u})(v, \mathsf{v})) \text{ iff } u = v \text{ and } \mathfrak{B}_{\mathsf{p}}(\mathsf{u}, \mathsf{v})$$

□

It should now be clear why the condition $\mathsf{tgl(u)} = \mathsf{tgl(u')}$ in **Here-Bisim**$_{\mathsf{p}}$, which does not occur for action models, is needed: since our programs can change the state, we only want to identify states that make the same changes. The reader should convince herself that our **Here-Bisim**$_{\mathsf{p}}$ clause is necessary to show the **Atom** case in Proposition 2.25.

The following corollary follows immediately from Propositions 2.20 and 2.25.

Corollary 2.26. *Let M be an epistemic model with vision function V. If $(M, s) \leftrightarrows (M', s')$ and $(\mathsf{M, s}) \leftrightarrows^V_{\mathsf{p}} (\mathsf{M', s'})$ then $(M \times \mathsf{M}, (s, \mathsf{s})) \leftrightarrows (M' \times \mathsf{M'}, (s', \mathsf{s'}))$.*

We now introduce the notion of program emulation, which is a weaker form of structural similarity compared to that of program bisimulation, but still guarantees bisimilarity of epistemic states, when programs that emulate each other, are executed on it. Roughly, this weakening is obtained as follows. Rather than requiring that two program points (as is program bisimulation) have exactly the same preconditions, for emulation, it is sufficient that one precondition entails the other— as long as the weaker condition is compensated for by a number of alternatives: see the requirements **Forth-Emul** and **Back-Emul** below.

Definition 2.27 (Emulation)**.** Given two pointed program models $(\mathsf{M, w}), (\mathsf{M', w'})$, and a vision function V, a *program emulation* between them is a relation $\mathfrak{E} \subseteq (\mathsf{M} \times \mathsf{M'})$ such that $\mathfrak{E}(\mathsf{w}, \mathsf{w'})$ and the following three conditions are met for each agent i (for arbitrary program points):

- **Here-Emul** If $\mathfrak{E}(\mathsf{u}, \mathsf{u'})$, then $\mathsf{pre(u)} \wedge \mathsf{pre(u')}$ is consistent and, moreover, $\mathsf{tgl(u)} = \mathsf{tgl(u')}$.
- **Forth-Emul** If $\mathfrak{E}(\mathsf{u}, \mathsf{u'})$ and $\mathsf{u} \approx^V_i \mathsf{v}$, then there are $\mathsf{v'_1}, ...\mathsf{v'_n} \in \mathsf{M'}$ such that for all $k = 1, ..., n$, $\mathfrak{E}(\mathsf{v}, \mathsf{v'_k})$, and such that $\mathsf{pre(v)} \models \mathsf{pre(v'_1)} \vee \cdots \vee \mathsf{pre(v'_k)}$ and $\mathsf{tgl(v)} = \mathsf{tgl(v'_1)} = \cdots = \mathsf{tgl(v'_n)}$.
- **Back-Emul** If $\mathfrak{E}(\mathsf{u}, \mathsf{u'})$ and $\mathsf{u'} \approx^V_i \mathsf{v'}$, then there are $\mathsf{v}_1, ...\mathsf{v}_n \in \mathsf{M}$ such that for all $k = 1, ..., n$, $\mathfrak{E}(\mathsf{v}_k, \mathsf{v'})$, and such that $\mathsf{pre(v')} \models \mathsf{pre(v}_1) \vee \cdots \vee \mathsf{pre(v}_n)$ and $\mathsf{tgl(v')} = \mathsf{tgl(v}_1) = \cdots = \mathsf{tgl(v}_n)$.

A *total emulation* $\mathfrak{E} : \mathsf{M} \rightleftarrows^V \mathsf{M}'$ is an emulation such that for each $\mathsf{w} \in \mathsf{M}$ there is a $\mathsf{w}' \in \mathsf{M}'$ with $\mathfrak{E}(\mathsf{w}, \mathsf{w}')$ and vice versa.

It should be clear that a program bisimulation is also a program emulation.

Example 2.8. *Consider the models* $\mathsf{M}_{!\varphi_0 \vee \psi_0} = \{\mathsf{e} = (\varphi_0 \vee \psi_0, \emptyset)\}$ *and* $\mathsf{M}_{!\varphi_0 \cup !\psi_0} = \{\mathsf{e}_1 = (\varphi_0, \emptyset), \mathsf{e}_2 = (\psi_0, \emptyset)\}$. *The reader can easily check* $\mathfrak{E} = \{(e, e_1), (e, e_2)\}$ *is a total emulation.*

The following theorem shows that program emulation guarantees bisimulation of results:

Theorem 2.28. *Let* M *be an epistemic model with vision function* V. *If* $\mathsf{M} \rightleftarrows^V \mathsf{M}'$ *then* $(M \times \mathsf{M}) \leftrightarroweq (M \times \mathsf{M}')$.

Proof. The proof of this theorem is similar to that of [19, Proposition 6.30]. The bisimulation that we are after is defined as

$$\mathfrak{B}((w, \mathsf{w}), (w, \mathsf{w}')) \text{ iff } w = v \text{ and } (\mathsf{M}, \mathsf{w}) \rightleftarrows^V (\mathsf{M}', \mathsf{w}')$$

We leave it to the reader that $\mathfrak{B}$ indeed is a bisimulation between static epistemic models. □

In our setting preconditions are only propositional and, by altering the proof of the related property [24, Proposition 2] accordingly, we get the following extra.

Theorem 2.29. *Let* M *be an epistemic model with vision function* V. *If* $(M \times \mathsf{M}) \leftrightarroweq (M \times \mathsf{M}')$ *then* $\mathsf{M} \rightleftarrows^V \mathsf{M}'$.

It is also natural to consider when two programs are 'the same' directly with respect to what agents can actually reason about – namely, formulas.

Definition 2.30. Let us write (in the meta language) $\pi_1 \sim \pi_2$ to mean that for all $\varphi \in \mathcal{L}$, we have $\models [\mathsf{M}_{\pi_1}]\varphi \leftrightarrow [\mathsf{M}_{\pi_2}]\varphi$.

This approach has its own interest even outside the context of an agent's epistemic reasoning. Let M be a model and $w \in M$. Applying models M_{π_1} and M_{π_2} will result in worlds of the form (w, u_1) and (w, u_2) with u_1 and u_2 being points of the first and second program model respectively. So we have a partition of the resulting static models based on the first component. The definition then says that the programs are 'equal' if: φ is valid in a part of $M \times \mathsf{M}_{\pi_1}$ iff it is valid in the respective part of $M \times \mathsf{M}_{\pi_2}$. It is therefore evident that this definition is a stronger version of model equivalence.

Theorem 2.31. $\pi_1 \sim \pi_2$ *iff for all functions* $V \in \mathcal{V}is$, $\mathsf{M}_{\pi_1} \rightleftarrows^V \mathsf{M}_{\pi_2}$.

Proof. From left to right: Let V be a vision function and M a model with that vision function. As we pointed out before, we have $(M \times \mathsf{M}_{\pi_1}) \equiv (M \times \mathsf{M}_{\pi_2})$. In our setting all models are finite, so (cf. [11]) we further have $(M \times \mathsf{M}_{\pi_1}) \leftrightarroweq (M \times \mathsf{M}_{\pi_2})$. By Theorem 2.29

we have $\mathsf{M}_{\pi_1} \rightleftarrows^V \mathsf{M}_{\pi_2}$. From right to left: Let $\mathfrak{E}$ be the assumed emulation. By the proof of Theorem 2.28, we know that there is a bisimulation between $(M \times \mathsf{M}_{\pi_1}, (w_1, \mathsf{u}_1))$ and $(M \times \mathsf{M}_{\pi_2}, (w_2, \mathsf{u}_2))$ if $w_1 = w_2$ and $\mathfrak{E}(\mathsf{u}_1, \mathsf{u}_2)$. Now assume $M, w \models [\mathsf{M}_{\pi_1}]\varphi$ i.e., for every $\mathsf{u} \in \mathsf{M}_{\pi_1}$ such that $M, w \models pre(\mathsf{u})$, it holds that $M \times \mathsf{M}_{\pi_1}, (w, \mathsf{u}) \models \varphi$. Because $\mathfrak{E}$ is a total emulation and because of the fact that bisimilarity implies point-wise equivalence, we also have $M, w \models [\mathsf{M}_{\pi_2}]\varphi$. □

Example 2.9. *We have the following equivalences.*

- $(!\varphi_0 \cup !\psi_0) \sim !(\varphi_0 \vee \psi_0)$
- $\updownarrow x \sim (x := \neg x)$

Remark 2.32. In our program models, agent i cannot distinguish points u and w if from the variables he sees, those affected by u are the same as those affected by w. This may seem weird, because it might be that u would change x from $false$ to $true$, while w does the opposite. However, this is harmless, because those points will have preconditions ($\neg x$ and x, respectively) that i can distinguish.

However, for the Boolean domain one can (and for domains with more than two values, one needs to) model change as follows. Consider $x := \varphi$ again. In the Boolean domain, we take three program points for the program model. One of those three points is similar to s, representing 'stay the same'. Then, we have a state t_1, representing 'change from $false$ to $true$', with precondition $(\neg x \wedge \varphi)$ and a program point t_2, representing 'change from $true$ to $false$', with precondition $(x \wedge \neg\varphi)$. We replace tgl by a set upd. For instance, we would have $\mathsf{upd}(\mathsf{t}_1) = \{x \mapsto \top\}$ and $\mathsf{upd}(\mathsf{t}_2) = \{x \mapsto \bot\}$, where $\{x \mapsto \top\}\varphi_0$ would then be defined as $\varphi_0[\top/x]$.

2.4 Related Work

Our work is most closely related to that of van Benthem et al [8] which studies information change combined with 'factual alteration'. This work places the emphasis on reduction axioms and the underlying system 'is capable of expressing all model-shifting operations with finite action models' [8, page 1]. However, as explained on [8, page 6], an approach where one has regular operations (choice, sequence and iteration) to make structured programs starting from simple actions is not examined. Moreover, there is no notion of visibility in [8].

Our work is also related to 'Dynamic Epistemic Logic with Assignment' [18]. However, there are some important differences. We aim at modelling a computational environment, where *program variables* are first class citizens: they receive values during a commonly known computation, while agents derive their knowledge from the variables they observe and the program that is being executed. The motivation in [18] is more general: its starting point is DEL, to which an assignment of the form $p := \varphi$ is added, that is, some atomic propositions receive a new value. The logic of [18] includes common

knowledge and more general update operators than just public announcement. However, there is a price to pay for this generality: [18] gives only one relevant semantic principle, whereas we provide a complete axiomatisation. Moreover, in our framework, the notion of *visibility* (of variables, by agents) plays a central role. This makes it possible to exactly formalise the interaction between assignments, visibility, and knowledge. No such principles are given in [18]. One might argue that $V_i x$ (agent i sees x) in our framework might be mimicked in [18] as $K_i x \vee K_i \neg x$. However, this still does not give the same behaviour in our system: in [18], even if the agent knows the value of x, and he observes the public program $x := y$, if he does not know the value of y, he will 'forget' the value of x! Of course, a formula of the form $V_i x \leftrightarrow K_i x \vee K_i \neg x$ is not valid in our semantics (an agent may know more than is implied by just the variables he sees).

Another line of relevant work is the "Dynamic Logic of Propositional Assignments" ([4]), or DL-PA for short. We claim that for the case of a finite set of variables $\mathcal{V}ar$ we can embed this language into to our framework. We make the comparison firstly based on the language used. In DL-PA the language (let us call it $\mathcal{L}_{DLPA}$) comprises programs $\pi ::= +p \mid -p \mid \pi;\pi \mid \pi \cup \pi \mid \pi^* \mid ?\varphi$ and formulas $\varphi ::= p \mid \top \mid \bot \mid \neg\varphi \mid \varphi \vee \varphi \mid [\pi]\varphi$. Atoms $+p$ (and $-p$ respectively) can be translated in our setting to programs $[p := \top]$ and the rest of the program connectives translate as expected. Another subtle difference is that the tests of DL-PA can include programs, however it is also the case in DL-PA that all formulas are equivalent to some program-free formula, and so we can use this reduct for our version of tests. Regarding semantics, the models of DL-PA are those of propositional logic, i.e., valuations. The translation of this to our setting would be any model $M = \langle W, R, V, f\rangle$, with W the set of all valuations, $f(w) = w$ and for each $i \in \mathcal{AG}, V(i) = \emptyset$. The accessibility relation R could be anything again due to the lack epistemic modalities, and for each agent i, $V(i)$ has to be empty so that $\cup$ would indeed be non-deterministic and no agent finds a way around it using his vision. Finally, for $t(\varphi)$ being the translation we mention above, we claim that for every $\varphi \in \mathcal{L}_{DLPA}$, φ is valid in DL-PA iff $t(\varphi)$ is valid in the logic of semi-public environments. We ought to mention that this claim is not true between DL-PA and PDL (with atomic programs those of propositional assignment).

We should also mention the relationship of our work to the logic AC of action models for DEL [19, Chapter 6]. It should be the case that since DEL and Semi-Public Environments share (almost) the same class of static models, and program models have the capacity for both epistemic and ontic change, all reasoning performed in AC is also possible in semi-public environments. This would be true, were it not for the fact that public announcements in our setting only involve term formulas. We could start by translating action points to program points with empty toggle sets. But action models have the flexibility of having any kind of indistinguishability relation between their action points and by making all program points have empty toggle sets we lose that flexibility. It is then, that one can decide to add 'dummy' variables to the toggle sets, mentioned only therein, and fix the vision functions appropriately, so as to achieve the

indistinguishability relation one wants. Regarding the common trait of action and program models, that is, their ability to change the knowledge of an agent: in epistemic models the worlds represent the agent's epistemic alternatives. By using the preconditions of action or program models, one can eliminate alternatives by making worlds 'vanish'. But with program models one can eliminate alternatives also by changing the worlds themselves. The exact relation between the effects of these two approaches, and always within a multi-agent environment, is worth investigating both from a technical and philosophical aspect.

In [47] Levesque gives a version of 'only-knowing' using 'knowing-at-least' and 'knowing-at-most' modalities. $V_i x$ can be thought of as "agent i 'knows-at-least' formula $K_i x \vee K_i \neg x$". This also brings in mind the possibility to implement vision for other formulas as well, so we would have $V_i \varphi$ where φ is not necessarily a (propositional) variable.

In the work of [40] agents can *see* some of the propositional variables (as in our paper), but in addition they can *control* some, i.e., for each variable there is an agent who controls its truth-value. In fact, this work sits in a sequence of papers where the notion of control, and also that of the *transfer* of control is studied ([32, 38, 41]). It is interesting to note here that, no matter whether such an approach includes an epistemic component or not, the dynamics of those systems can be modelled using our program models.

The Situation Calculus studies scenarios like the ones we presented here ([52], and many related papers). Although the language and models in this line of work are similar to ours, more work is needed to establish the precise technical relationships.

It may be tempting to think that there is a connection between our notion of *vision* of variables, and the notion of *awareness*, which, in the context of epistemic logic, dates back to [27] (see also [54] for a contemporary overview). However, there are major differences. For an agent i who is not aware of x, the formula $K_i(x \vee \neg x)$ would typically be false. Compare that to an agent who does not know, and hence does not see the value of x. For such an agent, in our set-up, $K_i(x \vee \neg x)$ would hold. The agent is aware of x, or, put differently, the agent knows *about* x. Of course, the well-informed agent is he who even knows the value of x (possibly because it is seen by the agent): $(K_i x \vee K_i \neg x)$. In this case, the agent knows the value of x, or, the agent knows *whether* x.

Chapter 3

The Logic DELVO

3.1 Preliminaries

Throughout the chapter, we will be dealing with a finite set $Ag = \{1, \dots, m\}$ of agents, and a countable set of propositional variables $\mathcal{V}ar = \{x_1, x_2, \dots\}$. A valuation $\theta : \mathcal{V}ar \to \{true, false\}$ assigns a truth value to each variable $x \in \mathcal{V}ar$. The set of all valuations is denoted by Θ. Given a relation $R \subseteq A \times A$, $R^{\sim}$ is its reflexive, symmetric, and transitive closure. For $S \subseteq A$, $R(S) = \{w \in A \mid \exists v \in S \; s.t. \; vRw\}$.

Let us give a summary of the notation for the different classes of models we will be using, given that several logics are involved in this study.

	DEL	SPE	DELVO	PDL	PDLVE
Static Models	$\mathcal{EM}$	$\mathcal{EMV}$	$\mathcal{EMV}$	$\mathcal{DM}$	$\mathcal{DEM}$
Dynamic Models	$\mathbb{AM}$	$\mathbb{PM}$	$\mathbb{AMT}$	–	–

A model M from any of the above classes, has a set of worlds/points W. We then have *single-pointed models* to be pairs of the form (M, w), for $w \in W$, and *multi-pointed models* to be pairs of the form (M, S), for $S \subseteq W$. We will use the prefix σ or μ, and the name of a class in brackets, to denote the corresponding classes of single-pointed and multi-pointed models (e.g., $\mu(\mathbb{AM})$ stands for multi-pointed action models). To justify the abbreviations: $\mathcal{EM}$ stands for Epistemic Models, $\mathcal{EMV}$ for the Epistemic Models with Vision, $\mathcal{DM}$ for Dynamic Models and $\mathcal{DEM}$ for Dynamic Epistemic Models, $\mathbb{AM}$ for Action Models, $\mathbb{AMT}$ for Action Models with Toggle sets, $\mathbb{PM}$ for Program Models.

We recapitulate the basic notions of DEL, we refer the reader to [19, Ch. 6] for a comprehensive exposition. Let $\mathcal{L}$ be a logical language for given parameters Ag and $\mathcal{V}ar$. An **action model** M is a tuple $\langle \mathsf{W}, \mathsf{R}, \mathsf{pre} \rangle$ such that W is a finite set, $\mathsf{R} : Ag \to 2^{(\mathsf{W} \times \mathsf{W})}$ assigns an equivalence relation to each agent $i \in Ag$, and $\mathsf{pre} : \mathsf{W} \to \mathcal{L}$. We will denote the class of action models for $\mathcal{L}_{\mathrm{DEL}}$ with $\mathbb{AM}$. Language $\mathcal{L}_{\mathrm{DEL}}$ is given by syntax $\varphi ::= x \mid \neg\varphi \mid \varphi \wedge \varphi \mid K_i\varphi \mid [a]\varphi$, where $x \in \mathcal{V}ar, i \in Ag$ and $a ::= (\mathsf{M}, \mathsf{w}) \mid (a \cup a)$ where $w \in \mathsf{W}$. An **epistemic model** for $\mathcal{L}_{\mathrm{DEL}}$ is $M = \langle W, R, f \rangle$, where W is a (possibly empty) set, the set of *states*; $f : W \to \Theta$ assigns a valuation θ to each state in W; $R : Ag \to 2^{(W \times W)}$ assigns an equivalence relation to each agent $i \in Ag$. $\mathcal{EM}$

denotes the class of epistemic models. The model resulting from the application of an action model M to an epistemic model M, referred to here as **model product**, is the model $M \otimes \mathsf{M} = \langle W', R', f' \rangle$ such that $W' = \{(w, \mathsf{w}) \in W \times \mathsf{W} \mid M, w \models \mathsf{pre(w)}\}$, $(w, \mathsf{w})R'(u, \mathsf{u})$ iff wRu and wRu. and $f(w, \mathsf{w}) = f(w)$. Actions of DEL are defined by syntax $a ::= (M, w) \mid a \cup a$. Finally, for each action a, we have the **induced relation** $[\![a]\!]$: for $\mathsf{M}, \mathsf{w} \in \sigma(\mathbb{AM})$ we have $(M, w)[\![\mathsf{M}, \mathsf{w}]\!](M', w')$ iff $M, w \models \mathsf{pre(w)}$ and $(M', w') = (M \otimes \mathsf{M}, (w, \mathsf{w}))$, while $[\![a \cup b]\!] = [\![a]\!] \cup [\![b]\!]$. We have $M, w \models [a]\varphi$ iff for all $M', w' \in [\![a]\!](M, w)$ we have $M', w' \models \varphi$. The rest of the truth definitions are as usual.

Let us denote the actions of DEL with $\mathcal{L}_1^{Act}$. In the remainder of the section we define an equivalent presentation of DEL based on multi-pointed action models. This allows us to have a unified model structure to handle all actions of $\mathcal{L}_1^{Act}$, which facilitates later comparison with SPE. So let us consider the language $\mathcal{L}_2^{Act}$ where actions are only multi-pointed action models, that is, $a ::= \mathsf{M}, \mathsf{S}$ where $\mathsf{M}, \mathsf{S} \in \mu(\mathbb{AM})$; and the language of DEL using $\mathcal{L}_2^{Act}$ denoted by $\mathcal{L}'_{\mathrm{DEL}}$. Now the induced relation is defined as $[\![\mathsf{M}, \mathsf{S}]\!](M, w) = \{M \otimes \mathsf{M}, (w, \mathsf{w}) \mid \mathsf{w} \in \mathsf{S}\}$.

In what follows, "$\sqcup$" denotes the disjoint union of two action models, whose sets of worlds are, for simplicity, already assumed to be disjoint.

Definition 3.1 (Translation to $\mu(\mathbb{AM})$)**.** We define function $t : \mathcal{L}_1^{Act} \mapsto \mathcal{L}_2^{Act} = \mu(\mathbb{AM})$ as follows: If $\mathsf{M}, \mathsf{w} \in \sigma(\mathbb{AM})$, then $t(\mathsf{M}, \mathsf{w}) = \mathsf{M}, \{\mathsf{w}\}$. If $a, b \in \mathcal{L}_1^{Act}$ and $t(a) = \mathsf{M}_a, \mathsf{S}_a$, $t(b) = \mathsf{M}_b, \mathsf{S}_b$, then $t(a \cup b) = \mathsf{M}_a \sqcup \mathsf{M}_b, \mathsf{S}_a \sqcup \mathsf{S}_b$.

The fact that this translation is truth-preserving is formally stated in the following Theorem.

Theorem 3.2. *For all $\varphi \in \mathcal{L}_{DEL}$ let $t(\varphi)$ denote the formula that is φ but for each action a all its occurrences in φ are replaced by $t(a)$. For all $M, w \in \sigma(\mathcal{EM})$ and $\varphi \in \mathcal{L}_{DEL}$, we have $M, w \models \varphi$ iff $M, w \models t(\varphi)$.*

Proof. The proof is by induction on the complexity of φ. The base case and inductive step for all cases where φ is not of the form $[a]\psi$ is straightforward. We elaborate on the case where $\varphi = [a]\psi$.

We want to prove that, if the induction hypothesis holds for ψ (i.e. $M, w \models \psi$ iff $M, w \models t(\psi)$), the same is true for $[a]\psi$ (i.e. $M, w \models [a]\psi$ iff $M, w \models [t(a)]t(\psi)$) for all a. We prove this by induction on the complexity of a.

If $a = \mathsf{M}, \mathsf{w}$ we have $M, w \models [a]\psi \Leftrightarrow M, w \models [\mathsf{M}, \mathsf{w}]\psi \Leftrightarrow M \otimes \mathsf{M}, (w, \mathsf{w}) \models \psi \overset{\text{ind.hyp.}}{\Longleftrightarrow} M \otimes \mathsf{M}, (w, \mathsf{w}) \models t(\psi) \Leftrightarrow [\![\mathsf{M}, \mathsf{w}]\!](M, w) \models t(\psi) \Leftrightarrow M, w \models [\mathsf{M}, \{\mathsf{w}\}]t(\psi) \Leftrightarrow M, w \models [t(a)]t(\psi)$.

If $a = b \cup c$ we have $M, w \models [a]\psi$ iff $(M, w \models [b]\psi$ and $M, w \models [c]\psi)$. By the induction hypothesis for actions of less complexity than a, the last clause is equivalent to $(M, w \models [t(b)]t(\psi)$ and $M, w \models [t(c)]t(\psi))$, which by definition is equivalent to $[\![\mathsf{M}_b, \mathsf{S}_b]\!](M, w) \models t(\psi)$ and $[\![\mathsf{M}_c, \mathsf{S}_c]\!](M, w) \models t(\psi)$. By a known theorem on disjoint union ([11], p. 261) this is equivalent to $[\![\mathsf{M}_b \sqcup \mathsf{M}_c, \mathsf{S}_b \sqcup \mathsf{S}_c]\!](M, w) \models t(\psi)$. But $\mathsf{M}_b \sqcup \mathsf{M}_c, \mathsf{S}_b \sqcup \mathsf{S}_c = t(a)$ and so $(M, w) \models [t(a)]t(\psi)$. □

3.2 The Logic DELVO

The first traditional setting on which we would like to expand and incorporate both alternative versions of choice and composition mentioned in the introductory section is DEL. The difference in behaviour we have observed relies on vision—that is, what agents can observe—and ontic change—that is, the 'factual' differences among points of evaluation in the model. It thus seems reasonable to introduce a DEL-style logic, with epistemic models enriched with vision, and action models that allow for ontic change.

Definition 3.3 (DELVO Action Models)**.** A (DELVO) action model is a tuple $\mathsf{M} = \langle \mathsf{W}, \mathsf{R}, \mathsf{pre}, \mathsf{tgl} \rangle$, where $\mathsf{W} \neq \emptyset$ is finite, $\mathsf{R} : Ag \to 2^{(\mathsf{W} \times \mathsf{W})}$ assigns an equivalence relation to each agent $i \in Ag$, $\mathsf{pre} : \mathsf{W} \to \mathcal{L}$, $\mathsf{tgl} : \mathsf{W} \to \mathcal{P}(\mathcal{V}ar)$ and each $\mathsf{tgl}(\mathsf{w})$ is finite. A pair (M, S), where M is an action model and $\emptyset \subset \mathsf{S} \subseteq \mathsf{W}$, is called a *(Multi-)Pointed Action Model*. When clear from context, instead of $\mathsf{M}, \{\mathsf{w}\}$ we may also write M, w. The class of all action models is denoted by $\mathbb{AMT}$.

In comparison with the standard manifestation of post-condition effects in DEL, which uses assignments (see for example [8]), our action models use toggle sets, as in [34]. The reason is that it allows for a more elegant handling of the notion of vision, since the agent successfully resolves uncertainty based on the *difference in ontic changes* caused by the actions. The two formulations are technically equivalent: one can simulate 'assignments' to variables using toggle sets [34]. Pairs (M, S) now are essentially the multi-pointed action models of DEL, but with each point w carrying additional information about a finite toggle set $\mathsf{tgl}(\mathsf{w})$. Finally the **syntax** of formulas $\varphi \in \mathcal{L}_{\text{DELVO}}$ is:

$$\varphi ::= \top \mid x \mid V_i x \mid \neg\varphi \mid \varphi \wedge \varphi \mid [\mathsf{M}, \mathsf{S}]\varphi \mid K_i\varphi$$

where $x \in \mathcal{V}ar$, $i \in Ag$, $(\mathsf{M}, \mathsf{S}) \in \mu(\mathbb{AMT})$.

Definition 3.4 (DELVO models)**.** An *epistemic model* (with vision) M for $\mathcal{L}_{\text{DELVO}}$ is a tuple $M = \langle W, R, V, f \rangle$ where $\langle W, R, f \rangle \in \mathcal{EM}$, $V : Ag \to 2^{\mathcal{V}ar}$, and

- wR_iu implies $(f(w) \triangle f(u)) \cap V(i) = \emptyset$.

$\mathcal{EMV}$ denotes the class of epistemic models with vision. The set of all vision functions V is denoted by $\mathcal{V}is$.

So the models of DELVO are standard epistemic models enhanced with a vision function recording information about which propositional variables each agent observes. If an agent considers two worlds indistinguishable, then it must be the case that he cannot *observe* a difference in their valuation.

There exists the possibility that the indistinguishability relation of an action model is not compatible with the vision function of an epistemic model. The way we address these cases when trying to calculate a model product, is by 'rewiring' the action models to be 'in sync' with the agents' vision, but without them hiding any information at all, or giving information not acquired by vision.

Definition 3.5. Let $V \in \mathcal{V}is$ and $\mathsf{M} \in \mathbb{AMT}$. We define the action model $\mathsf{M}^V = \langle \mathsf{W}^V, \mathsf{R}^V, \mathsf{pre}^V, \mathsf{tgl}^V \rangle$ as follows:

- $\mathsf{W}^V = \mathsf{W}$; $\mathsf{pre}^V = \mathsf{pre}$; $\mathsf{tgl}^V = \mathsf{tgl}$;
- $\mathsf{w}\mathsf{R}_i^V\mathsf{u}$ iff $\big(\mathsf{w}\mathsf{R}_i\mathsf{u}$ and $\big(\mathsf{tgl}(\mathsf{w}) \triangle \mathsf{tgl}(\mathsf{u})\big) \cap V(i) = \emptyset\big)$.

Now that we have everything else in place, we proceed to define the semantics of formulas and actions in DELVO.

Definition 3.6. Let $M = \langle W, R, V, f \rangle \in \mathcal{EMV}$. The truth conditions are as for DEL, plus $(M, w) \models V_i x$ iff $x \in V(i)$.

Relation $[\![\mathsf{M}, \mathsf{S}]\!] \subseteq \sigma(\mathcal{EMV}) \times \sigma(\mathcal{EMV})$ is defined again as in DEL: $[\![\mathsf{M}, \mathsf{S}]\!](M, w) = \{M \otimes \mathsf{M}, (w, \mathsf{w}) \mid \mathsf{w} \in \mathsf{S}\}$, while $(M \otimes \mathsf{M})$ is now the epistemic model $M' = \langle W', R', V', f' \rangle$ defined as:

- $W' = \{(w, \mathsf{w}) \mid w \in W, \mathsf{w} \in \mathsf{W}$ & $(M, w) \models \mathsf{pre}(\mathsf{w})\}$;
- $(w, \mathsf{w})R_i'(u, \mathsf{u})$ iff wR_iu and $\mathsf{w}\mathsf{R}_i^V\mathsf{u}$;
- $V' = V$, $f'((w, \mathsf{w})) = \updownarrow\big(\mathsf{tgl}(\mathsf{w})\big)\big(f(w)\big)$.

We conclude the section thus far by introducing a notion of *bisimulation* between DELVO models [12].

Definition 3.7. Let $M, M' \in \mathcal{EMV}$. A non-empty relation $Z \subseteq W \times W'$ is a bisimulation iff for all $(w, w') \in Z$ and $i \in Ag$:

Atoms For all $x \in Var$, $x \in f(w)$ iff $x \in f'(w')$, and $x \in V(i)$ iff $x \in V'(i)$;

Forth For all $v \in W$, if wR_iv then there is a $v' \in W'$ such that $w'R_i'v'$ and $(v, v') \in Z$;

Back For all $v' \in W'$, if $w'R_i'v'$ then there is a $v \in W$ such that wR_iv and $(v, v') \in Z$.

Two multi-pointed action models (M, S) and (M', S') are bisimilar (written $(M, S) \leftrightarrows (M', S')$) iff there is a bisimulation Z between M and M', such that for any $w \in S$, there is a $w' \in S'$ with $(w, w') \in Z$, and vice versa.

3.3 Axiomatisation of DELVO

The axiomatisation of DELVO is comprised of the axioms and rules for logic S5, axioms for vision $V_ix \to (K_ix \vee K_i\neg x)$ and $V_ix \to K_jV_ix$, and reduction axioms that allow any formula that includes dynamic modalities to be reduced to one that does not. These reduction axioms are identical to those of SPE [34], except the slightly changed reduction action for knowledge: $[\mathsf{M}, \mathsf{w}]K_i\varphi \leftrightarrow \big(\mathsf{pre}(\mathsf{w}) \to \bigwedge_{V \in \mathcal{V}is^{\mathsf{M}}}(\chi_V \to \bigwedge_{\mathsf{w}\mathsf{R}_i^V\mathsf{u}} K_i[\mathsf{M}^{\mathsf{V}}, \mathsf{u}]\varphi)\big)$, with χ_V being a characteristic formula of vision function V over the variables that appear in M. Soundness is derived easily from the definitions, and completeness is proven along the lines of DEL and SPE via the standard reduction technique; completeness is then further reduced to that of logic S5+V, that is, the multi-agent epistemic logic $\mathrm{S5}_n$ [28], with the two additional axioms for vision.

3.4 Opaque Choice and Composition

The logic DELVO provides us with a setting in which to properly capture not only the traditional forms of choice and composition (cf. Theorem 3.2) but also their *opaque* variants—opaque choice and composition—we motivated in the introductory section.

Let the set of actions Act be defined inductively as:
$Act \ni a ::= \varphi! \mid \varphi? \mid \updownarrow x \mid a \cup a \mid a \uplus_T a \mid a; a \mid a;; a,$
where $\varphi \in \mathcal{L}$, $x \in \mathcal{V}ar$ and $T \subseteq Ag$.

The constructs $\uplus_T$ and $;;$ denote the opaque variants of the standard $\cup$ and $;$. For $T = Ag$ we write $\uplus$ omitting the subscript. Each of the above actions we recursively define in terms of suitable DELVO action models. By writing action "a" in a formula we will mean the multi-pointed action model that corresponds to a.

Definition 3.8.

1. (Announcement) Let $\mathsf{M}_{\varphi!} = \langle \mathsf{W}, \mathsf{R}, \mathsf{pre}, \mathsf{tgl} \rangle$, where $\mathsf{W} = \{\mathsf{e}\}$, $\mathsf{R}_i = \{(\mathsf{e}, \mathsf{e})\}$, $\mathsf{pre}(\mathsf{e}) = \varphi$ and $\mathsf{tgl}(\mathsf{e}) = \{\}$. The multi-pointed action model for $\varphi!$ is $\mathsf{M}_{\varphi!}, \{\mathsf{e}\}$.
2. (Toggle) Let $\mathsf{M}_{\updownarrow x} = \langle \mathsf{W}, \mathsf{R}, \mathsf{pre}, \mathsf{tgl} \rangle$, where $\mathsf{W} = \{\mathsf{t}\}$, $\mathsf{R}_i = \{(\mathsf{t}, \mathsf{t})\}$, $\mathsf{pre}(\mathsf{t}) = \top$ and $\mathsf{tgl}(\mathsf{t}) = \{x\}$. The multi-pointed action model for $\updownarrow x$ is $\mathsf{M}_{\updownarrow x}, \{\mathsf{t}\}$.
3. (Test) Let $\mathsf{M}_{\varphi?} = \langle \mathsf{W}, \mathsf{R}, \mathsf{pre}, \mathsf{tgl} \rangle$, where $\mathsf{W} = \{\mathsf{w}, \mathsf{v}\}$, $\mathsf{R}_i = \mathsf{W} \times \mathsf{W}$, $\mathsf{pre}(\mathsf{w}) = \varphi$, $\mathsf{pre}(\mathsf{v}) = \neg\varphi$ and $\mathsf{tgl}(\mathsf{w}) = \mathsf{tgl}(\mathsf{v}) = \{\}$. The multi-pointed action model for $\varphi?$ is $\mathsf{M}_{\varphi?}, \{\mathsf{w}\}$.

In order to define test as a multi-pointed action model, we have based ourselves on the archetype that the test relation represents, in PDL models. Specifically, given a PDL model M, then $[\![\varphi?]\!] = \{(w, w) \in M \times M \mid M, w \models \varphi\}$. The relation induced by action models (in our case Def. 3.6) is not between worlds of the same model, but between the original epistemic model, and the model product. The reader can check that with the action model for test in Definition 3.8, we have, given $M, w \in \sigma(\mathcal{EMV})$, if $M, w \not\models \varphi$ then $[\![\varphi?]\!](M, w) = \emptyset$, and if $M, w \models \varphi$ then $[\![\varphi?]\!](M, w) = M \otimes \mathsf{M}_{\varphi?}, (w, \mathsf{w})$, and M, w is isomorphic to $M \otimes \mathsf{M}_{\varphi?}, (w, \mathsf{w})$.

The action model for standard non-deterministic choice is a disjoint union of action models, as in DEL (Def. 3.1).

Definition 3.9 (Opaque Non-det. Choice)**.** Let $a, b \in Act$. The action model for $a \uplus_T b$ is $\mathsf{M}_{a \uplus_T b}, \mathsf{S}_{a \uplus_T b} = \langle \mathsf{W}', \mathsf{R}', \mathsf{pre}', \mathsf{tgl}' \rangle, \mathsf{S}'$ where, if $\mathsf{M}_{a \cup b}, \mathsf{S}_{a \cup b} = \langle \mathsf{W}, \mathsf{R}, \mathsf{pre}, \mathsf{tgl} \rangle, \mathsf{S}$:

- $\mathsf{W}' = \mathsf{W}$; $\mathsf{pre}' = \mathsf{pre}$; $\mathsf{tgl}' = \mathsf{tgl}$; $\mathsf{S}' = \mathsf{S}$
- For $i \in T$, $\mathsf{R}'_i = (\ \mathsf{R}_i \cup (\mathsf{S}' \times \mathsf{S}')\ )^{\sim}$. For $i \notin T$, $\mathsf{R}'_i = \mathsf{R}_i$.

The intuition behind opaque choice compared to standard choice, is that, for agents in T, it 'overrides' any information gained by the distinction between points in S. It can be read as "non-deterministically execute one of the action points in S, but, after execution you will only be able to tell between them if your vision allows it". To do this, we simply take the action model of the standard non-deterministic choice, and

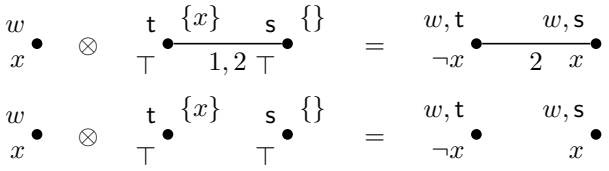


FIGURE 3.1: Epistemic model M (left) for which $x \in V(1), x \notin V(2)$, action models $(\updownarrow x \uplus \top!)$ (top) and $(\updownarrow x \cup \top!)$ (bottom), and the resulting model products (right).

then connect, for the agents in T, the worlds in S. As mentioned earlier, an agent's uncertainty is bound by the information it acquires by vision (recall $M \otimes \mathsf{M}$ in Def. 3.6).

Example 3.1. Consider the epistemic model M with a single world w with $x \in V(1), x \notin V(2)$. In Figure 3.1 we compare the action models $\updownarrow x \cup \top!$ and $\updownarrow x \uplus \top!$, along with the resulting model products. Note that for opaque choice, $M, w \models [\updownarrow x \uplus \top!]\neg K w_2 x$ while for standard choice $M, w \models [\updownarrow x \cup \top!] K w_2 x$. And for agent 1 that 'sees' x we have $M, w \models [\updownarrow x \uplus \top!] K w_1 x \wedge [\updownarrow x \cup \top!] K w_1 x$.

Example 3.1 demonstrates that *both* directions of validity (1.1) fail for opaque choice. Indeed, we have $M, w \models [\updownarrow x] K w_2 x \wedge [\top!] K w_2 x$ but $M, w \models \neg[\updownarrow x \uplus \top!] K w_2 x$, and similarly $M, w \models [\updownarrow x \uplus \top!]\neg K w_2 x$ but $M, w \models \neg[\updownarrow x]\neg K w_2 x$. The uncertainty of the agent described by the intuition above and portrayed by this example can be described more generally by the following formula: let $(\mathsf{M}, \mathsf{w}), (\mathsf{M}', \mathsf{w}') \in \mu(\mathbb{AMT})$ and V be the vision function for which agent i can see none of the variables in $\mathsf{tgl}(\mathsf{w}) \triangle \mathsf{tgl}(\mathsf{w}')$. Then we have:

$$\left.\begin{array}{l} \chi_V \wedge [\mathsf{M}, \mathsf{w}] x \wedge [\mathsf{M}', \mathsf{w}']\neg x \\ \wedge(\mathsf{pre}(\mathsf{w}) \leftrightarrow \mathsf{pre}(\mathsf{w}')) \end{array}\right\} \rightarrow [\mathsf{M}, \mathsf{w} \uplus \mathsf{M}', \mathsf{w}']\neg K w_i x \tag{3.1}$$

Example 3.2. Consider action $\alpha = \varphi! \uplus_1 \neg\varphi!$. The multi-pointed action model for α is $(\mathsf{M}_\alpha, \mathsf{S}_\alpha)$, where $\mathsf{W} = \{\mathsf{e}_1, \mathsf{e}_2\}$, $\mathsf{R}_1 = W \times W$, $\mathsf{R}_2 = \{(\mathsf{e}_1, \mathsf{e}_1), (\mathsf{e}_2, \mathsf{e}_2)\}$, $\mathsf{pre}(\mathsf{e}_1) = \varphi$, $\mathsf{pre}(\mathsf{e}_2) = \neg\varphi$, $\mathsf{tgl}(\mathsf{e}_1) = \mathsf{tgl}(\mathsf{e}_2) = \{\}$, and $\mathsf{S} = \mathsf{W}$.

In words, the action would be "either it is announced that φ or it is announced that $\neg\varphi$, agent 1 does not 'hear' the announcement, while agent 2 does", and both agents know that this executed. It comes as no surprise that (for all $(M, w) \in \sigma(\mathcal{EMV})$) we have $M, w \models K_2 \neg K w_1 \varphi$. At the same time and when restricted to R_1, we have that $\mathsf{M}_\alpha = \mathsf{M}_{\varphi?}$, but with the difference that $\mathsf{S}_\alpha \neq \mathsf{S}_{\varphi?}$. It is easily understood that such a difference is not to be overlooked. For example, when $\varphi = x$ we have $M, w \models \neg x \rightarrow [x?]\bot$, whereas $M, w \not\models \neg x \rightarrow [a]\bot$.

Let us further comment on Def 3.9 by noticing that combining two multi-pointed action models with opaque choice, means that any previous indistinguishability relation between the points the opaque choice affects are ignored. As an example consider three action models, $(\mathsf{M}_1, \{\mathsf{w}_1\})$,$(\mathsf{M}_2, \{\mathsf{w}_2\})$,$(\mathsf{M}_3, \{\mathsf{w}_3\})$. We have that $(\mathsf{M}_1, \{\mathsf{w}_1\} \cup \mathsf{M}_2, \{\mathsf{w}_2\}) \uplus \mathsf{M}_3, \{\mathsf{w}_3\} = \mathsf{M}_1, \{\mathsf{w}_1\} \uplus \mathsf{M}_2, \{\mathsf{w}_2\} \uplus \mathsf{M}_3, \{\mathsf{w}_3\}$. This might seem unintuitive at first but is to be expected, if opaque choice is to stray true to its archetype (to connects points in action models) and we consider that indistinguishability relations that are transitive.

There is no way to keep w_1 and w_2 not connected (the intention of $\mathsf{M}_1, \{\mathsf{w}_1\} \cup \mathsf{M}_2, \{\mathsf{w}_2\}$) while we connect both with w_3. The standard choice still retains its characteristic to keep things disconnected in the absence of a *later* application of opaque choice: for example note that $\mathsf{M}_1, \{\mathsf{w}_1\} \uplus \mathsf{M}_2, \{\mathsf{w}_2\} \uplus \mathsf{M}_3, \{\mathsf{w}_3\} \neq (\mathsf{M}_1, \{\mathsf{w}_1\} \uplus \mathsf{M}_2, \{\mathsf{w}_2\}) \cup \mathsf{M}_3, \{\mathsf{w}_3\}$.

We now move on to the subject of sequential composition, which poses some more difficulties. It becomes evident that it is not possible to define regular composition only as a function of the models $\mathsf{M}_a, \mathsf{M}_b$, without vision as a parameter; this is in contrast with the action models and the other constructs defined so far. As proof recall the actions of Example 1.1: if $V(i) = \emptyset$ then $\mathsf{v}_j \mathsf{R}_i \mathsf{v}_k$ for $1 \leq j, k \leq 4$ in the action model, and for the resulting worlds also $(w, \mathsf{v}_j) R_i (w, \mathsf{v}_k)$. If we set $V(i) = \{x\}$ then the action model would be the same but it should not be the case e.g. that $(w, \mathsf{v}_2) R_i (w, \mathsf{v}_3)$. We are therefore led to the following definition.

Definition 3.10 (Composition)**.** Let $V \in \mathcal{V}is$, $\mathsf{M}_a = \langle \mathsf{W}_a, \mathsf{R}_a\ , \mathsf{pre}_a, \mathsf{tgl}_a \rangle, \mathsf{S}_a$ and $\mathsf{M}_b = \langle \mathsf{W}_b, \mathsf{R}_b, \mathsf{pre}_b, \mathsf{tgl}_b, \rangle, \mathsf{S}_b$. Their composition w.r.t. V is the pointed action model $\mathsf{M}^V_{a;b}, \mathsf{S}^V_{a;b} = \langle \mathsf{W}, \mathsf{R}, \mathsf{pre}, \mathsf{tgl} \rangle, \mathsf{S}$ where: $\mathsf{W} = \mathsf{W}_a \times \mathsf{W}_b$, $\mathsf{S} = \mathsf{S}_a \times \mathsf{S}_b$,

- $(\mathsf{w}, \mathsf{w}') \mathsf{R}_i (\mathsf{v}, \mathsf{v}')$ iff $\mathsf{w} \mathsf{R}^V_{a_i} \mathsf{v}$ & $\mathsf{w}' \mathsf{R}^V_{b_i} \mathsf{v}'$;
- $\mathsf{pre}(\mathsf{w}, \mathsf{v}) = \langle \mathsf{M}_a, \mathsf{w} \rangle \mathsf{pre}_b(\mathsf{v})$, $\mathsf{tgl}(\mathsf{w}, \mathsf{v}) = \mathsf{tgl}_a(\mathsf{w}) \triangle \mathsf{tgl}_b(\mathsf{v})$

We need not advocate the correctness of the definition regarding W, pre, tgl, and S, as they are what is expected for the composition of two action models in DEL as well as SPE. Vision influences only the accessibility relation R, and regarding that we need only note that it is again the one normally expected, with the exception that the agent has the opportunity to apply his vision after execution of actions a and b. Hence $\mathsf{R}^V_{a_i}$ is used (Def. 3.5) and not R_{a_i}. Finally, let us point out that with this definition a vision function V is linked to an action model directly, and an epistemic model M may have a vision function $V' \neq V$. In that case, a formula using such an action model cannot be interpreted in M.

We do not run into the same kind of problem with opaque composition as it uses the same idea behind opaque choice: it disregards previous connections (or more precisely, the lack thereof) in the model. Intuitively speaking, the agent is absent while actions a and then b are executed, 'missing' any announcements made and also the opportunity to apply his vision as changes occur to the variables.

Definition 3.11 (Opaque Composition)**.** Let $a, b \in A$. The pointed action model for $a;;b$ is $\mathsf{M}_{a;;b}, \mathsf{S}_{a;;b} = \langle \mathsf{W}, \mathsf{R}, \mathsf{pre}, \mathsf{tgl} \rangle, \mathsf{S}$ defined as in Definition 3.10 but $\mathsf{R}_i = \mathsf{W} \times \mathsf{W}$.

We can now see how to falsify property (2) from the introduction. Consider again the epistemic model of Example 3.1. Action model for $(\updownarrow x \cup \top!);;\top!$ is the action model of $(\updownarrow x \uplus \top!)$, found in the first row of Figure 3.1, while in the second row we have the action model of $(\updownarrow x \cup \top!)$ and the resulting epistemic model. Applying action $\top!$ does not change anything, and so $M, w \models \neg[(\updownarrow x \cup \top!);;\top!] K w_2 x$, while $M, w \models [\updownarrow x \cup \top!][\top!] K w_2 x$.

Before moving on to the next section, let us take the opportunity to make the distinction between the set of multi-pointed actions models that correspond to the set of actions *Act* defined in the beginning of this section, and $\mu(\mathbb{AMT})$, that is in general used in DELVO. In fact, the latter is strictly larger than the former. To see this consider an action model as follows: $(\mathsf{M}_\alpha, \mathsf{S}_\alpha)$, where $\mathsf{W} = \{\mathsf{w}_1, \mathsf{w}_2\}$, $\mathsf{R} = W \times W$, $\mathsf{pre}(\mathsf{w}_1) = x$, $\mathsf{pre}(\mathsf{w}_2) = y$, $\mathsf{tgl}(\mathsf{w}_1) = \mathsf{tgl}(\mathsf{w}_2) = \{\}$, and $\mathsf{S} = \{\mathsf{w}_1\}$. Any attempt to build this model inductively using actions *Act* would have to introduce point w_2 that has $\mathsf{pre}(\mathsf{w}_2) = y$. This could be done either through $y?$ or $y!$. In the first case precondition $\neg y$ would have to appear somewhere, but it does not. In the second case in order to include this point with precondition y in M, it means we would have to include the same point in S. But this does not happen either.

3.5 Relation to DEL and SPE

Not coincidentally, the class of DELVO (static) models—i.e. epistemic models with vision—is exactly the class of models of SPE, and subsumes the class of standard epistemic models (without vision, that is, where vision is empty), which are the models of DEL. In this section we define the relevant translations among these classes of models and state, without proof, and state the two truth-preserving embeddings.

Definition 3.12. We define function $h : \mathcal{EM} \to \mathcal{EMV}$ as follows: Let $M = \langle W, R, f\rangle \in \mathcal{EM}$. Then $h(M) = \langle W, R, V, f\rangle$, where V is such that for all $i \in \mathcal{AG}$, $V(i) = \emptyset$.

Definition 3.13. For all $\varphi \in \mathcal{L}'_{DEL}$ let $g(\varphi)$ denote the formula that is φ but for each action (M, S) all its occurrences in φ are replaced by $g(\mathsf{M}, \mathsf{S})$ as defined below.

Let $(\mathsf{M}, \mathsf{S}) = (\langle \mathsf{W}, \mathsf{R}, \mathsf{pre}\rangle, \mathsf{S})$. We define $g : \mu(\mathbb{AM}) \to \mu(\mathbb{AMT})$ as $g(\mathsf{M}, \mathsf{S}) = (\langle \mathsf{W}, \mathsf{R}, g(\mathsf{pre}), \mathsf{tgl}\rangle, \mathsf{S})$, where tgl is the function such that for all $\mathsf{w} \in \mathsf{W}$ it holds $\mathsf{tgl}(\mathsf{w}) = \emptyset$, and $g(\mathsf{pre})$ is a formula that has already been constructed in a previous stage of the inductively defined hierarchy.

Theorem 3.14. *Let $M, w \in \sigma(\mathcal{EM})$ and $\varphi \in \mathcal{L}'_{DEL}$. We have $M, w \models \varphi$ iff $h(M), w \models g(\varphi)$.*

Proof. The proof is by induction on the complexity of φ. Specifically, if the complexity of φ is n and the complexity of ψ is $k < n$, then the induction hypothesis is that, for all $M, w \in \sigma(\mathcal{EM})$, $M, w \models \psi$ iff $h(M), w \models g(\psi)$

Let $\varphi = x \in \mathcal{V}ar$. $M, w \models x$ iff $f(w)(x) = true$ iff $h(M), w \models x$. Also $\varphi = x = g(x) = g(\varphi)$ so $h(M), w \models g(\varphi)$.

Let $\varphi = \neg\psi$. $M, w \models \neg\psi$ iff $M, w \not\models \psi$. By induction hypothesis this is equivalent to $h(M), w \not\models g(\psi)$ and in turn this is equivalent to $h(M), w \models \neg g(\psi)$. We have that $\neg g(\psi) = g(\neg\psi)$ and so, equivalently, $h(M), w \models g(\varphi)$.

Let $\varphi = \psi_1 \wedge \psi_2$. $M, w \models \psi_1 \wedge \psi_2$ iff $M, w \models \psi_1$ and $M, w \models \psi_2$. By induction hypothesis this conjunction is equivalent to $h(M), w \models g(\psi_1)$ and $h(M), w \models g(\psi_2)$

and so $h(M), w \models g(\psi_1) \wedge g(\psi_2)$. We have that $g(\psi_1) \wedge g(\psi_2) = g(\psi_1 \wedge \psi_2)$ and so $h(M), w \models g(\varphi)$.

Let $\varphi = K_i\psi$. $M, w \models K_i\psi$ iff for all $v \in W$ such that wR_iv we have $M, v \models \psi$. By induction hypothesis, $M, v \models \psi$ iff $h(M), v \models g(\psi)$, and so $M, w \models K_i\psi$ iff for all $v \in W$ such that wR_iv we have $h(M), v \models g(\psi)$. Models M and $h(M)$ both have set of worlds W and accessibility relation R, so we have $h(M), w \models K_i g(\psi)$. Note that $K_i g(\psi) = g(K_i\psi)$. So equivalently we have $h(M), w \models g(\varphi)$.

Let $\varphi = [\mathsf{M}, \mathsf{S}]\psi$, where $\mathsf{M}, \mathsf{S} \in \mu(\mathbb{AM})$. $M, w \models [\mathsf{M}, \mathsf{S}]\psi$ iff for all $v \in [\![\mathsf{M}, \mathsf{S}]\!](M, w)$ we have $M \otimes \mathsf{M}, v \models \psi$. By induction hypothesis, $h(M \otimes \mathsf{M}), v \models g(\psi)$. Now note that $h(M \otimes \mathsf{M}), v = h(M) \otimes g(\mathsf{M}), v$. To further explain this, observe that the only difference in the model product definitions of DEL and DELVO (Def. 2.3 and 3.6), is the existence of a vision function V, vision's presence in $w\mathsf{R}_i^V v$, and the effect of tgl in the clause $f'((w, \mathsf{w})) = \updownarrow\big(\mathsf{tgl}(\mathsf{w})\big)\big(f(w)\big)$. In both cases we have the same V ($V(i) = \emptyset$ for all i). Also $wR_i^V v$ iff $\big(\mathsf{wR}_i\mathsf{u}$ and $\big(\mathsf{tgl}(\mathsf{w}) \triangle \mathsf{tgl}(\mathsf{u})\big) \cap V(i) = \emptyset\big)$, and because $V(i) = \emptyset$, we simply have $w\mathsf{R}_i^V v$ iff $w\mathsf{R}_i v$. Also $\mathsf{tgl}(\mathsf{w}) = \emptyset$ for all w, so $f'((w, \mathsf{w})) = \updownarrow\big(\mathsf{tgl}(\mathsf{w})\big)\big(f(w)\big) = f(w)$. So getting back to the initial equivalences: for all $v \in [\![\mathsf{M}, \mathsf{S}]\!](M, w)$ we have $M \otimes \mathsf{M}, v \models \psi$ iff for all $v \in [\![g(\mathsf{M}, \mathsf{S})]\!](h(M), w)$ we have $h(M \otimes g(\mathsf{M}), v) \models g(\psi)$. This means that $h(M), w \models [g(\mathsf{M}, \mathsf{S})]g(\psi)$, i.e., $h(M), w \models g(\varphi)$. □

We work similarly for SPE; its models are exactly those of DELVO, and its action models can be seen as a special case of DELVO action models.

Definition 3.15. We define function $g : \sigma(\mathbb{PM}) \mapsto \mu(\mathbb{AMT})$ as follows: Let $\mathsf{M} = \{w_1 = (\varphi_1, X_1), \ldots, w_n = (\varphi_n, X_n)\} \in \mathbb{PM}$ and $\mathsf{w} \in \mathsf{M}$. Then $g(\mathsf{M}, \mathsf{w}) = \langle \mathsf{W}, \mathsf{R}, \mathsf{pre}, \mathsf{tgl}\rangle, \{\mathsf{w}\}$ where $\mathsf{W} = \{\mathsf{w}_1, \ldots, \mathsf{w}_n\}, \mathsf{pre}(\mathsf{w}_i) = \varphi_i, \mathsf{tgl}(\mathsf{w}_i) = X_i$ and $\mathsf{R}_j = \mathsf{W} \times \mathsf{W}$.

Theorem 3.16. *For all $\varphi \in \mathcal{L}_{SPE}$ let $g(\varphi)$ denote the formula that is φ but for each action M, w all its occurrences in φ are replaced by $g(\mathsf{M}, \mathsf{w})$. Let $M, w \in \sigma(\mathcal{EMV})$ and $\varphi \in \mathcal{L}_{SPE}$. We have $M, w \models \varphi$ iff $M, w \models g(\varphi)$.*

Proof. We will be referring to the semantics of SPE in [34]. The proof is by induction on the complexity of φ. Specifically, if the complexity of φ is n and the complexity of ψ is $k < n$, then the induction hypothesis is that, for all $M, w \in \sigma(\mathcal{EMV})$, $M, w \models \psi$ iff $M, w \models g(\psi)$. Note that the base case as well as the induction steps, except for $\varphi = [\mathsf{M}, \mathsf{w}]\psi$ (where $\mathsf{M}, \mathsf{w} \in \sigma(\mathbb{PM})$), are similar to the respective parts in the proof of Theorem 3.14.

Let $\varphi = x \in \mathcal{V}ar$. $\varphi = x = g(x) = g(\varphi)$ so $M, w \models x$ iff $M, w \models g(\varphi)$. Similarly for $\varphi = V_i x$.

Let $\varphi = \neg\psi$. $M, w \models \neg\psi$ iff $M, w \not\models \psi$. By induction hypothesis this is equivalent to $M, w \not\models g(\psi)$ and in turn this is equivalent to $M, w \models \neg g(\psi)$. We have that $\neg g(\psi) = g(\neg\psi)$ and so, equivalently, $M, w \models g(\varphi)$.

Let $\varphi = \psi_1 \wedge \psi_2$. $M, w \models \psi_1 \wedge \psi_2$ iff $M, w \models \psi_1$ and $M, w \models \psi_2$. By induction hypothesis this conjunction is equivalent to $M, w \models g(\psi_1)$ and $M, w \models g(\psi_2)$ and so $M, w \models g(\psi_1) \wedge g(\psi_2)$. We have that $g(\psi_1) \wedge g(\psi_2) = g(\psi_1 \wedge \psi_2)$ and so $M, w \models g(\varphi)$.

Let $\varphi = K_i\psi$. $(M, w) \models K_i\psi$ iff for all $v \in W$ such that wR_iv we have $M, v \models \psi$. By induction hypothesis, $M, v \models \psi$ iff $M, v \models g(\psi)$, and so $M, w \models K_i\psi$ iff for all $v \in W$ such that wR_iv we have $M, v \models g(\psi)$ and so we have $M, w \models K_ig(\psi)$. Note that $K_ig(\psi) = g(K_i\psi)$. So equivalently we have $M, w \models g(\varphi)$.

Let $\varphi = [\mathsf{M}, \mathsf{S}]\psi$, where $\mathsf{M}, \mathsf{w} \in \sigma(\mathbb{PM})$. $M, w \models [\mathsf{M}, \mathsf{w}]\psi$ iff $(w, \mathsf{w}) \in M \otimes \mathsf{M}$ and $M \otimes \mathsf{M}, (w, \mathsf{w}) \models \psi$. By induction hypothesis, $M \otimes \mathsf{M}, (w, \mathsf{w}) \models g(\psi)$. Now note that $M \otimes \mathsf{M} = M \otimes g(\mathsf{M})$. To further explain this, observe that the only difference in the model product definitions of SPE and DELVO, is the clause for the accessibility relation: In SPE $(w, \mathsf{w})R'_i(v, \mathsf{v})$ iff wR_iu and $\mathsf{tgl(w)} \triangle \mathsf{tgl(u)} \cap V(i) = \emptyset$, while in DELVO, $(w, \mathsf{w})R'_i(v, \mathsf{v})$ iff wR_iu and wRv and $\mathsf{tgl(w)} \triangle \mathsf{tgl(u)} \cap V(i) = \emptyset$. But in our case, wRv is always true, and so the two clauses are equivalent. So getting back to the initial equivalences: $M \otimes \mathsf{M}, (w, \mathsf{w}) \models g(\psi)$ iff $M \otimes g(\mathsf{M}), (w, \mathsf{w}) \models g(\psi)$. By definition, this is equivalent to $M \models [g(\mathsf{M}), \mathsf{w}]g(\psi)$ and so $M \models g(\varphi)$. □

Chapter 4

Satisfiability for DEL and DELVO

In this chapter we will examine the computation complexity of the two main decision problems associated with DEL and DELVO, namely satisfiability and model checking. It is proven that the satisfiability problem is NEXPTIME-complete and the model checking problem is PSPACE-complete.

For both of these problems, we need a measure for the size of a formula $\varphi \in \mathcal{L}_{\mathrm{DEL}}$. But φ may contain action operators and so we need a size for them as well:

Definition 4.1. Let α be an action of DEL. Its size $|\alpha|$ is defined as:

- if $a = (\mathsf{M}, \mathsf{w})$ then $|a| = |\mathsf{M}|$;
- if $a = b \cup c$ then $|a| = 1 + |b| + |c|$

The size of an action model $\mathsf{M} = \langle \mathsf{W}, \mathsf{R}, \mathsf{pre} \rangle \in \mathbb{AM}$ is defined as

$$|\mathsf{M}| = card(\mathsf{W}) + \sum_{i \in Ag} card(\mathsf{R}_i) + \sum_{\mathsf{w} \in \mathsf{W}} |\mathsf{pre}(\mathsf{w})|$$

For $\varphi \in \mathcal{L}_{\mathrm{EL}}$ size is defined as usual; the sum of the symbols of the formula. For $\varphi \in \mathcal{L}_{\mathrm{DEL}}$ we need to define an extra induction case: $|[a]\varphi| = 1 + |a| + |\varphi|$.

The *satisfiability problem* of DEL is the decision problem defined as follows:

Input: A formula $\varphi \in \mathcal{L}_{\mathrm{DEL}}$

Output: YES iff φ is satisfiable, i.e., there exists $(M, w) \in \sigma(\mathcal{EM})$ such that $M, w \models \varphi$.

For any logic $\mathcal{L}$ we abbreviate the associated satisfiability problem with $\mathcal{L}$-SAT. In the following sections it is shown that DEL-SAT and DELVO-SAT are NEXPTIME-complete. To do this we need to show that the problems above are in NEXPTIME and are NEXPTIME-hard. The proofs for the above are inspired by the relevant proofs in [2]. there, a version of DEL is considered that uses arbitrary accessibility relations in its static and dynamic models, instead of equivalences (and so we can essentially say the underlying logic is K_n instead of $\mathrm{S5}_n$). For that version of DEL, it is then shown that DEL-SAT is NEXPTIME-complete. Naturally, the proofs here has been be modified to take into account that we use $\mathrm{S5}_n$ instead of K_n, as the underlying logic.

4.1 Lower Bound for DEL-SAT

In this section it is proven that DEL-SAT is NEXPTIME-hard by reducing the following tiling problem to it.

Let C be a countable and infinite set of colours. A *tile type* t is a 4-tuple of colours, denoted $t = (\mathit{left}(t), \mathit{right}(t), \mathit{up}(t), \mathit{down}(t)) \in C^4$. The tiling problem we will use is defined as follows:

Input: A finite set T of tile types, and a natural number k written in its binary form[1].

Output: Yes iff there exists a function $\tau : \{0, \ldots, k-1\}^2 \mapsto T$ satisfying the following constraints. For all $x \in \{0, \ldots, k-1\}$ and $y \in \{0, \ldots, k-2\}$:

$$\mathit{up}(\tau(x, y)) = \mathit{down}(\tau(x, y+1)); \tag{4.1}$$

for all $x \in \{0, \ldots, k-2\}$ and $y \in \{0, \ldots, k-1\}$:

$$\mathit{right}(\tau(x, y)) = \mathit{left}(\tau(x+1, y)); \tag{4.2}$$

In other words, the problem is to decide whether we can tile a $(k \times k)$-grid (positions range from 0 to $k-1$) with the tile types of T. Any function $\tau' : \{0, \ldots, k-1\}^2 \mapsto T$ is called a *tiling* for the grid. This tiling problem is known to be NEXPTIME-complete [26][2].

Theorem 4.2. *DEL-SAT with $S5_n$ is NEXPTIME-hard, for any $n \geq 1$.*

Proof. We can assume that $k = 2^n$ [3] and so the binary representation up to number $k-1$ requires n digits. Note that the size of the input is at least $n + |T|$[4]. Given an instance of the tiling problem described above (i.e., a tiling τ) the goal is to provide a polynomial translation from this instance, to an instance of DEL-SAT (i.e., a formula $\varphi \in \mathcal{L}_{\mathrm{DEL}}$), such that τ is correct for the grid iff φ is satisfiable in DEL.

The idea[5] is to encode τ twice (using propositional variables) so that we can compare two neighbouring tiles within $\mathcal{L}_{\mathrm{DEL}}$; one tile being from the 'first' tiling, the other from the 'second' tiling (first and second tiling being in fact the same). As such, a world in a model $M \in \mathcal{EM}$ will point to two pairs of coordinates – (x_1, y_1) and (x_2, y_2) – dictated by the world's valuation. The two identical encodings are being represented by atomic propositions $p_1, \ldots, p_{4n}$ (as mentioned we need n for each coordinate). More

[1] Not using a representation of k in unary important, as that would make the problem NP-complete.

[2] To be more precise, in [26] defined is the problem "BOUNDED TILING", which also includes colouring of the perimeter of the grid. In page 12, line 4, it is stated however that we obtain the same completeness result without the explicit boundary conditions.

[3] The proof in [26] implies that the problem remains as hard even with this extra assumption.

[4] We need not provide an upper bound for the size of the input; if the reduction we provide is polynomial wrt to some size, it is also polynomial wrt to a larger size.

[5] One can read this proof that follows accompanied by Example 4.1 found next, that has been formulated to follow the proof as close as possible.

precisely, set $X_1 = \{p_1, ..., p_n\}$ contains the atomic propositions encoding the binary representation of the integer x_1, $Y_1 = \{p_{n+1}, \ldots, p_{2n}\}$, and so on with sets X_2, Y_2.

Additionally, the tile types of the first tiling are represented by propositional variables 1_t and the tile types of the second tiling are represented by propositional variables $2_{t'}$, where t and t' range over T. The intention is that in a world that points for example to (x_1, y_1) and (x_2, y_2), 1_t is true in that world iff the tile type of (x_1, y_1) is t (and similarly $2_{t'}$ is true in that world iff the tile type of (x_2, y_2) is t').

So to begin with, we would like to guarantee the existence of all valuations over $X_1 \cup Y_1 \cup X_2 \cup Y_2$. This is easier to do when the underlying logic is K; the way is a well known technique used in [46] where we require the construction of a model in the form of a binary tree; each of the leaves in the tree corresponds to a valuation. A modification of this idea would be needed when using S5; that is found in [36]. This modification though, requires the presence of at least *two* agents to create a tree in S5. In an effort to use only *one* agent, and thus show that DEL-SAT with $S5_n$, for any $n \geq 1$, is NEXPTIME-hard, a new reduction is shown here; for the following we indeed assume the existence of only one agent.

The power of the dynamic aspect of DEL once again comes into play; instead of creating a tree in the epistemic 'dimension', we do so in the dynamic one: consider the action models $\mathsf{M}_i = \langle \mathsf{W}, \mathsf{R}, \mathsf{pre}_i \rangle \in \mathbb{AM}$ for $i \in \{1, \ldots, 4n\}$, where $\mathsf{W} = \{\mathsf{w}, \mathsf{v}\}$; $\mathsf{R} = \{(\mathsf{w}, \mathsf{w}), (\mathsf{v}, \mathsf{v})\}$; $\mathsf{pre}_i(\mathsf{w}) = \neg q \vee p_i$, $\mathsf{pre}_i(\mathsf{v}) = \neg q \vee \neg p_i$.

Suppose that formula φ – the formula we are trying to build – is satisfied at a world w in a (non-empty) single-agent epistemic model M. By applying any of the action models M_i we non-deterministically choose to keep the q-worlds where p_i is true, or the q-worlds where p_i is false, in the cluster[6] (this is similar to the approach in Model Checking section; there however, the worlds in which p_i was false were eliminated). More precisely, we get a model product comprised of 2 clusters, one only with p_i in its q-worlds, the other one only with $\neg p_i$ in its q-worlds. Applying all of the models in sequence would give us 2^n clusters; one for each valuation over $p_1, \ldots, p_{4n}$, or in other words, one for each pair of coordinates in our grid. At the same time, world w acts as our reference point throughout any of the model products, and to ensure its survival we require that $M, w \models \neg q$. As such, w survives (it is still the case that in any world occurring from w in the model products we still have $\neg q$; DEL as presented in this thesis has no ontic change anyway), but it happens regardless of its valuation with respect to any of the p_i and so it can have no 'saying' when it comes to coordinates and tile types. The latter would be true for any $\neg q$-world. Thus, to signal a tile type we have to require the existence of some q-world. So far all of the above is reflected in the following formula:

$$\neg q \wedge [\mathsf{M}_1] \ldots [\mathsf{M}_{4n}] \langle K \rangle q \tag{4.3}$$

[6]Reminder: the class of single-agent epistemic models is comprised of clusters, and we need only refer to the one that contains w.

Next: in order to check constraints 4.1 and 4.2 in the definition of a tiling, one refers to the tile located to the right or to the left of a given position in a tiling, and also to the tile located above or below it. The following formulas encode the fact that any pair of coordinates (x_1, x_2) and (y_1, y_2) of the two tilings satisfy the properties $x_1 = x_2$, $x_1 = x_2 + 1$, $y_1 = y_2$, and $y_1 = y_2 + 1$, respectively:

$$(x_1 = x_2) = \bigwedge_{1 \leq j \leq n} (p_j \leftrightarrow p_{j+2n}) \tag{4.4}$$

$$(y_1 = y_2) = \bigwedge_{n+1 \leq j \leq 2n} (p_j \leftrightarrow p_{j+2n}) \tag{4.5}$$

$$(x_1 = x_2 + 1) = \bigvee_{1 \leq k \leq n} \Big(\bigwedge_{1 \leq j < k} (p_{j+2n} \leftrightarrow p_j) \wedge \neg p_{k+2n} \wedge p_k \wedge \bigwedge_{k < j \leq n} (p_{j+2n} \wedge \neg p_j) \Big) \tag{4.6}$$

$$(y_1 = y_2 + 1) = \bigvee_{n+1 \leq k \leq 2n} \Big(\bigwedge_{n \leq j < k} (p_{j+2n} \leftrightarrow p_j) \wedge \neg p_{k+2n} \wedge p_k \wedge \bigwedge_{k < j \leq 2n} (p_{j+2n} \wedge \neg p_j) \Big) \tag{4.7}$$

Next, we have formulas 4.8, 4.9 to encode that in each cluster of $M \otimes \mathsf{M}_1 \otimes \cdots \otimes \mathsf{M}_{4n}$, and when restricted to q-worlds, there is exactly one tile type for the first tiling and exactly one tile type for the second tiling. Formula 4.10 below encodes the fact that when these two pairs of coordinates coincide then the tile type of the first tiling and the tile type of the second tiling are identical (as explained earlier, first and second tiling are in fact the same).

$$[\mathsf{M}_1] \ldots [\mathsf{M}_{4n}] K \Big(q \rightarrow \big(\bigvee_{t \in T} 1_t \wedge \bigvee_{t \in T} 2_t \big) \Big) \tag{4.8}$$

$$[\mathsf{M}_1] \ldots [\mathsf{M}_{4n}] K \Big(q \rightarrow \bigwedge \{ (1_t \rightarrow \neg 1_{t'}) \wedge (2_t \rightarrow \neg 2_{t'}) \mid t, t' \in T, t \neq t' \} \Big) \tag{4.9}$$

$$[\mathsf{M}_1] \ldots [\mathsf{M}_{4n}] K \Big(\big(q \wedge (x_1 = x_2) \wedge (y_1 = y_2) \big) \rightarrow \bigwedge_{t \in T} (1_t \leftrightarrow 2_t) \Big) \tag{4.10}$$

Furthermore, while two different clusters never refer to the same pair of positions, two or more clusters can refer to the same position of the first tiling (respectively, the second). It should be the case that these clusters encode the same tile type for that position. Formula 4.11 and 4.12 below, enforce this when the common position is of the first tiling and second tiling, respectively:

$$[\mathsf{M}_1] \ldots [\mathsf{M}_{2n}] \bigvee_{t \in T} [\mathsf{M}_{2n+1}] \ldots [\mathsf{M}_{4n}] K (q \rightarrow 1_t) \tag{4.11}$$

$$[\mathsf{M}_{2n+1}] \ldots [\mathsf{M}_{4n}] \bigvee_{t \in T} [\mathsf{M}_1] \ldots [\mathsf{M}_{2n}] K (q \rightarrow 2_t) \tag{4.12}$$

So for example 4.11 says: for each cluster/position of the first tiling (part $[\mathsf{M}_1] \ldots [\mathsf{M}_{2n}]$), either, for all clusters/positions of the second tiling (part $[\mathsf{M}_{2n+1}] \ldots [\mathsf{M}_{4n}]$) – with the first position fixed – the first tile type in the cluster is t, or, for all clusters/positions of the second tiling – with the first position fixed – the first tile type in the cluster is t',

and so on and so forth, for each tile type in T.

Thus far we managed to encode tiling τ twice. To conclude with the constraints, we express that 4.1 and 4.2 of the definition of a tiling hold, with formulas 4.13 and 4.14 respectively.

$$[\mathsf{M}_1]\dots[\mathsf{M}_{4n}]K\Big(\big(q\wedge(x_1=x_2)\wedge(y_1=y_2+1)\big)\rightarrow \bigwedge_{t\in T}\Big\{1_t\rightarrow\bigvee\{2_{t'}\mid t'\in T, \mathit{down}(t')=\mathit{up}(t)\}\Big\}\Big) \tag{4.13}$$

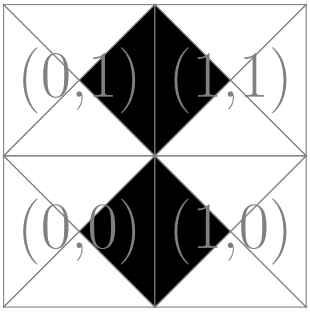
$$[\mathsf{M}_1]\dots[\mathsf{M}_{4n}]K\Big(\big(q\wedge(x_1=x_2+1)\wedge(y_1=y_2)\big)\rightarrow \bigwedge_{t\in T}\Big\{1_t\rightarrow\bigvee\{2_{t'}\mid t'\in T, \mathit{left}(t')=\mathit{right}(t)\}\Big\}\Big) \tag{4.14}$$

We define formula φ the conjunction of formulas 4.3, and 4.8 to 4.14. As shown in the above, φ is satisfiable in DEL if and only if there exists a tiling for the instance of the tiling problem.

Finally, we have that the whole reduction is indeed polynomial in the size of the instance of the tiling problem; it suffices to look at the formulas involved in the conjunction that is φ. Note that each $[\mathsf{M}_i]$ is $O(1)$ in size. We can readily see that 4.3 is of size $O(n)$. 4.4 and 4.5 have size $O(n)$, 4.6 and 4.7 have size $O(n^2)$. Thus 4.10 is of size $O(n^2+|T|)$. 4.8 has size $O(n+|T|)$, 4.9 size $O(n+|T|^2)$. 4.11 and 4.12 also have size $O(n|T|)$. And so we have that φ is polynomial with respect to the size of the input. □

It is especially important to note that in the proof, in order to speak about an exponential number of clusters using a formula of linear size, we had to use sequences of the form $[\mathsf{M}_i]\dots[\mathsf{M}_j]$, and because all models in these sequences have two worlds w,v this is in fact $[(\mathsf{M}_i,\mathsf{w})\cup(\mathsf{M}_i,\mathsf{v})]\dots[(\mathsf{M}_j,\mathsf{w})\cup(\mathsf{M}_j,\mathsf{v})]$. This makes apparent that the existence of the choice operator $\cup$ is crucial for the fact that DEL-SAT even for a single agent is NEXPTIME-complete. Another especially useful way to see this is to compare with the complexity of the satisfiability problem for a single agent in Public Announcement Logic (PAL) which however does not include a choice operator; that problem is NP-complete [48] - much lower than NEXPTIME and in fact of the same complexity as $\mathrm{S5}_1$-SAT [36].

Example 4.1. Let $k=2$ (and so $n=1$). Also, let $T=\{t_1=(\mathit{white},\mathit{black},\mathit{white},\mathit{white}),\ t_2=(\mathit{black},\mathit{white},\mathit{white},\mathit{white})\}$. That is, T contains two tile types: t_1 for which only its right part is *black* and the rest are *white*, and t_2 for which only its left part is *black*, and the rest are *white*. We can easily see the answer to this instance of the tiling problem is YES, as there exists a suitable tiling; one example is the function with $\tau(0,0)=\tau(0,1)=t_1$ and $\tau(1,0)=\tau(1,0)=t_2$ (illustration shown below).



According to the reduction we need variables $p_1, \ldots, p_4$ to encode the positions, variables $1_{t_1}, 1_{t_2}, 2_{t_1}, 2_{t_2}$ for the tile types, and of course variable q. Also we need action models $\mathsf{M}_i = \langle \mathsf{W}, \mathsf{R}, \mathsf{pre}_i \rangle \in \mathbb{AM}$ for $i \in \{1, \ldots, 4\}$, as described in the proof. That is: $\mathsf{W} = \{\mathsf{w}, \mathsf{v}\}$; $\mathsf{R} = \{(\mathsf{w}, \mathsf{w}), (\mathsf{v}, \mathsf{v})\}$; $\mathsf{pre}_i(\mathsf{w}) = \neg q \vee p_i$, $\mathsf{pre}_i(\mathsf{v}) = \neg q \vee \neg p_i$.

The end formula φ (the conjunction of formulas 4.3, and 4.8 to 4.14 for $n = 1$ and T as above) is indeed satisfiable in a single-agent epistemic model. To see this consider the model of Figure 4.1: $M = \langle W, R, f \rangle$ where $W = \{w\} \cup \{w_{i_1 i_2}^{i_3 i_4} \mid i_1, i_2, i_3, i_4 \in \{0, 1\}\}$, $R = W \times W$, $f(q) = W \setminus \{w\}$. Regarding variables $p_1, \ldots, p_4$: the subscript (respectively the superscript) of each world refers to the coordinates of the position in the first tiling (respectively the second). Formally, for all $j \in \{1, \ldots, 4\}$, $f(p_j) = \{w_{i_1 i_2}^{i_3 i_4} \mid i_j = 1\}$. Finally regarding the atoms for the tile types: according to the solution portrayed we define $f(1_{t_1}) = \{w_{00}^{i_3, i_4}, w_{01}^{i_3 i_4} \in W\}$, $f(1_{t_2}) = \{w_{10}^{i_3 i_4}, w_{11}^{i_3 i_4} \in W\}$, $f(2_{t_1}) = \{w_{i_1 i_2}^{00}, w_{i_1 i_2}^{01} \in W\}$, $f(2_{t_2}) = \{w_{i_1 i_2}^{10}, w_{i_1 i_2}^{11} \in W\}$, again i_1, i_2, i_3, i_4 range over $\{0, 1\}$.

We have that $M, w \models \varphi$. Note that model $M \otimes \mathsf{M}_1 \otimes \cdots \otimes \mathsf{M}_4$ is comprised of 16 clusters, each one with two worlds, one of the form (w, q_1, q_2, q_3, q_4), and one of the form $(w_{i_1 i_2}^{i_3 i_4}, q_1, q_2, q_3, q_4)$ with $q_i \in \{\mathsf{w}, \mathsf{v}\}$. Indeed each of the conjuncts is satisfied in w:

4.3 : $M, w \models q$ and furthermore in each of the clusters in $M_\varphi \otimes \mathsf{M}_1 \otimes \cdots \otimes \mathsf{M}_4$ there exists a world in which q is true; world $(w_{11}^{11}, \mathsf{w}, \mathsf{w}, \mathsf{w}, \mathsf{w})$ etc.
4.8 and 4.9 : Also true; all q-worlds in $M \otimes \mathsf{M}_1 \otimes \cdots \otimes \mathsf{M}_4$ have exactly one atom true between $\{1_{t_1}$ and 1_{t_2}, and exactly one true between $\{2_{t_1}$ and 2_{t_2}. For example in $(w_{11}^{11}, \mathsf{w}, \mathsf{w}, \mathsf{w}, \mathsf{w})$ only 1_{t_2} and 2_{t_2} are true.
4.10 : Indeed, worlds that correspond to the same position for the first and for the second tiling have the same tile type. For example we have $(w_{00}^{00}, \mathsf{v}, \mathsf{v}, \mathsf{v}, \mathsf{v}) \in f(1_{t_1}), f(2_{t_1})$ and $(w_{00}^{00}, \mathsf{v}, \mathsf{v}, \mathsf{v}, \mathsf{v}) \notin f(1_{t_2}), f(2_{t_2})$.
4.11 and 4.12 : This is also easy to check (strictly speaking, 4.11 is evaluated in $M \otimes \mathsf{M}_1 \otimes \mathsf{M}_2 \otimes \mathsf{M}_3 \otimes \mathsf{M}_4$ while 4.12 is evaluated in $M \otimes \mathsf{M}_3 \otimes \mathsf{M}_4 \otimes \mathsf{M}_1 \otimes \mathsf{M}_2$, but these two are isomorphic). For example, all worlds that refer to $(0, 0)$ for the first tiling have the same tile type: $(w_{00}^{00}, \mathsf{v}, \mathsf{v}, \mathsf{v}, \mathsf{v}), (w_{00}^{01}, \mathsf{v}, \mathsf{v}, \mathsf{v}, \mathsf{w}), (w_{00}^{10}, \mathsf{v}, \mathsf{v}, \mathsf{w}, \mathsf{v}), (w_{00}^{11}, \mathsf{v}, \mathsf{v}, \mathsf{w}, \mathsf{w}) \in f(1_{t_1})$.
4.13: Let us abbreviate the instance of 4.13 with $[\mathsf{M}_1][\mathsf{M}_2][\mathsf{M}_3][\mathsf{M}_4]K\psi$. In $(M \otimes \mathsf{M}_1 \otimes \mathsf{M}_2 \otimes \mathsf{M}_3 \otimes \mathsf{M}_4)$ we discern three cases for its worlds.

- The world is of the form (w, q_1, q_2, q_3, q_4); it is not a q-world and so the implication ψ is vacuously true.
- The world is of the form $(w_{i_1 i_2}^{i_3 i_4}, q_1, q_2, q_3, q_4)$ and refers to an inconsequential pair of coordinates, for example $(w_{00}^{00}, \mathsf{v}, \mathsf{v}, \mathsf{v}, \mathsf{v})$. Again the implication ψ is vacuously true.

- The world is of the form $(w^{i_3 i_4}_{i_1 i_2}, q_1, q_2, q_3, q_4)$ and refers to a pair of coordinates that offers meaningful comparison, for example $(w^{01}_{00}, \mathsf{v}, \mathsf{v}, \mathsf{v}, \mathsf{w})$. The antecedent of ψ is true and so we must check that $\big(1_{t_1} \to (2_{t_1} \vee 2_{t_2})\big) \wedge \big(1_{t_2} \to (2_{t_1} \vee 2_{t_2})\big)$; note that $\{2_{t'} \mid t' \in T, down(t') = up(t_1)\} = \{2_{t'} \mid t' \in T, down(t') = up(t_2)\} = \{2_{t_1}, 2_{t_2}\}$. Indeed, in $(w^{01}_{00}, \mathsf{v}, \mathsf{v}, \mathsf{v}, \mathsf{w})$, 1_{t_2} is false but 1_{t_1} and 2_{t_1} are true.

So we have that $(M \otimes \mathsf{M}_1 \otimes \mathsf{M}_2 \otimes \mathsf{M}_3 \otimes \mathsf{M}_4) \models \psi$, therefore $(M \otimes \mathsf{M}_1 \otimes \mathsf{M}_2 \otimes \mathsf{M}_3 \otimes \mathsf{M}_4) \models K\psi$, and so for all $q_1, q_2, q_3, q_4 \in \{\mathsf{w}, \mathsf{v}\}$ we have $\big((M \otimes \mathsf{M}_1 \otimes \mathsf{M}_2 \otimes \mathsf{M}_3 \otimes \mathsf{M}_4), (w, q_1, q_2, q_3, q_4)\big) \models K\psi$, and so $M, w \models [\mathsf{M}_1][\mathsf{M}_2][\mathsf{M}_3][\mathsf{M}_4]K\psi$.
4.14: Similarly to 4.13.

Differences with the G. Aucher and F. Schwarzentruber approach.

To conclude this section let us highlight some more the differences between our approach here with the approach taken in [2]. There is a series of different choices made as a matter of presentation, but the essential differences stem from the fact that [2] deals with the complexity of DEL with underlying logic K, whereas our results use S5 as the underlying logic. Having said that, the actual need for a novel idea comes when we need to find the lower bound of the complexity in the case S5 is restricted to the single-agent case. The novel idea being relying on a tree not in the epistemic dimension, but in the dynamic one.

The first notable difference is in the formula that enforces the existence of such a tree, in our approach this is formula 4.3. In [2] the respective formula (apperance/symbols have changed a bit to ease readability within the context of the thesis) is: $\bigwedge_{l \leq 4n} K^l\Big(\langle K\rangle p_l \wedge \langle K\rangle \neg p_l \wedge \bigwedge_{i<l} \big((p_i \to Kp_i) \wedge (\neg p_i \to K\neg p_i)\big)\Big)$. One can note that 4.3 is much shorter, but it is in fact not that simpler, since a lot is hidden in the definition of models $\mathsf{M}_1, \ldots, \mathsf{M}_{4n}$ and the definition of the model product. In our technique there is also the distinction between q-worlds and $\neg q$-worlds; the initial world is a $\neg q$-world.

Second, we have formulas 4.8, 4.9, 4.10. In [2] the formula corresponding to 4.8 is: $K^{4n}\Big(\bigvee_{t \in T} 1_t \wedge \bigvee_{t \in T} 2_t\Big)$. This formula is of the form $K^{4n}\varphi$; intuitively speaking, we need φ to be true at the leaves of the tree (and the leaves happen to be in depth $4n$). In our case we need to reach the leaves of a 'dynamic' tree, and so we have replaced K^{4n} with $[\mathsf{M}_1] \ldots [\mathsf{M}_{4n}]$. Furthermore the leaves are not single worlds but clusters, so instead of φ we have $K\varphi$. And finally, because we need φ to be true in the q-worlds of that cluster we in fact have $K(q \to \varphi)$. A similar transformation is applied to formulas of [2] to end up with formulas 4.9 and 4.10.

Third, we have formulas 4.11 and 4.12. The corresponding formulas in [2] are: $[\mathsf{M}_{p_1} \cup \mathsf{M}_{\neg p_1}] \ldots [\mathsf{M}_{p_{2n}} \cup \mathsf{M}_{\neg p_{2n}}] \bigvee_{t \in T} K^{4n} 1_t$ and $[\mathsf{M}_{p_{2n+1}} \cup \mathsf{M}_{\neg p_{2n+1}}] \ldots [\mathsf{M}_{p_{4n}} \cup \mathsf{M}_{\neg p_{4n}}] \bigvee_{t \in T} K^{4n} 2_t$. Remember that in [2] we deal with a tree in the logic K. In the first formula for example, the sequence of the dynamic boxes picks a valuation over $X_1 \cup Y_1$ and selects the branches of the tree whose leaves satisfy this valuation (the action models are a bit complicated and we refrain from defining them here). Having selected the branches, K^{4n} is again to be expected to reach the leaves of the branches. In our case, in the

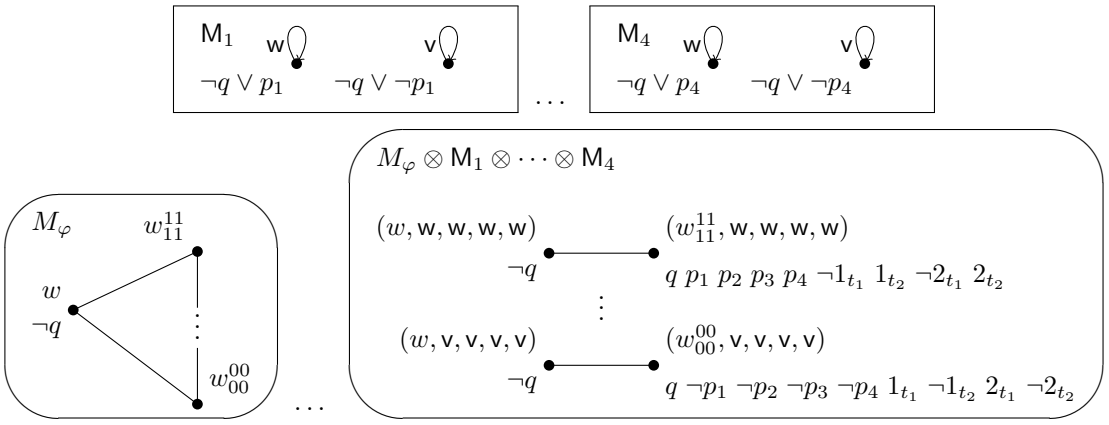


FIGURE 4.1: The epistemic and action models involved in the reduction from the tiling problem to Satisfiability of DEL. Reflexive and transitive arrows in the epistemic models are not drawn.

corresponding formula 4.11, in order to have a tree of depth $4n$ we need in a sequence the presence of all $4n$ action models (as opposed to the formulas above, where each time we have $2n$ dynamic boxes), and speaking about the leaves is also job of that sequence. Because of the latter, we have the big disjunction intersect the middle of the sequence, in order to refer to the range of valuations that we want.

4.2 Upper Bound for DEL-SAT

4.2.1 Tableaux Description

In this section a tableau system is presented, for determining the satisfiability of a formula of S5-based DEL. Arguably, tableaux are a more intuitive and practical proof system than that of a Hilbert-style axiomatisation, but more importantly they allow us to show that the complexity of DEL-SAT with S5 is in NEXPTIME. The idea for this comes from [2], as it is there that a tableau system for K-based DEL is developed for the purpose of determining complexity. The tableaux we will be creating draw, naturally, from [2] and from previous similar work by the author ([44, 45]); also from [30, 31], [42], and [36], to which one can refer for a more general exposition on (prefixed) tableaux systems[7]. A presentation of some terminology is in order:

A world prefix σ is just a natural number. The intention for the world prefixes is that they denote worlds in an epistemic model. An action prefix Σ is a finite –perhaps empty– sequence of $(\mathsf{M}, \mathsf{w}) \in \sigma(\mathbb{A}\mathbb{M})$ separated by semi-colons. A prefix is an expression $\sigma\ \Sigma$. A term is one of the following:

[7]Other than several differences in presentation, the important technical difference from [2] is the modification of rule $[K_\nu]$ that follows into its more general form that works for $S5$ and not only K.

- An expression $\sigma\ \Sigma\ \varphi$, where $\varphi \in \mathcal{L}_{\text{DEL}}$. The intended meaning is that Σ is executable in the world denoted by σ and that φ is true in the corresponding world of the model product.
- An expression $\sigma\ \Sigma\ \checkmark$, to signal that Σ is executable in σ.
- An expression $\sigma\ \Sigma\ \times$, to signal that Σ is *not* executable in σ.
- An expression $\sigma R_i \sigma'$, to say that if w and w' are the worlds denoted by σ and σ' respectively, then wR_iw'.

The tableau itself is a rooted tree, where its nodes are sets of terms. A branch is a path from the root of the tableau tree to one of its leaves.

The way we begin a tableau stems from a very important characteristic of theirs, namely that they are a refutation proof system; if I want to prove formula φ I assume that $\neg\varphi$ is satisfiable and *attempt* to derive a contradiction. Practically, this means that we *start any tableau* for φ with $1\ \neg\varphi$ at the root.

Then, nodes branch out using the set of rules below. All of these rules are of the form:

$$\frac{A}{B_1 \mid \ldots \mid B_n}$$

where $A, B_1, \ldots, B_n$ are sets of terms. The instruction given by these rules is that if the set of terms A appears in a node, then we branch out the tableau tree with nodes $B_1, \ldots, B_n$ on each branch respectively, if they do not already appear on the branch.

A branch that includes $\bot$ is closed, otherwise it is open. A tableau is closed if all of its branches are closed. A tableau proof for φ is a closed tableau that begins with $1\ \neg\varphi$.

Propositional rules:

$$[\neg\neg]\ \frac{\sigma\ \Sigma\ \neg\neg\varphi}{\sigma\ \Sigma\ \varphi} \qquad [\wedge]\ \frac{\sigma\ \Sigma\ \varphi \wedge \psi}{\begin{array}{c}\sigma\ \Sigma\ \varphi \\ \sigma\ \Sigma\ \psi\end{array}} \qquad [\vee]\ \frac{\sigma\ \Sigma\ \neg(\varphi \wedge \psi)}{\sigma\ \Sigma\ \neg\varphi \mid \sigma\ \Sigma\ \neg\psi}$$

$$[\bot]\ \frac{\begin{array}{c}\sigma\ \Sigma\ \varphi \\ \sigma\ \Sigma\ \neg\varphi\end{array}}{\bot} \qquad [\leftarrow_p]\ \frac{\sigma\ \Sigma\ p}{\sigma\ p} \qquad [\leftarrow_{\neg p}]\ \frac{\sigma\ \Sigma\ \neg p}{\sigma\ \neg p}$$

S5$_n$ rules:

$$[K_\nu]\ \frac{\sigma\ \mathsf{M}_1, \mathsf{w}_1; \ldots; \mathsf{M}_n, \mathsf{w}_n\ K_i\varphi}{\begin{array}{l}\sigma'\ \mathsf{M}_1, \mathsf{u}_1; \ldots; \mathsf{M}_n, \mathsf{u}_n\ \checkmark \\ \sigma'\ \mathsf{M}_1, \mathsf{u}_1; \ldots; \mathsf{M}_n, \mathsf{u}_n\ \varphi\end{array} \mid \sigma'\ \mathsf{M}_1, \mathsf{u}_1; \ldots; \mathsf{M}_n, \mathsf{u}_n\ \times}$$

where σ' already appears and is i-accessible from σ, and $\mathsf{u}_1, \ldots, \mathsf{u}_n$ are such that for all $j \in \{1, \ldots, n\}$ $\mathsf{w}_j \mathsf{R}_i^j \mathsf{u}_j$.

$$[K_\pi]\ \frac{\sigma\ \mathsf{M}_1, \mathsf{w}_1; \ldots; \mathsf{M}_n, \mathsf{w}_n\ \neg K_i\varphi}{\begin{array}{l|c|l} \sigma\ R_i\ \sigma_{\text{new}} & & \sigma\ R_i\ \sigma_{\text{new}} \\ \sigma_{\text{new}}\ \mathsf{M}_1, \mathsf{u}_1; \ldots; \mathsf{M}_n, \mathsf{u}_n\ \checkmark & \ldots & \sigma_{\text{new}}\ \mathsf{M}_1, \mathsf{u}'_1; \ldots; \mathsf{M}_n, \mathsf{u}'_n\ \checkmark \\ \sigma_{\text{new}}\ \mathsf{M}_1, \mathsf{u}_1; \ldots; \mathsf{M}_n, \mathsf{u}_n\ \neg\varphi & & \sigma_{\text{new}}\ \mathsf{M}_1, \mathsf{u}'_1; \ldots; \mathsf{M}_n, \mathsf{u}'_n\ \neg\varphi \end{array}}$$

where σ_{new} is new to the branch and $\mathsf{u}_1, \ldots, \mathsf{u}_n$ (respectively $\mathsf{u}'_1, \ldots, \mathsf{u}'_n$) are such that for all $j \in \{1, \ldots, n\}$ $\mathsf{w}_j \mathsf{R}_i^j \mathsf{u}_j$ (respectively $\mathsf{w}_j \mathsf{R}_i \mathsf{u}_j$).

Dynamic rules:

$$[(\mathsf{M}, \mathsf{w})_\nu]\ \frac{\sigma\ \Sigma\ [\mathsf{M}, \mathsf{w}]\varphi}{\begin{array}{l|l} \sigma\ \Sigma; \mathsf{M}, \mathsf{w}\ \checkmark & \sigma\ \Sigma; \mathsf{M}, \mathsf{w}\ \times \\ \sigma\ \Sigma; \mathsf{M}, \mathsf{w}\ \varphi & \end{array}} \qquad [(\mathsf{M}, \mathsf{w})_\pi]\ \frac{\sigma\ \Sigma\ \neg[\mathsf{M}, \mathsf{w}]\varphi}{\begin{array}{l} \sigma\ \Sigma; \mathsf{M}, \mathsf{w}\ \checkmark \\ \sigma\ \Sigma; \mathsf{M}, \mathsf{w}\ \neg\varphi \end{array}}$$

$$[\checkmark]\ \frac{\sigma\ \Sigma; \mathsf{M}, \mathsf{w}\ \checkmark}{\begin{array}{l} \sigma\ \Sigma\ \mathsf{pre}(\mathsf{w}) \\ \sigma\ \Sigma\ \checkmark \end{array}} \qquad [\times]\ \frac{\sigma\ \Sigma; \mathsf{M}, \mathsf{w}\ \times}{\begin{array}{l|l} \sigma\ \Sigma\ \checkmark & \sigma\ \Sigma\ \times \\ \sigma\ \Sigma\ \neg\mathsf{pre}(\mathsf{w}) & \end{array}} \qquad [\epsilon]\ \frac{\sigma\ \times}{\bot} \qquad [\checkmark\times]\ \frac{\begin{array}{l} \sigma\ \Sigma\ \checkmark \\ \sigma\ \Sigma\ \times \end{array}}{\bot}$$

$$[\cup_1]\ \frac{\sigma\ \Sigma\ [\alpha \cup \beta]\varphi}{\begin{array}{l} \sigma\ \Sigma\ [\alpha]\varphi \\ \sigma\ \Sigma\ [\beta]\varphi \end{array}} \qquad [\cup_2]\ \frac{\sigma\ \Sigma\ \neg[\alpha \cup \beta]\varphi}{\sigma\ \Sigma\ \neg[\alpha]\varphi \mid \sigma\ \Sigma\ \neg[\beta]\varphi}$$

Propositional rules are mostly self-explenatory: Rule $[\vee]$ is the one propositional rule by which we have to split a branch, to cater for each of the two possibilities that make the disjunction true. Rules $[\leftarrow_p]$ and $[\leftarrow_{\neg p}]$ can be easily explained if we remember that actions in DEL are not capable of ontic change. Furthermore, let us also mention here that rule $[\bot]$ is a bit stronger than what is necessary, because we can replace φ and $\neg\varphi$ with p and $\neg p$ respectively. Intuitively speaking, this is because with all the other tableaux rules we expect the implications of each formula being true to be analysed down to atoms. However, since assuming that φ and $\neg\varphi$ are true is a contradiction, there is no reason to keep analysing these formulas and apply more rules and so we keep its current form for convenience.

We then have the $\mathrm{S5}_{\boldsymbol{n}}$ rules. It is true that tableaux mostly follow the semantics of the logic we use them for, but for them to qualify as a symbolic proof system we need them to do so in a systematic way. Based on DEL semantics we can of course understand what $K_i\varphi$ means, in some world of some model product far far away of the

initial epistemic model: that φ is true in the worlds accessible with R_i. But we would then have to calculate which worlds those are, firstly by seeing which world even exists in that model product, and then which are actually seen through R_i. The novelty of [2] is that it offers a way to break this procedure down (and so that we are not lost and able to do it in a systematic way), by being able to write down all possible possible[8] worlds that could be seen through R_i, and branch out as to whether these worlds actually exist (so as to whether $(\mathsf{M}_1, \mathsf{u}_1), \ldots, (\mathsf{M}_n, \mathsf{u}_n)$ is executable or not). For rule $[K_\pi]$ we encounter a similar idea. In some accessible world $\neg\varphi$ is true, that much we know. But we do not know how that world came to be – what sequence of pointed models is responsible for its existence. Hence we make a new branch for each of the possibilities, and the implications are to be seen at a later step.

Last we have the Dynamic rules; they are easily explained as well having in mind the semantics of DEL.

4.2.2 Soundness

In this section the result is:

Theorem 4.3. *Let $\varphi \in \mathcal{L}_{DEL}$. If φ has a tableau proof then φ is valid in DEL.*

The proof of soundness for a tableaux proof system resembles that for Hilbert style systems, in the sense that it involves proving that any rules we could potentially apply cannot 'spoil' the correctness of what we have built (in the proof system) so far. For that notion of correctness we need the following:

Definition 4.4. Suppose B is a set of terms. We say that B is satisfiable if it does not contain $\bot$ and there exists $M = \langle W, R, V\rangle \in \mathcal{EM}$ and a function θ from the set of world prefixes to W such that:

- If $\sigma R_i \sigma'$ appears in B then $\theta(\sigma) R_i \theta(\sigma')$.
- If $\sigma\ \mathsf{M}_1, \mathsf{w}_1; \ldots; \mathsf{M}_n, \mathsf{w}_n\ \varphi$ appears in B then $(\mathsf{M}_1, \mathsf{w}_1), \ldots, (\mathsf{M}_n, \mathsf{w}_n)$ is executable in $\theta(\sigma)$ and $\theta(\sigma) \models \varphi$.
- If $\sigma\ \mathsf{M}_1, \mathsf{w}_1; \ldots; \mathsf{M}_n, \mathsf{w}_n\ \checkmark$ appears in B then $(\mathsf{M}_1, \mathsf{w}_1), \ldots, (\mathsf{M}_n, \mathsf{w}_n)$ is executable in $\theta(\sigma)$.
- If $\sigma\ \mathsf{M}_1, \mathsf{w}_1; \ldots; \mathsf{M}_n, \mathsf{w}_n\ \times$ appears in B then $(\mathsf{M}_1, \mathsf{w}_1), \ldots, (\mathsf{M}_n, \mathsf{w}_n)$ is not executable in $\theta(\sigma)$.

Furthermore we say that a tableau is satisfiable if one of its branches is satisfiable. Note that closed tableaux/branches cannot be satisfiable. Now to prove the validity of the tableaux rules:

Proposition 4.5. *If a rule is applied to a satisfiable tableau, the result is again a satisfiable tableau.*

[8] This is not a typo

Proof. Let T be such a satisfiable tableau, and let B be a satisfiable branch. We can assume that the rule we apply is applied to B, otherwise if B remains unchanged then T trivially remains satisfiable.

First let us argue that rules $[\bot], [clash], [\epsilon]$ cannot be applied to a satisfiable branch. To apply rule $[\bot]$ we would have $\sigma\ \Sigma\ \varphi \in B$ and $\sigma\ \Sigma\ \neg\varphi \in B$. Because B is satisfiable this means Σ is executable in $\theta(\sigma)$ and in the resulting world of the model product φ and $\neg\varphi$ hold, which is a contradiction. The argument is similar for $[clash]$ and $[\epsilon]$.

We continue with the rest of the rules:

• Rule $[\wedge]$: We have $\sigma\ \Sigma\ \varphi \wedge \psi \in B$ and we add $\sigma\ \Sigma\varphi$ and $\sigma\ \Sigma\ \psi$ to get branch B'. B is satisfiable and so there exists $M \in \mathcal{EM}$ and appropriate function θ. Suppose the pointed model we get by executing Σ in $\theta(\sigma)$ is (M', w'). Because B is satisfiable we have $M', w' \models \varphi \wedge \psi$, therefore $M', w' \models \varphi$ and $M', w' \models \psi$. B' is satisfiable using the same M and θ.

• Rule $[\vee]$: We have $\sigma\ \Sigma\ \varphi \vee \psi \in B$ and get branches B_1 and B_2 with $\sigma\ \Sigma\varphi \in B_1$ and $\sigma\ \Sigma\ \psi \in B_2$. B is satisfiable and so there exists $M \in \mathcal{EM}$ such that $\theta(\sigma) \models \varphi \vee \psi$, therefore $\theta(\sigma) \models \varphi$ or $\theta(\sigma) \models \psi$. So B_1 or B_2 is satisfiable using the same M and θ, in either case T is satisfiable.

• Rule $[\leftarrow_p]$: To apply this rule we have $\sigma\ \Sigma\ p \in B$ and we add $\sigma\ p$ to get branch B'. Because B is satisfiable this means Σ is executable in $\theta(\sigma)$ and in the resulting world of the model product p is true. We know DEL does not include ontic change and so $\theta(\sigma) \models p$. Therefore B' is satisfiable using the same M and θ.

• Rule $[K\nu]$: To apply this rule we have $\sigma\ \mathsf{M}_1, \mathsf{w}_1; \ldots; \mathsf{M}_n, \mathsf{w}_n\ K_i\varphi \in B$. We then split the branch into branches B' and B'', and for some appropriate σ' and $\mathsf{u}_1, \ldots, \mathsf{u}_n$ we add $\sigma'\ \mathsf{M}_1, \mathsf{u}_1; \ldots; \mathsf{M}_n, \mathsf{u}_n\ \checkmark$ and $\sigma'\ \mathsf{M}_1, \mathsf{u}_1; \ldots; \mathsf{M}_n, \mathsf{u}_n\ \varphi$ to branch B', and $\sigma'\ \mathsf{M}_1, \mathsf{u}_1; \ldots; \mathsf{M}_n, \mathsf{u}_n\ \times$ to B''. Now either (1) $(\mathsf{M}_1, \mathsf{u}_1), \ldots, (\mathsf{M}_n, \mathsf{u}_n)$ is executable in $\theta(\sigma')$ (note that σ' already appeared on B and so $\theta(\sigma')$ is defined) or (2) it is not executable in $\theta(\sigma')$. In case (1) branch B' is satisfiable whereas in case (2) B'' is satisfiable - using the same M and θ. In either case T is satisfiable.

• Rule $[K\pi]$: To apply this rule we have $\sigma\ \mathsf{M}_1, \mathsf{w}_1; \ldots; \mathsf{M}_n, \mathsf{w}_n\ \neg K_i\varphi \in B$. We then split B into branches B_1 to B_k (one for each appropriate sequence $\mathsf{u}_1, \ldots, \mathsf{u}_n$), and to each of those we add $\sigma\ R_i\ \sigma_{\text{new}}$, $\sigma_{\text{new}}\ \mathsf{M}_1, \mathsf{u}_1; \ldots; \mathsf{M}_n, \mathsf{u}_n\ \checkmark$, and $\sigma_{\text{new}}\ \mathsf{M}_1, \mathsf{u}_1; \ldots; \mathsf{M}_n, \mathsf{u}_n\ \neg\varphi$ (for the corresponding $\mathsf{u}_1, \ldots, \mathsf{u}_n$ each time). Because B is satisfiable we have $M \otimes \mathsf{M}_1 \otimes \cdots \otimes \mathsf{M}_n, (\theta(\sigma), \mathsf{w}_1, \ldots, \mathsf{w}_n) \models \neg K_i\varphi$, therefore there exists $(w', \mathsf{u}_1, \ldots, \mathsf{u}_n) \in M \otimes \mathsf{M}_1 \otimes \cdots \otimes \mathsf{M}_n$ such that (1) $\mathsf{u}_1, \ldots, \mathsf{u}_n$ is executable in $\theta(\sigma)$, (2) $(w, \mathsf{w}_1, \ldots, \mathsf{w}_n) R_i (w', \mathsf{u}_1, \ldots, \mathsf{u}_n)$ (therefore wR_iw'), and (3) $\neg\varphi$ is true there. We extend θ to θ' for the new world prefix we introduced; $\theta'(\sigma_{new}) = w'$. Some branch among B_1 to B_k has the sequence $\mathsf{u}_1, \ldots, \mathsf{u}_n$. Using M and θ' that branch is satisfiable, and so T remains satisfiable.

• Rule $[(\mathsf{M}, \mathsf{w})_\nu]$: We have $\sigma\ \Sigma\ [\mathsf{M}, \mathsf{w}]\varphi \in B$ and get branch B' with $\sigma\ \Sigma; \mathsf{M}, \mathsf{w}\ \checkmark$ and $\sigma\ \Sigma; \mathsf{M}, \mathsf{w}\ \varphi$, and branch B'' with $\sigma\ \Sigma; \mathsf{M}, \mathsf{w}\ \times$. Because B is satisfiable there exist appropriate model M and function θ. Now either (1) $\Sigma; \mathsf{M}, \mathsf{w}$ is executable in $\theta(\sigma)$ or (2) it is not executable in $\theta(\sigma)$. In case (1) also because B is satisfiable and so (suppose

the pointed model we get by executing Σ in $\theta(\sigma)$ is (M', w')) we have $M', w' \models [\mathsf{M}, \mathsf{w}]\varphi$ we have that $M' \otimes \mathsf{M}, (w', \mathsf{w}) \models \varphi$ and so B' is satisfiable. In case (2) we have that B'' is satisfiable. In both cases T is satisfiable.

• Rule $[(\mathsf{M}, \mathsf{w})_\pi]$: We have $\sigma\ \Sigma\ \neg[\mathsf{M}, \mathsf{w}]\varphi \in B$ and we add $\sigma\ \Sigma; \mathsf{M}, \mathsf{w}\ \checkmark$ and $\sigma\ \Sigma; \mathsf{M}, \mathsf{w}\ \neg\varphi$. Because B is satisfiable we have that (suppose the pointed model we get by executing Σ in $\theta(\sigma)$ is (M', w')) $M', w' \models \neg[\mathsf{M}, \mathsf{w}]\varphi$. This means that (M, w) is executable in M', w' *and* $M' \otimes \mathsf{M}, (w', \mathsf{w}) \models \neg\varphi$. Therefore T remains satisfiable using the same M and θ.

• The rest of the rules – $[\neg\neg]$,$[\leftarrow_{\neg p}]$,$[\checkmark]$,$[\times]$,$[\epsilon]$,$[\cup_1]$, and $[\cup_2]$ – are treated similarly.

□

We are now ready to conclude the proof for soundness. For the sake of contradiction, suppose φ has a tableau proof but it is not valid in DEL, and thus $\neg\varphi$ is satisfiable. I.e., there exists $(M, w) \in \sigma(\mathcal{EM})$ such that $M, w \models \varphi$. Thus a tableau containing only $\{1\ \varphi\}$ is satisfiable, simply by defining $\theta(1) = w$. Therefore any tableau beginning with $\{1\ \varphi\}$ and extended by any of the tableaux rules is satisfiable. But the tableau proof for φ is such a tableau and includes $\bot$ therefore it cannot be satisfiable. Hence the contradiction.

4.2.3 Completeness

In this section we prove:

Theorem 4.6. *Let $\varphi \in \mathcal{L}_{DEL}$. If φ is valid in DEL then φ has a tableau proof.*

The plan is in fact to work with the (equivalent) contrapositive statement: for all φ, if φ is unprovable using our tableaux system, then φ is not valid (i.e., $\neg\varphi$ is satisfiable).

By definition φ is unprovable iff there is no closed tableau for φ. Note that all of the tableaux rules *expand* a tableau and it is impossible to render a rule non-applicable by having applied other rules previously. Therefore, there exists a (the) *maximum* tableau for φ, in which all rules are *saturated*. By definition we have that there is no closed tableau for φ iff the maximum tableau for φ is open.

So, we begin with the assumption that the maximum tableau for φ is open. Therefore one of its branches its open; let us take the leftmost one, we will call it B. We define a so called *counter-model* $M = \langle W, R, f\rangle$ as follows:

- $W = \{\sigma \mid \sigma \text{ appears on } B\}$.
- R_i is the reflexive, transitive, symmetric closure of $\{(\sigma, \tau) \mid \sigma R_i \tau \in B\}$.
- $f(p) = \{\sigma \mid \sigma\ p \in B\}$.

We first prove the following Lemma:

Lemma 4.7. *Let $\varphi \in \mathcal{L}_{DEL}$ and $(\mathsf{M}_1, \mathsf{w}_1), \ldots, (\mathsf{M}_k, \mathsf{w}_k) \in \sigma(\mathbb{A}\mathbb{M})$. If $\sigma\ \mathsf{M}_1, \mathsf{w}_1; \ldots; \mathsf{M}_k, \mathsf{w}_k\ \varphi \in B$ and $\sigma\ \mathsf{M}_1, \mathsf{w}_1; \ldots; \mathsf{M}_k, \mathsf{w}_k\ \checkmark \in B$, then the sequence $(\mathsf{M}_1, \mathsf{w}_1), \ldots, (\mathsf{M}_k, \mathsf{w}_k)$ is executable in σ and $M \otimes \mathsf{M}_1 \otimes \cdots \otimes \mathsf{M}_k, (\theta(\sigma), \mathsf{w}_1, \ldots, \mathsf{w}_k) \models \varphi$.*

Proof. We prove the Lemma by induction on $n = \sum_{i=1}^{k} |\mathsf{M}_i| + |\varphi|$.
For $n = 1$**:** In that case we necessarily have $\varphi = p$ for some $p \in \mathcal{V}ar$, and $k = 0$. Consequently we have $\sigma\ p \in B$. Indeed, the empty sequence is - trivially - executable in σ and by definition of f in M we have $(M, \sigma) \models p$.
For $n{+}1$**:** If $k \neq 0$ then by saturation of rule (pre) we have that $\sigma\ \mathsf{M}_1, \mathsf{w}_1, \ldots, \mathsf{M}_{k-1}, \mathsf{w}_{k-1}\ \mathsf{pre}(\mathsf{w}_k) \in B$. From Def. 4.1 we have $|\mathsf{pre}(\mathsf{w}_k)| < |\mathsf{M}_k|$ and so $\sum_{i=1}^{k-1} |\mathsf{M}_i| + |\mathsf{pre}(\mathsf{w}_k)| < \sum_{i=1}^{k} |\mathsf{M}_i| < \sum_{i=1}^{k} |\mathsf{M}_i| + |\varphi|$. So by induction hypothesis we have that the sequence $(\mathsf{M}_1, \mathsf{w}_1), \ldots, (\mathsf{M}_{k-1}, \mathsf{w}_{k-1})$ is executable in σ and $M \otimes \mathsf{M}_1 \otimes \cdots \otimes \mathsf{M}_{k-1}, (\sigma, \mathsf{w}_1, \ldots, \mathsf{w}_{k-1}) \models \mathsf{pre}(\mathsf{w}_k)$, and thus the whole $(\mathsf{M}_1, \mathsf{w}_1), \ldots, (\mathsf{M}_k, \mathsf{w}_k)$ is executable in σ. It remains to show that $M \otimes \mathsf{M}_1 \otimes \cdots \otimes \mathsf{M}_k, (\theta(\sigma), \mathsf{w}_1, \ldots, \mathsf{w}_k) \models \varphi$. We do this by distinguishing cases for the structure of φ but no new induction is needed.
• If $\varphi = p$: then by saturation of rule ($\leftarrow_p$) we have $\sigma\ p \in B$ and by definition of f in M we have $M, \sigma \models p$. By definition of the DEL model product valuations remain unchanged and so $M \otimes \mathsf{M}_1 \otimes \cdots \otimes \mathsf{M}_k, (\theta(\sigma), \mathsf{w}_1, \ldots, \mathsf{w}_k) \models p$.
• If $\varphi = \varphi_1 \wedge \varphi_2$: then by saturation of rule ($\wedge$) we have $\sigma\ \mathsf{M}_1, \mathsf{w}_1; \ldots; \mathsf{M}_k, \mathsf{w}_k\ \varphi_1 \in B$ and $\sigma\ \mathsf{M}_1, \mathsf{w}_1; \ldots; \mathsf{M}_k, \mathsf{w}_k\ \varphi_2 \in B$. by definition of f in M we have $M, \sigma \models p$. We have $|\varphi_1| < |\varphi|$ and $|\varphi_2| < |\varphi|$ and so by induction hypothesis $M \otimes \mathsf{M}_1 \otimes \cdots \otimes \mathsf{M}_k, (\theta(\sigma), \mathsf{w}_1, \ldots, \mathsf{w}_k) \models \varphi_1$ and $M \otimes \mathsf{M}_1 \otimes \cdots \otimes \mathsf{M}_k, (\theta(\sigma), \mathsf{w}_1, \ldots, \mathsf{w}_k) \models \varphi_2$. Overall $M \otimes \mathsf{M}_1 \otimes \cdots \otimes \mathsf{M}_k, (\theta(\sigma), \mathsf{w}_1, \ldots, \mathsf{w}_k) \models \varphi$.
• If $\varphi = K_i\psi$: Let $(\sigma', \mathsf{u}_1, \ldots, \mathsf{u}_k) \in R_i'((\sigma, \mathsf{w}_1, \ldots, \mathsf{w}_k))$; this means that $\sigma' \in R(\sigma), \mathsf{u}_1 \in \mathsf{R}_i^1(\mathsf{w}_1), \ldots, \mathsf{u}_k \in \mathsf{R}_i^k(\mathsf{w}_k)$ and that sequence $(\mathsf{M}_1, \mathsf{u}_1), \ldots, (\mathsf{M}_k, \mathsf{u}_k)$ is executable in σ'. We need to show that $M \otimes \mathsf{M}_1 \otimes \cdots \otimes \mathsf{M}_k, (\sigma', \mathsf{u}_1, \ldots, \mathsf{u}_k) \models \psi$. By definition of R in M, σ' is accessible from σ and so by saturation of rule ($K\nu$), either (1) $\sigma'\ \mathsf{M}_1, \mathsf{u}_1; \ldots; \mathsf{M}_k, \mathsf{u}_k \times \in B$ or (2) $\big(\sigma'\ \mathsf{M}_1, \mathsf{u}_1; \ldots; \mathsf{M}_k, \mathsf{u}_k \checkmark \in B$ and $(\sigma'\ \mathsf{M}_1, \mathsf{u}_1; \ldots; \mathsf{M}_k, \mathsf{u}_k \psi \in B\big)$. In case (1), by saturation of rule ($\times$) either (1i) there is $j < k$ such that $\big(\sigma'\ \mathsf{M}_1, \mathsf{u}_1; \ldots; \mathsf{M}_j, \mathsf{u}_j\ \neg\mathsf{pre}(\mathsf{u}_{j+1} \in B$ and $\sigma'\ \mathsf{M}_1, \mathsf{u}_1; \ldots; \mathsf{M}_j, \mathsf{u}_j\ \checkmark \in B$ or (1ii) $\sigma' \times \in B$. Case (1ii) is excluded, as B is open. For case (1i) Note that (from 4.1) $\sum_{i=1}^{j} |\mathsf{M}_i| + |\neg\mathsf{pre}(\mathsf{w}_{j+1})| = \sum_{i=1}^{j} |\mathsf{M}_i| + |\mathsf{pre}(\mathsf{w}_{j+1})| + 1 < \sum_{i=1}^{k} |\mathsf{M}_i| < \sum_{i=1}^{k} |\mathsf{M}_i| + |\varphi|$ and so by induction hypothesis we have that sequence $(\mathsf{M}_1, \mathsf{u}_1), \ldots, (\mathsf{M}_j, \mathsf{u}_j)$ is executable in σ but $(\mathsf{M}_1, \mathsf{u}_1), \ldots, (\mathsf{M}_{j+1}, \mathsf{u}_{j+1})$ is not. And consequently $(\mathsf{M}_1, \mathsf{u}_1), \ldots, (\mathsf{M}_k, \mathsf{u}_k)$ is not executable in σ' which leads to a contradiction. Therefore only case (2) is viable. Because $\sum_{i=1}^{k} |\mathsf{M}_i| + |\psi| < \sum_{i=1}^{k} |\mathsf{M}_i| + |\varphi|$ the induction hypothesis again applies and we have $M \otimes \mathsf{M}_1 \otimes \cdots \otimes \mathsf{M}_k, (\sigma', \mathsf{u}_1, \ldots, \mathsf{u}_k) \models \psi$.
• If $\varphi = [\mathsf{M}, \mathsf{w}]\psi$ then by saturation of rule ($[(\mathsf{M}, \mathsf{w})\nu]$) we have (1) $\sigma\ \mathsf{M}_1, \mathsf{w}_1; \ldots; \mathsf{M}_k, \mathsf{w}_k; \mathsf{M}, \mathsf{w} \times \in B$ or (2) $\sigma\ \mathsf{M}_1, \mathsf{w}_1; \ldots; \mathsf{M}_k, \mathsf{w}_k; \mathsf{M}, \mathsf{w}\ \checkmark \in B$ and $\sigma\ \mathsf{M}_1, \mathsf{w}_1; \ldots; \mathsf{M}_k, \mathsf{w}_k; \mathsf{M}, \mathsf{w}\ \psi \in B$. For case (1) by saturation of rule ($\times$) we have either (1i) $\sigma\ \mathsf{M}_1, \mathsf{w}_1; \ldots; \mathsf{M}_k, \mathsf{w}_k \times \in B$ or (1ii) $\sigma\ \mathsf{M}_1, \mathsf{w}_1; \ldots; \mathsf{M}_k, \mathsf{w}_k\ \checkmark \in B$ and $\sigma\ \mathsf{M}_1, \mathsf{w}_1; \ldots; \mathsf{M}_k, \mathsf{w}_k\ \neg\mathsf{pre}(\mathsf{w}) \in B$. Case (1i) is impossible because by our very initial assumption $\sigma\ \mathsf{M}_1, \mathsf{w}_1; \ldots; \mathsf{M}_k, \mathsf{w}_k\ \checkmark \in B$ and by saturation of rule (*clash*) branch B would be closed. Therefore case (1ii) remains. We have that $|\neg\mathsf{pre}(\mathsf{w})| = |\mathsf{pre}(\mathsf{w})| + 1 < |\mathsf{M}| < |\mathsf{M}| + |\psi| + 1 = |\varphi|$ and so $\sum_{i=1}^{k} |\mathsf{M}_i| + |\neg\mathsf{pre}(\mathsf{w})| < \sum_{i=1}^{k} |\mathsf{M}_i| + |\varphi|$. Therefore by induction hypothesis $M \otimes \mathsf{M}_1 \otimes \cdots \otimes \mathsf{M}_k, (\theta(\sigma), \mathsf{w}_1, \ldots, \mathsf{w}_k) \models \neg\mathsf{pre}(\mathsf{w})$ and so for case (1) trivially $M \otimes \mathsf{M}_1 \otimes$

$\cdots \otimes \mathsf{M}_k, (\theta(\sigma), \mathsf{w}_1, \ldots, \mathsf{w}_k) \models [\mathsf{M}, \mathsf{w}]\psi$. In case (2): We have that $\sum_{i=1}^{k} |\mathsf{M}_i| + |\mathsf{M}| + |\psi| < \sum_{i=1}^{k} |\mathsf{M}_i| + |\mathsf{M}| + |\psi| + 1 = \sum_{i=1}^{k} |\mathsf{M}_i| + |\varphi|$ and so again by induction hypothesis the sequence $(\mathsf{M}_1, \mathsf{w}_1), \ldots, (\mathsf{M}_k, \mathsf{w}_k), (\mathsf{M}, \mathsf{w})$ is executable in σ and $M \otimes \mathsf{M}_1 \otimes \cdots \otimes \mathsf{M}_k \otimes \mathsf{M}, (\sigma, \mathsf{w}_1, \ldots, \mathsf{w}_k, \mathsf{w}) \models \psi$. Therefore also for case (2) $M \otimes \mathsf{M}_1 \otimes \cdots \otimes \mathsf{M}_k, (\theta(\sigma), \mathsf{w}_1, \ldots, \mathsf{w}_k) \models [\mathsf{M}, \mathsf{w}]\psi$.

• If $\varphi = [\alpha \cup \beta]\psi$ then by saturation of (*choice*) we have $\sigma\ \mathsf{M}_1, \mathsf{w}_1; \ldots; \mathsf{M}_k, \mathsf{w}_k\ [\alpha]\psi \in B$ and $\sigma\ \mathsf{M}_1, \mathsf{w}_1; \ldots; \mathsf{M}_k, \mathsf{w}_k\ [\beta]\psi \in B$. By induction hypothesis we have $M \otimes \mathsf{M}_1 \otimes \cdots \otimes \mathsf{M}_k, (\theta(\sigma), \mathsf{w}_1, \ldots, \mathsf{w}_k) \models [\alpha]\psi$ and $M \otimes \mathsf{M}_1 \otimes \cdots \otimes \mathsf{M}_k, (\theta(\sigma), \mathsf{w}_1, \ldots, \mathsf{w}_k) \models [\beta]\psi$ therefore $M \otimes \mathsf{M}_1 \otimes \cdots \otimes \mathsf{M}_k, (\theta(\sigma), \mathsf{w}_1, \ldots, \mathsf{w}_k) \models [\alpha \cup \beta]\psi$.

• If $\varphi = \neg\psi$: We need to further distinguish sub-cases for ψ.

– If $\psi = p$ then by saturation of rule $(\leftarrow_{\neg p})$ we have $\sigma\ \neg p \in B$ and by definition of f in M we have $M, \sigma \models \neg p$. By definition of the DEL model product valuations remain unchanged and so $M \otimes \mathsf{M}_1 \otimes \cdots \otimes \mathsf{M}_k, (\theta(\sigma), \mathsf{w}_1, \ldots, \mathsf{w}_k) \models \neg p$.

– If $\psi = \neg\psi'$ and so $\varphi = \neg\neg\psi'$ then by saturation of rule $(\neg\neg)$ we have $\sigma\ \mathsf{M}_1, \mathsf{w}_1; \ldots; \mathsf{M}_k, \mathsf{w}_k\ \psi' \in B$. Because $|\psi'| < |\varphi|$ by induction hypothesis we have $M \otimes \mathsf{M}_1 \otimes \cdots \otimes \mathsf{M}_k, (\theta(\sigma), \mathsf{w}_1, \ldots, \mathsf{w}_k) \models \psi'$ and so, by propositional reasoning we have $M \otimes \mathsf{M}_1 \otimes \cdots \otimes \mathsf{M}_k, (\theta(\sigma), \mathsf{w}_1, \ldots, \mathsf{w}_k) \models \varphi$.

– If $\psi = \psi_1 \wedge \psi_2$ then by saturation of rule $(\vee)$ we that $\sigma\ \mathsf{M}_1, \mathsf{w}_1; \ldots; \mathsf{M}_k, \mathsf{w}_k\ \psi_1 \in B$ or $\sigma\ \mathsf{M}_1, \mathsf{w}_1; \ldots; \mathsf{M}_k, \mathsf{w}_k\ \psi_2 \in B$. We have $|\psi_1| < |\varphi|$ and $|\psi_2| < |\varphi|$ and so by induction hypothesis $M \otimes \mathsf{M}_1 \otimes \cdots \otimes \mathsf{M}_k, (\theta(\sigma), \mathsf{w}_1, \ldots, \mathsf{w}_k) \models \psi_1$ or $M \otimes \mathsf{M}_1 \otimes \cdots \otimes \mathsf{M}_k, (\theta(\sigma), \mathsf{w}_1, \ldots, \mathsf{w}_k) \models \psi_2$. Overall we have $M \otimes \mathsf{M}_1 \otimes \cdots \otimes \mathsf{M}_k, (\theta(\sigma), \mathsf{w}_1, \ldots, \mathsf{w}_k) \models \varphi$.

– If $\psi = K_i\psi'$ and so $\varphi = \neg K_i\psi'$: By saturation of rule $(K\pi)$ we have $\sigma R_i \sigma_{new} \in B$ for some world prefix σ_{new} (note this also implies $\sigma R_i \sigma_{new}$ by definition of M), and $\sigma\ \mathsf{M}_1, \mathsf{u}_1; \ldots; \mathsf{M}_k, \mathsf{u}_k\ \checkmark \in B$, $\sigma\ \mathsf{M}_1, \mathsf{u}_1; \ldots; \mathsf{M}_k, \mathsf{u}_k\ \neg\psi' \in B$ for some $\mathsf{u}_1 \in \mathsf{R}_i^1(\mathsf{w}_1), \ldots, \mathsf{u}_k \in \mathsf{R}_i^k(\mathsf{w}_k)$. We have that $\sum_{i=1}^{k} |\mathsf{M}_i| + |\neg\psi'| < \sum_{i=1}^{k} |\mathsf{M}_i| + |\varphi|$ and so by induction hypothesis $(\mathsf{M}_1, \mathsf{u}_1), \ldots, (\mathsf{M}_k, \mathsf{u}_k)$ is executable in σ_{new} and $M \otimes \mathsf{M}_1 \otimes \cdots \otimes \mathsf{M}_k, (\sigma_{new}, \mathsf{u}_1, \ldots, \mathsf{u}_k) \models \psi'$. Note that $(\sigma_n ew, \mathsf{u}_1, \ldots, \mathsf{u}_k) \in R_i'((\sigma, \mathsf{w}_1, \ldots, \mathsf{w}_k))$ and so $\otimes \models \neg K_i\psi'$.

– If $\psi = [\mathsf{M}, \mathsf{w}]\psi'$ and so $\varphi = \neg[\mathsf{M}, \mathsf{w}]\psi'$: By saturation of rule $([(\mathsf{M}, \mathsf{w})\pi])$ we have $\sigma\ \mathsf{M}_1, \mathsf{w}_1; \ldots; \mathsf{M}_k, \mathsf{w}_k; \mathsf{M}, \mathsf{w}\ \checkmark \in B$ and $\sigma\ \mathsf{M}_1, \mathsf{w}_1; \ldots; \mathsf{M}_k, \mathsf{w}_k; \mathsf{M}, \mathsf{w}\ \neg\psi' \in B$. We have that $\sum_{i=1}^{k} |\mathsf{M}_i| + |\mathsf{M}| + |\neg\psi| < \sum_{i=1}^{k} |\mathsf{M}_i| + |\mathsf{M}| + |\neg\psi| + 1 = \sum_{i=1}^{k} |\mathsf{M}_i| + |\varphi|$ and so by induction hypothesis the sequence $(\mathsf{M}_1, \mathsf{w}_1), \ldots, (\mathsf{M}_k, \mathsf{w}_k), (\mathsf{M}, \mathsf{w})$ is executable in σ and $M \otimes \mathsf{M}_1 \otimes \cdots \otimes \mathsf{M}_k \otimes \mathsf{M}, (\sigma, \mathsf{w}_1, \ldots, \mathsf{w}_k, \mathsf{w}) \models \neg\psi'$. Therefore $M \otimes \mathsf{M}_1 \otimes \cdots \otimes \mathsf{M}_k, (\theta(\sigma), \mathsf{w}_1, \ldots, \mathsf{w}_k) \models \neg[\mathsf{M}, \mathsf{w}]\psi'$.

– If $\psi = [\alpha \cup \beta]\psi'$ and so $\varphi = \neg[\alpha \cup \beta]\psi'$: Then by saturation of rule (*choice*′) we that $\sigma\ \mathsf{M}_1, \mathsf{w}_1; \ldots; \mathsf{M}_k, \mathsf{w}_k\ \neg[\alpha]\psi' \in B$ or $\sigma\ \mathsf{M}_1, \mathsf{w}_1; \ldots; \mathsf{M}_k, \mathsf{w}_k\ \neg[\beta]\psi' \in B$. We have $|\neg[\alpha]\psi'| < |\varphi|$ and $|\neg[\beta]\psi'| < |\varphi|$ and so by induction hypothesis $M \otimes \mathsf{M}_1 \otimes \cdots \otimes \mathsf{M}_k, (\theta(\sigma), \mathsf{w}_1, \ldots, \mathsf{w}_k) \models \neg[\alpha]\psi'$ or $M \otimes \mathsf{M}_1 \otimes \cdots \otimes \mathsf{M}_k, (\theta(\sigma), \mathsf{w}_1, \ldots, \mathsf{w}_k) \models \neg[\beta]\psi'$. Overall we have $M \otimes \mathsf{M}_1 \otimes \cdots \otimes \mathsf{M}_k, (\theta(\sigma), \mathsf{w}_1, \ldots, \mathsf{w}_k) \models \neg[\alpha \cup \beta]\psi'$.

□

Having proved the lemma we get back to our initial train of thought: we assume that the maximum tableau for φ is open. That tableau starts with 1 $\neg\varphi$ and so all of

its branches, including its leftmost open branch B, contain 1 $\neg\varphi$. We then consider the counter-model M produced from B. Based on Lemma 4.7 we have that $M, 1 \models \neg\varphi$, thus $\neg\varphi$ is satisfiable and thus φ is not valid. This concludes the completeness proof.

4.2.4 Complexity of Tableaux

Theorem 4.8. *DEL-SAT is in NEXPTIME.*

Having established soundness and completeness we now know that with a tableau proof for $\neg\varphi$ (and so starting the tableau with $\neg\neg\varphi$), after we have applied rules until the tableau is saturated we can determine whether $\neg\neg\varphi$ is satisfiable or not (i.e., whether φ is satisfiable). For simplicity let us call the formula at root of the tableau φ. Let $|\varphi| = m$. With respect to m we first prove the following:

Claim 4.9. *For a given world prefix σ at most an exponential number of terms appear in the tableaux.*

Proof. All terms are of the form $\sigma\ \Sigma\ \psi$. With a simple inspection of the rules we can see that ψ is some subformula or φ or a negation of one (these include any preconditions of action models that appear in φ). So the possible choices for ψ are $O(m)$. We would then like to count the possibilities for Σ. Each possible sequence when applied in σ is essentially the number of pointed epistemic models in which we would have to evaluate some subformula of φ. We already know the number of these and the crucial role that the non-deterministic choice operator plays in that, they are $O(2^m)$ in number. So overall, given some world prefix σ, we have at most an exponential number of terms. □

Claim 4.10. *From a given world prefix σ and agent i, there is at most an exponential number of world prefixes accessible from it through R_i.*

Proof. For a single cluster of world prefixes for agent i (world prefixes that are accessible through R_i), worlds are created from rule $[K_\pi]$, that is with formulas $\neg K_i\psi'$, whether it is with $\Sigma = \epsilon$ and thus the initial epistemic model, or with some other sequence of models. From Claim 4.9 we know that there exists an exponential number of possibilities for $\Sigma\ \psi$ and a fortiori for $\Sigma\ \neg K_i\psi'$. It does not matter if we have terms $\sigma\ \Sigma\ \neg K_i\psi'$ or $\sigma'\ \Sigma\ \neg K_i\psi'$ for some other σ' in the same cluster, we will only apply rule $[K_\pi]$ if $\neg\psi'$ does not already appear for an accessible world prefix. □

Claim 4.11. *For any world prefix σ we define its distance from prefix 1 to be the minimum amount of changes in accessibility relations we must traverse to reach it. The maximum distance over all σ is linear with respect to m.*

Proof. Let σ be a world prefix and its distance from 1 be k. This means that there are k formulas of the form $\neg K_i$ in φ. But such formulas can be at most $O(m)$. □

We can see that the number of world prefixes that appear in a tableau tree are exponential in number: Starting from the root that is prefix 1 we count itself and all its

immediate children for some relation R_i, those are exponential in number. Then for each of the children we count their children for accessibility relations R_j where $j \neq i$. The depth of such a tree is linear based on Claim 4.11 and its arity is exponential based on Claim 4.10. All of the world prefixes are nodes in that tree, and so they are exponential in number with respect to m.

Overall we have an exponential number of σ and an exponential number of $\Sigma\ \psi$. Therefore in the tableau tree there can be at most an exponential number of terms $\sigma\ \Sigma\ \psi$. The output of the tableau rules on a single branch is a finite number of terms, therefore for a single branch we apply an exponential number of rules, and this takes exponential time. Choosing the branch depends on the non-deterministic choices we make while applying the tableau rules, and so the whole procedure is in NEXPTIME.

4.3 Lower Bound for DELVO-SAT

Given Theorem 4.2 we have the option of reducing DEL-SAT to DELVO-SAT, in order to show that the latter is NEXPTIME-HARD. This is exactly what we will be doing.

Theorem 4.12. *DELVO-SAT is NEXPTIME-hard.*

Let φ be a formula of $\mathcal{L}_{\text{DEL}}$. We would like a formula $\varphi' \in \mathcal{L}_{\text{DELVO}}$ such that φ is satisfiable in a DEL model iff φ' is satisfiable in a DELVO model. To this end, recall Theorem 3.14; if $\varphi' = g(\varphi)$ 3.13 we at least have the 'left to right' direction. For the inverse we need consider that DELVO models are DEL models, but with the different vision functions as a parameter. To limit ourselves to the vision function that corresponds to DEL models, that is the 'empty' vision function, we need only say so with its characteristic formula. That is, we finally define $\varphi' = g(\varphi) \wedge \chi_\emptyset$.

Suppose indeed φ is true in some $(M, w) \in \sigma(\mathcal{EM})$. Then by Theorem 3.14, $(h(M), w) \models g(\varphi)$. Furthermore, we have that $h(M)$ is a model with the empty vision function, and so $h(M) \models \chi_\emptyset$. Overall $h(M), w \models g(\varphi) \wedge \chi_\emptyset$ and so φ' is satisfiable in DELVO.

Now suppose $\varphi' = g(\varphi) \wedge \chi_\emptyset$ is true in some $(M, w) \in \sigma(\mathcal{EMV})$, where $M = \langle W, R, V, f\rangle$. For $\chi_\emptyset$ to be true somewhere in M, this means that M has the empty vision function and we can see that model $M' = \langle W, R, f\rangle$ is in $\mathcal{EM}$ and $h(M') = M$. We have thus assumed that $h(M'), w \models g(\varphi)$. By the inverse direction in Theorem 3.14 we have that $M', w \models \varphi$ and so φ is satisfiable in DEL.

Finally, regarding the size of the reduction: the reduction involves only writing formula φ'. Suppose $|\varphi| = n$. Formula $g(\varphi)$ in turn involves only adding an empty toggle set tgl (information of constant size) to each of the points in the action models – those are at most n. Also, $\chi_\emptyset$ is a conjunction of formulas $\neg V_i x$ (constant given i and x), for each $i \in Ag$ (Ag is finite), and for each x that appears in φ (also at most n in number). Therefore the reduction needs size $O(n)$; linear to the size of the input.

4.4 Upper Bound for DELVO-SAT

The method to show the upper bound for $DELVO - SAT$ is again through tableau, we will be modifying the tableaux method we built for DEL in Section 4.2.

4.4.1 Tableaux Description

To cater for the existence of a vision function in the DELVO semantics: we first non-deterministically choose a vision function V; let χ_V be its characteristic formula. We start a tableau proof for φ with term 1 $\neg\varphi^V \wedge \chi_V$, with φ^V being the formula in which if M is an occurrence of an action model, it has been changed to M^V.

Furthermore, we need to take into account that DELVO action models are capable of change propositional variables (but not vision atoms), so we keep rules $[\leftarrow_p]$ and $[\leftarrow_{\neg p}]$ only for vision atoms and we add:

$$[O]\ \frac{\sigma\ \mathsf{M}_1, \mathsf{w}_1; \ldots; \mathsf{M}_n, \mathsf{w}_n\ p}{\sigma\ p} \text{ if } p \notin \triangle\{\mathsf{tgl}(\mathsf{w}_1), \ldots, \mathsf{tgl}(\mathsf{w}_n)\}.$$

$$\frac{\sigma\ \mathsf{M}_1, \mathsf{w}_1; \ldots; \mathsf{M}_n, \mathsf{w}_n\ p}{\sigma\ \neg p} \text{ if } p \in \triangle\{\mathsf{tgl}(\mathsf{w}_1), \ldots, \mathsf{tgl}(\mathsf{w}_n)\}.$$

$$[O^*]\ \frac{\sigma\ \mathsf{M}_1, \mathsf{w}_1; \ldots; \mathsf{M}_n, \mathsf{w}_n\ \neg p}{\sigma\ \neg p} \text{ if } \neg p \notin \triangle\{\mathsf{tgl}(\mathsf{w}_1), \ldots, \mathsf{tgl}(\mathsf{w}_n)\}.$$

$$\frac{\sigma\ \mathsf{M}_1, \mathsf{w}_1; \ldots; \mathsf{M}_n, \mathsf{w}_n\ \neg p}{\sigma\ p} \text{ if } p \in \triangle\{\mathsf{tgl}(\mathsf{w}_1), \ldots, \mathsf{tgl}(\mathsf{w}_n)\}.$$

Last, we have to take into account that the vision function impacts not only the action models but the accessibility relations in the epistemic models. It also holds that vision functions are common knowledge. For the above we add the rules:

$$[V]\ \frac{\begin{array}{c}\sigma\ V_i p \\ \sigma'\ p \\ \sigma''\ \neg p\end{array}}{\bot} \qquad \text{where } \sigma' \text{ is i-accessible from } \sigma''$$

4.4.2 Complexity

Choosing a vision function is another non-deterministic choice (of polynomial space). All of the modifications we made for DELVO, writing χ_V, changing M to M^V, checking if a variable belongs to the symmetric difference, can all be done in exponential time. There is therefore no change in complexity moving from DEL to DELVO:

Theorem 4.13. *DELVO-SAT is in NEXPTIME.*

Chapter 5

Model Checking for DEL and DELVO

5.1 Model Checking

The other important decision problem we will examine to pinpoint its complexity is that of Model Checking. For $\mathcal{L}_{\mathrm{DEL}}$ it is defined as follows:

Input: A model $(M, w) \in \sigma(\mathcal{EM})$ and a formula $\varphi \in \mathcal{L}_{\mathrm{DEL}}$

Output: YES iff $M, w \models \varphi$.

The sections that follow show that this problem is PSPACE-complete.

5.1.1 Lower bound for DEL Model Checking

We would like to reduce a PSPACE-complete problem using a polynomial time reduction, to DEL Model Checking, to show that the latter is PSPACE-hard. The problem used is the Quantified Boolean Formula (abbreviated as QBF) problem, known to be PSPACE-complete [51, p. 455]. The QBF problem is defined as follows:

Input: A formula $\varphi = Q_1 p_1 \dots Q_n p_n \varphi_0$, where φ_0 is a propositional formula with variables $p_1, \dots, p_n$, and for $(i = 1, \dots, n)$, Q_i is either $\forall$ or $\exists$.

Output: YES iff φ is true.[1]

Regarding the intuition behind the reduction that will follow: when checking whether $M, w \models_{\mathrm{DEL}} \varphi$ we need to evaluate the truth of some subformulas of φ in a number of epistemic models; either the original one, or one of the model products. But this is something that we can already do with model checking for S5 (which has lower complexity than PSPACE, namely PTIME); model products are epistemic models themselves. A part of the added value from the use of action models and the non-deterministic choice operator in the logical language, is that we can require the evaluation of truth in a 'large'

[1] In other pieces of literature the reader might see "satisfiable" instead of "true". The latter is probably preferable given that we make clear that the only variables of φ are amongst $p_1, \dots, p_n$.

number (relative to the size of the input) of worlds or model products. This is done by using multi-pointed action models (or the choice between single-pointed action models) in sequence. Finally, it should be noted that only DEL with a single agent suffices for the reduction, therefore the latter is proven to be PSPACE-hard. But model checking for single-agent DEL is obviously a subproblem of multi-agent DEL with k agents, and so we have the following theorem[2]

Theorem 5.1. *The model checking problem of $\mathcal{L}_{DEL}$ with k agents ($k \geq 1$) is PSPACE-hard.*

Proof. Let $\varphi = Q_1 p_1 \dots Q_n p_n \ \varphi_0$. As the theorem gives away, we will be using DEL with one agent. We define $M = M(n) = \langle W, R, f \rangle \in \mathcal{EM}$, where $W = \{w_0, \dots, w_n\}$; $R = W \times W$; and for $i = \{1 \dots, n\}$ $f(w_i) = \{p_i\}$, and $f(w_0) = \emptyset$.

Furthermore, for each variable p_i we define $\mathsf{M}_i = \langle \mathsf{W}, \mathsf{R}, \mathsf{pre}_i \rangle \in \mathbb{AM}$, where $\mathsf{W} = \{\mathsf{w}, \mathsf{v}\}$; $\mathsf{R} = \{(\mathsf{w}, \mathsf{w}), (\mathsf{v}, \mathsf{v})\}$; $\mathsf{pre}_i(\mathsf{w}) = \top$, $\mathsf{pre}_i(\mathsf{v}) = \neg p_i$.

The epistemic model M starts with a single cluster of worlds, with each world w_i acting as the witness to variable p_i being true somewhere in the cluster. By applying any of the action models M_i we non-deterministically keep or eliminate p_i in the cluster. More precisely, we get a model product comprised of 2 clusters, one with p_i, the other one without. Applying all of the models in sequence would give us 2^n clusters; one for each subset over the variables at hand (or *valuations* over the variables at hand). It now becomes more straightforward that we should be using box modalities for $\forall$ quantifiers, and diamond modalities for $\exists$ quantifiers. Furthermore, we have connected valuations with the existence (or lack thereof) of variables in the cluster.

To realise this connection, we define $t(\varphi)$ to be φ where he have substituted every $\forall p_i$ with $[M_i]$, every $\exists p_i$ with $\langle M_i \rangle$ (shorthand for $\neg[M_i]\neg$), and every instance of p_i in φ_0 (the propositional part of φ) is substituted with $\langle K \rangle p_i$ ($\langle K \rangle$ being shorthand for $\neg K \neg p_i$).

We show that for any QBF φ, we have $\varphi \Leftrightarrow M, w_0 \models t(\varphi)$. We can prove this by induction on the number of variables. We skip the base case for $n = 1$ as the arguments are a subset of those for the induction step.

For n: Suppose the induction hypothesis is true for any QBF with at most $n-1$ variables. Suppose $\varphi = Q_1 p_1 \dots Q_{n-1} p_{n-1} \forall p_n \ \varphi_0$. Consider formula $\varphi' = Q_1 p_1 \dots Q_{n-1} p_{n-1}$ $(\varphi_0[p_n \leftarrow \top] \wedge \varphi_0[p_n \leftarrow \bot])$[3] . It only takes propositional reasoning to see that $\varphi \Leftrightarrow \varphi'$. Based on the induction hypothesis $\varphi' \Leftrightarrow M(n-1), w_0 \models t(\varphi')$. We also have $M(n-1), w_0 \models t(\varphi') \Leftrightarrow M(n), w_0 \models t(\varphi')$, since as far as variables $p_1, \dots, p_{n-1}$ are concerned, the extra world w_n has the same valuation as w_0, all those variables are false (w_0 and w_n only differ at p_n). I henceforth refer to $M(n)$ as M. It now remains to show that $M, w_0 \models t(\varphi') \Leftrightarrow M, w_0 \models t(\varphi)$.

[2]It should be noted that the same result with a similar proof appears in [35]. Work by the respective authors was conducted independently.

[3]We skip the case where $Q_n = \exists$. In that case we define $\varphi' = Q_1 p_1 \dots Q_{n-1} p_{n-1}$ $(\varphi_0[p_n \leftarrow \top] \vee \varphi_0[p_n \leftarrow \bot])$

We have that $t(\varphi') = \mathsf{Q}_1 \dots \mathsf{Q}_{n-1}\psi'_0$ where $\psi'_0 = \psi'_\top \wedge \psi'_\bot$, $\psi'_\top = \varphi_0[p_1 \leftarrow \langle K\rangle p_1] \dots [p_{n-1} \leftarrow \langle K\rangle p_{n-1}][p_n \leftarrow \top]$, $\psi'_\bot = \varphi_0[p_1 \leftarrow \langle K\rangle p_1] \dots [p_{n-1} \leftarrow \langle K\rangle p_{n-1}][p_n \leftarrow \bot]$, and Q_i is either $[\mathsf{M}_i]$ or $\langle \mathsf{M}_i\rangle$. Also $t(\varphi) = \mathsf{Q}_1 \dots \mathsf{Q}_{n-1}[\mathsf{M}_n]\psi_0$ where $\psi_0 = (\varphi_0[p_1 \leftarrow \langle K\rangle p_1] \dots [p_n \leftarrow \langle K\rangle p_n]$. As $t(\varphi')$ and $t(\varphi)$ share the same prefix $\mathsf{Q}_1 \dots \mathsf{Q}_{n-1}$ it suffices to show that $M \otimes \mathsf{M}_1 \otimes \dots \otimes \mathsf{M}_{n-1}, (w_0, \mathsf{u}_1, \dots, \mathsf{u}_{n-1}) \models \psi'_0$ iff $M \otimes \mathsf{M}_1 \otimes \dots \otimes \mathsf{M}_{n-1}, (w_0, \mathsf{u}_1, \dots, \mathsf{u}_{n-1}) \models [\mathsf{M}_n]\psi_0$. We can in fact show that $(w_0, \mathsf{u}_1, \dots, \mathsf{u}_{n-1}) \models \psi'_\top$ iff $(w_0, \mathsf{u}_1, \dots, \mathsf{u}_{n-1}, \mathsf{w}_n) \models \psi_0$, and $(w_0, \mathsf{u}_1, \dots, \mathsf{u}_{n-1}) \models \psi'_\bot$ iff $(w_0, \mathsf{u}_1, \dots, \mathsf{u}_{n-1}, \mathsf{v}_n) \models \psi_0$.

Note that $(w_0, \mathsf{u}_1, \dots, \mathsf{u}_{n-1}) \models \langle K\rangle p_n$, as this is the case in the original model M and none of the points in $\mathsf{M}_1, \dots, \mathsf{M}_{n_1}$ have $\neg p_n$ as precondition. By then applying M_n we have $(w_0, \mathsf{u}_1, \dots, \mathsf{w}_n) \models \langle K\rangle p_n$ and $(w_0, \mathsf{u}_1, \dots, \mathsf{w}_n) \not\models \langle K\rangle p_n$. Therefore $(w_0, \mathsf{u}_1, \dots, \mathsf{w}_n) \models \langle K\rangle p_n \leftrightarrow \top$ and $(w_0, \mathsf{u}_1, \dots, \mathsf{v}_n) \models \langle K\rangle p_n \leftrightarrow \bot$. Furthermore for $i \neq n$, $(w_0, \mathsf{u}_1, \dots, \mathsf{u}_{n-1}) \models \langle K\rangle p_i$ iff $(w_0, \mathsf{u}_1, \dots, \mathsf{w}_n) \models \langle K\rangle p_i$ iff $(w_0, \mathsf{u}_1, \dots, \mathsf{v}_n) \models \langle K\rangle p_i$. Because of the above and that $\psi'_\top = \psi_0[\langle K\rangle p_n \leftarrow \top]$ we indeed have $(w_0, \mathsf{u}_1, \dots, \mathsf{u}_{n-1}) \models \psi'_\top$ iff $(w_0, \mathsf{u}_1, \dots, \mathsf{u}_{n-1}, \mathsf{w}_n) \models \psi_0$. Similarly for $\psi'_\bot$.

Finally, note that the reduction can be done in time polynomial, with respect to the size of the input $|\varphi| = m$. We need to write down $M, w \models t(\varphi)$. The variables in φ are at most m. For the epistemic model M we need to write down $m + 1$ worlds, we may want to write down the pairs in R (so all of the pairs) and a valuation for each of these worlds. Respectively we would need time $O(m)$, $O(m^2)$, $O(m) * O(m)$; the sum of these being $O(m^2)$. And for $t(\varphi)$ we have that each action model we use has constant size and there are m action models so to write those we need time $O(m)$. Then for the propositional part of the QBF we substitute at most m symbols of size 1 (the variables) with something again constant in size, that would take time $O(m)$. For the sum of all of the above we would need time $O(m^2)$. □

As with the satisfiability problem, it becomes apparent that also in model checking the existence of the non-deterministic choice operator $\cup$ plays a crucial part. With it we can come up with a formula that includes evaluation a formula in an exponential number of worlds, as the reduction shows, and therefore, certainly polynomial would not be enough. This also explains why it is natural for model checking of DEL, even restricted to a single agent, to be of higher complexity than model checking of Public Announcement Logic (which as mentioned in the satisfiability section does not include a choice operator), which is in P [43][4].

Example 5.1. Let $\varphi = \forall p_1 \exists p_2\ (p_1 \wedge \neg p_2)$. Then, in accordance with the proof, we have $t(\varphi) = [M_1]\langle M_2\rangle(\langle K\rangle p_1 \wedge \neg\langle K\rangle p_2)$. Equivalently $[M_1]\langle M_2\rangle(\langle K\rangle p_1 \wedge K\neg p_2)$.

Furthermore $M_\varphi = \langle W_\varphi, R_\varphi, f_\varphi\rangle$ where $W_\varphi = \{w_0, w_1, w_2\}, R = W \times W$, $f(w_1) = \{p_i\}$, $f(w_2) = \{p_2\}$, and $f(w_0) = \emptyset$.

Finally $\mathsf{M}_1 = \langle \mathsf{W}, \mathsf{R}, \mathsf{pre}_1\rangle$, where $\mathsf{W} = \{\mathsf{w}, \mathsf{v}\}$; $\mathsf{R} = \{(w, w), (v, v)\}$; $\mathsf{pre}_1(\mathsf{w}) = \top$, $\mathsf{pre}_1(\mathsf{v}) = \neg p_1$. Similarly for p_2. The models above can be seen in Figure 5.1.

[4] In fact from [43] we have that model checking of PAL with underlying logic K is in P, but it is then obvious that model checking of PAL with S5 is in P as well.

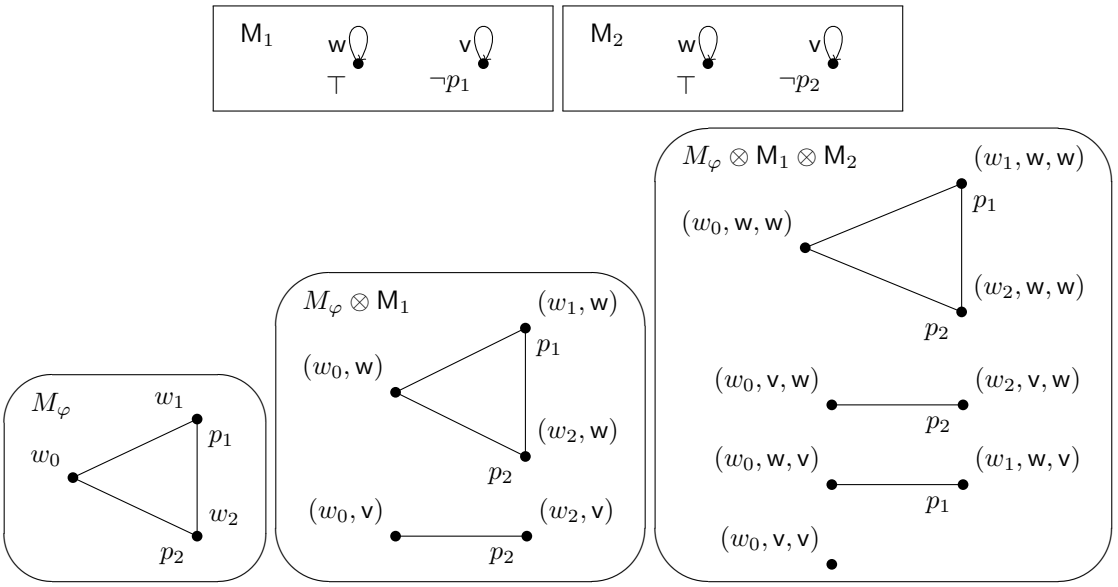


FIGURE 5.1: The epistemic and action models involved in the reduction from the QBF problem to Model Checking of DEL. Reflexive arrows in the epistemic models are not drawn.

Breaking down $t(\varphi)$ using the semantics of DEL we have $M, w \models t(\varphi)$ iff

$$\big(M \otimes \mathsf{M}_1 \otimes \mathsf{M}_2, (w_0, \mathsf{w}, \mathsf{w}) \models \langle K \rangle p_1 \wedge K \neg p_2$$

or

$$M \otimes \mathsf{M}_1 \otimes \mathsf{M}_2, (w_0, \mathsf{w}, \mathsf{v}) \models \langle K \rangle p_1 \wedge K \neg p_2\big)$$

and

$$\big(M \otimes \mathsf{M}_1 \otimes \mathsf{M}_2, (w_0, \mathsf{v}, \mathsf{w}) \models \langle K \rangle p_1 \wedge K \neg p_2$$

or

$$M \otimes \mathsf{M}_1 \otimes \mathsf{M}_2, (w_0, \mathsf{v}, \mathsf{v}) \models \langle K \rangle p_1 \wedge K \neg p_2\big).$$

In Figure 5.1, the submodels generated from $(w_0, \mathsf{w}, \mathsf{w}), (w_0, \mathsf{v}, \mathsf{w}), (w_0, \mathsf{w}, \mathsf{v}), (w_0, \mathsf{v}, \mathsf{v})$ each corresponds respectively to valuations $(T, T), (F, T), (T, F), (F, F)$ for (p_1, p_2). Observe that the four clauses above is each evaluated in a different submodel. It is trivial to check that from these four clauses only the third is true (and so $M, w_0 \not\models t(\varphi)$). This is in agreement with what we would expect by directly solving the QBF problem for φ. If p_1 is true, then only if p_2 is false ($p_1 \wedge \neg p_2$) holds. If on the other hand p_1 is false then ($p_1 \wedge \neg p_2$) is false regardless of the value of p_2.

5.1.2 Upper bound for DEL Model Checking

Algorithm 1 is a deterministic algorithm M-CHECK($w \quad \mathsf{M}_1, \mathsf{w}_1; \ldots; \mathsf{M}_i, \mathsf{w}_i, \ \varphi$) ($i$ can be zero), that checks whether we have $(M \otimes \mathsf{M}_1 \otimes \cdots \otimes \mathsf{M}_i, (\mathsf{w}_1, \ldots, \mathsf{w}_i)) \models \varphi$; where $(M, w) \in \sigma(\mathcal{EM})$ and for all $j \in \{1, \ldots, i\}$, $(\mathsf{M}_j, \mathsf{w}_j) \in \sigma(\mathbb{AM})$. Therefore, to check

whether $M, w \models \varphi$ we just execute M-CHECK(w, φ). This algorithm, along with the proof that it is in PSPACE, is originally found in [2]. The context of [2] is DEL with arbitrary accessibility relations, but trivially the same algorithm also works for DEL with equivalences as accessibility relations.

Theorem 5.2. *The model checking problem of $\mathcal{L}_{DEL}$ with k agents ($k \geq 1$), is in PSPACE.*

```
function M-Check (w  M_1, w_1; ...; M_i, w_i,  φ)
    case p:
        return p ∈ f(w)
    case ¬ψ:
        return not M-Check(w  M_1, w_1; ...; M_i, w_i,  φ)
    case ψ_1 ∧ ψ_2:
        return M-Check(w  M_1, w_1; ...; M_i, w_i,  ψ_1) and
          M-Check(w  M_1, w_1; ...; M_i, w_i,  ψ_2)
    case K_i ψ:
        for u ∈ R_a(w) do
            for u_1 ∈ R_a(w_1) do
                if M-Check(u,  pre(w_1)) then
                    ⋮
                    for u_i ∈ R_a(u_i) do
                        if M-Check(u  M_1, u_1; ...; M_{i-1}, u_{i-1},  pre(u_i)) then
                            if not M-Check(u  M_1, u_1; ...; M_i, u_i,  ψ) then
                                return false
                            end
                        end
                    end
                end
            end
        end
        return true
    case [M, w]ψ:
        if M-Check(w  M_1, w_1; ...; M_i, w_i,  pre(w)) then
            return M-Check(w  M_1, w_1; ...; M_i, w_i; M, w,  ψ)
        end
        return true
    case [α ∪ β]ψ:
        return M-Check(w  M_1, w_1; ...; M_i, w_i,  [α]ψ) and
          M-Check(w  M_1, w_1; ...; M_i, w_i,  [β]ψ)
endfunction
```

Algorithm 1: PSPACE algorithm for model checking

5.1.3 Lower and upper bound for DELVO Model Checking

While DEL Model Checking is not, strictly speaking, a subproblem of DELVO Model Checking, it becomes evident from Theorem 3.14 that we can easily (in polynomial time) reduce the former to the latter. The size of the input in this case, say m, comes from the model $M \in \mathcal{EM}$ and the formula $\varphi \in \mathcal{L}_{DEL}$. We only need add the empty vision function to M and empty toggle sets to the actions models of φ. The vision function needs constant time for each variable, and there can be at most m variables, so we need time $O(m)$. Each empty toggle set is of constant size, and there are at most m action points, so again we need time $O(m)$. The sum of these is $O(m)$. Due to Theorem 3.14 we need not argue the correctness of the reduction. The conclusion is that DELVO Model Checking is as hard as DEL Model Checking, and therefore PSPACE-hard.

On the other hand, we can very easily modify Algorithm 1 to work for DELVO, without any impact on its complexity. For M-CHECK$(w \ \ \mathsf{M}_1, \mathsf{w}_1; \ldots; \mathsf{M}_i, \mathsf{w}_i, \ \varphi)$ in the case for p it should now be

$$\textbf{return} \ \ p \in f(w) \ \ \textit{and} \ \ p \notin \triangle\{\mathsf{tgl}(\mathsf{w}_j) \mid j \in \{1, \ldots, i\}\}$$

The additional clause needs constant size; whilst going through the toggle sets if we find p inside we need only remember whether p was in the symmetric difference up to that point. Furthermore

$$\text{“}\textbf{for} \ \mathsf{u}_i \in \mathsf{R}_a(\mathsf{u}_i)\text{”} \text{ should be replaced with “}\textbf{for} \ \mathsf{u}_i \in \mathsf{R}_a^V(\mathsf{u}_i)\text{”}$$

Remember that the definition says $\mathsf{wR}_i^V\mathsf{u}$ iff $\big(\mathsf{wR}_i\mathsf{u}$ and $\big(\mathsf{tgl}(\mathsf{w}) \triangle \mathsf{tgl}(\mathsf{u})\big) \cap V(i) = \emptyset\big)$. Again the last extra clause needs constant size to compute; we need this intersection to be the empty set, and so we can check membership for each variable one at a time by reading $\mathsf{tgl}(\mathsf{w}), \mathsf{tgl}(\mathsf{u}), V(i)$ and stop if the set is non-empty, without any need to store the result for any of the previous variables.

Putting everything together we have:

Theorem 5.3. *The model checking problem of $\mathcal{L}_{DELVO}$ with k agents ($k \geq 1$), is PSPACE-complete.*

Chapter 6

The Logic PDLVE

6.1 Introduction

We now attempt a generalisation of opaque choice and composition to a more abstract PDL-like setting. For DELVO, a large part of the intuitions and formal techniques stem from the use of action models. But action models do not exist in PDL and it is interesting to examine how it would be possible to circumvent their use. Furthermore, actions of PDL with an epistemic operator are strictly more general; to see this consider that an atomic PDL action can connect a finite component of worlds to an infinite one - this cannot be achieved by any $[\![\mathsf{M}, \mathsf{w}]\!]$ as action models are finite. Therefore with PDLVE we attempt not only to represent the opaque constructs in the PDL paradigm, but also extent their use to a richer class of actions.

6.2 The Logic PDLVE

In addition to sets Ag and $\mathcal{V}ar$, let also a countable set of atomic actions $Atom = \{a_1, a_2, \dots\}$ be given.

The language of PDLVE is defined by the following **syntax**:

$$\varphi ::= \top \mid x \mid V_i x \mid \neg\varphi \mid \varphi \wedge \varphi \mid [a]\varphi \mid K_i\varphi$$

where $a ::= \pi \in Atom \mid a \cup a \mid a \uplus a \mid a;a \mid a;;a$ and $x \in \mathcal{V}ar,\ i \in Ag$. We avoid the generalisation for $T \subseteq Ag$ consciously, so as to not further encumber notation.

For regular choice and composition there is no doubt we should keep the definitions from PDL.

To capture opaque choice in PDLVE we rely on the basic intuition behind it. Given actions a, b, and world w, "either a or b is executed at w, but the agent does not know which, unless his vision allows him so". Based on the above, we first give an auxiliary definition: given $M \in \mathcal{EMV}$ and relations for $a, b \in \mathcal{A}ct$, this definition describes a representation of the epistemic model that should be the image of $a \uplus b$.

So let $M = \langle W, R, V, f\rangle \in \mathcal{EMV}$. For $S \subseteq W$ let $N(S)$ denote the *epistemic* submodel generated by S, and $N_w(S)$ the worlds of that model. We remind that for $E \subseteq W \times W$, $E(S) = \{w \in W \mid \exists v \in S \; s.t. \; vEw\}$.

Definition 6.1. Let $M = \langle W, R, V, f\rangle \in \mathcal{EMV}$, and $E \subseteq W \times W$. Let $M' = N(E(W)) = \langle W', R', V', f'\rangle$. Finally let E be injective and $E(W) = W'$. We then define $g(M, E) = \langle W'', R'', V'', f''\rangle$ such that: $W'' = W', V'' = V', f'' = f'$,

- for $u \in E(w), z \in E(v)$, we have $uR''z$ iff wRv and $\big((f(w) \triangle f''(u)) \triangle (f(v) \triangle f''(z))\big) \cap V(i) = \emptyset$

In words, given a model M and an injective relation E (that is later to play the role of choice or composition between two actions), we first get the epistemic submodel generated by worlds that are an image through E, namely M'. Note also that E is such that its co-domain is exactly the set of worlds of M'. Then we 'tinker' M' to get the model $g(M, E)$, that is the same as M' but the agents cannot distinguish between worlds that are images through E ($uR''z$), if and only if the origin worlds were indistinguishable to begin with (wRv) *and* their vision does not help them notice a difference in the changes caused among the possible executions; that is, $\big((f(w) \triangle f''(u)) \triangle (f(v) \triangle f''(z))\big) \cap V(i) = \emptyset\}$.

We also ought to explain why we require E to be injective and such that $E(W) = W'$. The reason is our goal to find a way to transfer our ideas from the DELVO setting but to circumvent the use of action models. The induced relation $[\![\mathsf{M}, \mathsf{w}]\!]$ from any pointed action model has these two properties: intuitively speaking, each one world in a model product is the result of applying the action model to one and only one world in the original model ($[\![\mathsf{M}, \mathsf{w}]\!]$ is injective), and also each world in a model product is indeed the result of some application of the action model to the original model ($[\![\mathsf{M}, \mathsf{w}]\!](W) = W'$). These properties usually elude our attention but are blatantly there; each world in a model product is of the form $(w, \mathsf{w}_1, \ldots, \mathsf{w}_n)$ and this form is unique. Furthermore, our whole endeavour of whether the agents can distinguish between two actions or not, relies on them being able to *observe* differences in the variables between the original world and the resulting world(s). So there should be an original world and it has to be unique, hence, for our purposes, the existence of these two properties cannot be avoided.

We can now define the class of PDLVE models (denoted by $\mathcal{DEM}$). For brevity, let $*$ denote either regular choice or composition, and $\circledast$ their opaque variant. Also let $(g(M, E), E(w)) = g(M, w, E)$.

Definition 6.2 (PDLVE Models)**.** Let $M = \langle W, R, E, V, f\rangle$ where $\langle W, R, V, f\rangle \in \mathcal{EMV}$ (Def. 3.4) and E is a family of relations $E_a \subseteq W \times W$, where a is either atomic or of the form "$b \uplus c$" or "$b; ; c$". Furthermore, E is such that:

- for $a = b \circledast c$, if E_{b*c} is an injection and $E_{b*c}(W) = N_w(E_{b*c}(W))$, then $N(E_a(w)), E_a(w) \leftrightarrows g(M, w, E_{b*c})$, otherwise $E_a = \emptyset$;

We explain the definition a bit more intuitively. First off, we keep the traditional/existing definitions of PDL untouched: we have a set of worlds as usual, indistinguishability

relations for the epistemic operator, and if α is atomic it can be arbitrary and we have no further requirements in our definition.

A natural question would then be, why do we have to involve opaque choice and composition in the definition of $\mathcal{DEM}$? This certainly does not happen with standard choice and composition. The answer to that, is that while relations $a \cup b$ and $a;b$ can always be defined using the worlds that are involved in relations a and b. For example $[\![a \cup b]\!] = [\![a]\!] \cup [\![b]\!]$. But this is not always true for the opaque variants, different worlds whose names we cannot dictate beforehand may be involved. What we do is therefore require that these worlds are at least 'correct', in the sense that they are bisimilar to the corresponding set of worlds in the model that would be the representation (Def. 6.1) of opaque choice or opaque composition.

In case not all of the requirements are met, for a relation to qualify as the opaque choice/composition of some other relations, then we simply say it is empty $E_a = \emptyset$.

We interpret formulas at pointed dynamic models as usual. And regarding standard choice and composition we define $E_{a\cup b} = E_a \cup E_b$ and $E_{a;b} = E_b \circ E_a$, again as normal.

Is the representation we have chosen a correct one? The next theorem shows that at least as far as a special subset of DELVO actions is concerned, this is indeed the case. Let, for $M \in \mathcal{EMV}$, $P_M = \{M, w \mid w \in M\}$.

Theorem 6.3. *Let $M, w \in \sigma(\mathcal{EMV})$ and $(\mathsf{M}_a, \mathsf{S}_a), (\mathsf{M}_b, \mathsf{S}_b) \in \mu(\mathbb{AMT})$ such that $\mathsf{M}_a, \mathsf{M}_b$ are disjoint, $[\![\mathsf{M}_{a*b}, \mathsf{S}_{a*b}]\!](P_M) = P_{M\otimes \mathsf{M}_{a*b}}$. Let $M' = M \cup (M \otimes \mathsf{M}_{a*b})$. Define wE_av iff $(M, w)[\![\mathsf{M}_a, \mathsf{S}_a]\!](M \otimes \mathsf{M}_{a*b}, v)$ and similarly for b. Also let $M'' = M' \sqcup g(M', E_{a\star b})$. Assume that for the disjoint union, every $w \in g(M', E_{a\star b})$ was renamed to w'. Finally, for $w \in M''$ let $E_{a\circledast b}(w) = \{v' \mid v \in E_{a*b}(w)\}$. Then $[\![\mathsf{M}_{a\circledast b}, \mathsf{S}_{a\circledast b}]\!](M, w) \leftrightarrows M'', E_{a\circledast b}(w)$.*

Proof. Let $M \otimes \mathsf{M}_{a\uplus b} = \langle W_1, R_1, V_1, f_1\rangle, M \otimes \mathsf{M}_{a\cup b} = \langle W_2, R_2, V_2, f_2\rangle$, $g(M, E_a, E_b) = \langle W_3, R_3, V_3, f_3\rangle$. The relation $\mathcal{B} \subseteq W_1 \times \{w' \mid w \in W_3\}$ that will serve as witness for our claim is the one such that $w\mathcal{B}v$ iff $v = w'$.

atom For $w \in M \otimes M_{a\uplus b}$, $f_1(w) = f_2(w)$ by definition of internal choice in DELVO (Def. 3.9, $\mathsf{tgl}' = \mathsf{tgl}$). In turn, $f_2(w) = f_3(w)$—this is consequence of Def. 6.1 ($f'' = f'$). And finally, $f_3(w) = f(w')$ by definition of disjoint union. Also the vision function is constant everywhere; $V(w') = V_1(w) = V_2(w) = V_3(w)$.

forth Let $(w, \mathsf{w})R_1(v, \mathsf{v})$. If $(w, \mathsf{w}) \in [\![\mathsf{M}_{a\uplus b}, \mathsf{S}_{a\uplus b}]\!](M, w)$ and $(v, \mathsf{v}) \in [\![\mathsf{M}_{a\uplus b}, \mathsf{S}_{a\uplus b}]\!](M, v)$ then we have wRv and $\big(\mathsf{tgl}(\mathsf{w})\triangle\mathsf{tgl}(\mathsf{v})\big) \cap V(i) = \emptyset\big)$. This means wRv and $\big((f(w)\triangle f_1(w, \mathsf{w}))\triangle(f(v)\triangle f_1(v, \mathsf{v}))\big) \cap V(i) = \emptyset$. By Def. 6.1 we have $(w, \mathsf{w})R_3(v, \mathsf{v})$ and so $(w, \mathsf{w})'R(v, \mathsf{v})'$.

back A similar argument applies. □

With Theorem 6.3 we test the correctness of opaque choice between PDLVE actions a and b, by checking that if we apply PDLVE opaque choice instead of DELVO opaque choice - when we are given a pointed model $M, w \in \sigma(\mathcal{EMV})$, and actions a and b can

be thought of as actions of DELVO - we get the same (bisimilar) result. The same holds for composition.

Theorem 6.4. *Let* $M, w \in \sigma(\mathcal{EMV})$ *and* $(\mathsf{M}_a, \mathsf{S}_a), (\mathsf{M}_b, \mathsf{S}_b) \in \mu(\mathbb{AMT})$ *such that* $\mathsf{M}_a, \mathsf{M}_b$ *are disjoint,* $[\![M_{a;b}, S_{a;b}]\!](P_M) = P_{M \otimes \mathsf{M}_{a;b}}$. *Let* $M' = M \cup (M \otimes \mathsf{M}_{a;b})$. *Define* wE_av *iff* $(M, w)[\![\mathsf{M}_a, \mathsf{S}_a]\!](M \otimes \mathsf{M}_{a;b}, v)$ *and similarly for* b. *Also let* $M'' = M' \sqcup g(M', E_{a;b})$. *Assume that for the disjoint union, every* $w \in g(M', E_{a;b})$ *was renamed to* w'. *Finally, for* $w \in M''$ *let* $E_{a;;b}(w) = \{v' \mid v \in E_{a;b}(w)\}$. *Then*

$$[\![\mathsf{M}_{a;;b}, \mathsf{S}_{a;;b}]\!](M, w) \leftrightarrows M'', E_{a;;b}(w).$$

Proof. The proof is similar to that of Theorem 6.3. Let $M \otimes \mathsf{M}_{a;;b} = \langle W_1, R_1, V_1, f_1\rangle, M \otimes \mathsf{M}_{a;b} = \langle W_2, R_2, V_2, f_2\rangle$, $g(M, E_a, E_b) = \langle W_3, R_3, V_3, f_3\rangle$. The relation $\mathcal{B} \subseteq W_1 \times \{w' \mid w \in W_3\}$ that will serve as witness for our claim is the one such that $w\mathcal{B}v$ iff $v = w'$.
atom For $w \in M \otimes M_{a;;b}$, $f_1(w) = f_2(w)$ by definition of internal composition in DELVO (Def. 3.11, tgl is defined as in Def. 3.10). In turn, $f_2(w) = f_3(w)$—this is consequence of Def. 6.1 ($f'' = f'$). And finally, $f_3(w) = f(w')$ by definition of disjoint union. Also the vision function is constant everywhere; $V(w') = V_1(w) = V_2(w) = V_3(w)$.
forth Let $(w, \mathsf{w})R_1(v, \mathsf{v})$. If $(w, \mathsf{w}) \in [\![\mathsf{M}_{a \uplus b}, \mathsf{S}_{a \uplus b}]\!](M, w)$ and $(v, \mathsf{v}) \in [\![\mathsf{M}_{a \uplus b}, \mathsf{S}_{a \uplus b}]\!](M, v)$ then we have wRv and $\big(\mathsf{tgl}(\mathsf{w}) \triangle \mathsf{tgl}(\mathsf{v})\big) \cap V(i) = \emptyset\big)$. This means wRv and $\big((f(w) \triangle f_1(w, \mathsf{w})) \triangle (f(v) \triangle f_1(v, \mathsf{v}))\big) \cap V(i) = \emptyset$. By Def. 6.1 we have $(w, \mathsf{w})R_3(v, \mathsf{v})$ and so $(w, \mathsf{w})'R(v, \mathsf{v})'$.
back A similar argument applies. □

6.3 Relation to DELVO

We now have two logics for describing knowledge, and epistemic & ontic change. A natural question is whether the actions of PDLVE are more general than those of DELVO. We find that, under mild restrictions, they are not.

As one would expect, the transition from DELVO to PDLVE can be done simply enough. After all DELVO can be reduced to S5+V, and epistemic models with vision are part of PDLVE models. To evaluate a given DELVO formula φ in a model $M, w \in \sigma(\mathcal{EMV})$, one would have to potentially evaluate subformulas of φ in M and in the respective model products. The idea for the transition from DELVO to PDLVE, is to define a single PDLVE model by putting together all such models.

Definition 6.5. Let $M \in \sigma(\mathcal{EMV})$ and $\varphi \in \mathcal{L}_{\text{DELVO}}$ (M may be omitted when clear from context) by induction on the complexity of φ as follows: $T_1(\top) = T_1(x) = T_1(V_i x) = \{M\}$, $T_1(\neg\varphi) = T_1(K_i\varphi) = T_1(\varphi)$, $T_1(\varphi \wedge \psi) = T_1(\varphi) \cup T_1(\psi)$, $T_1([\mathsf{M}, \mathsf{S}]\varphi) = \{N \otimes \mathsf{M} \mid N \in T_1(\varphi)\}$.

We then define $T_2^M(\varphi) = \{M' \in \mathcal{EMV} \mid \exists n \geq 0$ and $\mathsf{M}_1, \ldots, \mathsf{M}_n \in \mu(\mathbb{AMT})$ s.t. $M' \otimes \mathsf{M}_1 \cdots \otimes \mathsf{M}_n \in T_1(\varphi)\}$

$T_2^M(\varphi)$ therefore contains a tree with all the models involved in the process of evaluating φ in some $w \in M$. This idea is not new, see e.g. [10].

Theorem 6.6. *Let $\varphi \in \mathcal{L}_{DELVO}$ and $M, w \in \sigma(\mathcal{EMV})$. Let $T_3^M(\varphi) = \langle W, R, E, V, f\rangle$ such that $\langle W, R, V, f\rangle = \bigsqcup T_2^M(\varphi)$ and regarding E, for all multi-pointed action models $\mathsf{M}_i, \mathsf{S}_i$ that appear in φ it holds that, for $w \in M$ and $w' \in M'$, $wE_{a_i}v'$ iff $(M, w)[\![\mathsf{M}_i, \mathsf{S}_i]\!](M', w')$. Finally let $T_4^M(\varphi)$ be a model that has $T_3^M(\varphi)$ as a submodel but contains appropriate definitions for opaque choice and composition, so that $T_4^M(\varphi) \in \mathcal{DEM}$. Let $t : \mathcal{L}_{DELVO} \to \mathcal{L}_{PDLVE}$ be a function such that $t(\varphi)$ is φ, but for all action models $\mathsf{M}_i, \mathsf{S}_i$ that appear in φ, all its occurrences are replaced with a_i. Then it holds that $M, w \models \varphi$ iff $T_4^M(\varphi), w \models t(\varphi)$.*

For the direction "PDLVE to DELVO", the problem we will be trying to solve is, given $M, w \in \sigma(\mathcal{DEM})$ and $\varphi \in \mathcal{L}_{\text{PDLVE}}$, is there a translation t for φ, where actions of PDL have been translated into multi-pointed action models, such that $N(w), w \models t(\varphi)$? Such a translation is impossible to define homogeneously for all models of PDLVE, as action a changes depending on its relation, in each PDLVE model M. So let $M \in \mathcal{DEM}$, and $a \in Act$. Action a is perhaps executable in a number of worlds, and for each such world, the action may have a number of different possible executions. When focusing on the case where a single world w is connected through a with a world v, then the question is: is there a *multi-pointed action model* to transform $N(w), w$ into $N(v), v$?

This question relates to epistemic planning, which is undecidable for the multi-agent case [1], and it has been answered in [20] for finite S5 models (and other models under certain restrictions) and using action models with assignment to achieve ontic change. Our construct will follow the same idea, but we will be focusing on PDLVE models that

(i) have finite W

(ii) are an epistemic bisimulation contraction (cf. [11]), and

(iii) for which wE_av implies that for all $v' \in N_w(v)$, $f(w) \triangle f(v')$ is finite.

Note that for any $M, w \in \sigma(\mathcal{DEM})$ satisfying (i)-(ii), there exists a characteristic formula $\varphi_w \in \mathcal{L}_{\text{DELVO}}$ [7], i.e., for $v \in M$ we have $M, v \models \varphi_w$ iff $v = w$.

Definition 6.7. Let $M, w \in \sigma(\mathcal{DEM})$, satisfying (i)-(iii) and wE_av. We define $h(M, w, v) = \langle \mathsf{W}, \mathsf{R}, \mathsf{pre}, \mathsf{tgl}\rangle \in \mathbb{AMT}$ as:

- $\mathsf{W} = N_w(v)$, $\mathsf{R} = R|_{N_w(v)}$;
- for all $\mathsf{w} \in \mathsf{W}$, $\mathsf{pre}(\mathsf{w}) = \varphi_w$, $\mathsf{tgl}(\mathsf{w}) = f(w) \triangle f(\mathsf{w})$.

Back to the more general problem; action a can lead a world w to a multitude of worlds v_i, finite in number. A way to get an equivalent result by using action models, is to put 'side by side', via disjoint union, models $h(M, w, v_i)$. Furthermore, we can have the case where action a is executable in a multitude of worlds w_j, again finite in number. Again, this can be solved by taking the disjoint union of $h(M, w_j, v)$.

Definition 6.8. Let $M, w \in \sigma(\mathcal{DEM})$, $\varphi \in \mathcal{L}_{\text{PDLVE}}$ and a an action that appears in φ. Let M satisfy (i)-(iii). Define:

$$\mathsf{M}_a, \mathsf{S}_a = \bigsqcup_{w \in M} \bigsqcup_{v \in E_a(w)} h(M, w, v), v.$$

Theorem 6.9. *Let $M, w \in \sigma(\mathcal{DEM})$ as above and $\varphi \in \mathcal{L}_{PDLVE}$. Let $t : \mathcal{L}_{PDLVE} \mapsto \mathcal{L}_{DELVO}$ be a function such that $t(\varphi)$ is φ, but for all actions "a" that appear in φ, all its occurrences are replaced with $\mathsf{M}_a, \mathsf{S}_a$. Then it holds that $M, w \models \varphi$ iff $N(w), w \models t(\varphi)$.*

Proof. The proof is by induction on the complexity of φ. We examine only the case where $\varphi = [a]\psi$. We have that $M, w \models \varphi$ iff for all $v \in E_a(w)$, $M, v \models \psi$. By induction hypothesis, this is equivalent to saying for all $v \in E_a(w)$, $N(v), v \models t(\psi)$. It is trivial to check that $M, w \otimes \mathsf{M}_a, \mathsf{S}_a \in \mu(\mathcal{EMV})$ is epistemically bisimilar to $M, E_a(w)$, and because $t(\psi)$ is a DELVO formula, for all $v \in E_a(w)$ $N(v), v \models t(\psi)$ is equivalent to $M, w \models [\mathsf{M}_a, \mathsf{S}_a]t(\psi)$. The latter formula is $t(\varphi)$, and so $M, w \models t(\varphi)$. □

Chapter 7

Conclusion

We have provided a sound and complete axiomatisation for a logic of semi-public environments, showing that it is equivalent to epistemic logic enriched with a notion of *vision*. We introduced program models which cater for ontic and epistemic change, which also have a grounded semantics, based on the vision of the agents. Using those program models, we then gave a realistic interpretation of non-deterministic choice, and analysed the non-standard behaviour of sequential composition and iteration within the context of Semi-Public Environments. In contrast to action models, program models can be fully conceived of as syntactic objects, releasing one from the need to justify including them in the object language. Nevertheless, it is possible to take a semantic view of program models, which allowed us to compare them as mathematical structures, in particular, giving sufficient conditions on two program points being 'the same'.

There are many lines of possible future research. A rich vein of possible future work would involve lifting the various restrictions on our model. We might start with the restriction that preconditions are propositional formulas. Another restriction which might be worth softening, is the assumption that it is common knowledge which variables are seen by whom (note that this assumption is also taken in the interpreted systems paradigm: there, it is common knowledge that agents know the value of their local variables, or what the program under execution is). Those assumptions seem related, and removing them may well be a way to reason about *Knowledge-based Programs* [29], where the programs are distributed over the agents, and where it would be possible to branch in a program depending on the knowledge of certain agents. We also ought to consider actions that change vision itself, introduce higher order vision, as well as challenge the automatic connection between vision and knowledge; an agent seeing something might not necessarily mean that he realises it. An aspect of this form of omniscience is present for example in axiom $V_i x \to K_i V_i x$. Van Eijck ([23]) makes such a distinction between actual perception and capability of perception, and includes actions for commonly known changes to vision, unobserved change, and observation of an action witnessed by a group.

We pointed to two different ways to understand the interaction between knowledge dynamics and non-determinism, starting from logics DEL, PDL, and SPE. We presented a multi-pointed action model formulation of DEL to express non-deterministic choice.

We defined the logic DELVO, which is capable of handling epistemic and ontic change (for DEL-based work that also combines epistemic and ontic change see [21, 22], [8], and [20]), along with the notion of vision, and allows for a richer set of actions than those of DEL and SPE. Within DELVO, we defined both standard and opaque forms of choice and composition. We also provided a sound and complete axiomatisation for it, and a number of embedding results situating it among its related formalisms. Finally, we provided complexity results for the satisfiability and model checking problems of DEL and DELVO built on S5.

We also explored a more abstract logic, capable of epistemic and ontic change, with an even richer class of actions: PDLVE. Its semantics are based on PDL models with epistemic relations and vision atoms. Here, our contribution is the definition of opaque non-deterministic choice and composition in this PDL-like setting. Finally, we provided theorems that compare DELVO and PDLVE, thus also investigating the relation between action models, and atomic actions defined arbitrarily.

Several opportunities for future work exist; for DELVO we would like vision atoms to exist for other formulas as well, not only variables. For PDLVE there is the obvious need for a proof system. Furthermore, two papers that in our opinion are especially relevant are [23] and [14]. The former distinguishes between actual use of vision and the capability to use it. In the latter, vision is necessarily common knowledge and the notions of higher order and joint vision are introduced. One may also consider the agents not being aware of the actions being executed or some of the variables [16]. Another worthwhile endeavour would be a systematic comparison between our framework and the Situation Calculus (cf. [17, 53]) and the standard and opaque variants. The difference between the variants is demonstrated by the difference between MDP and POMDP (see for example [13, 15] and [3]).

Bibliography

[1] G. Aucher and T. Bolander, *Undecidability in epistemic planning*, Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI), 2013, pp. 3–9.

[2] G. Aucher and F. Schwarzentruber, *On the complexity of dynamic epistemic logic*, CoRR **abs/1310.6406** (2013).

[3] F. Bacchus, J. Y. Halpern, and H. J. Levesque, *Reasoning about noisy sensors and effectors in the situation calculus*, Artificial Intelligence **111** (1999), no. 1, 171–208.

[4] P. Balbiani, A. Herzig, and N. Troquard, *Dynamic logic of propositional assignments: A well-behaved variant of PDL*, Proceedings of the 2013 28th Annual ACM/IEEE Symposium on Logic in Computer Science (Washington, DC, USA), LICS '13, IEEE Computer Society, 2013, pp. 143–152.

[5] A. Baltag and L.S. Moss, *Logics for epistemic programs*, Synthese **139** (2004), 165–224.

[6] A. Baltag, L.S. Moss, and S. Solecki, *The logic of common knowledge, public announcements, and private suspicions*, Proceedings of the 7th conference on theoretical aspects of rationality and knowledge (TARK 98) (I. Gilboa, ed.), 1998, pp. 43–56.

[7] J. Barwise and L. S. Moss, *Vicious circles : on the mathematics of non-wellfounded phenomena*, CSLI Publications, Stanford, 1996.

[8] J. van Benthem, J. van Eijck, and B. Kooi, *Logics of communications and change*, Information and Computation **204** (2006), 1620–1662.

[9] J. van Benthem, J. Gerbrandy, T. Hoshi, and E. Pacuit, *Merging frameworks for interaction*, Journal of Philosophical Logic **38** (2009), no. 5, 491–526.

[10] J. van Benthem and E. Pacuit, *The tree of knowledge in action: towards a common perspective*, Advances in Modal Logic (AiML), 2006.

[11] P. Blackburn, J. van Benthem, and F. Wolter, *Handbook of modal logic*, Studies in logic and practical reasoning, Elsevier, 2007.

[12] P. Blackburn, M. de Rijke, and Y. Venema, *Modal logic*, Cambridge University Press: Cambridge, England, 2001.

[13] C. Boutilier, R. Reiter, M. Soutchanski, and S. Thrun, *Decision-theoretic, high-level agent programming in the situation calculus*, Proceedings of the 17th National Conference on Artificial Intelligence and 12th Conference on Innovative Applications of Artificial Intelligence, AAAI Press, 2000, pp. 355–362.

[14] T. Charrier, A. Herzig, E. Lorini, F. Maffre, and F. Schwarzentruber, *Building epistemic logic from observations and public announcements*, International Conference on Principles of Knowledge Representation and Reasoning (KR), AAAI Press, 2016, pp. 268–277.

[15] J. P. Delgrande and H. J. Levesque, *A formal account of nondeterministic and failed actions*, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, IJCAI '13, AAAI Press, 2013, pp. 861–868.

[16] H. van Ditmarsch, T. French, and F. R. Velázquez-Quesada, *Action models for knowledge and awareness*, International Conference on Autonomous Agents and Multiagent Systems (W. van der Hoek, L. Padgham, V. Conitzer, and M. Winikoff, eds.), IFAAMAS, 2012, pp. 1091–1098.

[17] H. van Ditmarsch, A. Herzig, and T. De Lima, *From situation calculus to dynamic epistemic logic*, Journal of Logic and Computation **21** (2011), no. 2, 179.

[18] H. P. van Ditmarsch, W. van der Hoek, and B. P. Kooi, *Dynamic epistemic logic with assignment*, Proceedings of the 4th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS) (New York, USA) (F. Dignum, V. Dignum, S. Koenig, S. Kraus, M.P. Singh, and M. Wooldridge, eds.), ACM Inc, 2005, pp. 141–148.

[19] H. P. van Ditmarsch, W. van der Hoek, and B. P. Kooi, *Dynamic epistemic logic*, Springer-Verlag: Berlin, Germany, 2007.

[20] H. P. van Ditmarsch and B. P. Kooi, *Semantic results for ontic and epistemic change*, Logic and the Foundations of Game and Decision Theory (G. Bonanno, W. van der Hoek, and M. Wooldridge, eds.), Amsterdam University Press, 2008, pp. 87–117.

[21] J. van Eijck, *Dynamic epistemic modelling*, 2004, Technical Report SEN-E0424.

[22] J. van Eijck, *Guarded actions*, 2004.

[23] J. van Eijck, *Perception and change in update logic*, Games, Actions and Social Software (J. van Eijck and R. Verbrugge, eds.), Springer-Verlag, Berlin, Heidelberg, 2012, pp. 119–140.

[24] J. van Eijck and J. Ruan, *Action emulation*, Tech. report, CWI, Amsterdam, the Netherlands, 2004, `http://homepages.cwi.nl/~jve/papers/04/ae/ae.pdf`.

[25] J. van Eijck, J. Ruan, and T. Sadzik, *Action emulation*, Synthese **185** (2012), no. 1, 131–151 (English).

[26] P. van Emde Boas, *The convenience of tilings*, Complexity, Logic, and Recursion Theory, Marcel Dekker Inc, 1997, pp. 331–363.

[27] R. Fagin and J. Halpern, *Belief, awareness, and limited reasoning*, Artificial Intelligence **34** (1988), 39–76.

[28] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi, *Reasoning about knowledge*, The MIT Press: Cambridge, MA, 1995.

[29] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi, *Knowledge-based programs*, Distrib. Comput. **10** (1997), no. 4, 199–225.

[30] M. Fitting, *Proof methods for modal and intuitionistic logics*, D. Reidel Publishing Company: Dordrecht, The Netherlands, 1983, (Synthese library volume 169).

[31] M. Fitting and R. L. Mendelsohn, *First-order modal logic*, Kluwer Academic Publishers, Norwell, MA, USA, 1999.

[32] J. Gerbrandy, *Logics of propositional control*, Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS) (Hakodate, Japan), 2006, pp. 193–200.

[33] R. Goldblatt, *Logics of time and computation (csli lecture notes number 7)*, Center for the Study of Language and Information, Ventura Hall, Stanford, CA 94305, 1987, (Distributed by Chicago University Press).

[34] D. Grossi, W. van der Hoek, C. Moyzes, and M. Wooldridge, *Program models and semi-public environments*, Journal of Logic and Computation (2016).

[35] R. de Haan and I. van de Pol, *On the computational complexity of model checking for dynamic epistemic logic with S5 models*, CoRR **abs/1805.09880** (2018).

[36] J. Y. Halpern and Y. Moses, *A guide to completeness and complexity for modal logics of knowledge and belief*, Artificial Intelligence **54** (1992), 319–379.

[37] D. Harel, D. Kozen, and J. Tiuryn, *Dynamic logic*, The MIT Press: Cambridge, MA, 2000.

[38] A. Herzig, T. de Lima, E. Lorini, and N. Troquard, *A computationally grounded dynamic logic of agency, with an application to legal actions*, Deontic Logic in Computer Science (T. Ågotnes, J. Broersen, and D. Elgesem, eds.), Springer Berlin Heidelberg, 2012, pp. 170–183.

[39] W. van der Hoek, P. Iliev, and M. Wooldridge, *Knowledge and action in semi-public environments*, Logic, Rationality, and Interaction (LORI) (J. Lang H. van Ditmarsch and S. Ju, eds.), LNCS, vol. 6953, 2011, pp. 97–110.

[40] W. van der Hoek, N. Troquard, and M. Wooldridge, *Knowledge and control*, Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS) (K. Tumer, P. Yolum, L. Sonenberg, and P. Stone, eds.), 2011, pp. 719–726.

[41] W. van der Hoek, D. Walther, and M. Wooldridge, *Reasoning about the transfer of control*, JAIR **37** (2010), 437–477.

[42] G. E. Hughes and M. J. Cresswell, *Introduction to modal logic*, Methuen and Co., Ltd., 1968.

[43] B. Kooi and J. van Benthem, *Reduction axioms for epistemic actions*, Advances in Modal Logic (2004), 197–211.

[44] C. D. Koutras, C. Moyzes, C. Nomikos, and Y. Zikos, *On the 'in many cases' modality: tableaux, decidability, complexity, variants*, Artificial Intelligence: Methods and Applications - 8th Hellenic Conference on AI (SETN). Proceedings (Cham) (A. Likas, K. Blekas, and D. Kalles, eds.), Springer International Publishing, 2014, pp. 207–220.

[45] C. D. Koutras, C. Moyzes, and Y. Zikos, *A modal logic of knowledge, belief, and estimation*, Logics in Artificial Intelligence: 14th European Conference (JELIA). Proceedings (Cham) (E. Fermé and J. Leite, eds.), Springer International Publishing, 2014, pp. 637–646.

[46] R. E. Ladner, *The computational complexity of provability in systems of modal propositional logic*, SIAM Journal of Computing (1977).

[47] H. J. Levesque, *All I know: a study in autoepistemic logic*, Artificial Intelligence **42** (1990), no. 2–3, 263–309.

[48] C. Lutz, *Complexity and succinctness of public announcement logic*, Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems (New York, NY, USA), AAMAS '06, ACM, 2006, pp. 137–143.

[49] J.-J. Ch. Meyer and W. van der Hoek, *Epistemic logic for AI and computer science*, Cambridge University Press: Cambridge, England, 1995.

[50] G. Mints, *A short introduction to modal logic*, CSLI Lecture Notes, no. 30, CSLI Publications, Stanford, USA, 1992.

[51] C. H. Papadimitriou, *Computational complexity*, Addison-Wesley, 1994.

[52] R B. Scherl and H. J. Levesque, *The frame problem and knowledge-producing actions*, Artificial Intelligence **144** (2003), no. 1-2, 1–39.

[53] R. B. Scherl and H. J. Levesque, *Knowledge, action, and the frame problem*, Artificial Intelligence **144** (2003), no. 1, 1–39.

[54] B. Schipper, *Awareness and knowledge*, Handbook of Epistemic Logic (J. Y. Halpern, H. van Ditmarsch, W. van der Hoek, and B. Kooi, eds.), College Publications, 2015.

[55] F. Sietsma and J. van Eijck, *Action emulation between canonical models*, Journal of Philosophical Logic **42** (2013), no. 6, 905–925.