Complexity Study of Knowledge and Public Observation

Avijeet Ghosh

Supervisor: Prof. Sourav Chakraborty
Prof. Sujata Ghosh

Submission Date: 12th December 2024

Advanced Computing and Microelectronics Unit
Indian Statistical Institute
203 B. T. Road, Kolkata-700108

A thesis submitted in partial fulfillment of the requirements for the degree of *Doctor of Philosophy* in *Computer Science* at *Indian Statistical Institute*

Dedicated to my family

Acknowledgements

Abstract

Automated planning has been a steady branch of research in the field of Artificial Intelligence. A very interesting branch of such planning studies is epistemic planning. Epistemic plans are such plans where the attained goal revolves around knowledge of some intelligent agents. One of the more popular modeling techniques and underlying language to handle knowledge of intelligent agents is provided by dynamic epistemic logic (DEL). It uses Kripke models that have possible states of truth and relations to model knowledge. It also uses a similar technique to model actions or events that update the knowledge state. Since DEL deals with how actions updating knowledge, it provides a natural way to deal with epistemic planning. Besides DEL, there are several other logical frameworks as well dealing with change in knowledge. One of them is epistemic temporal logic LTL_k. Deciding *plan-existence* problem using such logical frameworks are quite complicated and lie in the higher complexity classes. Epistemic planning has been proven to be undecidable. This is because in order to find a finite sequence of actions from a finite set, the updated knowledge models become arbitrarily large. The main motivation behind this thesis is this: Coming up with a decidable and much easier fragment of epistemic planning.

In order to do that, we have taken the help of the modeling techniques and language of the logical framework called public observation logic (POL). POL deals with how public observations change knowledge of intelligent agents. This framework uses a similar modelling technique as DEL that uses Kripke models. However, in POL, each possible state of the model has a regular expression assigned to it. This expression expresses the set of expected observations in that particular possible state. The formulas also use operators with regular expressions to denote change in knowledge after some public observation. This thesis majorly deals with two problems:

The first problem is the model-checking problem of POL. Here this thesis investigates the decidability of the problem. Moreover it also gives a tight complexity result. In addition to that, it also deals with fragments whose model-checking lie in much easier complexity classes. This problem finds application in epistemic planning. Considering the atomic observations as actions, using the Kleene star operator in regular expressions, one can decide whether there is a finite sequence of actions or a plan to achieve some change in knowledge.

The second problem is the satisfiability problem of POL. Just like the model-checking problem, this thesis also investigates various fragments. Infact, it provides tight complexity results for the satisfiability of such fragments. For the full language decidability, this thesis uses careful application of automata techniques to build an automata like model which is equivalent to a POL model.

Even though model-checking decides existence of a plan in a specific model, satisfiability finds utility in deciding the plan-existence problem for not just a specific model, but a class of models. In addition to all of this, this thesis also provides interesting connections of POL with LTL_k as well as much popular public announcement logic.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

Building intelligent systems has been an important aim of Artificial Intelligence (AI) research from the beginning. From recommendations in YouTube, to far more important domains such as finance and healthcare, building intelligent automated systems has been a priority.

For the last few decades, designing various logical systems, and studying their computational behaviors have been of utmost importance in building such automated systems [50]. Formal logical systems that reason about various aspects of intelligent agents, such as observability in the environment, state of the system, actions of the agents, and others, garnered much importance in such studies.

Among these aspects, one of the most important ones for the agents to reason on is knowledge. In particular, such logical frameworks formalise the notion of what an agent *knows*. For example, the *epistemic logic* [37]. This framework formalises knowledge using what is called Kripke or possible world models. Intuitively, each world is a possible description of facts, and the agents cannot distinguish between certain possibilities using an *indistinguishability* relation. This will be formalised in following chapters, but for the time being, consider an example:

Example 1. *Agents* A, B, C *pick three face-down cards* $1, 2, 3$*. Each agent can see its own cards and not others. Consider A got card number* 2*. Hence A cannot distinguish between the possibility where B has* 1 *and C has* 3 *and the possibility where B has* 3 *and C has* 1*.*

This example, or at least a version of it, has been well studied in [60] to motivate Dynamic Epistemic Logic. On the face of it, Epistemic logic gives ability to reason about the knowledge and possibilities considered by agents. But the setting seems very static. In particular, take the example 1. What if B tells A about its card? Assuming all agents are truthful, this *event* causes a change in the knowledge of agents, especially of A. Because A now knows exactly which agents posess what cards. As is evident, epistemic logic does not take such events into

account. To introduce such dynamic nature of knowledge, an extension of epistemic logic, called the *dynamic epistemic logic* (DEL) by van Ditmarsch et al. [60] is used. In addition to representing indistinguishability among facts or truths to represent knowledge, DEL uses similar indistinguishability among *events* to represent event models. These event models are used to update the knowledge introducing a dynamic perspective. It is to be noted that, various *simpler* dynamic extensions of epistemic logics were also introduced such as the *public announcement logic* (PAL) by Plaza [48]. Here, the event was modelled as a formula which represented an *announcement* of a fact to all agents publicly.

Another set of logical systems in a similar line of study and development is the collection of *temporal logics*. These logical systems are mostly used for specification and verification of systems.

Humans interact with innumerable amount of devices and programs that fulfill their tasks in an automated fashion. But, what is the evidence that such devices or programs shall maintain *suitable* properties after any interaction? What is the proof that they are *safe* at every point of time? The difficulty arises more from the fact that most of such programs and devices are supposed to run ideally forever along with numerous interactions with the users. Take for example, an Operating system. Even after the computer is turned off, an OS saves its *state* (relevant information about memory and programs) and starts from that saved state after it is turned back on. So, technically, an OS is projected to run for an infinite amount of time in the future. An user interacts with it any number amount of time. But what is the guarantee that this OS will not run into any deadlock **ever**?

To verify such safety properties, temporal logics, especially *linear temporal logic* (LTL) by Pnueli [49], come into play. In order to reason about certain safety properties of a system, *runs* of that system are modeled, which are just infinite transition structures where each node is a state of the system. To check whether a certain safety property always satisfies in such a system, such infinite run is represented using finite transition structures like an automata and the property is verified.

Both DEL and temporal logics have found utility in one of the most important branch of studies in Artificial Intillegence: Automated Planning. In simple terms, a *plan* is a sequence of actions to be performed in order to reach a certain goal. One of the most important decision problem studied in automated planning is the *plan-existence* problem. Formally, given an initial scenario and a goal, deciding whether there is a plan to reach the goal is called the plan-existence problem.

Classical planning [30] has been around for quite some time. It essentially uses a state transition system. Each state is characterised by truths represented by propositions and functions over variables. In addition, there is a set of actions defined along with their

pre-conditions. Pre-conditions of an action decides the states on which action can be implemented.

Classical planning with propositional fact-checking has been proven to be decidable, whereas the use of functional symbols lead to semi-decidability: if there is a plan, the decision algorithm will stop and return yes, if not it might stop and return no or might run forever.

With the introduction of DEL, another line of research in the planning domain, viz. *epistemic planning* [13] became quite popular . In epistemic planning, the main goal is to attain a knowledge state of agents. Since DEL can verify properties like "after a certain action, an agent knows some fact", the framework became one of the most important ones in automated planning. For single-agent, the plan-existence problem of epistemic planning was proved to decidable [38] . However, even for three agents, the problem has been proven to be undecidable by Bolander et al. [14] even though model-checking of DEL has been proven to be PSPACE-complete by Aucher and Schwarzentruber [6]. Intuitively, this is because while talking about whether there exist some finite iteration of some set of actions in epistemic planning (composition by Kleene-star), the product model can become arbitrarily large. This does not happen in DEL since actions are not composed using Kleene-star.
With respect to these existing results, this thesis answers the following questions:

Question 1. *Can we find a restricted framework which gives a decidable fragment of Epistemic Planning? How about we consider atomic actions instead of event models that have multiple possible actions?*

In 2014, van Ditmarsch et al. [62] introduced the *public observation logic* (POL) to reason about how public observations about the environment can change the knowledge of agents. Note that, public actions are also dealt in PAL as well, but finite iterations of public actions can also be dealt using POL.

This thesis shows that the model-checking problem of POL can be used in Epistemic planning. It is also showed that the problem is indeed PSPACE-complete and involves composition of atomic action using Kleene Star, since observations are regular expressions.

Along with the model-checking query, this thesis also answers the questions involving existence of such plans in a class of models using the satisfiability problem of POL.

Let us note that Linear temporal logic (LTL) is a framework used to verify whether at a certain point in time, some propositional fact holds. Extending this phenomena, to reason about the concurrent processes and knowledge, *epistemic linear temporal logic* (LTL_K) [34] is used. This framework can verify whether in distant future, certain knowledge change occurs in the processes or agents.

A very useful, and also integral to POL, assumption is the *no-forgetting* [34, 41] and *synchronous knowledge* [34] assumption. The first one suggests once an agent gains a

knowledge, it never forgets. The second assumption suggests that all the agents are aware of each time unit changing, that is, has access to a global clock. With common knowledge, this logic becomes undecidable, whereas, without common knowledge, the satisfiability problem of the logic lies in the non-elementary complexity region [34, 33]. In addition to the first question, this thesis also tries to answer the following question as well:

Question 2. *Does there exist any fragment of* LTL_K *with no-forgetting and synchronous assumptions that lies far below the non-elementary region?*

As it turns out, POL gives one such fragment of LTL_K and its satisfiability lies in $2-\mathsf{EXPSPACE}$, which is way below non-elementary classes of complexities. In addition to that, due to the extra power of regular expression, it can verify whether at some *even* point of time in the future, a knowledge change occurs. LTL_K cannot count even or odd number of points in the future. This thesis also looks into a decidability proof of model checking of an extension of POL called *epistemic protocol logic* (EPL) [62]. .

Key Contributions

The following are the key contribution that this thesis makes:

1. **Model-Checking Complexity of Public Observation Logic**: The POL model-checking, just like DEL, plays an important role in the plan-existence problem of epistemic planning. In this thesis, we have a tight bound for this problem. The upper-bound proof uses an oracle that efficiently traverses along updated models.

2. **Satisfiability of Public Observation Logic**: When the question is the plan-existence problem for a class of scenarios unlike a specific scenario, satisfiability of POL is a much more relevant question to answer rather than the model-checking. Here, the class of scenario is characterised by a property expressed in the POL syntax. We have tight bound for the satisfiability without Kleene-star and for the entire POL, we give decidability in double-exponential space.

3. **Model-Checking of Epistemic Protocol Logic** (EPL): In addition to POL, we look into an extension of POL called EPL as well. We study the decidability of the model-checking problem of this extension.

Outline of Chapters

Chapter 1 gives a brief introduction to the literature of planning and verification. It also lists the questions this thesis tackles with respect to planning and verification.

Chapter 2 is the basics and preliminary. This chapter introduces the basic definitions of the logical frameworks : *public observation logic* (POL), *epistemic protocol logic* (EPL). It introduces the syntax and semantics of these logic, including the models and different components such as regular expressions, protocols with verifiers, that they use. Along with that, this chapter introduce the principle running examples this thesis use to motivate its contribution.

Chapter 3 is on the model-checking study of POL. This chapter dives into a brute-force decidability proof after introduction and motivation. After giving a dynamic programming approach to the problem, it goes on to give the efficient algorithm along with a matching lower bound.

Chapter 4 is on model-checking of fragments of POL. This chapter introduces constraints over the input of the model-checker and proves their complexity. It also discusses application of the model checking algorithm in Epistemic planning and implementation using an example.

Chapter 5 is on fragments of the satisfiability of POL. This chapter does not include the full POL satisfiability problem. It uses the popular tableau method to prove the upper bound before proving matching lower bounds.

Chapter 6 is the full satisfiability problem of POL. This chapter also shows the connection between POL and Epistemic Temporal Logic (EPL).

Chapter 7 concludes this thesis along with a discussion of decidability of the model-checking of Epistemic Protocol Logic (EPL).

Chapter 2

Background and Preliminaries

In this chapter, we provide definitions to all the notions and frameworks that are being used in this thesis. We start with introducing the *dynamic epistemic logic* (DEL) [60]. This logic is the principle driving force behind the domain of *epistemic planning* [14]. We also introduce the *epistemic linear temporal logic* (LTL_K) [34], which is another framework used in this line of research. Then we move on to introduce the logical frameworks whose computational complexity has been studied in detail in this thesis: *public observation logic* (POL) [62] and *epistemic protocol logic* (EPL) [62]. For notational reasons, we first specify the symbols used throughout the thesis.

Let $\mathbf{I}$ be a finite set of agents, $\mathbf{P}$ be a countable set of propositions describing the facts about the world and $\mathbf{\Sigma}$ be a finite set of actions. Below, we will not differentiate between the action of observing a phenomenon and the phenomenon itself. The same notations for agents, propositions and actions will be used while defining the other frameworks as well, if not specified otherwise.

2.1 Dynamic epistemic logic (DEL)

Dynamic epistemic logic (DEL) [60] is a logical framework extending the well-known *epistemic Logic* [37]. This logic reasons about the dynamics of knowledge of multiple intelligent agents based on the events surrounding these agents. Recall example 1 in Chapter 1. To reason about the knowledge of agents A, B, and C regarding the cards that the other agents have, it is essential to model all the possibilities of hands. This forms the basic approach of *epistemic models* defined below.

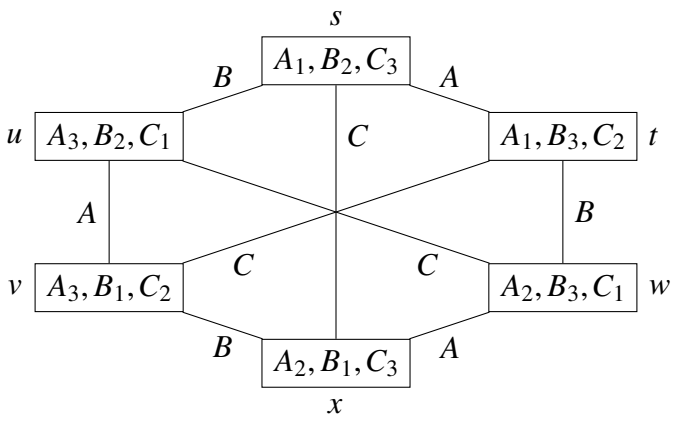

Fig. 2.1 Epistemic model $\mathcal{M}_{cards}$ with respect to example 1.

2.1.1 Epistemic models

In this section, the epistemic models are introduced, which is essentially the basic model that is being used in describing the planning scenarios in this thesis.

Definition 1 (Epistemic model)**.** *Given a finite set of agents* **I***, a set of propositions* **P***, an epistemic model* $\mathcal{M} = \langle W, R, V \rangle$ *has three components:*

- W *is a countable set of possible worlds or states.*
- $R = \{\sim_i\}_{i \in \mathbf{I}}$ *is a set of equivalence relations which are called indistinguishability relations. Each* $\sim_i$ *denotes the indistinguishability relation of agent* i*.*
- $V : W \to 2^{\mathbf{P}}$ *is a valuation function that assigns a set of propositions to a world.*

Take for example, figure 2.1 for the epistemic model describing example 1 in Chapter 1. The agents is $\mathbf{I} = \{A, B, C\}$ and the set of propositions is $\mathbf{P} = \{A_1, A_2, A_3, B_1, B_2, B_3, C_1, C_2, C_3\}$. A proposition, say for example, A_1 denotes that agent A has card 1. The model $\mathcal{M}_{cards} = \langle W, R, V \rangle$ contains every possible hand. These are modelled by the worlds s, t, u, v, w, x and the valuations in them. The indistinguishability relations are given by the edges labelled by agents. For example, since A has card 1 in both the worlds s and t and does not know what cards the others have, A cannot distinguish between the worlds where B has 2 or B has 3.

2.1.2 Action models and updates

Recall the discussion after example 1. An event can change the knowledge state of the agents. To model this, action models and updates are proposed [60]. Action models use a similar approach of possibilities and indistinguishability.

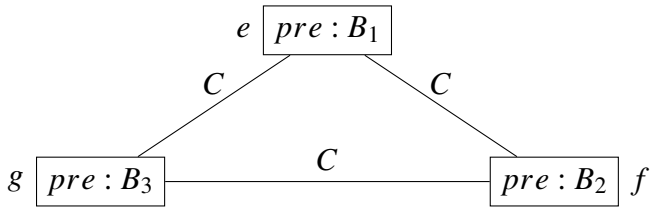

Fig. 2.2 Action model $\mathcal{A}_{cards}$ with respect to example 2.

Definition 2 (Action model)**.** *Given a set of propositions* **P** *and and a set of agents* **I**, *an action model* $\mathcal{A} = \langle E, R, pre \rangle$ *consists of three components:*

- E *is a countable set of possible events.*
- $R_e = \{\sim_i^e\}_{i \in \mathbf{I}}$ *is a set of equivalence relations, which are indistinguishability relations,* $\sim_i^e$ *of each agent* i.
- $pre : E \to \mathcal{L}(\mathbf{P})$ *is a pre-condition function which assigns a Boolean formula over the set of propositions* **P**, *where* $\mathcal{L}(\mathbf{P})$ *is the set of Boolean formulas over* **P**.

Intuitively, each event comes equipped with a pre-condition that tells us whether that event can be *executed* in a world. The following example makes it clearer.

Example 2. *Consider the action of "B tells A its own card". The action model* $\mathcal{A}_{cards}$ *depicting this event is in figure 2.2. For each card B can have, there can be an event, thus there are three possible events* $E = \{e, f, g\}$. *The event that B tells A privately that it has card* 1 *corresponds to the event labelled* e, *the event* f *corresponds to a similar phenomenon when B has card* 2 *and* g *corresponds to B having card* 3. *Since these events can only be executed when B actually has the respective cards, the pre-conditions for the events state as much. Note that, B has told about the card to A privately, in the sense that C has not heard it. Hence, C cannot distinguish between* e, f *and* g.

We have now modelled actions and we move on to consider how to reflect the update of knowledge in these epistemic models after a certain action is performed. This phenomenon is called an *action update* or a *knowledge update*. We describe it formally:

Definition 3 (Knowledge update)**.** *Given an action model* $\mathcal{A} = \langle E, R_e = \{\sim_i^e\}_{i \in \mathbf{I}}, pre \rangle$ *and an epistemic model* $\mathcal{M} = \langle W, R = \{\sim_i\}_{i \in \mathbf{I}}, V \rangle$, *the knowledge update operation* $\mathcal{M} \otimes \mathcal{A} = \langle W', R' = \{\sim_i'\}, V' \rangle$ *produces an epistemic model as follows:*

- $W' = \{(s, e) \mid s \in W, e \in E, V(s) \vDash pre(e)\}$
- $R' = \{\sim_i'\}_{i \in \mathbf{I}}$, *where each* $\sim_i' = \{((s, e), (t, f)) \mid s \sim_i t \text{ and } e \sim_i^e f\}$

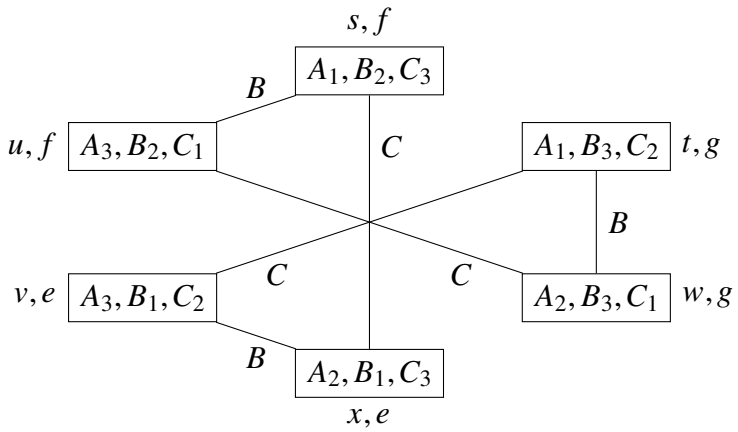

Fig. 2.3 Updated epistemic model $\mathcal{M}_{cards} \otimes \mathcal{A}_{cards}$ for examples 1 and 2 .

- $V'((s,e)) = V(s)$

Since a valuation at a world s, say, $V(s)$, is basically a set of propositions, the statement $V(s) \vDash \varphi$ for any propositional formula φ, expresses the fact: when all the propositions in $V(s)$ hold, the formula φ holds as well. In what follows, we will use the terms *knowledge update* and *action update* interchangeably.

Going back to examples 1 and 2, the model $\mathcal{M}_{cards} \otimes \mathcal{A}_{cards}$ turns out to be as depicted in figure 2.3. The worlds are labelled as pairs of worlds (in the original epistemic model) and events (executed on them). For example, the world s, f in the updated model is the result of the event f being executed on the world s. Note that, since A was told about the card owned by B privately, in the model (cf. figure 2.3), there is no indistinguishability relation with respect to A. This is because of the fact that they all knew what cards are there on the deck and can see their own card. Thus, the moment the information about another agent's card came to A, it would know what cards all the three agents have.

2.1.3 Syntax of DEL

We have introduced the concepts of epistemic and action models and their interactions, aka knowledge update models. We are now ready to describe a logical language to express properties of knowledge, actions and the updates. In this section, we introduce the language of DEL. A property expressed in the language is termed as a formula. The formal definition of DEL formulas is given below.

Definition 4 (DEL Syntax)**.** *Given a finite set of agents* **I***, a set of propositions* **P***, a* DEL *formula* φ *can be of the following form:*

$$\varphi \quad ::= \quad \top \mid \ p \in \mathbf{P} \mid \varphi \wedge \varphi \mid \neg\varphi \mid \hat{K}_i\varphi \mid \langle \mathcal{A}, e\rangle\varphi,$$

where $\mathcal{A}$ *is an action model and* e *is an event in it.*

In addition to Boolean formulas, DEL includes formulas of the form $\hat{K}_i\varphi$ which can be read as 'agent i considers φ a possibility'. We represent the dual notion of this operator as $K_i\varphi$ ($K_i\varphi \equiv \neg\hat{K}_i\neg\varphi$) which expresses 'agent i knows φ'. . The formula $\langle \mathcal{A}, e\rangle\varphi$ is used to express the following: there is an occurrence of the event e in $\mathcal{A}$ at the current world after which φ holds. For example, $\langle \mathcal{A}_{cards}, f\rangle K_A(C_3 \wedge B_2)$ expresses the fact that after B tells A it has card 2 (expressed by the modal operator $\langle \mathcal{A}_{cards}, f\rangle$), A knows that C has 3 and B has 2. The dual notion of this is represented as the *box operator* $[\mathcal{A}, e]\varphi$, where $[\mathcal{A}, e]\varphi = \neg\langle \mathcal{A}, e\rangle\neg\varphi$.

2.1.4 Semantics for DEL

We now define the interpretations of these formulas in terms of the models introduced earlier. Essentially, given an epistemic model, a formula is evaluated at a world/state in the model.

Definition 5 (Truth Conditions of DEL)**.** *Given an epistemic model* $\mathcal{M} = \langle W, R = \{\sim_i\}_{i\in\mathbf{I}}, V\rangle$*, a world* $s \in W$ *(pointed model) and a* DEL *formula* φ*, the truth definition of* φ *at the pointed model* $\mathcal{M}, s$*, denoted by* $\mathcal{M}, s \vDash \varphi$*, is given inductively as follows:*

$$\begin{aligned}
\mathcal{M}, s \vDash p \Leftrightarrow & p \in V(s) \\
\mathcal{M}, s \vDash \neg\varphi \Leftrightarrow & \mathcal{M}, s \nvDash \varphi \\
\mathcal{M}, s \vDash \psi \wedge \chi \Leftrightarrow & \mathcal{M}, s \vDash \psi \text{ and } \mathcal{M}, s \vDash \chi \\
\mathcal{M}, s \vDash \hat{K}_i\psi \Leftrightarrow & \exists t : s \sim_i t \text{ and } \mathcal{M}, t \vDash \psi \\
\mathcal{M}, s \vDash \langle \mathcal{A}, e\rangle\psi \Leftrightarrow & V(s) \vDash pre(e) \text{ and } \mathcal{M} \otimes \mathcal{A}, (s, e) \vDash \psi
\end{aligned}$$

Coming back to the examples 1 and 2, recall the models $\mathcal{M}_{cards}$ and $\mathcal{A}_{cards}$, and the formula $\langle \mathcal{A}_{cards}, f\rangle K_A(C_3 \wedge B_2)$. It holds at the pointed model $\mathcal{M}_{cards}, s$. Formally, $\mathcal{M}_{cards}, s \vDash \langle \mathcal{A}_{cards}, f\rangle K_A(C_3 \wedge B_2)$. with respect to definition 5, since $B_2 \in V(s)$, f is indeed executable on s. Moreover, $\mathcal{M}_{cards} \otimes \mathcal{A}_{cards}$ is the model given in figure 2.3. Now, all that is left to check is whether $\mathcal{M}_{cards} \otimes \mathcal{A}_{cards}, (s, f) \vDash K_A(C_3 \wedge B_2)$. Note that, as given in figure 2.3, (s, f) is the only world A cannot distinguish from (s, f) in the model $\mathcal{M}_{cards} \otimes \mathcal{A}_{cards}$ (cf. figure 2.3) and both C_3 and B_2 hold there.

2.1.5 Model checking problem for DEL

An interesting computational problem regarding DEL is the model checking problem. Given a pointed epistemic model $\mathcal{M}, s$ and a DEL formula φ, the model checking problem for DEL is to check whether $\mathcal{M}, s \vDash \varphi$. This problem has been shown to be PSPACE-complete [6].

As mentioned earlier, for obvious reasons, the model-checking problem of DEL plays a huge role in epistemic planning. The fact that DEL formulas can express whether after an execution of some action updates certain agents' knowledge plays a crucial role in planning. Aucher and Schwarzentruber [6] considered composing actions in the modalities with non-deterministic choice ($\langle\langle(\mathcal{A}_1, e_1) + (\mathcal{A}_2, e_2)\rangle\varphi$) and sequential execution ($\langle\langle(\mathcal{A}_1, e_1) \cdot (\mathcal{A}_2, e_2)\rangle\varphi$). They showed that the model checking problem for this extended language is PSPACE-complete as well.

However, the most important question that is tackled in epistemic planning is "whether there exists a sequence of actions such that after executing them, a certain knowledge formula holds". To incorporate such an expressibility, the DEL syntax needs to be extended with a form of iteration, Kleene star, for example ($\langle\langle((\mathcal{A}_1, e_1) + (\mathcal{A}_2, e_2))^\star\rangle\varphi$). Unfortunately, this extension makes the model checking problem undecidable since it has been shown that such a model checking problem can be reduced to epistemic planning [5] which is undecidable [14].

This thesis, however, studies a logical frameworks, POL, which is capable of expressing such iterations while keeping the model checking problem decidable. We will introduce these frameworks in later sections. Before that, we introduce another logical framework, LTL_k, which is also used in epistemic planning.

2.2 Epistemic temporal logic (LTL_k)

Consider the following general scenario given by [34]: A system of m processors run concurrently. As previously, these processes can be considered as agents. We want to reason about the knowledge these processes or agents have of the system and of other agents and the temporal effect on such knowledge, that is, how knowledge changes with time, as the overall state of the system changes.

2.2.1 Epistemic temporal models

An *epistemic temporal model* is similar to the models previously introduced, the only difference being that the indistinguishability relations are defined on points in *runs*. To formally introduce the model, we first introduce some terms.

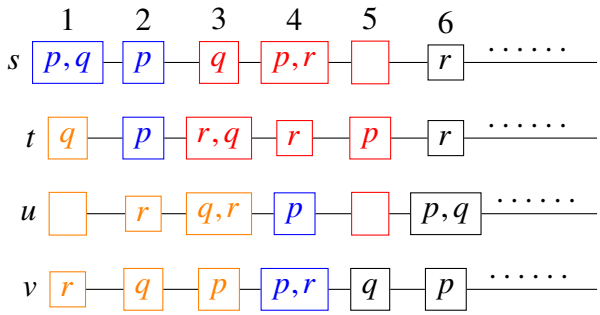

Fig. 2.4 An epistemic temporal model $\mathcal{M}_{\mathsf{LTL}_k}$ of *one* process.

Definition 6 (Epistemic Temporal Model)**.** *Given a system of m distributed processes (agents) and a set of propositions* $\mathbf{P}$*, a system is represented by a countable set* $\mathcal{R}$ *of* runs*. Each* $r \in \mathcal{R}$ *is a function* $r : \mathbb{N} \to L$*, where* L *is a countable set (possibly infinite) of labels. A single point in a run* r *at* $n \in \mathbb{N}$ *is denoted by* $r(n)$*. An interpreted model* $\langle \mathcal{R}, V \rangle$ *consists of the set of runs,* $\mathcal{R}$ *and an evaluation function* $V : \mathcal{R} \times \mathbb{N} \to 2^{\mathbf{P}}$ *assigning truth value to points in the runs. An epistemic temporal model is given by* $\mathcal{M} = \langle \mathcal{R}, R = \{\sim_i\}_{i \in \mathbf{I}}, V \rangle$*, where* $\mathbf{I} = [m]$*,* $\sim_i$*'s give the indistinguishability relations on the points in runs with* $(r(n), r'(n')) \in \sim_i$ *denoting that agent* i *cannot distinguish among the* n^{th} *point of run* r *and the* n'^{th} *point of run* r'*.*

Essentially, the set $\mathcal{R}$ consists of all possible runs that a system can form. Each run is a (possibly) infinite linear string of labels or points, each point representing a state of the entire distributed system. The agents (m processes) may not distinguish among different points over the runs. The function V assigns truth values to each of these points.

For example, figure 2.4 represents an epistemic temporal model with *one* process. Here, $R = \{s, t, u, v\}$ is the set of all possible runs the process can achieve. Each node (e.g., $s, 1$ or $t, 4$ in the grid) is a run-point and the valuation is mentioned inside those points, for example $V(s, 1) = \{p, q\}$. The indistinguishability (equivalence) relation here has been given by equivalent class of $\sim_1$. Each color represents one equivalent class, that is, if the system is in point $(u, 3)$, it cannot distinguish this point and every orange points.

2.2.2 Epistemic temporal logic (LTL_k) syntax

We now introduce a language to reason about such temporal models. Since the questions we want to ask out of these models involve both "what an agents knows" and "properties satisfying in some finite future", we have knowledge operators as well as *temporal* operators.

Definition 7. *Given a finite set of agents or processes* $\mathbf{I}$*, a set of propositions* $\mathbf{P}$*, an* LTL_k *formula* φ *can be of the form:*

$$\varphi \quad ::= \quad \top \mid \ p \in \mathbf{P} \mid \varphi \wedge \varphi \mid \neg\varphi \mid \hat{K}_i\varphi \mid X\varphi \mid \varphi U\varphi$$

Besides the usual propositions and the knowledge operators, this syntax also uses additional operators for verifying temporal properties. The formula $X\varphi$ expresses that φ holds in the next run point. The formula $\varphi U\psi$ expresses that φ holds in the run until a point arrives where ψ holds. There are other operators that have also been used in literature, such as $F\varphi = \top U\varphi$, which expresses that at some future point in the run, the formula φ holds. Also there is a global operator $G\varphi = \neg F\neg\varphi$ which expresses that φ will always hold.

For example, the formula $(p \vee q \vee r)U\neg(p \vee q \vee r)$ expresses that one of p, q or r should conmtinue holding until there will be a point where none of them holds. Also, $(\hat{K}p)U((Kp) \wedge X(\neg Kp)$ says that the process will keep considering p a possibility until a point comes when it knows p but loses the knowledge in the next point.

2.2.3 Semantics for LTL_k

The only question that remains in LTL_k is how are the properties that are expressed using the above language in definition 7 are interpreted in the model in definition 6.

Definition 8 (Truth Conditions of LTL_k)**.** *Given an epistemic temporal model* $\mathcal{M} = \langle \mathcal{R}, R = \{\sim_i\}_{i\in\mathbf{I}}, V\rangle$ *and a point* (r,i) *where* r *is a run and* $i \in \mathbb{N}$ *and given an* LTL_k *formula* φ*, whether the formula* φ *holds in* $\mathcal{M}, r, i$*, noted as* $\mathcal{M}, r, i \vDash \varphi$ *is defined inductively as:*

$$\begin{aligned}
\mathcal{M}, r, i \vDash p &\Leftrightarrow p \in V(r,i)\\
\mathcal{M}, r, i \vDash \neg\varphi &\Leftrightarrow \mathcal{M}, r, i \nvDash \varphi\\
\mathcal{M}, r, i \vDash \psi \wedge \chi &\Leftrightarrow \mathcal{M}, r, i \vDash \psi \text{ and } \mathcal{M}, r, i \vDash \chi\\
\mathcal{M}, r, i \vDash \hat{K}_i\psi &\Leftrightarrow \exists (r', i') : (r,i) \sim_i (r', i') \text{ and } \mathcal{M}, r', i' \vDash \psi\\
\mathcal{M}, r, i \vDash X\psi &\Leftrightarrow \mathcal{M}, r, i+1 \vDash \psi\\
\mathcal{M}, r, i \vDash \psi U\chi &\Leftrightarrow \exists j > i : \text{ for all } i' \in [i, j-1], \mathcal{M}, r, i' \vDash \psi \text{ and } \mathcal{M}, r, j \vDash \chi
\end{aligned}$$

Take for example the model $\mathcal{M}_{\mathsf{LTL}_k}$ in figure 2.4 and the formulas discussed after definition 7.

Note that, $\mathcal{M}_{\mathsf{LTL}_k}, s, 1 \vDash (p \vee q \vee r)U\neg(p \vee q \vee r)$, since at $s, 5$, nothing holds true and it is the first point to do so.

Also, $\mathcal{M}_{\mathsf{LTL}_k}, v, 1 \vDash (\hat{K}p)U((Kp) \wedge X(\neg Kp))$. This is so because in all the initial points in v from 1, the process considers p possible, that is it lies in the orange region. But as soon as it enters $v, 4$, the process knows p because p is true in all the blue points. Moreover, as soon as it moves to the next point which is a white one, p is not a knowledge then. Note that, since the example involves only one process, the operators $\hat{K}$ and K does not have any subscript to it. Although, one can use $\hat{K}_1$ and K_1 in such single process cases to be precise, we choose to ignore it.

2.3 Epistemic Propositional Dynamic Logic (EPDL)

In this section, we discuss another logical framework, EPDL, that is used in epistemic planning. This logic is a combination of propositional dynamic logic (PDL) and epistemic logic. The syntax and semantics of PDL is not discussed here, but can be found in [35]. For detailed discussions on EPDL, we refer the reader to [41].

Definition 9 (EPDL Syntax)**.** *Given a finite set of agents* $\mathbf{I}$*, a set of propositions* $\mathbf{P}$*, a finite set of actions* $\mathbf{\Sigma}$*, an* EPDL *formula* φ *and a program* π *can be of the following form:*

$$\begin{aligned} \varphi &::= \top \mid p \in \mathbf{P} \mid \varphi \wedge \varphi \mid \neg\varphi \mid K_i\varphi \mid [\pi]\varphi, \\ \pi &::= a \in \mathbf{\Sigma} \mid \varphi? \mid \pi \cdot \pi \mid \pi + \pi \mid \pi^\star \end{aligned}$$

Here, the expressions π are treated as *programs* and the formula $[\pi]\varphi$ expresses that after every possible execution of the program π, the formula φ holds. The program π can be an action $a \in \mathbf{\Sigma}$, a test $\varphi?$ (which checks whether φ holds in the current state), a sequential execution of two programs $(\pi_1 \cdot \pi_2)$, a non-deterministic choice between two programs $(\pi_1 + \pi_2)$ or an iteration of a program $(\pi^\star)$.

2.3.1 Semantics for EPDL

First we introduce the model for EPDL formulas. The model is an extension of the epistemic models introduced in definition 1. In addition to the indistinguishability relations, the model also includes transition relations for each action in $\mathbf{\Sigma}$.

Definition 10 (EPDL Model)**.** *Given a finite set of agents* $\mathbf{I}$*, a set of propositions* $\mathbf{P}$ *and a finite set of actions* $\mathbf{\Sigma}$*, an* EPDL *model is a tuple* $\mathcal{M} = \langle W, R = \{\sim_i\}_{i\in\mathbf{I}}, \{\rightarrow_a\}_{a\in\mathbf{\Sigma}}, V\rangle$ *where* $W, \sim_i$*'s and* V *are as in definition 1 and each* $\rightarrow_a \subseteq W \times W$ *is a transition relation for action* $a \in \mathbf{\Sigma}$*. Further more this transition relation can be extended to programs as follows:*

- $s \rightarrow_{\varphi?} t$ *iff* $s = t$ *and* $\mathcal{M}, s \vDash \varphi$ *as per definition 11.*

- $s \to_{\pi_1;\pi_2} t$ *iff there exists a* $u \in W$ *such that* $s \to_{\pi_1} u$ *and* $u \to_{\pi_2} t$
- $s \to_{\pi_1+\pi_2} t$ *iff* $s \to_{\pi_1} t$ *or* $s \to_{\pi_2} t$
- $s \to_{\pi^\star} t$ *iff there exists a sequence of worlds* $s_0, s_1, \ldots, s_n$ *such that* $s = s_0$, $t = s_n$ *and for each* $0 \leq i < n$, $s_i \to_\pi s_{i+1}$

Now we give the truth conditions for EPDL formulas.

Definition 11 (Truth Conditions of EPDL)**.** *Given an* EPDL *model* $\mathcal{M} = \langle W, R = \{\sim_i\}_{i\in\mathbf{I}}, \{\to_a\}_{a\in\mathbf{\Sigma}}, V\rangle$*, a world* $s \in W$ *(pointed model) and an* EPDL *formula* φ*, the truth definition of* φ *at the pointed model* $\mathcal{M}, s$*, denoted by* $\mathcal{M}, s \vDash \varphi$*, is given inductively as follows:*

$$\begin{aligned}
\mathcal{M}, s \vDash p &\Leftrightarrow p \in V(s)\\
\mathcal{M}, s \vDash \neg\varphi &\Leftrightarrow \mathcal{M}, s \nvDash \varphi\\
\mathcal{M}, s \vDash \psi \wedge \chi &\Leftrightarrow \mathcal{M}, s \vDash \psi \text{ and } \mathcal{M}, s \vDash \chi\\
\mathcal{M}, s \vDash K_i\psi &\Leftrightarrow \forall t : s \sim_i t, \mathcal{M}, t \vDash \psi\\
\mathcal{M}, s \vDash [\pi]\psi &\Leftrightarrow \forall t : s \to_\pi t, \mathcal{M}, t \vDash \psi
\end{aligned}$$

As we will see later, the syntax of EPDL is very similar to the logical framework POL that is studied in this thesis. Hence EPDL is another such logic that can talk about finite iterations. Although, as it can be seen in the previous definition, how the knowledge changes with execution of programs is explicitly mentioned in the EPDL model. Hence, even though model-checking is computationally easier, the input itself is quite space consuming.

Moreover, the programs are not fine-grained enough. To be precise, unlike DEL, the programs do not carry a formal mechanism of how the knowledge is changing. It is just hardcoded in the model. Whereas in POL, such programs (*observations* in POL) has a formal mechanism of knowledge update.

2.4 The Public Observation Logic (POL) and an Extension

First a few of the running examples are introduced which are used in this entire thesis. We introduce scenarios that we model and interpret using the logical frameworks we use, especially POL and EPL framework. Following which we formally define the logic, its syntax and semantics.

2.4.1 Running Examples

Let us first provide two examples showing the diverse nature of such reasoning phenomena.

- Consider a person traveling from Switzerland to France in a car. Here is one way she would know whether she is in France. According to her expectations based on the traffic light signals of the different states, if she observes the sequence of (green*-amber-red*)* ($*$ denotes the continuance of such sequences), she would know that she is in France, whereas if she observes (green*-amber-red*-amber)*, she would know that she is not.

- Consider three agents denoted by Sender (S), Receiver (R) and Attacker (A). Suppose S and R have already agreed that if S wants to convey that some decision has been taken, S would send a message, say m, to R; otherwise, S would send some other message, say m', to R. Suppose also that A has no information about this agreement. Then upon getting a message from S, there would be a change in the knowledge state of R but not A.

The first example concerns a certain rule that we follow in our daily life, and the second example brings in the flavour of coded message-passing under adversarial attacks. Expectations about the moves and strategies of other players also occur naturally in game theory, and possible behaviours of players are represented in these terms. Moving from theory to actual games, in the strategy video game Starcraft[1], one player may know/expect that the other player will attack her base as soon as possible, and thus may play accordingly. Games like Hanabi[2], and Colored Trails [24] also consider the connection between expectations and observations regarding the moves and strategies of the other players. Besidees these examples that is used to motivate the definitions, three (very similar) example scenarios are used to motivate the utility of the algorithms in this complexity study. The examples follow an automated robot (cleaning bot or a drone) and two agents whose knowledge vary as per their observations on how the drone moves.

Example 3. *Let us consider an automatic farming drone that is moving in a field represented as a grid (see Figure 2.5). Two agents a and b help the farming drone. The system is adaptive so the global behaviour is not hard-coded but learned. We suppose that the drone moves on a grid and agents a and b may observe one of the four directions:* $\Sigma := \{\blacktriangleright, \blacktriangleleft, \blacktriangle, \blacktriangledown\}$. *For instance, observing* $\blacktriangleleft$ *means that the drone moves one-step left. For this example, we suppose that agent a has learned that there are three possible expectations for the drone:*

[1] https://en.wikipedia.org/wiki/StarCraft_(video_game)
[2] https://en.wikipedia.org/wiki/Hanabi_(card_game)

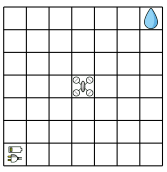

Fig. 2.5 Field and an automatic farming drone.

1. *the drone may go up-right searching for water, but the drone can make up to one wrong direction (▼ or ◄).*

2. *the drone may go down-left searching for power supply, but the drone can make up to one wrong direction (▲ or ►).*

3. *the drone is patrolling making clockwise squares.*

Although agent a considers all three of the above possibilities, agent b has more information and knows that the behaviour of the drone would include either searching for water or power supply. Agents a is programmed so that if she knows that the drone is searching for water then she will turn on the valve, and if she knows that the drone is searching for power, she will prepare the power supply. Agent b is programmed in the same way.

Example 4. *Let us consider a robotic vacuum cleaner (vbot) that is moving on a floor represented as a 7×7 grid (see Figure 2.6). On the top right of the floor, there is a debris-disposal area, and on the bottom left, there is a power source to recharge. Two children Alice and Bob are awed by this new robotic cleaner. They are watching it move and trying to guess which direction it is moving. The system is adaptive, thus the global behaviour is not hard-coded but learned. We suppose that vbot moves on a grid and the children may observe one of the four directions: right (►), left (◄), up(▲) or down(▼), and of course, combinations of them. Note that, for example, observing ◄ means that the bot moves one step left. Let Alice be aware of a glitch in the bot. Then her expectations regarding the vbot's movements include the following possibilities:*

1. *The bot may go up or right for debris-disposal, but may make an erroneous move, that is, a down or a left move.*

2. *The bot may go towards power source without error.*

The only difference between Bob's expectation and that of Alice is that Bob does not consider the bot to make an error while moving towards debris-disposal since he is unaware of the glitch.

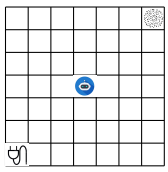

Fig. 2.6 A robotic vacuum cleaner on the floor (in the middle of the grid). The power source is at bottom left, whereas the debris-disposal area is at top right.

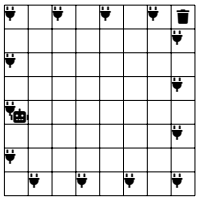

Fig. 2.7 A robot, some power sources, and the debris-disposal area.

Suppose the vbot is indeed moving towards power. Hence if the bot makes one left move, ◄, Bob would know that the bot is moving towards power whereas Alice would still consider moving towards debris-disposal a possibility.

Example 5. *Consider a cleaning robot in an* 8×8 *square maze with power sources on every alternate border cells (Figure 2.7). Suppose the robot is currently in a charging station, the third one from the top on the left. Also, it disposes debris on the top left corner of the maze. Here the assumption is that robot can either go horizontally or vertically, and blink a flag on display when it stops (*$\Sigma = \{◄, ►, ▲, ▼, ⚑\}$*).*

Consider two agents A and B, observing the robot moves and guessing the goal of the robot. The robot can either go to dispose off the debris, one cell at a time, or for a charging cell, in which case the robot will cover two cells at each step. Some of the charging cells turning off and on at regular intervals, so that they do not waste power. Thus, the robot, when going for the charging, she needs to keep on looking until it finds a suitable station.

Agents A and B both consider the two goals as possible. In addition, B is also aware that while going to the debris disposal area, the robot might face a glitch and then it may blink a flag after even number of moves towards a charging station and afterwards move on towards the debris area.

Consider that the robot is actually moving towards debris. Hence, if the robot, from the current position (cf. Figure 2.7), makes an even number of moves to a charging point and then blinks a flag, agent A would be sure that the robot is going for a charging point, yet B, since it is aware of the glitch, would still consider debris area as a possible goal.

2.4.2 Observations

For our purposes, we assume *observations* to be finite strings of actions. In the traffic example, an observation may be *green-amber-red-green* (abbreviated as $garg$) or, *green-amber-red-amber-green* (abbreviated as $garag$), among others, whereas, in the message-passing example, an observation is either m or m'. An agent may expect different (even infinitely many) potential observations at a given state, but to model agent expectations, they are described in a finitary way by introducing the *observation expressions* (as regular expressions over $\mathbf{\Sigma}$):

Definition 12 (Observation expressions)**.** *Given a finite set of action symbols* $\mathbf{\Sigma}$*, the language* $\mathcal{L}_{obs}$ *of* observation expressions *is defined by the following BNF:*

$$\pi \quad ::= \quad \emptyset \mid \varepsilon \mid a \mid \pi \cdot \pi \mid \pi + \pi \mid \pi^*$$

where $\emptyset$ *denotes the empty set of observations, the constant* ε *represents the empty string, and* $a \in \mathbf{\Sigma}$*.*

In the traffic example, the observation expression $(g^*ar^*)^*$ models the traveller's expectation of traffic signals in case she is in France. In the other one, the expression m models the expectation of the receiver in case a decision is made.

In the cleaner bot example 4, the observation expression $(\blacktriangleleft \cdot \blacktriangledown + \blacktriangleright \cdot \blacktriangle)$ models the expectation of the bot's movement in either way, towards the power source or the debris-disposal area, whereas $(\blacktriangleleft)^3 \cdot (\blacktriangledown)^3$ models the expectation of moving towards the power source.

Definition 13 (Size of observation expressions)**.** *Given an observation expression* π*, the size of* π*,* $|\pi|$ *can be defined inductively as follows:*

$|\varepsilon| = |\emptyset| = 0$
$|a| = 1$
$|\pi \cdot \pi'| = |\pi + \pi'| = |\pi| + |\pi'| + 1$
$|\pi^*| = |\pi| + 1$

In the traffic example, the observation expression $(g^*ar^*)^*$ models the traveller's expectation of traffic signals in case she is in France. Evidently, in this case, the size of the observation expression, $(g^*ar^*)^*$, is 6. The semantics for the observation expressions are given by *sets of observations* (strings over $\mathbf{\Sigma}$), similar to those for regular expressions [54].

Definition 14 (Semantics of observation expressions)**.** *Given an observation expression* π*, the corresponding* set of observations*, denoted by* $\mathcal{L}(\pi)$*, is the set of finite strings over* $\mathbf{\Sigma}$ *defined as follows:*

$\mathcal{L}(\emptyset) = \emptyset$
$\mathcal{L}(\varepsilon) = \{\epsilon\}$
$\mathcal{L}(a) = \{a\}$
$\mathcal{L}(\pi \cdot \pi') = \{wv \mid w \in \mathcal{L}(\pi) \text{ and } v \in \mathcal{L}(\pi')\}$
$\mathcal{L}(\pi + \pi') = \mathcal{L}(\pi) \cup \mathcal{L}(\pi')$
$\mathcal{L}(\pi^*) = \{\varepsilon\} \cup \bigcup_{n>0}(\mathcal{L}(\underbrace{\pi \cdots \pi}_{n}))$

For example, $\mathcal{L}(m) = \{m\}$, and $\mathcal{L}((g^*ar^*)^*) = \{\varepsilon, a, ga, ar, gar, gargar, \ldots\}$. Before going any further, let us first introduce the notion of *residue* which play an important role in providing the semantics of POL:

Definition 15 (Residue)**.** *Given an observation expression π and a word (finite string over $\mathbf{\Sigma}$) w, a residue of π with respect to w, denoted by $\pi\backslash w$, is an observation expression defined with an auxiliary output function o from the set of observation expressions over $\mathbf{\Sigma}$ to $\{\emptyset, \varepsilon\}$. If $\varepsilon \in \mathcal{L}(\pi)$, then o maps π to ε; otherwise,it maps π to $\emptyset$:*

$\varepsilon\backslash a = \delta\backslash a = b\backslash a = \delta \quad (a \neq b)$
$a\backslash a = \varepsilon$
$(\pi + \pi')\backslash a = \pi\backslash a + \pi'\backslash a$
$(\pi \cdot \pi')\backslash a = (\pi\backslash a) \cdot \pi' + o(\pi) \cdot (\pi'\backslash a)$
$\pi^*\backslash a = \pi\backslash a \cdot \pi^*$
$\pi\backslash a_0 \cdots a_n = \pi\backslash a_0 \backslash a_1 \ldots \backslash a_n$

Here δ is regular expression that represents the empty language ($\mathcal{L}(\delta) = \emptyset$). In this thesis, we use δ and $\emptyset$ interchangeably in case of regular expressions.

Note that, since the auxiliary output function o is a conditional function, the residue on concatenation ($(\pi.\pi')\backslash a$) can also be given as a conditional function. This thesis generally follows the convention in [62].

We recall that by Thomson's construction [54], there is a non-deterministic finite automaton (NFA) of polynomial size in $|\pi|$, that recognizes the language $\mathcal{L}(\pi)$. For simplicity in notations, that NFA is also denoted by π.

Given a word w, $\pi\backslash w$ denotes the regular language $\{v \mid wv \in \mathcal{L}(\pi)\}$. $\pi\backslash w$ corresponds to right residuation with respect to the monoid $(\mathbf{\Sigma}^*, \cdot, \varepsilon)$. The representation of $\pi\backslash w$ is the NFA π augmented by the subset of states of π that one reaches after having read w from the initial states of π. Computing the representation of $\pi\backslash w$ is polynomial in $|\pi|$ and $|w|$. In the sequel, $DFA(\pi)$ denotes the minimal deterministic finite automaton for π (unique up to isomorphism). Computing $DFA(\pi)$ is exponential in $|\pi|$ in the worst-case. The regular language $Pre(\pi)$ is the set of prefixes of words in $\mathcal{L}(\pi)$, that is, $w \in Pre(\pi)$ iff $\exists v \in \mathbf{\Sigma}^*$ such that $wv \in \mathcal{L}(\pi)$ (namely, $\mathcal{L}(\pi\backslash w) \neq \emptyset$).

Example 6. $(g^*ar^*a)^*\backslash(garaga) = r^*(g^*ar^*a)^*$ *denotes the language of words* $\{v : garaga \cdot v \in \mathcal{L}((g^*ar^*)^*)\}$. *The set* $Pre((g^*ar^*a)^*)$ *contains* $garaga$. *However,* $garg$ *is not in* $Pre((g^*ar^*a)^*)$ *and* $(g^*ar^*a)^*\backslash(garg)$ *is empty.*

Example 7. $(\blacktriangleleft \cdot \blacktriangledown)\backslash \blacktriangleleft = (\blacktriangleleft \cdot \blacktriangledown + \blacktriangleright \cdot \blacktriangle)\backslash \blacktriangleleft = \blacktriangledown$, *and* $Pre(\blacktriangleleft \cdot \blacktriangledown + \blacktriangleright \cdot \blacktriangle) = \{\varepsilon, \blacktriangleleft, \blacktriangleleft \blacktriangledown, \blacktriangleright, \blacktriangleright \blacktriangle\}$.

2.4.3 Protocols

Here we introduce the syntax and semantics of a protocol and also suggest transformers that takes a protocol and gives a regular expression with respect to a propositional valuation.

Definition 16 (Protocol Expression)**.** *Given a finite set of action symbols* $\mathbf{\Sigma}$*, and propositions* $\mathbf{P}$ *the language* $\mathcal{L}_{prot}$ *of* protocol expressions *is defined by the following BNF:*

$$\eta \quad ::= \quad \emptyset \mid \varepsilon \mid a \mid \varphi? \mid \eta \cdot \eta \mid \eta + \eta \mid \eta^*$$

where $\emptyset$ *denotes the empty set of expressions, the constant* ε *represents the empty string,* $a \in \mathbf{\Sigma}$ *and* φ *is a boolean formula over* $\mathbf{P}$.

For example, the expression $p? \cdot a^\star + \neg p? \cdot b$. This expressions interprets the protocol that if p holds, observing finite iterations of a is expected. If not then b is expected to be observed.

The basic idea of interpreting protocols from a regular expression point of view is that the string represented by a protocol has a propositional evaluation guarding each action. For example, a word or a string of observations from POL is of the form $a_1a_2\ldots a_k$ where each $a_i \in \Sigma$ is a letter in the alphabet.

In contrast to that, a word or a string or a *guarded observation* is of the form $\rho a_1 \rho a_2 \ldots a_k \rho$ where $\rho \subseteq \mathbf{P}$. Intuitively this represents the expected observation remains step by step as long as ρ is satisfied. Here in this work we assume no factual change, that is we do not consider actions that changes the valuation (ρ) (Guarded observation of the form $\rho_1 a_1 \rho_2 a_2 \ldots \rho_k a_k$).

In order to do that, we introduce the following definition:

Definition 17 (Language of a Protocol)**.** *Given a protocol expression* η*, the function* $\mathcal{L}_g(\eta)$ *represents a set of guarded observations as:*

- $\mathcal{L}_g(\emptyset) = \emptyset$
- $\mathcal{L}_g(\varepsilon) = \{\rho \mid \rho \subseteq \mathbf{P}\}$
- $\mathcal{L}_g(a) = \{\rho a \rho \mid \rho \subseteq \mathbf{P}\}$

- $\mathcal{L}_g(\varphi?) = \{\rho \mid \rho \vDash \varphi, \rho \subseteq \mathbf{P}\}$
- $\mathcal{L}_g(\eta_1.\eta_2) = \{w \diamond v \mid w \in \mathcal{L}_g(\eta_1), v \in \mathcal{L}_g(\eta_2)\}$
- $\mathcal{L}_g(\eta_1 + \eta_2) = \mathcal{L}_g(\eta_1) \cup \mathcal{L}_g(\eta_2)$
- $\mathcal{L}_g(\eta^\star) = \{\rho \mid \rho \subseteq \mathbf{P}\} \cup \bigcup_{n>0}(\mathcal{L}_g(\eta^n))$

where $w \diamond v$ denotes $w'\rho v'$ where $w = w'\rho$ and $v = \rho v'$

Coming back to example, consider the previous expression $p? \cdot a^\star + \neg p? \cdot b$. Consider $\mathbf{P} = \{p\}$. Hence the language of the protocol is

$$\mathcal{L}_g(p? \cdot a^\star + \neg p? \cdot b) = \{\{\}b\{\}, \{p\}, \{p\}a\{p\}, \{p\}a\{p\}a\{p\}, \ldots\}$$

Now, we introduce the next definition of a transformer function which intuitively extracts the set of observations that satisfy a given valuation in form of a regular expression. The basic idea is to assign such expression as expectations to POL models (further explanation in later definitions).

Definition 18 (Observation Tranformer Function)**.** *Given a protocol expression η and propositional valuation $\rho \subseteq \mathbf{P}$, a observation tranformer function $f_\rho : \mathcal{L}_{prot} \to \mathcal{L}_{obs}$ is defined as:*

- $f_\rho(\emptyset) = \emptyset$
- $f_\rho(\varepsilon) = \varepsilon$
- $f_\rho(a) = a$
- $f_\rho(\varphi?) = \begin{cases} \varepsilon & \text{if } \rho \vDash \varphi \\ \emptyset & \text{otherwise} \end{cases}$
- $f_\rho(\eta_1.\eta_2) = f_\rho(\eta_1).f_\rho(\eta_2)$
- $f_\rho(\eta_1 + \eta_2) = f_\rho(\eta_1) + f_\rho(\eta_2)$
- $f_\rho(\eta^\star) = f_\rho(\eta)^\star$

Note that

Theorem 1. *[62] For any $\eta \in \mathcal{L}_{prot}$*

$$\mathcal{L}(f_\rho(\eta)) = \{w \mid w = a_1a_2\ldots a_k, \text{ where } a_i \in \Sigma \cup \{\varepsilon\} \text{ and } \rho a_1 \rho \ldots a_k \rho \in \mathcal{L}_g(\eta)\}$$

Take for example, if we want to exatract the expectation from the protocol $p? \cdot a^\star + \neg p? \cdot b$ with respect to when p holds, note that $f_{\{p\}}(p? \cdot a^\star + \neg p? \cdot b) = a^\star$.

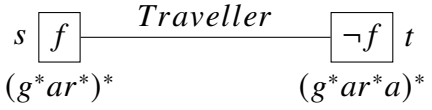

Fig. 2.8 $\mathcal{M}_{tl}$ (the traffic light model).

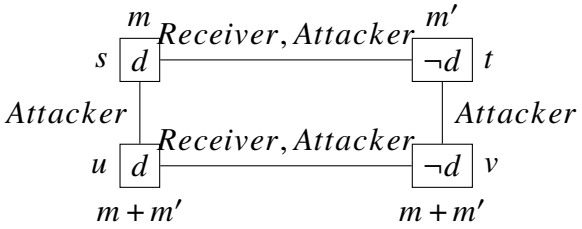

Fig. 2.9 $\mathcal{M}_{mp}$ (the message passing model).

2.4.4 Models

Epistemic expectation models [62] capture the expected observations of agents. They can be seen as epistemic models [27] together with, for each world, a set of potential or expected observations. Recall that an epistemic model is a tuple $\langle S, \sim, V\rangle$ where S is a non-empty set of worlds, $\sim$ assigns to each agent in $\mathbf{I}$ an equivalence relation $\sim_i \subseteq S \times S$, and $V : S \to 2^{\mathbf{P}}$ is a valuation function.

Definition 19 (Epistemic expectation model)**.** *An* epistemic expectation model $\mathcal{M}$ *is a quadruple* $\langle S, \sim, V, Exp\rangle$*, where* $\langle S, \sim, V\rangle$ *is an epistemic model and* $Exp : S \to \mathcal{L}_{obs}$ *is an expected observation function assigning to each world an observation expression* π *such that* $\mathcal{L}(\pi) \neq \emptyset$ *(non-empty set of finite sequences of observations). A pointed epistemic expectation model is a pair* $(\mathcal{M}, s)$ *where* $\mathcal{M} = \langle S, \sim, V, Exp\rangle$ *is an epistemic expectation model and* $s \in S$*.*

Intuitively, *Exp* assigns to each world a set of potential or expected observations. We now provide the model definitions of the examples mentioned in the introduction. The traffic light example where only one agent (the traveller) is involved can be depicted by the model $\mathcal{M}_{tl}$ (cf. Figure 2.8). Unless the traveller (T) observes the respective sequences of traffic signals, she would not know whether she is in France (f) or not ($\neg f$). Her uncertainty is represented by the (bi-directional) link between the two worlds s and t. For the sake of brevity, we do not draw the reflexive arrows. Similar representations are used in the message-passing example as well (cf. Figure 2.9). Here, the receiver would get to know about the decision depending on the message he receives, whereas, the attacker would be ignorant of the fact irrespective of the message (m or m') she receives.

The main idea for introducing this logic was to reason about agent knowledge via the matching of observations and expectations. In line of public announcement logic [48], it

is assumed that when a certain phenomenon is observed, people delete some impossible scenarios where they would not expect that observation to happen. To this end, the update of epistemic expectation models according to some observation $w \in \mathbf{\Sigma}^*$ is defined below. The idea behind an updated expectation model is to delete the worlds where the observation w could not have been happened.

Definition 20 (Update by observation)**.** *Let w be an observation over $\mathbf{\Sigma}$ and let $\mathcal{M} = \langle S, \sim, V, Exp\rangle$ be an epistemic expectation model. The updated model $\mathcal{M}|_w = \langle S', \sim', V', Exp'\rangle$ is defined by: $S' = \{s \in S \mid \mathcal{L}(Exp(s)\backslash w) \neq \emptyset\}$, $\sim'_i = \sim_i|_{S'\times S'}$, $V' = V|_{S'}$, and for all $s \in S'$, $Exp'(s) = Exp(s)\backslash w$. Here $\sim_i|_{S'\times S'}$ is the equivalence relation $\sim_i$ restricted only to the ordered pairs from $S' \times S'$ and $V|_{S'}$ is the valuation function V restricted only to the domain S'.*

In Definition 20, S' is the set of worlds s in S where the word w can be observed, i.e., $\mathcal{L}(Exp(s)\backslash w) \neq \emptyset$. The definitions of $\sim'$ and V' are given by usual restrictions to S'. The expectation at each world in S' gets updated by observing the word w: finite strings of actions that are of the form wu are replaced by u while strings that are not of the form wu get removed because they do not match the expectation.

Example 8. *Consider the model $\mathcal{M}_{tl}$ of Figure 2.8 and $w = garga$. The updated model $\mathcal{M}_{tl}|_w = \langle S', \sim', V', Exp'\rangle$ is such that $S' = \{s\}$: world t is removed because $garga$ is not a prefix of any word in $\mathcal{L}((g^*ar^*a)^*)$. The expectation $Exp(s)$ is replaced by $Exp'(s) = (g^*ar^*)^*\backslash(garga) = r^*(g^*ar^*)^*$. We have $\sim'= \{(s, s)\}$ and $V'(s) = \{f\}$.*

Example 9. *Consider the model $\mathcal{M}_{mp}$ of the message passing example stated earlier in Figure 2.9. It consists of four possible worlds:*

1. *s: the message m is expected to receive where decision d is made.*
2. *t: the message m' is expected to receive where decision d is NOT made.*
3. *u: the message m or m' is expected and decision d has been made.*
4. *v: the message m or m' is expected and decision d has not been made.*

Since the Attacker is unaware of the agreement between Sender and Receiver, it cannot distinguish between either of the four possibilities, whereas Receiver is only unable to distinguish between the fact that where the decision d is made or not made. Suppose the actual world is s. Now suppose m is received. $\mathcal{M}_{mp}|_m$ has the world t omitted since that world was expecting the message m'.

Now we are ready to introduce the *protocol models* and its updates and finally syntax and semantics of Epistemic Protocol Logic (EPL).

Fig. 2.10 $\mathcal{M}_{mpbef}$ (the message passing model before protocol defined).

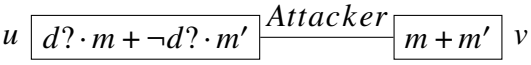

Fig. 2.11 $\mathcal{A}_{ifdm}$ (the message passing protocol model).

Definition 21 (Protocol Model)**.** *A Protocol model $\mathcal{A} = \langle E, \sim, Prot \rangle$ such that*

- *E is a finite set of events*
- *$\sim_i \subseteq E \times E$ is an equivalence relation for any agent $i \in \mathbf{I}$*
- *$Prot : E \to \mathcal{L}_{prot}$ is a function that assigns protocol to an event $e \in E$*

Protocol models are means to assign expectations to possible worlds in an expectation model. For example, let us take a step back through the example of message passing. Note that, before deciding upon what to expect in each world, a protocol was decided upon that "if the decidion d is made, m should be expected and if not m'". This protocol can be interpreted using the expression $d? \cdot m + \neg d? \cdot m'$. Note that, we also have to express the fact that the $Attacker$ should not be able to distinguish between this protocol and a general protocol $m + m'$ which interprets any one could be expected irrespective of the decision. The protocol model hence will look like Figure 2.11

Just like an event model and its update with epistemic models in DEL, we have model updates in EPL as well where a Protocol model is updated with an Epistemic expectation model, creating and assigning possible worlds with new expectations as per the valuation in the initial expectation model. Formally

Definition 22 (Product Update)**.** *Given an epistemic expectation model $\mathcal{M} = \langle W, \sim, V, Exp \rangle$ and a protocol model $\mathcal{A} = \langle E, \sim, Prot \rangle$, the product update $\mathcal{A} \otimes \mathcal{M} = \langle W', \sim', V', Exp' \rangle$ produces an expectation model such that:*

- $W' = \{(e, w) \mid \mathcal{L}(f_{V(w)}(Prot(e))) \neq \emptyset\}$
- $\sim_i' = \{((e, w), (e', w')) \mid e, e' \in E; w, w' \in W; e \sim_i e'; w \sim_i w'\}$
- $V'((e, w)) = V(w)$
- $Exp'((e, w)) = f_{V(w)}(Prot(e))$

Example 10. *Now consider the expectation model $\mathcal{M}_{mpbef}$ in Figure 2.10 and the protocol model $\mathcal{A}|_{ifdm}$ in Figure 2.11. Note that, the model $\mathcal{M}_{mpbef} \otimes \mathcal{A}_{ifdm}$ will give out the model $\mathcal{M}_{mp}$ in Figure 2.9 (labels changed).*

2.4.5 Public Observation Logic (POL)

To reason about agent expectations and observations, the language for POL is provided below.

Definition 23 (Syntax)**.** *The* formulas φ *of* POL *are given by:*

$$\varphi \quad ::= \quad \top \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid K_i\varphi \mid [\pi]\varphi,$$

where $p \in \mathbf{P}$, $i \in \mathbf{I}$, and $\pi \in \mathcal{L}_{obs}$.

Intuitively, $K_i\varphi$ says that 'agent i knows that φ', and $[\pi]\varphi$ says that 'after any observation in π, φ holds'. The other propositional connectives are defined in the usual manner. We also define $\langle\pi\rangle\varphi$ as $\neg[\pi]\neg\varphi$ and $\hat{K}_i\varphi$ as $\neg K_i\neg\varphi$. We will mostly use these modalities in our proofs.

Now we talk about various fragments, or special cases, of Public Observation Logic. Here various restrictions are promised on the syntax of regular languages or the syntax of the formulas.

Definition 24 (Negative Normal Form (NNF))**.** *Given a set of propositional letters* **P***, the Negative Normal Form of a* POL *formula is defined recursively as following:*

$$\varphi \quad ::= \quad \top \mid p \mid \neg p \mid \varphi \wedge \varphi \mid K_i\varphi \mid \hat{K}_i\varphi \mid [\pi]\varphi \mid \langle\pi\rangle\varphi$$

where $p \in \mathbf{P}$, $i \in \mathbf{I}$ and $\pi \in \mathcal{L}_{obs}$.

In simpler words, a POL formula is in a Negative Normal Form if and only if the negations ($\neg$) occur in the formula only before a propositional letter. Now we can define formally the various fragments of POL as motivated in the farming drone example previously.

Definition 25 (Star-Free Formula and Star-Free fragment of POL)**.** *Given a set of propositional letters* **P***, the* Star-Free *formulas of* POL *is defined recursively as following:*

$$\varphi \quad ::= \quad \top \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid K_i\varphi \mid [\pi]\varphi$$

where $p \in \mathbf{P}$, $i \in \mathbf{I}$ and $\pi \in \mathcal{L}_{obs}$ is defined recursively as:

$$\pi \quad ::= \quad a \in \Sigma \mid \varepsilon \mid \pi + \pi \mid \pi.\pi$$

The Star-Free *Fragment of* POL *consists only* Star-Free *formulas of* POL.

Note that, by Star-Free fragment, we do not mean the Star-Free language as defined in [25], where regular expressions allow complementation. The Star-Free fragment of POL just consists of formulas where the regular expression in the modalities does not contain any Kleene-Star in it.

Definition 26 (Existential Formula and Existential fragment of POL)**.** *Given a set of propositional letters* **P***, the* Existential *formulas of* POL *can be inductively defined as:*

$$\varphi \quad ::= \quad \top \mid p \mid \neg p \mid \varphi \wedge \varphi \mid \hat{K}_i \varphi \mid \langle \pi \rangle \varphi$$

where $p \in \mathbf{P}$*,* $i \in \mathbf{I}$ *and* $\pi \in \mathcal{L}_{obs}$*.*

The Existential *Fragment of* POL *consists only* Existential *formulas of* POL*.*

That is, the formulas of Existential Fragments are all in NNF and only contains $\hat{K}_i$ and $\langle \pi \rangle$ operators for an agent i and a regular expression π.

Definition 27 (Star-Free – Existential Formula and Star-Free – Existential fragment of POL)**.** *Given a set of propositional letters* **P***, the* Star-Free – Existential *formulas of* POL *can be inductively defined as:*

$$\varphi \quad ::= \quad \top \mid p \mid \neg p \mid \varphi \wedge \varphi \mid \hat{K}_i \varphi \mid \langle \pi \rangle \varphi$$

where $p \in \mathbf{P}$*,* $i \in \mathbf{I}$ *and* $\pi \in \mathcal{L}_{obs}$*, where* π *is inductively defined as:*

$$\pi \quad ::= \quad a \in \Sigma \mid \varepsilon \mid \pi + \pi \mid \pi . \pi$$

The Star-Free – Existential *Fragment of* POL *consists only* Star-Free – Existential *formulas of* POL*.*

Definition 28 (Word Formula and Word fragment of POL)**.** *Given a set of propositional letters* **P***, the* Word *formulas of* POL *is defined recursively as following:*

$$\varphi \quad ::= \quad \top \mid p \mid \neg \varphi \mid \varphi \wedge \varphi \mid K_i \varphi \mid [\pi] \varphi$$

where $p \in \mathbf{P}$*,* $i \in \mathbf{I}$ *and* $\pi \in \mathcal{L}_{obs}$ *is defined recursively as:*

$$\pi \quad ::= \quad a \in \Sigma \mid \varepsilon \mid \pi . \pi$$

The Word *Fragment of* POL *consists only* Word *formulas of* POL*.*

Therefore, in other words, the Word Fragment of POL contains only POL formulas where the regular expressions in the modalities are just words over Σ.

Definition 29 (Truth definition)**.** *Given an epistemic expectation model $\mathcal{M} = \langle S, \sim, V, Exp\rangle$, a world $s \in S$, and a* POL*-formula φ, the truth of φ at s, denoted by $\mathcal{M}, s \vDash \varphi$, is defined by induction on φ as follows:*

$$\begin{array}{rcl}
\mathcal{M}, s \vDash p & \Leftrightarrow & p \in V(s) \\
\mathcal{M}, s \vDash \neg\varphi & \Leftrightarrow & \mathcal{M}, s \nvDash \varphi \\
\mathcal{M}, s \vDash \varphi \wedge \psi & \Leftrightarrow & \mathcal{M}, s \vDash \varphi \text{ and } \mathcal{M}, s \vDash \psi \\
\mathcal{M}, s \vDash K_i\varphi & \Leftrightarrow & \text{for all } t : (s \sim_i t \text{ implies } \mathcal{M}, t \vDash \varphi) \\
\mathcal{M}, s \vDash [\pi]\varphi & \Leftrightarrow & \text{for all } w \in \mathcal{L}(\pi) \cap Pre(Exp(s)) \\
 & & \text{we have } \mathcal{M}|_w, s \vDash \varphi)
\end{array}$$

The truth of $K_i\varphi$ at s follows the standard possible world semantics of epistemic logic. The formula $[\pi]\varphi$ holds at s if for every observation w in the set $\mathcal{L}(\pi)$ that matches with the beginning of (i.e., is a prefix of) some expected observation in s, φ holds at s in the updated model $\mathcal{M}|_w$. Note that s is a world in $\mathcal{M}|_w$ because $w \in Pre(Exp(s))$. Similarly, the truth definition of $\langle\pi\rangle\varphi$ can be given as follows: $\mathcal{M}, s \vDash \langle\pi\rangle\varphi$ iff there exists $w \in \mathcal{L}(\pi) \cap Pre(Exp(s))$ such that $\mathcal{M}|_w, s \vDash \varphi$. Intuitively, the formula $\langle\pi\rangle\varphi$ holds at s if there is an observation w in $\mathcal{L}(\pi)$ that matches with the beginning of some expected observation in s, and φ holds at s in the updated model $\mathcal{M}|_w$. For the examples described earlier, we have:

- $\mathcal{M}_{tl}, s \models [g^*]\neg(K_T f \vee K_T \neg f)$. This example corresponds to a safety property: there is no leak of information when observing an arbitrary number of g's because it is compatible with both the expectation $g^*ar^*)^*$ of the French traffic light system, and the expectation $g^*ar^*a)^*$ of the non-French one.

- $\mathcal{M}_{tl}, s \models \langle(garg)^*\rangle(K_T f)$. This example in the Existential fragment shows that we can express the existence of a sequence of observations that reveals that the traveller is in France.

- $\mathcal{M}_{tl}, s \models \langle gar\rangle\neg(K_T f \vee K_T \neg f)$. This example in the Word fragment expresses that the sequence of observations gar would keep the traveller ignorant about her whereabouts.

- $\mathcal{M}_{mp}, s \models \langle m\rangle((K_R d \wedge \neg K_A d)$. This example, also in the Word fragment, expresses that after receiving the message m, the receiver gets to know about the decision but the attacker remains ignorant.

In this thesis, the main results that involve POL are the model-checking and satisfiability problem.

Model Checking for POL: Given a finite pointed epistemic expectation model $\mathcal{M}, s$, and a formula φ, does $\mathcal{M}, s \models \varphi$? We are interested in knowing the complexity of this problem. We will also consider restrictions of the model checking when the input formula φ is restricted to be in one of the syntactic fragments: Word, Star-Free, Existential and Star-FreeExistential.

Satisfiability for POL: Given a POL formula φ, does there exist a pointed expectation model that satisfies φ? Besides providing decidability of this problem, we have also given tight complexity results for various fragments of it.

2.4.6 Epistemic Protocol Logic (EPL)

Now we introduce the syntax of EPL:

Definition 30 (Syntax of EPL)**.** *The* formulas φ *of* EPL *are given by:*

$$\varphi \quad ::= \quad \top \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid K_i\varphi \mid [\pi]\varphi \mid [\mathcal{A}, e]\varphi,$$

where $p \in \mathbf{P}$*,* $i \in \mathbf{I}$*, and* $\pi \in \mathcal{L}_{obs}$*.*

Hence the syntax of EPL is POL extended by the formula $[\mathcal{A}, e]\varphi$ which is used to express that after the protocol model is *installed*, if event e is *executable* in the current possibility then whether φ holds true. This becomes clearer in the following truth definition

Definition 31 (Truth definition of EPL)**.** *Given an epistemic expectation model* $\mathcal{M} = \langle S, \sim, V, Exp\rangle$*, a world* $s \in S$*, and a* POL*-formula* φ*, the truth of* φ *at* s*, denoted by* $\mathcal{M}, s \vDash \varphi$*, is defined by induction on* φ *as follows:*

$$\begin{aligned}
\mathcal{M}, s \vDash p &\Leftrightarrow p \in V(s)\\
\mathcal{M}, s \vDash \neg\varphi &\Leftrightarrow \mathcal{M}, s \nvDash \varphi\\
\mathcal{M}, s \vDash \varphi \wedge \psi &\Leftrightarrow \mathcal{M}, s \vDash \varphi \text{ and } \mathcal{M}, s \vDash \psi\\
\mathcal{M}, s \vDash K_i\varphi &\Leftrightarrow \text{for all } t : (s \sim_i t \text{ implies } \mathcal{M}, t \vDash \varphi)\\
\mathcal{M}, s \vDash [\pi]\varphi &\Leftrightarrow \text{for all } w \in \mathcal{L}(\pi) \cap Pre(Exp(s))\\
&\quad \text{we have } \mathcal{M}|_w, s \vDash \varphi)\\
\mathcal{M}, s \vDash [\mathcal{A}, e]\varphi &\Leftrightarrow \text{if } \mathcal{L}(f_{V(s)}(Prot(e))) \neq \emptyset\\
&\quad \text{then we have } \mathcal{A} \otimes \mathcal{M}(e, s) \vDash \varphi
\end{aligned}$$

In addition to the truth definitions of POL, the truth definition of formulas of the form $[\mathcal{A}, e]\varphi$ has been added. The formula $[\mathcal{A}, e]\varphi$ holds at s in the expectation model $\mathcal{M}$ if φ

holds in the updated model $\mathcal{A} \otimes \mathcal{M}$ with the protocols in $\mathcal{A}$ from (e, s) (where s is updated with expectation from e as per the valuation in s) whenever the event e is *executable* at s ($\mathcal{L}(f_{V(s)}(Prot(e))) \neq \emptyset$). As per the message passing example:

- $\mathcal{M}_{mpbef}, s \vDash [\mathcal{A}_{ifdm}, u]\langle m \rangle (K_R d \wedge \neg K_A d)$. This example expresses that after the protocol that "if d then send m and if not then send m'" is installed at the world where decision d is made (s), along with the fact that $Attacker$ must not distinguish between such protocol and $m + m'$, whether after observing m, the $Reciever$ knows the decision d is made and $Attacker$ still considers $\neg d$ a possibility.

Model Checking for EPL: Given a finite pointed epistemic expectation model $\mathcal{M}, s$ and an EPL formula φ, decide whether $\mathcal{M}, s \vDash \varphi$. We only discuss the decidability of this decision problem. We leave its further complexity studies along with lower bounds as part of the future work.

2.5 Complexity Classes: A Brief Introduction

In this section, we introduce some very basic concepts of complexity classes that are used in this thesis. It is to be noted that these are very brief introduction. For much broader understanding, one may look into any well-known Computational Complexity materials such as by Arora Barak [4], by Papadimitrou [47] among many others.

2.5.1 Model of Computation

The study of complexity deals with how much computational resource a computer takes to compute a function. Hence, to define such measuring metrics, a notion is needed to be defined to measure *what is computable*. To be precise, when can a function be called computable and when it can not. To define such notion, several models of computations are used. One such model, which we use in this thesis, is the Turing machine.

Before procedding further to formal definitions, it is to be noted that the problems or functions that we consider are called *decision* problems. Given some input $x \in \Sigma^\star$, from some alphabet Σ and in some representation, $f(x) \in \{0, 1\}$. On the basis of this, over all such inputs, we define the notion of *language* of that decision function. An input $x \in \mathcal{L}_f$, where $\mathcal{L}_f$ is language of function f, if and only if $f(x) = 1$. Now we proceed to give a formal definition of Turing machine:

Definition 32. *A Turing machine $\mathcal{M} = \langle Q, \Sigma, q_0 \in Q, \delta, F \rangle$ is a finite state machine such that:*

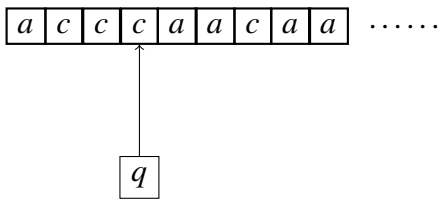

Fig. 2.12 A Turing Machine

- Q *is the set of finite states*
- Σ *is the set of letters called alphabet*
- q_0 *is the initial state when the machine starts*
- $F \subseteq Q$ *is the set of halting final states*
- $\delta : Q \times \Sigma \to Q \times \Sigma \times \{L, R\}$ *is called the transition function.*

The Turing machine, shown in figure 2.12, is imagined to have a tape, which is an infinite linear series of one-letter memory cell. There is a header which points at one cell of the tape and the machine is said to be reading that letter in the cell. For example, in the figure 2.12, the Turing machine is in state q, currently reading the letter c where the head (arrow edge) is pointing at. The head can move left or right one cell at each transition. For example, if the turing machine in the figure has the transition function such that $\delta(q, c) = (q', a, L)$, then in the next step, the head replaces the c with a, moves one cell to the left (L) and gets into state q'. A Turing machine is said to accept an input x iff it halts at some state in F.

Other than this, there are other variations in the Turing machine as well. For example, in some literature, the tape can be thought of multi-tape, that is not just one tape, but k many tapes. Also, there are variations on which way the tape is infinite, whether it is towards or right or left. It can be proven that each of these machines can be represented with the other with very small blow-up in the size of the machine.

2.5.2 Non-determinism

Another notion in the world of computation is the notion of non-determinism. Given a decision problem, or a language $\mathcal{L}$, and given an input $x \in \Sigma^\star$, to decide whether $x \in \mathcal{L}$, there are two things that have to be made sure of. Firstly, search whether there is a *witness* that can certify $x \in \mathcal{L}$. Secondly, if there is such a witness, verify if it is the correct witness. Non-determinism takes away the first part. Precisely, a non-deterministic machine does not

spend resource on searching for the existence of such a witness, it is just if such a witness exists it verifies and returns 1.

Formally, a non-deterministic Turing Machine is same as in definition 32, except the transition function is of the form $\delta(q,a) \subseteq Q \times \Sigma \times \{L,R\}$. Hence given a state $q \in Q$ and reading a letter $a \in \Sigma$, the machine chooses which state to go to, what to write on the tape and which way the head moves among some specified tuples in $Q \times \Sigma \times \{L,R\}$, such that if the input is in the language, the machine ends up halting at an accepting state.

2.5.3 Some Complexity Classes

Now we introduce the complexity classes. Although there are numerous such classes, we discuss the primary ones that are used in this thesis. A complexity class X contains languages as members. These classes are defined from the perspective of how many steps a Turing machine takes, or how much of the tape is used by the machine with respect to the input size, in order to decide the input belongs to in a member language. Before introducing such classes, let us define the notions of hardness and completeness

Definition 33 (Hardness)**.** *Given a complexity class X, and a language L, L is said to be X-hard if for any language $L' \in X$, there is a function $f : \Sigma^\star \to \Sigma^\star$ such that for any $x \in \Sigma^\star$ both the following conditions satisfy:*

- *$f(x) \in L$ iff $x \in L'$*
- *there is a polynomial p such that $f(x)$ can be computed by a Turing machine in at most $p(|x|)$ steps.*

The notion of hardness gives a lower-bound on the resources that has to be used in order to decide a language. The function f in the definition is called the *polynomial reduction* function. Essentially, when a language is L is X-hard, it means that for any algorithm or Turing machine that can be designed to decide L is bound to take at least the amount of resources (steps or tape space) that is taken by any language that is in X.

Definition 34 (Completeness)**.** *Given a complexity class X, and a language L, L is said to be X-complete if both the following condition satisfies*

- *$L \in X$*
- *L is X-hard.*

Completeness gives one the idea that to decide a language, this is as good as it gets in the worst case. Precisely, if a language L is X-complete, it means a turing machine can spend in terms of its resources in the worst case to decide L is X, also, any turing machine one can design, in the worst case it should take at least that much resource represented by X.

Next, we bring in the major complexity classes we use in the thesis.

The Class P

The class P is the set of all languages that can be decided in polynomial time. More formally

Definition 35 (P)**.** *A language $L \in \mathsf{P}$ if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ such that for any $x \in \Sigma^\star$ there is a Turing machine $\mathcal{M}$ that decides whether $x \in L$ and takes at most $p(|x|)$ many steps.*

Many model-checking problems such as model-checking DFA are problems that lie in P.

The Class NP

The class NP stands for non-deterministic polynomial. Here, just like languages in P, the decision is done by a Turing machine, but a non-deterministic one instead. Hence, the problems that lie in NP, it is not known whether it is polynomially possible to find a witness of belongingness, but if such a witness exists, it is small or polynomial in size. More formally,

Definition 36 (NP)**.** *A language $L \in \mathsf{P}$ if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ such that for any $x \in \Sigma^\star$ there is a non-deterministic Turing machine $\mathcal{M}$ that decides whether $x \in L$ and takes at most $p(|x|)$ many steps.*

The most classical problem that is in NP is the satisfiability problem for propositional logic. In fact,

Theorem 2 (Cook-Levin Theorem)**.** *The satisfiability problem of propositional logic is NP-complete.*

It is also to be noted that, any problem in P is bound to be in NP. But whether the inclusion is proper is a big open problem. In fact, the propositional satisfiability problem plays a major role here. The fact that it is NP-complete implies that if one can design a deterministic Turing machine that decides it in polynomial steps, then the inclusion becomes proper, that is $\mathsf{P} = \mathsf{NP}$, else $\mathsf{P} \subsetneq \mathsf{NP}$.

The Class PSPACE

Moving on from number of steps or time a machine takes, we now talk of how much space it uses. The class PSPACE is the class of languages that takes polynomial space to decide.

Definition 37 (PSPACE)**.** *A language $L \in$ PSPACE if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ such that for any $x \in \Sigma^\star$ there is a Turing machine $\mathcal{M}$ that decides whether $x \in L$ and takes at most $p(|x|)$ many tape cells.*

Just like NP, there is a classical PSPACE-complete language as well called the TQBF. A QBF or Quantified Boolean Formula is of the form $Q_1x_1Q_2x_2 \ldots Q_nx_n\varphi(x_1, x_2, \ldots, x_n)$, where $Q_i \in \{\exists, \forall\}$ and $\varphi(x_1, x_2, \ldots, x_n)$ is a propositional formula on variables $x_1, \ldots, x_n$. For example, $\forall x \exists y \forall z (x \vee \neg y) \wedge (y \vee z)$ is a QBF. Deciding whether, given such a formula, it can be made True or not is called TQBF. Just like NP, a corresponding complexity class is also defined for PSPACE that decides using non-deterministic Turing machines called the NPSPACE. But as it turns out, non-determinism does not give much power in terms of space:

Theorem 3 (Savitch's Theorem)**.** NPSPACE $\subseteq$ PSPACE

The basic idea why this is happening is because in time, the number of steps add up. But in case of space, there are spaces, or tape cells that has been used for some previous steps could be re-used. More clearly, whenever a machine makes a choice and it realises it was a bad choice, in terms that the initial choice will not guarantee the decision, when it moves on to make another choice, it simply reuses the space used for the previous choice.

Higher Complexity Classes

Just like in the polynomial level, there are similar classes that can be defined in exponential levels as well. Before formally defining them, let us introduce this expression $exp(m, n)$, where $m, n \in \mathbb{N}$, which is defined as

$$exp(0, n) = poly(n)$$
$$exp(m, n) = 2^{exp(m-1,n)}$$

where $poly(n)$ is a polynomial over n.

Now we define a series of complexity classes. The basic idea is, higher the m in $exp(m, n)$, one gets similar classes on that m level.

Definition 38 ($m - \mathsf{EXPTIME}$)**.** *A language* $L \in m - \mathsf{EXPTIME}$ *if there exists a polynomial* $p : \mathbb{N} \to \mathbb{N}$ *such that for any* $x \in \Sigma^\star$ *there is a Turing machine* $\mathcal{M}$ *that decides whether* $x \in L$ *and takes at most* $exp(m, p(|x|))$ *many steps, where* $m > 0$*.*

Similarly, using non-deterministic Turing Machines:

Definition 39 ($m - \mathsf{NEXPTIME}$)**.** *A language* $L \in m - \mathsf{NEXPTIME}$ *if there exists a polynomial* $p : \mathbb{N} \to \mathbb{N}$ *such that for any* $x \in \Sigma^\star$ *there is a non-deterministic Turing machine* $\mathcal{M}$ *that decides whether* $x \in L$ *and takes at most* $exp(m, p(|x|))$ *many steps, where* $m > 0$*.*

And lastly, from the perspective of space:

Definition 40 ($m - \mathsf{EXPSPACE}$)**.** *A language* $L \in m - \mathsf{EXPSPACE}$ *if there exists a polynomial* $p : \mathbb{N} \to \mathbb{N}$ *such that for any* $x \in \Sigma^\star$ *there is a Turing machine* $\mathcal{M}$ *that decides whether* $x \in L$ *and takes at most* $exp(m, p(|x|))$ *many tape cells, where* $m > 0$*.*

In this thesis, for $m = 1$, we dont specify m. Hence, $1 - \mathsf{EXPTIME} = \mathsf{EXPTIME}$, $1 - \mathsf{NEXPTIME} = \mathsf{NEXPTIME}$ and $1 - \mathsf{EXPSPACE} = \mathsf{EXPSPACE}$. And lastly, we have the following hierarchy:

$$\mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{PSPACE} \subseteq_{m>0} (m - \mathsf{EXPTIME} \subseteq m - \mathsf{NEXPTIME} \subseteq m - \mathsf{EXPSPACE})$$

It is to be noted that it is not yet known anything about the inclusivity among $\mathsf{P}, \mathsf{NP}, \mathsf{PSPACE}$, also among $m - \mathsf{EXPTIME}, m - \mathsf{NEXPTIME}, m - \mathsf{EXPSPACE}$, but it is known that $\mathsf{P} \neq \mathsf{EXPTIME} \neq 2 - \mathsf{EXPTIME} \ldots$ and so on.

Chapter 3

POL Model Checking

3.1 Introduction

In this chapter, we look into the model-checking problem of the Public Observation Logic. Precisely, given an epistemic expectation model $\mathcal{M} = \langle W, R, V, Exp \rangle$, as introduced in the last chapter (like the model $\mathcal{M}_{mp}$ in figure 2.9), pointed at some world $s \in W$, and given some POL formula φ, the model-checking problem is deciding whether $\mathcal{M}, s \vDash \varphi$.

We look into the upperbound on the time complexity in two different ways. First we look into a basic decidabilitity algorithm which takes exponential time. Later, we give a much efficient algorithm with tight bound. The main statement we are trying to prove in this chapter is:

Theorem 4. *The model checking of* POL *is* PSPACE*-complete.*

Besides an exhaustive exponential time decidability algorithm, the second algorithm we prove to be in PSPACE. Both the algorithms use techniques from automata theory, including reasoning using them for complexity analysis. We also prove matching lowerbounds from two syntactic fragments of POL.

A simple utility of such model-checking algorithm can be in verification of certain properties in the model [53]. Take the message passing example model $\mathcal{M}_{mp}$ in figure 2.9. To verify whether after message m is passed, although $Receiver$ knows about the decision, $Attacker$ still does not know it, one can run model-checking to decide whether $\mathcal{M}_{mp}, s \vDash \langle m \rangle K_{Receiver} d \wedge \hat{K}_{Attacker} \neg d$.

Outline. Section 3.2 deals with a brute force decidable model-checking algorithm that runs in exponential time. Section 3.3 proves the model checking poblem to be PSPACE-complete.

3.2 Decidability

In this section, we give a brute-force algorithm for the model-checking problem for POL. Furthermore, we also give an algorithm using a dynamic programming approach.

The unique case for POL model-checking, other than the epistemic logic operators is the case for model-checking the formula $\langle\pi\rangle\psi$. By the definition 29,

$$\mathcal{M}, s \vDash \langle\pi\rangle\psi \text{ iff } \exists w \in \mathcal{L}(\pi) \cap Pre(Exp(s)) : \mathcal{M}|_w, s \vDash \psi$$

Here $Pre(Exp(s))$ is the language of all the prefixes of $Exp(s)$ (defined in page 2.4.2). Hence, for a decision procedure of such formula, we need to find an upper bound on the size of such w such that the procedure can search all such words of at most that size and verify the rest.

Definition 41. *For any model M, starting state s and finite POL formula φ we define $\Lambda^{M,s,\varphi}$ recursively as follows:*

- ***If φ is a a propositional formula***

$$\Lambda^{M,s,\varphi} = \begin{cases} \mathcal{L}(Pre(Exp(s))) & \text{if } \mathcal{M}, s \vDash \varphi \\ \emptyset & \text{if } \mathcal{M}, s \nvDash \varphi \end{cases}$$

- ***If*** $\varphi = \neg\psi$

$$\Lambda^{M,s,\varphi} = \overline{\Lambda^{M,s,\psi}}$$

- ***If*** $\varphi = \psi' \vee \psi$

$$\Lambda^{M,s,\varphi} = \Lambda^{M,s,\psi'} \cup \Lambda^{M,s,\psi}$$

- ***If*** $\varphi = \hat{K}_i\psi$

$$\Lambda^{M,s,\varphi} = (\cup_{t:t\sim_i s}\Lambda^{M,t,\psi}) \cap \mathcal{L}(Pre(Exp(s)))$$

- ***If*** $\varphi = \langle\pi\rangle\psi$

$$\Lambda^{M,s,\varphi} = \left\{w \mid \exists w' \in \mathcal{L}(\pi) \text{ and } \mathcal{L}(Exp(s)\backslash ww') \neq \emptyset \text{ and } ww' \in \Lambda^{M,s,\psi}\right\}$$

As shown the following lemma, Intuitively, $\Lambda^{M,s,\varphi}$ is the set of words that are publicly observable in s and such that by observing them, φ still holds in s.

Lemma 5. *For any model M, starting state s and finite POL formula φ*

$$\Lambda^{M,s,\varphi} = \{w \mid M|_w, s \vDash \varphi\}$$

Proof. We prove this by induction on the size of φ. Consider $\Delta^{M,s,\varphi} = \{w \mid M|_w, s \vDash \varphi\}$
Base Case. φ is a propositional formula. Hence by definition 41,

$$\Lambda^{M,s,\varphi} = \begin{cases} \mathcal{L}(Pre(Exp(s))) & \text{if } \mathcal{M}, s \vDash \varphi \\ \varphi & \text{if } \mathcal{M}, s \nvDash \varphi \end{cases}$$

If $M, s \nvDash \varphi$ then $\Lambda^{M,s,\varphi}$ is empty and the result trivially stands true.
Now consider $M, s \vDash \varphi$. Let $w \in \Lambda^{M,s,\varphi}$.

$$\begin{aligned} w \in \Lambda^{M,s,\varphi} &\implies w \in \mathcal{L}(Pre(Exp(s))) \\ &\implies M|_w, s \vDash \varphi, \text{ since } \mathcal{L}(Exp(s)\backslash w) \neq \emptyset \\ &\implies w \in \Delta^{M,s,\varphi} \end{aligned}$$

Now let's consider $w \in \Delta^{M,s,\varphi}$.

$$\begin{aligned} w \in \Delta^{M,s,\varphi} &\implies M|_w, s \vDash \varphi \\ &\implies \mathcal{L}(Exp(s)\backslash w) \neq \emptyset \\ &\implies w \in \mathcal{L}(Pre(Exp(s))) \\ &\implies w \in \Lambda^{M,s,\varphi} \end{aligned}$$

Induction Hypothesis. For any finite model M, starting state s, any POL formula ψ where $|\psi| \leq m$, $\Lambda^{M,s,\psi} = \Delta^{M,s,\psi}$.

Inductive Step. Here we will go case by case on the form of φ.

- $\varphi = \neg\psi$.

$$\begin{aligned} w \in \Lambda^{M,s,\neg\psi} &\implies w \in \overline{\Lambda^{M,s,\psi}} \\ &\implies w \notin \Lambda^{M,s,\psi} \\ &\implies w \notin \Delta^{M,s,\psi}, \text{ by Induction Hypothesis} \\ &\implies M|_w, s \nvDash \psi \\ &\implies M|_w, s \vDash \neg\psi \\ &\implies \Delta^{M,s,\neg\psi} \end{aligned}$$

$$\begin{aligned} w \in \Delta^{M,s,\neg\psi} &\implies M|_w, s \vDash \neg\psi \\ &\implies M|_w, s \nvDash \psi \\ &\implies w \in \overline{\Delta^{M,s,\psi}} \\ &\implies w \in \overline{\Lambda^{M,s,\psi}}, \text{ by Induction Hypothesis.} \\ &\implies w \in \Lambda^{M,s,\neg\psi}, \text{ by definition 41} \end{aligned}$$

- $\varphi = \psi \vee \psi'$.

$$\begin{aligned} w \in \Lambda^{M,s,\varphi} &\implies w \in \Lambda^{M,s,\psi} \cup \Lambda^{M,s,\psi'} \\ &\implies w \in \Delta^{M,s,\psi} \cup \Delta^{M,s,\psi'}, \text{ by Induction Hypothesis} \\ &\implies w \in \Delta^{M,s,\psi} \text{ or } w \in \Delta^{M,s,\psi'} \\ &\implies M|_w, s \vDash \psi \text{ or } M|_w, s \vDash \psi' \\ &\implies M|_w, s \vDash \psi \vee \psi' \\ &\implies w \in \Delta^{M,s,\varphi} \end{aligned}$$

$$\begin{aligned} w \in \Delta^{M,s,\psi\vee\psi'} &\implies M|_w, s \vDash \psi \text{ or } M|_w, s \vDash \psi' \\ &\implies w \in \Delta^{M,s,\psi} \text{ or } w \in \Delta^{M,s,\psi'} \\ &\implies w \in \Lambda^{M,s,\psi} \text{ or } \Lambda^{M,s,\psi'}, \text{ by Induction Hyporthesis} \\ &\implies w \in \Lambda^{M,s,\psi} \cup \Lambda^{M,s,\psi'} \\ &\implies w \in \Lambda^{M,s,\psi\vee\psi'}, \text{ by definition 41} \end{aligned}$$

- $\varphi = \hat{K}_i\psi$.

$$
\begin{aligned}
w \in \Lambda^{M,s,\hat{K}_i\psi} &\implies w \in (\cup_{t:t\sim_i s}\Lambda^{M,t,\psi}) \cap \mathcal{L}(Pre(Exp(s))) \\
&\implies w \in (\cup_{t:t\sim_i s}\Delta^{M,t,\psi}) \cap \mathcal{L}(Pre(Exp(s))), \text{ by Induction Hypothesis} \\
&\implies (\exists t \sim_i s \text{ such that } w \in \Delta^{M,t,\psi}) \text{ and } w \in \mathcal{L}(Pre(Exp(s))) \\
&\implies (\exists t \sim_i s \text{ such that } M|_w, t \vDash \psi) \text{ and } \mathcal{L}(Exp(s)\backslash w) \neq \emptyset \\
&\implies M|_w, s \vDash \hat{K}_i\psi \\
&\implies w \in \Delta^{M,s,\hat{K}_i\psi}
\end{aligned}
$$

$$
\begin{aligned}
w \in \Delta^{M,s,\hat{K}_i\psi} &\implies M|_w, s \vDash \hat{K}_i\psi \\
&\implies \mathcal{L}(Exp(s)\backslash w) \neq \emptyset \text{ and } \exists t \sim_i s \text{ such that } M|_w, t \vDash \psi \\
&\implies w \in \mathcal{L}(Pre(Exp(s))) \text{ and } \exists t \sim_i s \text{ such that } w \in \Delta^{M,t,\psi} \\
&\implies w \in \mathcal{L}(Pre(Exp(s))) \text{ and } \exists t \sim_i s \text{ such that } w \in \Lambda^{M,t,\psi}, \text{ by IH} \\
&\implies w \in \mathcal{L}(Pre(Exp(s))) \text{ and } w \in (\cup_{t:t\sim_i s}\Lambda^{M,t,\psi}) \\
&\implies w \in \mathcal{L}(Pre(Exp(s))) \cap (\cup_{t:t\sim_i s}\Lambda^{M,t,\psi}) \\
&\implies w \in \Lambda^{M,s,\hat{K}_i\psi}, \text{ by definition 2.1}
\end{aligned}
$$

- $\varphi = \langle\pi\rangle\psi$

$$
\begin{aligned}
w \in \Lambda^{M,s,\langle\pi\rangle\psi} &\implies \exists w' \in \mathcal{L}(\pi) \text{ and } \mathcal{L}(Exp(s)\backslash ww') \neq \emptyset \text{ and } ww' \in \Lambda^{M,s,\psi} \\
&\implies \exists w' \in \mathcal{L}(\pi) \text{ and } \mathcal{L}(Exp(s)\backslash ww') \neq \emptyset \text{ and } ww' \in \Delta^{M,s,\psi}, \text{ by Induction Hypothesis} \\
&\implies \exists w' \in \mathcal{L}(\pi) \text{ and } \mathcal{L}(Exp(s)\backslash ww') \neq \emptyset \text{ and } M|_{ww'}, s \vDash \psi \\
&\implies M|_w, s \vDash \langle\pi\rangle\psi \\
&\implies w \in \Delta^{M,s,\langle\pi\rangle\psi}
\end{aligned}
$$

$$
\begin{aligned}
w \in \Delta^{M,s,\langle\pi\rangle\psi} &\implies M|_w, s \vDash \langle\pi\rangle\psi \\
&\implies \exists w' \in \mathcal{L}(\pi) \text{ such that } \mathcal{L}(Exp(s)\backslash ww') \neq \emptyset \text{ and } M|_{ww'}, s \vDash \psi \\
&\implies \exists w' \in \mathcal{L}(\pi) \text{ and } \mathcal{L}(Exp(s)\backslash ww') \neq \emptyset \text{ and } ww' \in \Delta^{M,s,\psi} \\
&\implies \exists w' \in \mathcal{L}(\pi) \text{ and } \mathcal{L}(Exp(s)\backslash ww') \neq \emptyset \text{ and } ww' \in \Lambda^{M,s,\psi}, \text{ by Induction Hypothesis} \\
&\implies w \in \Lambda^{M,s,\langle\pi\rangle\psi}
\end{aligned}
$$

□

Hence, while model-checking formulas of type $\langle\pi\rangle\psi$, the publicly observable $w \in \mathcal{L}(\pi)$ will always belong in the set $\Lambda^{\mathcal{M},s,\langle\pi\rangle\psi}$. But note that, such a set can have countably infinite many words. However, such a set is mostly built out of regular languages (since π as well as the expectations *Exp* of any state of world of the model is regular). Hence, we prove the following lemma.

Lemma 6. *For any model M, starting state s and finite POL formula φ the set $\Lambda^{M,s,\varphi}$ is regular.*

Proof. We prove this by inducting on the size of φ.
Base Case. Consider φ to be propositional formula. By definition 41:

$$
\Lambda^{M,s,\varphi} = \begin{cases} \mathcal{L}(Pre(Exp(s))) & \text{if } \mathcal{M}, s \vDash \varphi \\ \varphi & \text{if } \mathcal{M}, s \nvDash \varphi \end{cases}
$$

The language of empty set is regular. Also, by definition of any finite model M, for a state s, $Exp(s)$ is regular. Hence the language of every possible prefix of $Exp(s)$, that is $\mathcal{L}(Pre(Exp(s)))$ is also regular (Given the DFA of $Exp(s)$, mark every state that can reach the accepting state in it as an accepting state, including the start state.).

Induction Hypothesis. For any finite model M, starting state s and a POL formula ψ, such that $|\psi| \leq m$, for a positive integer m, $\Lambda^{M,s,\psi}$ is regular.

Inductive Step.

- $\varphi = \langle\pi\rangle\psi$.

  Let $\mathcal{A}_1 = (Q^{\mathcal{A}_1}, \Sigma, \delta^{\mathcal{A}_1}, q_0^{\mathcal{A}_1}, F^{\mathcal{A}_1})$ be the DFA accepting the language

$$
\{w \mid w \in \mathcal{L}(Pre(Exp(s))) \text{ and } w \in \Lambda^{M,s,\psi}\}.
$$

Let $\mathcal{A}_\pi = (Q^{\mathcal{A}_\pi}, \Sigma, \delta^{\mathcal{A}_\pi}, q_0^{\mathcal{A}_\pi}, F^{\mathcal{A}_\pi})$ be the DFA accepting $\mathcal{L}(\pi)$.

And let $\mathcal{A}_2 = (Q^{\mathcal{A}_2}, \Sigma, \delta^{\mathcal{A}_2}, q_0^{\mathcal{A}_2}, F^{\mathcal{A}_2})$ be the product DFA accepting the language

$$\{w \mid w \in \mathcal{L}(Pre(Exp(s))) \text{ and } w \in \Lambda^{M,s,\psi}\} \cap \mathcal{L}(\pi).$$

That is $Q^{\mathcal{A}_2} = Q^{\mathcal{A}_1} \times Q^{\mathcal{A}_\pi}$ and $F^{\mathcal{A}_2} = F^{\mathcal{A}_1} \times F^{\mathcal{A}_\pi}$. Let T be the set of states in $\mathcal{A}_2$ from where the final states are reachable.

Consider the DFA $\mathcal{B} = (Q^{\mathcal{A}_1}, \Sigma, \delta^{\mathcal{A}_1}, q_0^{\mathcal{A}_1}, F^{\mathcal{B}})$ whose states and transition functions are same as $\mathcal{A}_1$ and the set of final states is defined as follows:

$$F^{\mathcal{B}} = \{t \in Q^{\mathcal{A}_1} \mid (t, q_0^{\mathcal{A}_\pi}) \in T\}.$$

Claim 7. *The DFA $\mathcal{B}$ is the DFA accepting the language*

$$\left\{w \mid \exists w' \in \mathcal{L}(\pi) \text{ and } \mathcal{L}(Exp(s)\backslash ww') \neq \emptyset \text{ and } ww' \in \Lambda^{M,s,\psi}\right\}.$$

Proof. A state t is in $F^{\mathcal{B}}$ iff $(t, q_0^{\mathcal{A}_\pi}) \in T$, that is (by definition of T) means there exists w' such that $\widehat{\delta^{\mathcal{A}_1}}(t, w')^1 \in F^{\mathcal{A}_1}$ and $\widehat{\delta^{\mathcal{A}_\pi}}(s, w')^1 \in F^{\mathcal{A}_\pi}$. The second condition means $w' \in \mathcal{L}(\pi)$.

Intuitively, this means that any state t is a final state which is tracking the prefix of $Exp(s)$ as well as once can start verifying π in the product automata as well ($q_0^{\mathcal{A}_\pi}$).

So w is accepted by $\mathcal{B}$ iff $\widehat{\delta^{\mathcal{A}_1}}(q_0^{\mathcal{A}_1}, w) \in F^{\mathcal{B}}$ that is there exists w' such that $\widehat{\delta^{\mathcal{A}_1}}(q_0^{\mathcal{A}_1}, ww') = \widehat{\delta^{\mathcal{A}_1}}(\widehat{\delta^{\mathcal{A}_1}}(q_0^{\mathcal{A}_1}, w), w') \in F^{\mathcal{A}_1}$ and $w' \in \mathcal{L}(\pi)$. □

- $\varphi = \neg\psi$. Here $\Lambda^{M,s,\varphi} = \overline{\Lambda^{M,s,\psi}}$. By Induction hypothesis, $\Lambda^{M,s,\psi}$ is regular and since regular language is closed under complementation, $\Lambda^{M,s,\varphi}$ is regular.
- $\varphi = \psi \vee \psi'$. Henre $\Lambda^{M,s,\psi\vee\psi'} = \Lambda^{M,s,\psi} \cup \Lambda^{M,s,\psi'}$. By Induction hypothesis, $\Lambda^{M,s,\psi}$ and $\Lambda^{M,s,\psi'}$ is regular. Since regular language is closed under union, $\Lambda^{M,s,\varphi}$ is regular.
- $\varphi = \hat{K}_i\psi$. Here $\Lambda^{M,s,\varphi} = (\cup_{t\sim_i s}\Lambda^{M,t,\psi}) \cap \mathcal{L}(Pre(Exp(s)))$. Since regular language is closed under finite union and intersection, $\Lambda^{M,s,\varphi}$ is regular.

□

Decide (Algorithm 1) is an algorithm for checking if $M, s \vDash \varphi$.

[1]Here $\widehat{\delta}$, where δ is a usual transition function of a DFA, is the conventional extension of the transition function from letter to words as defined in [40].

Algorithm 1 Decide(M, s, φ)

1: **procedure** Decide$(\mathcal{M}, s, \varphi)$
2: **if** φ is a propositional formula **then**
3: Return True if $\varphi \in V(s)$; False otherwise
4: **if** $\varphi = \psi' \vee \psi$ **then**
5: Return Decide(M, s, ψ) or Decide(M, s, ψ')
6: **if** $\varphi = \langle\pi\rangle\psi$ **then**
7: **if** there exist $|w| \leq 2^{|\pi|} \times 2^{|Exp(s)|} \times 2^{\max_{s\in S}(|Exp(s)|)\times O(m^{n-1})}$, where m is the size of M in terms of the states and relation, $n = |\varphi|$, such that $w \in \mathcal{L}(\pi)$ and $w \in \mathcal{L}(Pre(Exp(s)))$ and Decide$(M|_w, s, \psi)$ **then**
8: Return True
9: **else**
10: Return False
11: **if** $\varphi = \hat{K}_i\psi$ **then**
12: **if** if there exists a $t \in S$ such that $t \sim_i s$ and Decide(M, t, ψ) **then**
13: Return True
14: **else**
15: Return False
16: **if** $\varphi = \neg\psi$ **then**
17: Return not Decide(M, s, ψ)

Definition 42 (POL Syntactic Tree)**.** *The POL Syntactic tree for a POL formula φ is constructed as:*

- *For a node φ, where $\varphi \in \mathcal{P}$, where $\mathcal{P}$ is the set of propositional formulas, keep the node as it is.*
- *For a node $\varphi = \hat{K}_i\psi$, create POL syntactic subtree for node ψ rooted at node $\hat{K}_i$.*
- *For a node $\varphi = \langle\pi\rangle\psi$, create POL syntactic subtree for node ψ rooted at $\langle\pi\rangle$.*
- *For a node $\varphi = \neg\psi$, where $\psi \notin \mathcal{P}$, create POL syntactic subtree for node ψ rooted at $\neg$.*
- *For a node $\varphi = \psi \vee \psi'$ where $\psi \notin \mathcal{P}$ and where $\psi' \notin \mathcal{P}$, create POL syntactic subtree for left node ψ and right node ψ' rooted at $\vee$*

Also, consider $P_M = max_{s\in S}(|Exp(s)|)$.

Next we prove the size of the DFA of the regular language $\Lambda^{\mathcal{M},s,\varphi}$ in order to gain an upper-bound on the search-space of the words (bound on the word size in line 7 of algorithm 1) and prove the correctness of the algorithm Decide.

Lemma 8. *For any finite model M, starting state s and a finite POL formula φ, the size of DFA of $\Lambda^{M,s,\varphi}$, $D_{\Lambda^{M,s,\varphi}}$, is given by $|D_{\Lambda^{M,s,\varphi}}| \leq 2^{P_M(O(m^{(n-1)}))} \times 2^{P_M}$, where n is the depth of POL Syntactic tree of φ and $|M| = m$*

Proof. Given a finite model M,

$$S_M(n) = \max_{s\in S,\varphi:|\varphi|\leq n} \Big\{ \text{the size of the DFA accepting the regular language } \Lambda^{M,s,\varphi} \Big\}$$

Now we will form a recursion of $S_M(n)$ for each case as in definition 41:

- φ is propositional case.
  $Exp(s)$ is by model definition a regular expression. Hence the size of the DFA of $Exp(s)$ is $D_{Exp(s)} \leq 2^{|Exp(s)|}$. Hence $S_M(1) \leq 2^{P_M}$.
- $\varphi = \neg\psi$. Here $\Lambda^{M,s,\varphi} = \overline{\Lambda^{M,s,\psi}}$. Hence $S_M(n) = S_M(n-1)$.
- $\varphi = \psi \vee \psi'$. Here $\Lambda^{M,s,\varphi} = \Lambda^{M,s,\psi} \cup \Lambda^{M,s,\psi'}$. Hence $S_M(n) \leq S_M(n-1)^2$
- $\varphi = \hat{K}_i\psi$. Here $\Lambda^{M,s,\varphi} = (\cup_{t\sim_i s}(\Lambda^{M,t,\psi})) \cap \mathcal{L}(Pre(Exp(s)))$. Considering the maximum adjacency a state s can have is $|M|$, then $S_M(n) \leq S_M(n-1)^m \times 2^{P_M}$.
- $\varphi = \langle\pi\rangle\psi$. By definition 41, $\Lambda^{M,s,\varphi} = \{w \mid \exists w' \in \mathcal{L}(\pi)$ such that $\mathcal{L}(Exp(s)\backslash ww') \neq \emptyset$ and $ww' \in \Lambda^{M,s,\psi}\}$. In Lemma 6, a DFA $\mathcal{B}$ was constructed for this language using a product DFA of $Pre(Exp(s))$ and $\Lambda^{M,s,\psi}$. Hence $S_M(n) \leq 2^{P_M} \times S_M(n-1)$.

From the above cases it can be deduced that the last two cases produce the largest sized DFAs. Hence, $S_M(n) \leq (S_M(n-1)^m \times 2^{P_M})$.

$$S_M(n) \leq S_M(n-1)^m \times 2^{P_M}$$

Observation 9. $S_M(n) \leq 2^{P_M \times O(m^{n-1})} \times 2^{P_M}$

Proof. We will prove it by induction on n.
Base Case. $n = 1$. Therefore φ is propositional.Hence $S_M(1) \leq 2^{P_M} \leq 2^{P_M \times O(1)} \times 2^{P_M}$.

Induction Hypothesis. For every integer $i \leq n$, $S_M(n) \leq 2^{P_M \times O(m^{i-1})} \times 2^{P_M}$.

Inductive Step. Consider $S_M(n+1)$.

$$\begin{aligned} S_M(n+1) &\leq S_M(n)^m \times 2^{P_M} \\ &\leq (2^{P_M \times O(m^{n-1})} \times 2^{P_M})^m \times 2^{P_M}, \text{ by Induction Hypothesis} \\ &\leq 2^{P_M(O(m^n)+m)} \times 2^{P_M} \\ &\leq 2^{P_M(O(m^n))} \times 2^{P_M} \end{aligned}$$

□

Hence $S_M(n) \leq 2^{P_M(O(m^{(n-1)}))} \times 2^{P_M}$ □

Finally, we prove the correctness.

Theorem 10. *The above algorithm* $\mathsf{Decide}(M, s, \varphi)$ *correctly decides if* $M, s \vDash \varphi$.

Proof. We will prove $\mathsf{Decide}(M, s, \varphi)$ returns True iff $M, s \vDash \varphi$ by induction on the size of φ.

Base Case. Consider φ to be propositional formula. $M, s \vDash \varphi$ iff the propositional variables are assigned according to $V(s)$ iff $\mathsf{Decide}(M, s, \varphi)$ returns True.

Induction Hypothesis. For any POL formula $|\psi| \leq m$, any finite model M and any state s, $\mathsf{Decide}(M, s, \psi)$ returns True iff $M, s \vDash \psi$.

Inductive Step. We will now go case by case over the form of φ.

- $\varphi = \psi \vee \psi'$.

$$\begin{aligned} M, s \vDash \psi \vee \psi' \text{ iff } & M, s \vDash \psi \text{ or } M, s \vDash \psi' \\ & \text{iff } \mathsf{Decide}(M, s, \psi) = \mathsf{True} \\ & \quad \text{or } \mathsf{Decide}(M, s, \psi') = \mathsf{True} \text{ , by Induction Hypothesis} \\ & \text{iff } \mathsf{Decide}(M, s, \varphi) = \mathsf{True}, \text{ as per line 4- 5} \end{aligned}$$

- $\varphi = \hat{K}_i \psi$.

$$\begin{aligned} M, s \vDash \hat{K}_i \psi \text{ iff } & \exists t \sim_i s \text{ such that } M, t \vDash \psi \\ & \text{iff } \exists t \sim_i s \text{ such that } \mathsf{Decide}(M, t, \psi) = \mathsf{True} \\ & \text{iff } \mathsf{Decide}(M, s, \varphi) = \mathsf{True}, \text{ as per line 12-15} \end{aligned}$$

- $\varphi = \langle\pi\rangle\psi$.

$$M, s \vDash \langle\pi\rangle\psi \text{ iff } \exists w \in \mathcal{L}(\pi) \text{ such that } \mathcal{L}(Exp(s)\backslash w) \neq \emptyset \text{ and } M|_w, s \vDash \psi$$
$$\text{iff } \exists w \in (\mathcal{L}(\pi) \cap \mathcal{L}(Pre(Exp(s))) \cap \Lambda^{M,s,\psi}), \text{ by Lemma 5}$$

Now, by Lemma 6, $\Lambda^{M,s,\psi}$ is regular, and by Lemma 8, the size of the DFA of $\Lambda^{M,s,\psi}$ is at most $2^{P_M(O(m^{n-1}))} \times 2^{P_M}$. Also π and $Pre(Exp(s))$ being regular, hence the size of DFA of $\mathcal{L}(\pi) \cap \mathcal{L}(Pre(Exp(s))) \cap \Lambda^{M,s,\psi}$ will be at most $2^{|\pi|} \times 2^{|Exp(s)|} \times 2^{\max_{s\in S}(|Exp(s)|)\times O(m^{n-1})}$. Therefore from above deduction

$$M, s \vDash \langle\pi\rangle\psi \text{ iff } \exists w \in (\mathcal{L}(\pi) \cap \mathcal{L}(Pre(Exp(s))) \cap \Lambda^{M,s,\psi})$$
$$\text{iff } \exists w, |w| \leq 2^{|\pi|} \times 2^{|Exp(s)|} \times 2^{\max_{s\in S}(|Exp(s)|)\times O(m^{n-1})}$$
$$\text{and } w \in \mathcal{L}(\pi) \text{ and } w \in \mathcal{L}(Pre(Exp(s))) \text{ and}$$
$$\mathsf{Decide}(M|_w, s, \psi) = \mathsf{True}$$

- $\varphi = \neg\psi$

$$M, s \vDash \neg\psi \text{ iff } M, s \nvDash \psi$$
$$\text{iff } \mathsf{Decide}(M, s, \psi) = \mathsf{False}, \text{ by Induction Hypothesis}$$
$$\text{iff } \mathsf{Decide}(M, s, \varphi) = \mathsf{True}, \text{ as in Line 17}$$

□

Hence we have an algorithm that correctly decides the model-checking problem for POL (Decide). Also, by Lemma 8, the algorithm terminates. In the next subsection, we look into further optimising this algorithm. Note that, by Lemma 8, for formula $\langle\pi\rangle\psi$, the size of witness w can range at most upto double exponential in size with respect to input (line 7). We take a dynamic programming approach to reduce that bound in the next subsection.

3.2.1 A Dynamic Approach

In this section we will be computing an upper bound on the POL model checking problem using a dynamic approach.

Observation 11. *For a finite model $\mathcal{M} = \langle S, \sim, V, Exp\rangle$, a world $s \in S$, and for every pair of words $w, w' \in \Sigma^*$, if w and w' are simulated in the $DFA(Exp(s)) = (Q, \Sigma, \delta, q^0, F)$, with both simulations ending in the state $q \in Q$, then $DFA(Exp(s)\backslash w) = DFA(Exp(s)\backslash w')$.*

Proof. From the definition of residue (definition 15) we have $u \in \mathcal{L}(DFA(Exp(s)\backslash w))$ iff $wu \in \mathcal{L}(DFA(Exp(s)))$. Note that,

$$\begin{aligned} & u \in \mathcal{L}(DFA(Exp(s)\backslash w)) \\ \textit{iff } & wu \in \mathcal{L}(DFA(Exp(s))) \\ \textit{iff } & \widehat{\delta}(q^0, wu) \in F \\ \textit{iff } & \widehat{\delta}(\widehat{\delta}(q^0, w), u) \in F \\ \textit{iff } & \widehat{\delta}(\widehat{\delta_s}(q^0, w'), u) \in F, \end{aligned}$$

the last if and only if holds as by assumption $\widehat{\delta}(q^0, w) = \widehat{\delta}(q^0, w')$. Finally note that,

$$\begin{aligned} \widehat{\delta}(\widehat{\delta}(q^0, w'), u) \in F \textit{ iff } & w'u \in \mathcal{L}(DFA(Exp(s))) \\ \textit{iff } & u \in \mathcal{L}(DFA(Exp(s)\backslash w')) \end{aligned}$$

□

Hence we prove that, even though there are countable infinite many unique words in $\Sigma^\star$, given a regular expression, there are similar DFA representation of residue over different words. This helps us bounding the number of *unique* projections $\mathcal{M}|_w$ given $\mathcal{M}$ over any $w \in \Sigma^\star$. Using Observation 11 we can obtain an upper bound on the size of the set $\mathcal{M}_{\Sigma^\star} = \{\mathcal{M}|_w \mid w \in \Sigma^*\}$.

Lemma 12. *Given a finite* POL *model $\mathcal{M} = \langle S, \sim, V, Exp\rangle$, the size of $|\mathcal{M}_{\Sigma^\star}| \leq \Pi_{t \in \mathcal{M}} 2^{|Exp(t)|}$.*

Proof. For any given world $s \in S$ and $DFA(Exp(s)) = (Q_s, \Sigma, \delta_s, q_s^0, F_s)$, relation $Z_s^{\mathcal{M}} \subseteq \Sigma^* \times \Sigma^*$ is defined as:

$$(w, u) \in Z_s^{\mathcal{M}} \text{ iff } \widehat{\delta_s}(q_s^0, w) = \widehat{\delta_s}(q_s^0, u)$$

Clearly, $Z_s^{\mathcal{M}}$ is an equivalence relation, hence creates a partition over Σ^*. Therefore by Observation 11, for any pair $(w, w') \in Z_s^{\mathcal{M}}$, $DFA(Exp(s)\backslash w) = DFA(Exp(s)\backslash w')$. In other words, any w from a single partition $[w]_s$ over Σ^* by $Z_s^{\mathcal{M}}$, will produce the same $DFA(Exp(s)\backslash w)$. Therefore, number of partitions over Σ^* by $Z_s^{\mathcal{M}}$ is at most the number of states in $DFA(Exp(s))$, that is, $2^{|Exp(s)|}$.

For the model $\mathcal{M}$, let $n = |S|$. Consider the following n-tuple $T^w_{\mathcal{M}} = (D^w_{s_1}, \ldots, D^w_{s_n})$ for all $w \in \Sigma^*$, where $D^w_{s_i} = DFA(Exp(s_i)\backslash w)$ for the world $s_i \in S$. Note that $|\mathcal{M}_{\Sigma^\star}| = |\{T^w_{\mathcal{M}} \mid w \in \Sigma^*\}|$, because for every $w \in \Sigma^*$, $T^w_{\mathcal{M}}$ is the tuple enumerating the Exp function of $\mathcal{M}|_w$ according to the worlds of $\mathcal{M}$ (Note that, if a world vanishes in $\mathcal{M}|_w$ for some w, the corresponding DFA will be of empty language).

For each world s_i, the total number of $D^w_{s_i}$ possible is at most $2^{|Exp(s_i)|}$, and hence the total number of such tuples possible is $\Pi_{t\in\mathcal{M}} 2^{|Exp(t)|}$. □

Consider $|Q_M| = \max_{s\in S}(|Q_s|)$, where for every $s \in S$, Q_s is the set of states in the $DFA(Exp(s))$. Consider the following definition required to prove the following theorem.

Definition 43. *Given a model M, for any $w \in \Sigma^*$, define $\Delta^w_M = \{w' \mid M|_{w'} = M|_w\}$.*

Lemma 13. *Given a model $M =< S, R, V, Exp >$, for any $\bar{w} \in \Sigma^*$, $\Delta^{\bar{w}}_M$ is regular. Moreover, the size of the DFA of $\Delta^{\bar{w}}_M$, $|DFA(\Delta^{\bar{w}}_M)| \leq |Q_M|^{|S|}$.*

Proof.

$$\begin{aligned} w \in \Delta^{\bar{w}}_M &\text{ iff } M|_{\bar{w}} = M|_w \\ &\text{iff } \forall s \in S, w \in [\bar{w}]_s \\ &\text{iff } w \in \cap_{s\in S}[\bar{w}]_s \end{aligned}$$

Given a state $s \in S$, by definition of the relation Z^M_s as in the proof of Lemma 12, $w \in [\bar{w}]_s$ if and only if $\widehat{\delta_s}(q^0_s, w) = \widehat{\delta_s}(q^0_s, \bar{w})$, where δ_s is the transition function of the $DFA(Exp(s)) = (Q_s, \Sigma, \delta_s, q^0_s, F_s)$. Therefore the $DFA([\bar{w}]_s) = (Q_s, \Sigma, \delta_s, q^0_s, F_{[\bar{w}]_s} = \{\widehat{\delta_s}(q^0_s, \bar{w})\})$. Informally, the language of $[\bar{w}]_s$ takes the DFA of $Exp(s)$, unmarks all its final/accepting states and marks the state where $\bar{w}$ reaches when simulated on it from the start state, as final. Hence $[\bar{w}]_s$ for any state $s \in S$ is regular.
Therefore, $\Delta^{\bar{w}}_M = \cap_{s\in S}[\bar{w}]_s$ is regular. Also $DFA([\bar{w}]_s)$ for any state $s \in S$ has $|Q_s|$ states, which is at most $|Q_M|$. Therefore $|DFA(\Delta^{\bar{w}}_M)| = |DFA(\cap_{s\in S}[\bar{w}]_s)| \leq |Q_M|^{|S|}$. □

Hence we have the following theorem:

Theorem 14. *For any POL formula φ, a finite pointed* POL *model M, s, there exists a $w \in \Sigma^*$, such that if $M|_w, s \vDash \varphi$ then $|w| \leq |Q_M|^{|S|}$.*

Proof. Create a graph structure $G^M(M_{\Sigma^\star}, E^M)$. E^M is a set of labelled directed edges (labelled with letters $a \in \Sigma$) between nodes from $M_{\Sigma^\star}$.

For every $a \in \Sigma$, $(M|_u, M|_{u'}, a) \in E^M$ iff $ua \in \Delta^{u'}_M$. Now consider the longest path in G^M from M (that is, $M|_\epsilon$). Each edge in that path contributes an alphabet to the string and

the biggest path that can occur in this graph is $|M_{\Sigma^\star}|$. Hence there exists a w such that $|w| = |M_{\Sigma^\star}| \leq |Q_M|^{|S|}$ □

Let $\Gamma_{[M]} = \{\Delta_M^{\bar{w}} \mid \bar{w} \in \Sigma^*\}$.

Lemma 15. *For any finite POL formula φ, finite model M, starting state s, there exists a subset $S \subseteq \Gamma_{[M]}$ such that $\Lambda^{M,s,\varphi} = \cup_{\Delta_M^{\bar{w}} \in S}(\Delta_M^{\bar{w}})$.*

Proof. By Lemma 5, $\Lambda^{M,s,\varphi} = \{w \mid M|_w, s \vDash \varphi\}$. Now, define a relation $Z^M \subseteq \Lambda^{M,s,\varphi} \times \Lambda^{M,s,\varphi}$, such that for any $w, w' \in \Lambda^{M,s,\varphi}$, $(w, w') \in Z^M$ if $M|_w = M|_{w'}$. Again, Z^M is an equivalence relation. Therefore, it forms a partition over $\Lambda^{M,s,\varphi}$. Let $[w]$ be one such partition with the string w in it.

Note that, $[w] = \{w' \mid M|_w = M|_{w'}\} = \Delta_M^w$. Hence there exists a subset $S \subseteq \Gamma_{[M]}$ such that $\Lambda^{M,s,\varphi} = \cup_{\Delta_M^{\bar{w}} \in S}(\Delta_M^{\bar{w}})$ □

Corollary 16. *Given a model $M =< S, R, V, Exp >$, a formula $\langle\pi\rangle\varphi$ for any POL formula φ and regular expression π over Σ, and a start state s, there exists a string in $\mathcal{L}(\pi) \cap \mathcal{L}(Pre(Exp(s))) \cap \Lambda^{M,s,\varphi}$ of length at most $2^{|\pi|+|Exp(s)|} \times |Q_M|^{O(|S|^2)}$.*

Proof. From Lemma 15, we know $\Lambda^{M,s,\varphi} = \cup_{\Delta_M^{\bar{w}} \in S}(\Delta_M^{\bar{w}})$ for some $S \subseteq \Gamma_{|M|}$. Also from Lemma 13, we know $|DFA(\Delta_M^{\bar{w}})| \leq |Q_M|^{|S|}$. Also $|M_{\Sigma^\star}| = |\Gamma_{[M]}| \leq |Q_M|^{|S|}$. Hence, $|DFA(\Lambda^{M,s,\varphi})| \leq |Q_M|^{O(|S|^2)}$. Therefore, the DFA of $\mathcal{L}(\pi) \cap \mathcal{L}(Pre(Exp(s))) \cap \Lambda^{M,s,\varphi}$ will have at most $2^{|\pi|+|Exp(s)|} \times |Q_M|^{O(|S|^2)}$ states. Hence there exists a string in $\mathcal{L}(\pi) \cap \mathcal{L}(Pre(Exp(s))) \cap \Lambda^{M,s,\varphi}$ of length at most $2^{|\pi|+|Exp(s)|} \times |Q_M|^{O(|S|^2)}$. □

In this section, we present the model checking decidability algorithm for POL. Given a finite POL formula φ, a finite POL model $M =< S, R, V, Exp >$, and a start state $s \in S$, decide $M, s \vDash \varphi$.

Notational Preliminaries

The algorithm CreateDelta(M) returns every possible tuple $(\Delta_M, \bar{w})$, where Δ_M is a DFA such that $\mathcal{L}(\Delta_M) = \{w \in \Sigma^* \mid M|_w = M|_{\bar{w}}\}$ and $\bar{w}$ is the smallest word in $\Delta_M^{\bar{w}}$. This tuple represents the same language $\Delta_M^{\bar{w}}$ defined in definition 43, and the DFA was constructed in the proof of Lemma 13.

The algorithm StringRepresent$(M, \pi, s, \Gamma_{[M]})$ returns every possible DFA of $\mathcal{L}(\pi) \cap \mathcal{L}(Pre(Exp(s))) \cap \Delta_M$ over all tuple $(\Delta_M, \bar{w}) \in \Gamma_{[M]}$. Each DFA is named $\beta^M_{\bar{w},\pi,s}$ over every possible $\bar{w}$, which, by Lemma 12, is $|Q_M|^{|S|}$ many. $w_{min}(\beta^M_{\bar{w},\pi,s})$ is the minimum length string accepted by $\beta^M_{\bar{w},\pi,s}$. Hence at most $|Q_M|^{|S|}$ such minimum strings (of length at most $2^{|\pi|+|Exp(s)|} \times |Q_M|^{|S|}$, by Lemma 13) are stored in the set $B^M_{\pi,s}$ and is returned.

Algorithm 2 CreateDelta($M =< S, R, V, Exp >$)

procedure CreateDelta($M =< S, R, V, Exp >$)
  $DFAMatrix[s_i] \leftarrow \emptyset$ for all $s_i \in S$
  for each $s_i \in S$ **do**
    $D_s = (Q_s, \Sigma, \delta_s, q_s^0, F_s) \leftarrow$ CreateDFAFromREG($Exp(s)$)
    $DFArray[q] \leftarrow \emptyset$ for all $q \in Q_s$
    for each $q \in Q_s$ **do**
      $D_s^q = (Q_s, \Sigma, \delta_s, q, F_s)$
      $DFArray[q] \leftarrow D_s^q$
    $DFAMatrix[s] \leftarrow DFArray$
  $\Gamma_{[M]} \leftarrow \emptyset$
  for each n-tuple $(q_{s_1}, q_{s_2} ... q_{s_n})$ for all $s_i \in S$ where $q_{s_i} \in Q_{s_i}$ a state in DFA of $Exp(s_i)$ **do**
    $\Delta_M \leftarrow \cap_{s_i \in S}(DFAMatrix[s_i][q_{s_i}])$
    $\bar{w} \leftarrow$ the string from shortest path from start state of Δ_M to a final state
    $\Gamma_{[M]} \leftarrow \Gamma_{[M]} \cup \{(\Delta_M, \bar{w})\}$
  Return $\Gamma_{[M]}$

The algorithm StoreStrings(M, φ) returns every possible $w_{min}(\beta^M_{\bar{w},\pi,s})$, over all $\bar{w}$, every π in the set of regular language modalities of φ (in the algorithm, the set is called φ_R), over every state $s \in S$. The 2-D matrix W_{min} is indexed by every $\pi \in \varphi_R$ and $s \in S$. $W_{min}[\pi][s]$ contains every possible DFA of $\mathcal{L}(\pi) \cap \mathcal{L}(Pre(Exp(s))) \cap \Delta_M$ over all tuple $(\Delta_M, \bar{w}) \in \Gamma_{[M]}$ in the state s, for the regular language π.

Finally, the Decide2(M, s, φ, W_{min}) decides whether $M, s \vDash \varphi$, as the algorithm previously stated. Only difference is in the case where $\varphi = \langle\pi\rangle\psi$. Here, instead of exhaustively searching for all strings in $\mathcal{L}(\pi) \cap \mathcal{L}(Pre(Exp(s)))$ upto a certain size (double exponential), a more memoization approach is taken. By StoreStrings, we store every possible small-sized $w_{min}(\beta^M_{\bar{w},\pi,s})$ and refer them accordingly. The argument W_{min} is passed as $\emptyset$. The StoreStrings subroutine is called once in a recursion subtree of Decide2, for the rest, we just refer to W_{min}, which is indexed by π and s. By Lemma 15, we know that there exists a $w \in \{w' | M|_w \vDash \psi\}$, such that it is in at least one of $\Delta^{\bar{w}}_M$. For any formula ψ, we search every possible (at most) single exponential many strings $w_{min}(\beta^M_{\bar{w},\pi,s})$ over all possible $\bar{w}$, each of which are of at most single exponential size.

Time Complexity

Lemma 17. *The algorithm* CreateDelta($M =< S, R, V, Exp >$) *takes at most* $O(2^{P_M} \times |S| + |Q_M|^{|S|})$ *steps to complete.*

Algorithm 3 StringRepresent$(M, \pi, s, \Gamma_{[M]})$

1: **procedure** StringRepresent$((M, \pi, s, \Gamma_{[M]}))$
2: $D_\pi \leftarrow CreateDFAFromREG(\pi)$
3: $D_{Exp(s)} = (Q_s, \Sigma, \delta_s, q_s^0, F_s) \leftarrow CreateDFAFromREG(Exp(s))$
4: $F_{Pre(Exp(s))} \leftarrow \emptyset$
5: **for** each $q \in Q_s$ such that q can reach a final state **do**
6: $F_{Pre(Exp(s))} \leftarrow F_{Pre(Exp(s))} \cup \{q\}$
7: $D_{Pre(Exp(s))} \leftarrow (Q_s, \Sigma, \delta_s, q_s^0, F_{Pre(Exp(s))})$
8: $D_{\pi,s} \leftarrow D_\pi \cap D_{Pre(Exp(s))}$
9: $B^M_{\pi,s} \leftarrow \emptyset$
10: **for** $(\Delta_M, \bar{w}) \in \Gamma_{[M]}$ **do**
11: $\beta^M_{\bar{w},\pi,s} \leftarrow D_{\pi,s} \cap \Delta_M$
12: $w_{min}(\beta^M_{\bar{w},\pi,s}) \leftarrow$ string representing the shortest path from start state to a final state in the DFA of $\beta^M_{\bar{w},\pi,s}$
13: $B^M_{\pi,s} \leftarrow B^M_{\pi,s} \cup \{w_{min}(\beta^M_{\bar{w},\pi,s})\}$
14: Return $B^M_{\pi,s}$

Algorithm 4 StoreStrings(M, φ)

1: **procedure** StoreStrings$((M, \varphi))$
2: $\Gamma_{[M]} \leftarrow$ CreateDelta(M)
3: $\varphi_R \leftarrow \{\pi \mid \langle\pi\rangle$is a modality in $\varphi\}$
4: $W_{min}[\pi_i][s_j] \leftarrow \emptyset$ for all $\pi_i \in \varphi_R$ and $s_i \in S$
5: **for** each $\pi \in \varphi_R$ **do**
6: **for** each $s \in S$ **do**
7: $W_{min}[\pi][s] \leftarrow$ StringRepresent$(M, \pi, s, \Gamma_{[M]})$
8: Return W_{min}

Proof. To create DFA from regular expression takes $O(2^{|Exp(s)|})$ steps, where $Exp(s)$ is the observation function of a state $s \in S$. Also, the said DFA has $O(2^{|Exp(s)|})$ number of states in it. Hence with the loop at line number 3 (goes over every state in model) and the loop inside at line number 6, line 3-9 takes at most $O(2^{P_M} \times |S|)$.

The number of such n-tuples possible in line 10 is at most $|Q_M|^{|S|}$, as each such tuple corresponds to one T^w_M in proof of Lemma 12.

Hence the algorithm takes at most $O(2^{P_M} \times |S| + |Q_M|^{|S|})$ steps. □

Lemma 18. *The algorithm* StringRepresent$(M, \pi, s, \Gamma_{[M]})$ *takes at most* $O(2^{|\pi|+P_M} \times |Q_M|^{|S|})$ *steps to complete.*

Algorithm 5 Decide2$(M, s, \varphi, W_{min} = \emptyset)$

1: **procedure** Decide2$((M, s, \varphi, W_{min} = \emptyset))$
2: **if** $W_{min} = \emptyset$ **then**
3: $W_{min} \leftarrow$ StoreStrings(M, φ)
4: **if** φ is a propositional formula **then**
5: Return True if $M, s \vDash \varphi$; False otherwise
6: **if** $\varphi = \psi' \vee \psi$ **then**
7: Return Decide2(M, s, ψ, W_{min}) or Decide2(M, s, ψ', W_{min})
8: **if** $\varphi = \langle\pi\rangle\psi$ **then**
9: **for** each $w_{min}(\beta^M_{\bar{w},\pi,s}) \in W_{min}[\pi][s]$ **do**
10: **if** Decide2$(M|_{w_{min}(\beta^M_{\bar{w},\pi,s})}, s, \psi, W_{min}) =$ True **then**
11: return True
12: return False
13: **if** $\varphi = \hat{K}_i\psi$ **then**
14: **if** if there exists a $t \in S$ such that $t \sim_i s$ and Decide2(M, t, ψ, W_{min}) **then**
15: Return True
16: **else**
17: Return False
18: **if** $\varphi = \neg\psi$ **then**
19: Return not Decide2(M, s, ψ, W_{min})

Proof. From lemma 12, $|\Gamma_{[M]}| \leq |Q_M|^{|S|}$ ($|\Gamma_{[M]}| = |M_{\Sigma^\star}|$). Hence loop in line 10 runs at most $|Q_M|^{|S|}$ times. To create $\beta^M_{\bar{w},\pi,s}$ takes at most $2^{|\pi|+P_M} \times |\Delta_M|$. Hence StringRepresent takes at most $O(2^{|\pi|+P_M} \times |Q_M|^{|S|})$, as by Lemma 13, $|\Delta_M| \leq |Q_M|^{|S|}$. □

Let π_M be the largest sized regular expression among all the regular modalities in a given POL formula φ.

Corollary 19. *The algorithm* StoreStrings(M, φ) *takes at most* $O(|\varphi| \times |S| \times 2^{|\pi_M|+P_M} \times |Q_M|^{|S|})$ *steps to complete.*

Proof. Line 1 takes $O(2^{P_M} \times |S| + |Q_M|^{|S|})$ steps to complete (Lemma 17). The loop from line 4-8 takes at most $|\varphi_R| \times |S| \times O(2^{|\pi_M|+P_M} \times |Q_M|^{|S|})$ (φ_R is the set of all regular expressions occuring in φ) steps to complete (using Lemma 18). Hence the algorithm takes at most $O(|\varphi| \times |S| \times 2^{|\pi_M|+P_M} \times |Q_M|^{|S|})$ steps to complete. □

Theorem 20. *The algorithm* Decide2$(M, s, \varphi, W_{min} = \emptyset)$ *takes at most* $O(|Q_M|^{|S|\times(n+1)} + (n \times |S| \times 2^{|\pi_M|+P_M} \times |Q_M|^{|S|}))$ *steps to complete.*

Proof. Note that subroutine StoreStrings(M, φ) only get called once, to populate the matrix W_{min}. Hence the case where $\varphi = \langle\pi\rangle\psi$, only takes time to refer to W_{min} and the loop in line 11-15 runs for at most $|W_{min}[\pi][s]|$ steps at worst case, which is the number of $(\Delta_M, \bar{w})$ in $\Gamma_{[M]}$,

which is $|Q_M|^{|S|}$. Assuming random access, the recurrence relation for time complexity stands:

$$T(n) \leq |Q_M|^{|S|} \times T(n-1),$$

where $T(n)$ is time taken by Decide2(M, s, φ) excluding the call to subroutine StoreStrings(M, φ) and n is the depth of the POL syntactic tree of the POL formula φ.
We shall now prove $T(n) \leq O(|Q_M|^{|S|\times n})$.
Base Case. For $n = 0$, the formula is a propositional formula, and hence $T(1) \leq O(|\varphi|) \leq O(|Q_M|^{|S|})$.
Induction Hypothesis. $T(i) \leq O(|Q_M|^{|S|\times i})$ for any integer $i \leq n$.
Inductive Step.

$$\begin{aligned} T(n+1) &\leq |Q_M|^{|S|} \times T(n) \\ &\leq |Q_M|^{|S|} \times O(|Q_M|^{|S|\times n}), \text{ by Induction Hypothesis} \\ &\leq O(|Q_M|^{|S|\times(n+1)}) \end{aligned}$$

Hence along with the call to subroutine StoreStrings(M, φ), Decide2(M, s, φ) takes at most $O(|Q_M|^{|S|\times(n+1)} + (n \times |S| \times 2^{|\pi_M|+P_M} \times |Q_M|^{|S|}))$ steps to complete. □

3.3 Complexity Results

The main complexity result that is proven is given below.

3.3.1 POL Model Checking is in PSPACE.

For proving the upper bound result, that is, showing that POL model checking is in PSPACE, we design the algorithm mcPOL (Algorithm 6). It takes as input a POL model $\mathcal{M} = \langle S, \sim, V, Exp \rangle$, an initial starting world $s \in S$, and a POL formula φ and returns True iff $\mathcal{M}, s \models \varphi$. We also prove mcPOL uses polynomial space. The recursive algorithm mcPOL is divided into various cases depending on the structure of φ. The subtle case is the observation modality $\langle \pi \rangle \psi$ (that is dealt with in lines 8 to 12). It follows from the truth definition that $\mathcal{M}, s \vDash \langle \pi \rangle \psi$ iff there exists a $w \in \mathcal{L}(\pi)$ such that $\mathcal{M}|_w, s \vDash \psi$. We observe that for any $\mathcal{M}$ and w the model $\mathcal{M}|_w$ can be represented by a string of size polynomial in the size of $\mathcal{M}$ (This is because $\mathcal{M}$ and $\mathcal{M}|_w$ just differ by their expected observation functions as follows: for any world t, $Exp'(t) = Exp(t) \backslash w$ and $Exp(t)$ share the same Non-deterministic Finite Automata (NFA), just the set of initial states is different.). Let us define a notation $\mathcal{M}_\pi$ in the following way: Given a model $\mathcal{M}$, and a regular expression π, define $\mathcal{M}_\pi = \{\mathcal{M}|_w \mid w \in \mathcal{L}(\pi)\}$.

Thus if we consider the set $\mathcal{M}_{\Sigma^\star} = \{\mathcal{M}|_w \mid w \in \Sigma^*\}$, the set of every updated model $\mathcal{M}|_w$, for a POL model $\mathcal{M}$, over all $w \in \Sigma^*$, we realize that all the models in $\mathcal{M}_{\Sigma^\star}$ has size polynomial in the size of $\mathcal{M}$. Thus, by using both the observations together, when mcPOL has to check if $\mathcal{M}, s \vDash \langle\pi\rangle\psi$ (in the **for** loop in lines 9 to 11) it goes over all models $\mathcal{M}'$ in $\mathcal{M}_{\Sigma^\star}$ and (in line 9) checks if $\mathcal{M}' = \mathcal{M}|_w$ for some $w \in \mathcal{L}(\pi)$ and finally (in line 11) calls mcPOL recursively to check if $\mathcal{M}|_w, s \vDash \psi$. Thus mcPOL needs to call a polynomial space subroutine to check if $\mathcal{M}' = \mathcal{M}|_w$ for some word $w \in \mathcal{L}(\pi)$. To prove that there exists such a polynomial space algorithm we present a slightly convoluted argument. Algorithm 7 provides a non-deterministic procedure running in polynomial space for deciding that $\mathcal{M}' = \mathcal{M}|_w$ for some word $w \in \mathcal{L}(\pi)$. By Savitch's theorem [52] which states that NPSPACE = PSPACE, we have that a polynomial space algorithm also exists. Algorithm 7 starts by guessing a word of exponential length, sufficiently long enough to explore all subsets of current states for NFAs of $Exp(t)$ for all worlds t in $\mathcal{M}$ and for the NFA of π. Then the algorithm guesses the word w letter by letter and it progresses in the NFAs (note that it does not store the word w as it can be of exponential length). Algorithm 7 accepts when $w \in \mathcal{L}(\pi)$ (i.e., $\epsilon \in \mathcal{L}(\pi')$) and $\mathcal{M} = \mathcal{M}'$. Otherwise, it rejects.

A crucial step in mcPOL is when (in line 10) it uses an oracle to check if $\mathcal{M}' = \mathcal{M}|_w$ for some word $w \in \mathcal{L}(\pi)$. So to prove that the algorithm mcPOL uses polynomial space, we need to prove the existence of a polynomial space subroutine to check if $\mathcal{M}' = \mathcal{M}|_w$ for some word $w \in \mathcal{L}(\pi)$. For this we design the Algorithm 7 (ResidueModelExistence) which is a non-deterministic polynomial space algorithm to check the same. And using Savitch's theorem we can conclude that a deterministic polynomial space algorithm must also exist. We start by proving the correctness and complexity of Algorithm 7 and then using this we present the proof of correctness and complexity of mcPOL.

Correctness and Complexity of Algorithm 7

Before we prove the correctness of mcPOL we need to first prove that there is a PSPACE algorithm (oracle) for checking if $\mathcal{M}' = \mathcal{M}|_w$ for some word $w \in \mathcal{L}(\pi)$. We will first prove that Algorithm 7 is a non-deterministic algorithm for checking if $\mathcal{M}' = \mathcal{M}|_w$ for some word $w \in \mathcal{L}(\pi)$ that takes polynomial space. For that we need to understand the following: for these two models $\mathcal{M}$ and $\mathcal{M}'$, what is the length of the smallest $w \in \mathcal{L}(\pi)$ such that $\mathcal{M}' = \mathcal{M}|_w$? Lemma 21 takes care of this query. We use Observation 11 to show that the length of the smallest such w is bounded by $2^\pi \times \Pi_{t\in\mathcal{M}} 2^{|Exp(t)|}$.

Now using the Lemma 12 we prove Lemma 21 that would be used to prove the correctness of the Algorithm 7.

Algorithm 6 mcPOL

procedure mcPOL($\mathcal{M} = \langle S, \sim, V, Exp\rangle, s \in S, \varphi$)
  if $\varphi = p$ is a propositional variable **then**
    return True if $p \in V(s)$; False otherwise
  if $\varphi = \neg\psi$ **then**
    return not mcPOL($\mathcal{M}, s, \psi$)
  if $\varphi = \psi' \vee \psi$ **then**
    return mcPOL($\mathcal{M}, s, \psi$) or mcPOL($\mathcal{M}, s, \psi'$)
  if $\varphi = \langle\pi\rangle\psi$ **then**
    for all models $\mathcal{M}'$ in $\mathcal{M}_{\Sigma^\star}$ **do**
      if $s \in \mathcal{M}'$ and the oracle ResidueModelExistence($\mathcal{M}, \mathcal{M}', \pi$) accepts **then**
        return mcPOL($\mathcal{M}', s, \psi$)
    return False
  if $\varphi = \hat{K}_i\psi$ **then**
    if $\exists t \in S$ such that $t \sim_i s$ and mcPOL($\mathcal{M}, t, \psi$) **then**
      return True
    else
      return False

Lemma 21. *Given a* POL *model* $\mathcal{M} = \langle S, \sim, V, Exp\rangle$*, a world* $s \in S$ *and a formula* $\langle\pi\rangle\psi$*,* $\mathcal{M}, s \vDash \langle\pi\rangle\psi$ *iff* $\exists w \in \mathcal{L}(\pi)$ *of length at most* $2^{|\pi|} \times \Pi_{t\in\mathcal{M}} 2^{|Exp(t)|}$ *such that* $\mathcal{M}|_w, s \vDash \psi$ *and the world* s *survives in* $\mathcal{M}_w$*.*

Proof. The $\Leftarrow$ direction is easy: if there exists a $w \in \mathcal{L}(\pi)$ such that $\mathcal{M}|_w, s \vDash \psi$ then by definition $\mathcal{M}, s \vDash \langle\pi\rangle\psi$.

Now for the $\Rightarrow$ direction, consider the edge graph $G^{\mathcal{M}}(\mathcal{M}_{\Sigma^\star}, E^{\mathcal{M}})$ on vertex set $\mathcal{M}_{\Sigma^\star}$ and and edge from vertex $\mathcal{M}|_u$ to vertex $\mathcal{M}|_{u'}$ is present if and only if there exists a $a \in \Sigma$ such that $\mathcal{M}|_{ua} = \mathcal{M}|_{u'}$. From Lemma 12 we know that the number of vertices in the graph is at most $\Pi_{t\in\mathcal{M}} 2^{|Exp(t)|}$. Thus it is easy to observe that for any two vertices $\mathcal{M}|_u, \mathcal{M}|_{u'} \in \mathcal{M}_{\Sigma^\star}$ the set $\Delta_{\mathcal{M}|_u, \mathcal{M}|_{u'}} := \{w \in \Sigma^* \mid \mathcal{M}|_{uw} = \mathcal{M}|_{u'}\}$ is a regular language accepted by a DFA of size at most $\Pi_{t\in\mathcal{M}} 2^{|Exp(t)|}$.

Let $\mathcal{M}, s \vDash \langle\pi\rangle\psi$. Then by definition there exists a $w_0 \in \mathcal{L}(\pi)$ such that $\mathcal{M}|_{w_0}, s \vDash \psi$. Note that $\mathcal{M}$ and $\mathcal{M}|_{w_0}$ are both vertices of the graph $G^{\mathcal{M}}$. So all the $\{w \in \mathcal{L}(\pi) \mid \mathcal{M}|_w = \mathcal{M}|_{w_0}\}$ is nothing but the the set $\mathcal{L}(\pi) \cap \Delta_{\mathcal{M}, \mathcal{M}|_{w_0}}$. So we know there is a $w \in \mathcal{L}(\pi) \cap \Delta_{\mathcal{M}, \mathcal{M}|_{w_0}}$ of size at most $2^{\pi} \times \Pi_{t\in\mathcal{M}} 2^{|Exp(t)|}$ and for that w, $\mathcal{M}|_w, s \vDash \psi$. □

From Lemma 21 we know that if $\mathcal{M}' = \mathcal{M}|_w$ for some $w \in \mathcal{L}(\pi)$ there exists a $w_0 \in \mathcal{L}(\pi)$ with $|w_0| \leq 2^{|\pi|} \times \Pi_{t\in S} 2^{|Exp(t)|}$ and $\mathcal{M}' = \mathcal{M}|_{w_0}$. Since Algorithm 7 guesses a $w = \alpha_1 \ldots, \alpha_j \ldots$ (one letter at a time) of length at most $2^{|\pi|} \times \Pi_{t\in S} 2^{|Exp(t)|}$ and $\mathcal{M}' = \mathcal{M}|_{w_0}$ and checks if

Algorithm 7 ResidueModelExistence: Non-deterministic procedure to decides that $\mathcal{M}' = \mathcal{M}|_w$ for some word $w \in \mathcal{L}(\pi)$

```
1:  procedure ResidueModelExistence(M = ⟨S, ∼, V, Exp⟩, M' ∈ M_{Σ⋆}, π)
2:      π' := π
3:      for i = 1 to 2^π × Π_{t∈S} 2^{|Exp(t)|} do
4:          if ϵ ∈ L(π') and M = M' then
5:              accept
6:          guess a letter a from Σ
7:          π' := π'\a
8:          for each world t in S do
9:              Exp(t) := Exp(t)\a                    // we modify M locally
10:     reject
```

$\mathcal{M}' = \mathcal{M}|_w$, from Lemma 21 we see that the algorithm is correct. Note that Algorithm 7 is a non-deterministic algorithm. The algorithm uses only polynomial space (in the size of $\mathcal{M}$), since at any point of time (say at the jth iteration of the **for** loop in Line 3) the algorithm only have to update the model from $\mathcal{M}|_{\alpha_1 \ldots \alpha_{j-1}}$ to $\mathcal{M}|_{\alpha_1 \ldots \alpha_j}$ which can be done using polynomial space. Note that the Algorithm 7 does not have to remember the string $\alpha_1 \ldots \alpha_j \ldots$ which can be of size exponential. So the Algorithm 7 is a non-deterministic polynomial space algorithm. Thus we have

Theorem 22. *Algorithm 7 (*ResidueModelExistence*) is a non-deterministic polynomial space algorithm that correctly checks if $\mathcal{M}' = \mathcal{M}|_w$ for some $w \in \mathcal{L}(\pi)$.*

By Savitch's Theorem [52],there is a deterministic polynomial space oracle for checking if $\mathcal{M}' = \mathcal{M}|_w$ for some $w \in \mathcal{L}(\pi)$. Thus we have

Theorem 23. *There is a deterministic polynomial space algorithm that correctly checks if $\mathcal{M}' = \mathcal{M}|_w$ for some $w \in \mathcal{L}(\pi)$.*

Correctness and Complexity of mcPOL

Using the Theorem 23 we now present the proof of correctness and complexity of mcPOL.

Lemma 24. $\mathsf{mcPOL}(\mathcal{M}, s, \varphi)$ *returns* True *iff* $\mathcal{M}, s \vDash \varphi$.

Proof. We will prove $\mathsf{mcPOL}(\mathcal{M}, s, \varphi)$ returns True iff $\mathcal{M}, s \vDash \varphi$ by induction on the size of φ.

Base Case. Consider φ to be a proposition. $\mathcal{M}, s \vDash \varphi$ iff $\varphi \in V(s)$ iff $\mathsf{mcPOL}(\mathcal{M}, s, \varphi)$ returns True.

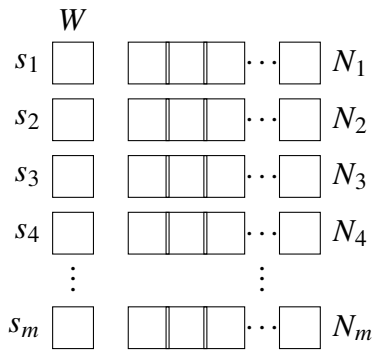

Fig. 3.1 Storing $\mathcal{M}_{\Sigma^\star}$

Induction Hypothesis. For any POL formula $|\psi| \leq m$, any finite model $\mathcal{M}$ and any world s, mcPOL$(\mathcal{M}, s, \psi)$ returns True iff $\mathcal{M}, s \vDash \psi$.

Inductive Step. We go case by case over the forms of φ. For all the cases except when $\varphi = \langle\pi\rangle\psi$, the inductive step is trivial. So we focus on the crucial case when $\varphi = \langle\pi\rangle\psi$.

By definition we know that $\mathcal{M}, s \vDash \langle\pi\rangle\psi$ iff there exists a $w \in \mathcal{L}(\pi)$ such that $\mathcal{M}|_w, s \vDash \psi$. In other words, $\mathcal{M}, s \vDash \langle\pi\rangle\psi$ iff there exists $\mathcal{M}' \in \mathcal{M}_{\Sigma^\star}$ such that $\mathcal{M}' = \mathcal{M}|_w$ for some $w \in \mathcal{L}(\pi)$ and the world s survives and $\mathcal{M}', s \vDash \psi$. By induction hypothesis $\mathcal{M}', s \vDash \psi$ iff mcPOL$(\mathcal{M}', s, \psi)$ is True. In the **for** loop Lines 9 to 11 the algorithm goes over all $\mathcal{M}' \in \mathcal{M}_{\Sigma^\star}$. For each of the $\mathcal{M}'$ the algorithm in Line 10 calls the oracle (Algorithm 7) to check if $\mathcal{M}' = \mathcal{M}|_w$ for some $w \in \mathcal{L}(\pi)$ and if the world survives calls mcPOL$(\mathcal{M}', s, \psi)$ recursively. Since we have already argued correctness of Algorithm 7 by Theorem 22 , the correctness of the algorithm follows. □

Next before proving the complexity of mcPOL, we prove a crucial result regarding Line 9. Note that to enumerate every model in the set $\mathcal{M}_{\Sigma^\star}$, exponential memory is required to store each $\mathcal{M}|_w$ with respect to size of $\mathcal{M}$ and π. This next results prove that we need only polynomial many bits to represent all the models given $\mathcal{M}$.

Lemma 25. *Given a* POL *model* $\mathcal{M}$*, the set* $\mathcal{M}_{\Sigma^\star}$ *needs polynomial many bits to be represented with respect to* $|\mathcal{M}|$.

Proof. Consider the model $\mathcal{M}$ has m many worlds $s_1, s_2, \ldots, s_k$. Since size of an NFA is bounded polynomially by the size of a regular expression, let n be the largest number of states of NFA of the expectations among all the worlds. Formally, let π_i be the NFA for world s_i. Hence $n = max_{i\in[m]}(|\pi_i|)$.

Consider a memory space W of m bits, each bit space $W[i]$ for each world s_i. In addition ot this, consider m many n bits, each n sized bitspace N_i corresponding to the NFA π_i.

The residuation of an NFA is just changing the start state of the NFA. For a model $\mathcal{M}|_w \in \mathcal{M}_{\Sigma^\star}$ we set the bits like this (Figure 3.1):

- $W[i] = 1$ iff $s_i \in \mathcal{M}|_w$
- For every $i \in [m]$, $N_i[j] = 1$ iff the state q_j of NFA of π_i is the start state in $Exp(s_i)$ of $\mathcal{M}|_w$.

Hence, given $\mathcal{M}$, every models $\mathcal{M}|_w \in \mathcal{M}_{\Sigma^\star}$ can be evaluated from at most mn many bits □

Note that, the mn bits in the previous lemma can also express models which are not in $\mathcal{M}_{\Sigma^\star}$. But the oracle ResidueModelExistence will never produce it, hence for such models it will reject.

Hence in the algorithm 6, the line 9 does not directly enumerate every model by storing all models in $\mathcal{M}_{\Sigma^\star}$ simultaneously. For each combination of mn bits, it evaluates an $\mathcal{M}'$ from the bits and $\mathcal{M}$, and then evaluates the rest of the loop.

Now we move on to prove that mcPOL uses polynomial amount of space. This (along with Lemma 24 and Theorem 23) would prove that mcPOL is in PSPACE.

Lemma 26. mcPOL *uses polynomial space.*

Proof. Since the algorithm is recursive, the formal argument (as in the proof of Lemma 24) should go via induction. It can be observed that in all the cases except when $\varphi = \langle\pi\rangle\psi$, mcPOL only uses a constant amount of space before making the recursive call. In the case when $\varphi = \langle\pi\rangle\psi$, since the algorithm goes over all $\mathcal{M}' \in \mathcal{M}_{\Sigma^\star}$ (**for** loop from Line 9 to Line 11), the algorithm will have to do some bookkeeping to keep a track on when $\mathcal{M}$ is being processed and to store the current $\mathcal{M}$. But since $\mathcal{M}_{\Sigma^\star}$ has size exponential (Lemma 12) and since all $\mathcal{M}' \in \mathcal{M}_{\Sigma^\star}$ can be represented in size polynomial in the size of $\mathcal{M}$ so it is possible to do the bookkeeping and tracking using only polynomial space (Lemma 25). For any $\mathcal{M}'$ inside the **for** loop the only non-trivial thing to do is the call to the oracle in Line 10. By Theorem 23, there exists an algorithm in PSPACE that given two models $\mathcal{M}$ and $\mathcal{M}'$ and regular expression π checks if there exists $w \in \mathcal{L}(\pi)$ such that $\mathcal{M}' = \mathcal{M}|_w$. This space can of course be reused for any iteration of the **for** loop in Line 9. So mcPOL uses at most polynomial space before making a recursive call and hence the total space used by the algorithm is polynomial. □

By combining Lemma 24 and 26 we have the following:

Theorem 27. POL *model-checking is in* PSPACE.

3.3.2 Model checking for POL is PSPACE-hard:

Interestingly, there are two reasons for the model checking to be PSPACE-hard: Kleene star in observation modalities as well as alternations in modalities (sequences of nested existential and universal modalities). We prove the PSPACE-hardness of model checking against the Existential fragment and the Star-Free fragment of POL respectively.

Theorem 28. *The model checking for the* Existential *fragment of* POL *is* PSPACE*-hard.*

Proof. Kozen [39] proved that the following problem, called the intersection non-emptiness problem, is PSPACE-complete: given a finite collection of DFAs $\mathcal{A}_1, \ldots, \mathcal{A}_n$, decide whether $\mathcal{L}(\mathcal{A}_1) \cap \cdots \cap \mathcal{L}(\mathcal{A}_n) \neq \emptyset$. Let us reduce this problem to the model checking for POL. For that we construct the instance $\mathsf{tr}(\mathcal{A}_1, \ldots, \mathcal{A}_n) = (\mathcal{M}, s_0, \varphi)$ as:

- $\mathcal{M} = (S, \sim, V, Exp)$ where
  - $S = \{0, 1, \ldots, n\} \cup \{0', 1', \ldots, (n-1)'\}$.
  - $\sim_1 = \{(i,i'), (i,i), (i',i'), (i',i) \mid i = 0, \ldots, n-1\}$, $\sim_2 = \{(i',i+1), (i',i'), (i+1,i+1), (i+1,i') \mid i = 0, \ldots, n-1\}$,
  - $V(s) = V(s') = \emptyset$ for all $s \leq n-1$, $V(n) = \{p\}$.
  - $Exp(0) = Exp(0') = \Sigma^*$, $Exp(i) = Exp(i') = \mathcal{A}_i$ for all $i = 1 \ldots n-1$ and $Exp(n) = \mathcal{A}_n$.
- $s_0 = 0$.
- $\varphi = \langle \Sigma^* \rangle (\hat{K}_1 \hat{K}_2)^{n+1} p$.

The model $\mathcal{M}$ is a chain of worlds: starting with two worlds $0, 0'$, labeled by an automaton for the universal language Σ^*. followed by worlds $1, 1'$ labelled by automaton $\mathcal{A}_1$, followed by worlds $2, 2'$ labelled by automaton $\mathcal{A}_2$, etc. It ends with worlds $n-1$, $(n-1)'$ labelled by automaton $\mathcal{A}_{n-1}$, followed by a world n labelled by $\mathcal{A}_n$. Proposition p is false in all worlds except n. The formula φ says that there exists a word w such that $(\hat{K}_1 \hat{K}_2)^{n+1} p$ holds in $\mathcal{M}|_w, 0$. As n should still be reachable, it means that the word must be in $\mathcal{L}(\mathcal{A}_i)$ for all $i \in [n]$. Now, $\mathsf{tr}(\mathcal{A}_1, \ldots, \mathcal{A}_n)$ is computable in polynomial time in the size of $(\mathcal{A}_1, \ldots, \mathcal{A}_n)$. Furthermore we have $\mathcal{L}(\mathcal{A}_1) \cap \cdots \cap \mathcal{L}(\mathcal{A}_n) \neq \emptyset$ iff $\mathcal{M}, s_0 \models \varphi$. □

Theorem 29. *The model checking for* POL *is* PSPACE*-hard, when the* POL *formulas are* Star-Free.

Proof. We shall prove by a reduction from TQBF. Given a Quantified Boolean formula $\varphi = Q_1 x_1 \dots Q_n x_n \gamma$, where γ is in CNF containing n variables $\{x_1, \dots, x_n\}$ and m clauses $C = \{c_1, \dots, c_m\}$ and $Q_i \in \{\exists, \forall\}$ for all $i \in [n]$, we shall construct our reduction. Consider the translations $\mathsf{tr}(x_i) = a_i$, $\mathsf{tr}(\neg x_i) = a_i'$, $\mathsf{tr}(\exists x_i) = \langle a_i + a_i' \rangle$ and $\mathsf{tr}(\forall x_i) = [a_i + a_i']$ for all $i \in [n]$. Now we present the model checking instance that we construct from φ.

- Alphabet: $\Sigma_\varphi = \cup_{i \in [n]} \{\mathsf{tr}(x_i), \mathsf{tr}(\neg x_i)\}$.
- Model: $\mathcal{M}_\varphi = \langle S_\varphi, \sim_\varphi, V_\varphi, Exp_\varphi \rangle$, where
  - $S_\varphi = \{1, \dots, m\}$.
  - A single agent 1 and $\forall i, j \in S_\varphi$, $i \sim_1 j$.
  - $V_\varphi(j) = \{p_j\}$ for all $j \in S_\varphi$.
  - For each clause $c_j \in C$, $Exp_\varphi(j) = (\sum_{\ell \in c_j} (\Sigma_\varphi \setminus \{\mathsf{tr}(\ell), \mathsf{tr}(\neg \ell)\})^* \mathsf{tr}(\ell) (\Sigma_\varphi \setminus \{\mathsf{tr}(\ell), \mathsf{tr}(\neg \ell)\})^*)$, where the sum is over the literals in the clause c_j.
- Formula: $\psi := \mathsf{tr}(Q_1 x_1) \dots \mathsf{tr}(Q_n x_n) \bigwedge_{i \in [m]} (\hat{K}_1 p_i)$.
- Starting world: s is any world from S.

We need to prove that the QBF φ is True iff $\mathcal{M}_\varphi, s \vDash \psi$. Let us start by proving the forward direction: that is if φ is True then we prove that $\mathcal{M}_\varphi, s \vDash \psi$.

Consider the set $\mathcal{T}$ all assignments of $(\ell_1, \dots, \ell_n)$ (with $\ell_i \in \{x_i, \neg x_i\}$ that make the CNF formula γ evaluates to True. Since we assumed that the QBF φ is True we observe that there exist a subset $\mathcal{T}' \subseteq \mathcal{T}$ that has the "structure" of $Q_1 x_1 \dots Q_n x_n$. By the construction of tr and the formula ψ we see that all we need to show is that for all assignment $(\ell_1, \dots, \ell_n) \in \mathcal{T}'$ $\mathcal{M}_\varphi|_{\mathsf{tr}(\ell_1) \dots \mathsf{tr}(\ell_n)}, s \vDash \psi$.

Let $w = \mathsf{tr}(\ell_1) \dots \mathsf{tr}(\ell_n)$ where $(\ell_1, \dots, \ell_n) \in \mathcal{T}'$. Consider any world j (corresponding to the clause $c_j = (\ell_p \vee \ell_q \vee \ell_r)$). What happens to the world j in $\mathcal{M}_\varphi|_w$? If we think of $\mathcal{M}_\varphi|_w$ as a series of n updates (namely, $\mathcal{M}_\varphi|_{\mathsf{tr}(y_1)}$, $\mathcal{M}_\varphi|_{\mathsf{tr}(y_1)\mathsf{tr}(y_2)} \dots$ $\mathcal{M}_\varphi|_{\mathsf{tr}(y_1) \dots \mathsf{tr}(y_n)}$) then note that if the variable x_i is not in the clause c_j then the update by $\mathsf{tr}(y_i)$ does not affect the world j. At the same time, since $y_1 \dots y_n$ is a satisfying assignment to the φ so at least one of ℓ_p, ℓ_q, ℓ_r is in the set $\{y_1, \dots, y_n\}$ and hence after the updating by $\mathsf{tr}(y_p), \mathsf{tr}(y_q)$ and $\mathsf{tr}(y_r)$ the world j survives. Wlog if we assume $\ell_p = y_p$ then the $Exp(j)$ will have $(\Sigma \setminus \{\mathsf{tr}(\ell_p), \mathsf{tr}(\neg \ell_p)\})^*$ added in the updated model, which guarantees the survival of the world in subsequent updates.

Thus for any world j (corresponding to the clause $c_j = (\ell_p \vee \ell_q \vee \ell_r)$), in the $\mathcal{M}_\varphi|_w$ the world j survives, that is, no world vanishes after the update. And since $s \sim_1 j$ for all j, hence, $\mathcal{M}_\varphi|_w, s \vDash \bigwedge_{i \in [m]} (\hat{K}_1 p_i)$ for any $1 \leq j \leq m$, since all the initial m worlds are in the

same equivalence class of $\sim_1$. Since this is true for any w corresponding to any satisfying assignment of γ, so $\mathcal{M}_\varphi, s \vDash \psi$.

Conversely, assume φ is unsatisfiable. Note that if for a set of literals $\ell := \ell_1, \ldots, \ell_n$ if γ evaluates to False then there exists i_ℓ such that the world i_ℓ does not survive in $\mathcal{M}_\varphi|_{\mathsf{tr}(\ell_1)\ldots\mathsf{tr}(\ell_n)}$. This is because there exist at least a clause, say c_{i_ℓ} in γ, that evaluates to False when the the literals ℓ is assignment True. Consider the The world i_ℓ in $\mathcal{M}_\varphi$ corresponding to c_{i_ℓ}, will have $Exp(i_\ell)\backslash w = \delta$. In that case, note that $\mathcal{M}_\varphi|_w, s \nvDash \hat{K}_1 p_{i_\ell}$. Thus there is a bijection between the set $\mathcal{T}$ of all satisfying assignments of γ and the set $\{w \mid \mathcal{M}_\varphi|_w, s \vDash \hat{K}_1 p_{i_\ell}\}$. Now from the construction of the formula ψ we see that φ is satisfiable iff $\mathcal{M}, s \vDash \psi$. □

Chapter 4

Model Checking Fragments and Application

4.1 Introduction

Although the model-checking of DEL is PSPACE-complete [6], deciding the plan-existence problem of Epistemic Planning is undecidable [14]. This is due to the reason that action model in DEL get composed using regular language operators including Kleene Star.

With respect to that, POL deals with public observations, which are actions which all the agents can distinguish among. That is, action model has a single action point. It is to be noted that POL model checking do involve all the regular language operators including Kleene Star that are composing the atomic observations. Hence POL model-checking has a natural utility in Epistemic Planning. In this chapter, we show such utility using example 3.

Although the model-checking algorithm is PSPACE-complete, for practical purposes, we investigate syntactic fragments that offer better complexities than reasoning in the full language of POL (see Figure 4.1), and are suitable for relevant verification tasks:

- the Word fragment, where any regular expression π is a *word*, is sufficient to verify that some given plan leads to a state satisfying some epistemic property;
- the Existential fragment, where the dynamic operators of POL are all *existential*, is suitable for epistemic planning (e.g., does there exist a plan?);
- the Star-Free fragment, where the regular expressions π are *star-free*, embeds *bounded* planning (in which sequences of observations to synthesize are bounded by some constant). In particular, the Star-FreeExistential fragment (i.e. the intersection of the Star-Free and the Existential fragments) is suitable for bounded epistemic planning.

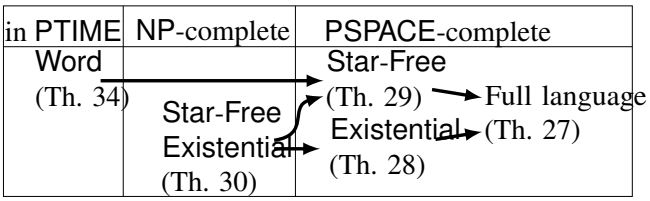

Fig. 4.1 Complexity results of model checking for different fragment of POL. (arrows represent inclusion of fragments).

Outline. Section 4.2 explores various syntactic fragments of POL model-checking problems and their complexity. Section 4.3 shows applications of POL model-checking in Epistemic planning using the farming drone example. In addition to that, there are also implementations of fragments on the example instances. In section 4.4, we discuss literature that are related with POL model-checking and its applications. Finally, in section 4.5, we conclude the POL model-checking work.

4.2 Model checking for Star-Free Existential and Word fragment of POL

While Theorems 28 and 29 proved the PSPACE-hardness of the model checking for the Existential fragment and the Star-Free fragment of POL, respectively, if we consider the Star-Free Existential fragment then we can show that the model checking is NP-complete. Finally, we also prove that the model checking for the Word fragment is in P. We start with proving that the model-checking problem for the Star-Free – Existential is in NP. To prove that the model-checking problem for the Star-Free – Existential is in NP we present an algorithm WorldsNP and in Lemma 32 we prove the correctness and complexity of the algorithm mcSF&EXIT. Note that, the algorithm WorldsNP is a non-deterministic algorithm, since it guesses letters from the alphabet Σ in line 20 of the algorithm.

In Lemma 33 we prove that the model-checking problem in NP-hard. Thus Lemma 32 and 33 combined gives us

Theorem 30. *The model checking problem for the* Star-Free Existential *fragment of* POL *is* NP*-complete.*

The algorithm mcSF&EXIT calls a non-deterministic subroutine WorldsNP that takes the model $\mathcal{M}$ and NNF formula φ, which is a Star-Free – Existential formula, and returns the set of all worlds s such that $\mathcal{M}, s \vDash \varphi$. The subroutine WorldsNP uses another subroutine

Algorithm 8 mcSF&EXIT

1: **procedure** mcSF&EXIT($\mathcal{M} = \langle S, R, V, Exp\rangle$, $s \in S$, φ, where Exp are NFA, and φ is a Star-Free – Existential Formula)
2: **if** $s \in$ WorldsNP$(\mathcal{M}, \varphi)$ **then**
3: Return True

ResidueByLetter that in turn uses a subroutine AuxOut. The goal of the subroutine ResidueByLetter is to take as input a regular expression π and a word a and output the residue $\pi\backslash a$ in polynomial time. Although the algorithm ResidueByLetter is straightforward for the sake of completeness we present the pseudo-code formally. The proof of correctness and the complexity of the algorithms (formally stated in Lemma 31) follows from standard arguments.

Lemma 31. *Given a regular expression π over Σ, and $a \in \Sigma$,* ResidueByLetter *returns $\pi\backslash a$ in polynomial time.*

Proof. The algorithm is a recursive one that directly follows the inductive definition 15 of Residue. Hence the correctness and the complexity of the algorithm follows. □

Lemma 32. *Given a finite* POL *model $\mathcal{M} = \langle S, R, V, Exp\rangle$, an $s \in S$ and a* Star-Free – Existential *formula φ, the algorithm* mcSF&EXIT *is a polynomial time non-deterministic algorithm that outputs* True *iff $\mathcal{M}, s \vDash \varphi$.*

In other words, the model-checking problem for the Star-Free – Existential *fragment of* POL *is in* NP.

Proof. We prove the lemma in two parts: first we will prove the correctness of the algorithm WorldsNP - that is we show that given a finite POL model $\mathcal{M} = \langle S, R, V, Exp\rangle$, an $s \in S$ and a Star-Free – Existential formula φ, $s \in$ WorldsNP$(\mathcal{M}, \varphi)$ iff $\mathcal{M}, s \vDash \varphi$. The correctness of the algorithm mcSF&EXIT follows immediately.

We then prove the complexity of the algorithm mcSF&EXIT.

Proof of Correctness of WorldsNP Let us start by proving the correctness of the algorithm WorldsNP. This can be proved by induction over the size of φ.

Base Case. Consider the case where $\varphi = p$, where $p \in \mathbf{P}$. In the scope of the IF case in line 2, the set S' is populated with all the worlds $s \in S$ where $p \in V(s)$. Hence, $\mathcal{M}, s \vDash \varphi$ iff $s \in$ WorldsNP$(\mathcal{M}, \varphi)$

Induction Hypothesis (IH). Given a finite POL model $\mathcal{M} = \langle S, R, V, Exp\rangle$, an $s \in S$ and a Star-Free – Existential formula φ, $s \in$ WorldsNP$(\mathcal{M}, \varphi)$ iff $\mathcal{M}, s \vDash \varphi$, where $|\varphi| \leq k$, for an integer k.

Inductive Step.

Algorithm 9 WorldsNP

procedure WorldsNP($\mathcal{M} = \langle S, R, V, Exp\rangle, \varphi$, where Exp are NFA, and φ is a Star-Free – Existential Formula)
  if $\varphi = p$ is a propositional variable **then**
    $S' = \emptyset$
    for $s \in S$ **do**
      if $p \in V(s)$ **then**
        $S' = S' \cup \{s\}$
    Return S'
  if $\varphi = \neg p$, where p is a propositional variable **then**
    Return $S \setminus$ WorldsNP$(\mathcal{M}, \psi)$
  if $\varphi = \psi_1 \vee \psi_2$ **then**
    Return WorldsNP$(\mathcal{M}, \psi_1) \cup$ WorldsNP$(\mathcal{M}, \psi_2)$
  if $\varphi = \psi_1 \wedge \psi_2$ **then**
    Return WorldsNP$(\mathcal{M}, \psi_1) \cap$ WorldsNP$(\mathcal{M}, \psi_2)$
  if $\varphi = \langle\pi\rangle\psi$ **then**
    $\pi' = \pi$
    $\mathcal{M}' = \mathcal{M}$
    for $i = 1$ to $|\pi|$ **do**
      if $\epsilon \in \pi'$ and $s \in S$ **then**
        Return WorldsNP$(\mathcal{M}', \psi)$
      Guess a letter $a \in \Sigma$
      $\pi' = \pi' \backslash a$ (using ResidueByLetter)
      for each state $s \in S$ **do**
        $Exp(s) = Exp(s) \backslash a$ (using ResidueByLetter)
  if $\varphi = \hat{K}_i\psi$ **then**
    $S' =$ WorldsNP$(\mathcal{M}, \psi)$
    Return $\{s \in S \mid \exists t \in S'$ and $t \sim_i s\}$

- $\varphi = \neg\psi$

$$
\begin{aligned}
&\mathcal{M}, s \vDash \neg\psi \\
\text{iff } &\mathcal{M}, s \nvDash \psi \\
\text{iff } &s \notin \text{WorldsNP}(\mathcal{M}, \psi), \text{ by IH} \\
\text{iff } &s \in S \setminus \text{WorldsNP}(\mathcal{M}, \psi) \\
\text{iff } &s \in \text{WorldsNP}(\mathcal{M}, \neg\psi)
\end{aligned}
$$

Algorithm 10 ResidueByLetter

```
1: procedure ResidueByLetter(Regular expression π and a letter a ∈ Σ)
2:     if π ∈ Σ ∪ {ϵ, δ} then
3:         if π = a then
4:             Return ϵ
5:         else
6:             Return δ
7:     if π = π1 + π2 then
8:         Return ResidueByLetter(π1, a) + ResidueByLetter(π2, a)
9:     if π = π1.π2 then
10:        Return ResidueByLetter(π1, a).π2 + AuxOut(π1).ResidueByLetter(π2, a)
11:    if π = (π1)* then
12:        Return ResidueByLetter(π1, a).(π1)*
```

Algorithm 11 AuxOut

```
1: procedure AuxOut( A regular expression π)
2:     Create A_π =< Q_π, Σ, δ_π, q_0^π, F_π >, the NFA for π
3:     if q_0^π ∈ F_π then
4:         Return ϵ
5:     else
6:         Return δ
```

- $\varphi = \psi_1 \vee \psi_2$

$$
\begin{aligned}
& \mathcal{M}, s \vDash \psi_1 \vee \psi_2 \\
\text{iff } & \mathcal{M}, s \vDash \psi_1 \text{ or } \mathcal{M}, s \vDash \psi_2 \\
\text{iff } & s \in \mathsf{WorldsNP}(\mathcal{M}, \psi_1) \\
& \qquad \text{or } s \in \mathsf{WorldsNP}(\mathcal{M}, \psi_2), \text{ by IH} \\
\text{iff } & s \in \mathsf{WorldsNP}(\mathcal{M}, \psi_1) \\
& \qquad \cup \mathsf{WorldsNP}(\mathcal{M}, \psi_2) \\
\text{iff } & s \in \mathsf{WorldsNP}(\mathcal{M}, \psi_1 \vee \psi_2)
\end{aligned}
$$

- $\varphi = \psi_1 \wedge \psi_2$

$$
\begin{aligned}
&\mathcal{M}, s \vDash \psi_1 \wedge \psi_2 \\
&\text{iff } \mathcal{M}, s \vDash \psi_1 \text{ and } \mathcal{M}, s \vDash \psi_2 \\
&\text{iff } s \in \mathsf{WorldsNP}(\mathcal{M}, \psi_1) \\
&\qquad \text{and } s \in \mathsf{WorldsNP}(\mathcal{M}, \psi_2), \text{ by IH} \\
&\text{iff } s \in \mathsf{WorldsNP}(\mathcal{M}, \psi_1) \\
&\qquad \cap \mathsf{WorldsNP}(\mathcal{M}, \psi_2) \\
&\text{iff } s \in \mathsf{WorldsNP}(\mathcal{M}, \psi_1 \wedge \psi_2)
\end{aligned}
$$

- $\varphi = \hat{K}_i \psi$

$$
\begin{aligned}
&\mathcal{M}, s \vDash \hat{K}_i \psi \\
&\text{iff } \exists t \sim_i s \text{ and } \mathcal{M}, t \vDash \psi \\
&\text{iff } \exists t \sim_i s \text{ and } t \in \mathsf{WorldsNP}(\mathcal{M}, \psi), \text{ by IH} \\
&\text{iff } \mathsf{WorldsNP}(\mathcal{M}, \hat{K}_i \psi)
\end{aligned}
$$

- $\varphi = \langle \pi \rangle \psi$

$$
\begin{aligned}
&\mathcal{M}, s \vDash \langle \pi \rangle \psi \\
&\text{iff } \exists w \in \mathcal{L}(\pi) : \mathcal{L}(Exp(s) \backslash w) \neq \emptyset \\
&\qquad \text{and } \mathcal{M}|_w, s \vDash \psi \\
&\text{iff } \exists w : |w| \leq |\pi| \text{ and } \mathcal{L}(Exp(s) \backslash w) \neq \emptyset \\
&\qquad \text{and } s \in \mathsf{WorldsNP}(\mathcal{M}|_w, \psi)
\end{aligned}
$$

Since π is star-free, hence all words in $\mathcal{L}(\pi)$ is size at most $|\pi|$. Loop in line 17 guesses $w \in \mathcal{L}(\pi)$ letter by letter, residues π (Line 21) and updates model (the for loop starting from Line 22), and recursively calls $\mathsf{WorldsNP}(\mathcal{M}|_w, \psi)$ once π is exhausted (condition in Line 18), that is $w \in \mathcal{L}(\pi)$. The residuation in Line 21 and the model updation can be done by $\mathsf{ResidueByLetter}$, which is correct by Lemma 31. Therefore, $\mathcal{M}, s \vDash \langle \pi \rangle \psi$ iff $s \in \mathsf{WorldsNP}(\mathcal{M}, \langle \pi \rangle \psi)$

Complexity of WorldsNP Now for the complexity of the algorithm, WorldsNP let us prove that the algorithm is a non-deterministic polytime algorithm. WorldsNP is a recursive algorithm that returns the worlds in $\mathcal{M}$ where φ holds. The algorithms labels each world s in the $\mathcal{M}$ with $\psi \subseteq \varphi$ iff ψ holds in s.

For each case of propositional operators, that is,

$$\varphi = p \mid \neg p \mid \psi_1 \vee \psi_2 \mid \psi_1 \wedge \psi_2$$

, assuming the worlds are labelled by the subformulas of φ, deciding the worlds where φ holds require linear steps with respect to the worlds in $\mathcal{M}$.

In case of $\varphi = \hat{K}_i\psi$, for each world s to be decided whether to label by φ, at worst case at most all the related worlds of s needs to be checked whether at least one of them is labelled by ψ. Hence, assuming all the worlds satisfying ψ are labelled, this step takes quadratic steps with respect to the number of worlds in $\mathcal{M}$.

In the case of $\varphi = \langle\pi\rangle\psi$, π is promised to be star-free. Hence, any word $u \in \mathcal{L}(\pi)$ is such that $|u| \leq |\pi|$. Hence, there exists a sequence of letters w of length at most $|\pi|$, such that $\mathcal{M}|_w, s \vDash \psi$ if and only if $\mathcal{M}, s \vDash \langle\pi\rangle\psi$. By Lemma 31, the residuation in Line 21 and Line 23 takes polynomial time. Also at each step, there are constant number of guesses (a letter from Σ).
Also, there can be at most $|\varphi|$ number of subformulas of φ, each of size at most $|\varphi|$.

Hence WorldsNP is the NP algorithm for model checking problem where the input formula is promised to be Star-Free – Existential. Thus the algorithm mcSF&EXIT is also a non-deterministic polytime algorithm.

□

Lemma 33. *The model-checking problem for the* Star-Free – Existential *fragment of* POL *is* NP*-hard.*

Proof. We shall prove this by a reduction from 3-SAT. Given a 3-SAT CNF-formula φ containing n variables $\{x_1, \ldots, x_n\}$ and m clauses $C = \{c_1, \ldots, c_m\}$, we shall construct our reduction. Consider the translations $\mathsf{tr}(x_i) = a_i$, $\mathsf{tr}(\neg x_i) = a_i'$ for all $i \in [n]$. Now we present the model checking instance that we construct from φ.

- Alphabet: $\Sigma_\varphi = \cup_{i\in[n]}\{\mathsf{tr}(x_i), \mathsf{tr}(\neg x_i)\}$.
- Model: $\mathcal{M}_\varphi = \langle S_\varphi, \sim_\varphi, V_\varphi, Exp_\varphi\rangle$, where
  - $S_\varphi = \{1, \ldots, m\}$.
  - A single agent 1 and $\forall i, j \in S_\varphi$, $i \sim_1 j$.

- $V_\varphi(j) = \{p_j\}$ for all $j \in S_\varphi$.
- For each clause $c_j \in C$, $Exp_\varphi(j) = (\sum_{l \in c_j}(\Sigma_\varphi \setminus \{\mathsf{tr}(\ell), \mathsf{tr}(\neg\ell)\})^* \mathsf{tr}(\ell)(\Sigma_\varphi \setminus \{\mathsf{tr}(\ell), \mathsf{tr}(\neg\ell)\})^*)$, where the sum is over the literals in the clause c_j.

- Formula: $\psi := \langle a_1 + a_1' \rangle \ldots \langle a_n + a_n' \rangle \bigwedge_{i \in [m]} (\hat{K}_1 p_i)$.
- Starting world: Any s from S.

Note that, since we are dealing with Star-Free – Existential, we have Kleene star only in the model, but not in the formula. Also, we only have $\langle\rangle$ and $\hat{K}_i$ operators in the formula. All we need to show now is that the CNF-formula φ is satisfiable iff $\mathcal{M}_\varphi, s \vDash \psi$.

Let us start by assuming φ is True. We have to prove that $\mathcal{M}_\varphi, s \vDash \psi$. Hence there exists an assignment $\sigma = (l_1, l_2, \ldots, l_n)$, where $l_i \in \{x_i, \neg x_i\}$, such that it evaluates φ to True. Now the consider the corresponding word $w = \mathsf{tr}(l_1)\mathsf{tr}(l_2)\ldots\mathsf{tr}(l_n)$. By construction, proving $\mathcal{M}_\varphi|_w, s \vDash \bigwedge_{i \in [m]} (\hat{K}_1 p_i)$ proves our claim.

We can consider $\mathcal{M}_\varphi|_w$ to be updated from a series of updates $\mathcal{M}_\varphi|_{\mathsf{tr}(l_1)}$, $\mathcal{M}_\varphi|_{\mathsf{tr}(l_1)\mathsf{tr}(l_2)}$, …, $\mathcal{M}_\varphi|_{\mathsf{tr}(l_1)\mathsf{tr}(l_2)\ldots\mathsf{tr}(l_n)}$. Let us consider a world j in the model $\mathcal{M}_\varphi$, corresponding to the clause $c_j = (l_p \vee l_q \vee l_r)$, where l_p, l_q, l_r are literals in the clause. Note that, if neither of the literals in c_j contains x_i, the update of the model with $\mathsf{tr}(x_i)$ or $\mathsf{tr}(\neg x_i)$ does not affect the world j. Now, on the other hand, without loss of generality, since σ is a satisfying assignment, consider literal l_p is in σ, since at least one of the literals in c_j has to be in the satisfying assignment. Note that after updating corresponding to $\mathsf{tr}(l_p)$, the $Exp(j)$ will have the residue regular expression $(\Sigma \setminus \{\mathsf{tr}(l_p), \mathsf{tr}(\neg l_p)\})^*$ which guarantees the survival of the world j in future updates.

Hence, since σ is a satisfying assignment, every clause will have at least one literal in σ, hence guaranteeing the survival of all the worlds in $\mathcal{M}_\varphi|_w$. Since all the world survives and $s \sim_1 t$, for every $t \in S_\varphi$, hence $\mathcal{M}_\varphi|_w, s \vDash \bigwedge_{j \in [m]} \hat{K}_j p_j$.

Conversely, let φ is unsatisfiable. Hence, for any assignment $\sigma = (l_1, l_2, \ldots, l_n)$, there exists at least one clause, say $c_j = (l_p \vee l_q \vee l_r)$, such that $\neg l_p, \neg l_q, \neg l_r$ is in the σ. Note update in the world j corresponding to clause c_j. After the update corresponding to $\mathsf{tr}(\neg l_p)$, the term $(\Sigma \setminus \{\mathsf{tr}(l_p), \mathsf{tr}(\neg l_p)\})^* \mathsf{tr}(l_p)(\Sigma \setminus \{\mathsf{tr}(l_p), \mathsf{tr}(\neg l_p)\})^*$ in $Exp(j)$ becomes δ (regular expression for the empty regular language). Same occurs for the update corresponding to $\mathsf{tr}(\neg l_q)$ and $\mathsf{tr}(\neg l_r)$. Hence the $Exp(j)$, after all the three updates, becomes δ, due to which the world j does not survive. Hence $\mathcal{M}_\varphi|_w, s \nvDash \bigwedge_{j \in [m]} \hat{K}_j p_j$, where $w = \mathsf{tr}(l_1)\mathsf{tr}(l_2)\ldots\mathsf{tr}(l_n)$.

□

Theorem 34. *Model checking for the* Word *fragment is in* P.

Algorithm 12 mcWords

procedure mcWords(φ)
  if $s \in$ mcWordPOL($\mathcal{M}, \varphi$) **then**
    Return True

Algorithm 13 mcWordPOL

procedure mcWordPOL($\mathcal{M} = \langle S, R, V, Exp \rangle$, φ, where φ is a Word Formula)
  if $\varphi = p$ is a propositional variable **then**
    $S' = \emptyset$
    for $s \in S$ **do**
      if $p \in V(s)$ **then**
        $S' = S' \cup \{s\}$
    Return S'
  if $\varphi = \neg\psi$ **then**
    Return $S \setminus$ mcWordPOL($\mathcal{M}, \psi$)
  if $\varphi = \psi_1 \vee \psi_2$ **then**
    Return mcWordPOL($\mathcal{M}, \psi_1$) $\cup$ mcWordPOL($\mathcal{M}, \psi_2$)
  if $\varphi = \langle w \rangle \psi$ **then**
    $\mathcal{M}' =< S', R', V', Exp' >= \mathcal{M}$
    for $s \in S'$ **do**
      $Exp'(s) = Exp(s) \setminus w$
      if $Exp'(s) = \delta$ **then**
        $S' = S' \setminus \{s\}$
        $R' = R' \setminus \{\{s, t\}\}$ for every $t \in S$
    Return mcWordPOL($\mathcal{M}', \psi$)
  if $\varphi = \hat{K}_i \psi$ **then**
    $S' =$ mcWordPOL($\mathcal{M}, \psi$)
    Return $\{s \in S \mid \exists t \in S'$ and $t \sim_i s\}$

Proof. The model checking algorithm for POL, when π's are words, can be designed in a similar way as the folklore recursive model checking algorithm for epistemic logic. Only modification is when, checking whether $\mathcal{M}, s \vDash \langle \pi \rangle \varphi$ recursively call $\mathcal{M}|w, s \vDash \varphi$ where $\mathcal{L}(\pi) = \{w\}$.

We use the algorithm mcWords for model checking the Word fragment. The algorithm mcWords calls the subroutine mcWordPOL which is a polytime algorithm that takes a model $\mathcal{M}$ and a word-formula φ and outputs the set of all states s such that $\mathcal{M}, s \vDash \varphi$. The correctness of the algorithm mcWords follows from the correctness of the algorithm mcWordPOL. The proof of correctness of the algorithm mcWordPOL is presented in Lemma 35. □

Lemma 35. *Given a finite* POL *model* $\mathcal{M} = \langle S, R, V, Exp \rangle$*, an* $s \in S$ *and a* Word *formula* φ*,* $s \in \mathsf{mcWordPOL}(\mathcal{M}, \varphi)$ *iff* $\mathcal{M}, s \vDash \varphi$*.*

Proof. This can be proved by induction over the size of φ.

Base Case. Consider the case where $\varphi = p$, where $p \in \mathbf{P}$. In the scope of the IF case in line 2, the set S' is populated with all the worlds $s \in S$ where $p \in V(s)$. Hence, $\mathcal{M}, s \vDash \varphi$ iff $s \in \mathsf{mcWordPOL}(\mathcal{M}, \varphi)$.

Induction Hypothesis. Given a finite POL model $\mathcal{M} = \langle S, R, V, Exp \rangle$, an $s \in S$ and a POL formula of the Word fragment φ, $s \in \mathsf{mcWordPOL}(\mathcal{M}, \varphi)$ iff $\mathcal{M}, s \vDash \varphi$, where $|\varphi| \leq k$, for an integer k.

Inductive Step.

- $\varphi = \neg\psi$

$$
\begin{aligned}
&\mathcal{M}, s \vDash \neg\psi \\
\text{iff } &\mathcal{M}, s \nvDash \psi \\
\text{iff } &s \notin \mathsf{mcWordPOL}(\mathcal{M}, \psi), \text{ by IH} \\
\text{iff } &s \in S \setminus \mathsf{mcWordPOL}(\mathcal{M}, \psi) \\
\text{iff } &s \in \mathsf{mcWordPOL}(\mathcal{M}, \neg\psi)
\end{aligned}
$$

- $\varphi = \psi_1 \vee \psi_2$

$$
\begin{aligned}
&\mathcal{M}, s \vDash \psi_1 \vee \psi_2 \\
\text{iff } &\mathcal{M}, s \vDash \psi_1 \text{ or } \mathcal{M}, s \vDash \psi_2 \\
\text{iff } &s \in \mathsf{mcWordPOL}(\mathcal{M}, \psi_1) \\
&\qquad \text{or } s \in \mathsf{mcWordPOL}(\mathcal{M}, \psi_2), \text{ by IH} \\
\text{iff } &s \in \mathsf{mcWordPOL}(\mathcal{M}, \psi_1) \cup \mathsf{mcWordPOL}(\mathcal{M}, \psi_2) \\
\text{iff } &s \in \mathsf{mcWordPOL}(\mathcal{M}, \psi_{\vee}\psi_2)
\end{aligned}
$$

- $\varphi = \hat{K}_i\psi$

$$
\begin{aligned}
&\mathcal{M}, s \vDash \hat{K}_i\psi \\
\text{iff } &\exists t \sim_i s \text{ and } \mathcal{M}, t \vDash \psi \\
\text{iff } &\exists t \sim_i s \text{ and } t \in \mathsf{mcWordPOL}(\mathcal{M}, \psi), \text{ by IH} \\
\text{iff } &\mathsf{mcWordPOL}(\mathcal{M}, \hat{K}_i\psi)
\end{aligned}
$$

- $\varphi = \langle w \rangle \psi$
  Since w is a word, $Exp(s)\backslash w$ is calculated in line 15, and hence for all $s \in S$ in the loop in 14-18. Also a certain state $t(\in S) \notin S'$ if $Exp(t)\backslash w = \delta$, that is, $\mathcal{L}(Exp(t)\backslash w) = \emptyset$. By Lemma 31, we have the correctness of the residuation. Hence after the termination of loop 14-18, $\mathcal{M}' = \mathcal{M}|_w$.

$$
\begin{aligned}
& \mathcal{M}, s \vDash \langle w \rangle \psi \\
\text{iff } & \mathcal{L}(Exp(s)\backslash w) \neq \emptyset \\
& \quad \text{and } \mathcal{M}|_w, s \vDash \psi, \text{ since } w \text{ is word} \\
\text{iff } & \mathcal{L}(Exp(s)\backslash w) \neq \emptyset \\
& \quad \text{and } s \in \mathsf{mcWordPOL}(\mathcal{M}', \psi), \text{ by IH} \\
\text{iff } & s \in \mathsf{mcWordPOL}(\mathcal{M}, \langle w \rangle \psi)
\end{aligned}
$$

□

4.3 Application

Recall example 3, that of farming drone introduced in chapter 2 (Figure 4.3). There were three possibilities, each having its own expectation.

1. For the possibility of the drone going up-right searching for water with at most one wrong direction (▼ or ◄), the corresponding set of expectations is captured by the regular expression $(\blacktriangleright \cup \blacktriangle)^*(\blacktriangledown \cup \blacktriangleleft \cup \varepsilon)(\blacktriangleright \cup \blacktriangle)^*$ where ε stands for the empty word regular expression.

2. For the possibility of the drone going down-left for power supply with at most one wrong direction (▲ or ►), the corresponding set of expectations is captured by the regular expression: $(\blacktriangleleft \cup \blacktriangledown)^*(\blacktriangle \cup \blacktriangleright \cup \varepsilon)(\blacktriangleleft \cup \blacktriangledown)^*$.

3. For the possibility of the drone patrolling making clockwise squares, the expectation is: $(\blacktriangleright^+ \blacktriangledown^+ \blacktriangleleft^+ \blacktriangle^+)^*$.

The model $\mathcal{M}$, depicted in Figure 4.2, can be obtained by techniques described in [62] (they use a mechanism from DEL for constructing the epistemic expectation model, by assigning the expectations at each world). The regular expressions (the expectations) may be learned by the agents after observing several executions (see for instance [9]) or might be computed by planning techniques [16].

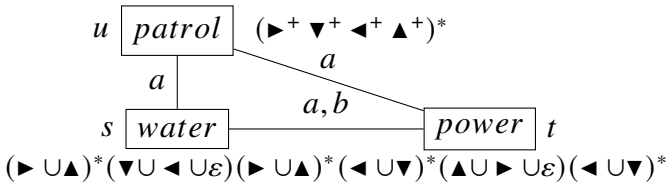

Fig. 4.2 Model describing the initial knowledge of the two agents a and b about the expectation of the automatic farming drone.

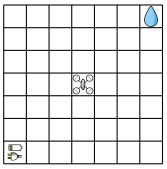

Fig. 4.3 Field and an automatic farming drone.

Now, verification tasks related to epistemic planning (e.g., verifying whether $K_a water$, $K_b water$, $K_a power$, or $K_b power$ is true after some observations) reduce to the POL model checking problem. Let us now discuss the expressivity of the fragments: Word, Existential, Star-Free – Existential and Star-Free.

In the Word fragment, words are fixed sequences of observations. The fragment thus enables to write formulas of the form $\langle w \rangle \varphi$, meaning that φ holds after the sequence w of observations (that can be considered as the observations produced by the plan executed by the system). Thus, this fragment enables to write formulas to verify properties after the execution of a plan.

Example 11 (verification of a plan, Word fragment)**.** *Does agent a know that the drone is searching for water after the sequence* ►►►*?*

$$\mathcal{M}, s \models \langle ►►► \rangle K_a water$$

Epistemic planning is the general problem of verifying whether there exists a plan leading to a state satisfying a given epistemic formula. In our setting, it can be expressed by a formula of the form $\langle \pi \rangle \varphi$ where π denotes the plan search space (more precisely the search space of sequences of observations produced by a plan).

Example 12 (epistemic planning, Existential fragment)**.** *Does there exist a plan for the drone such that agent b would know that the drone is searching for water while agent a would still consider patrolling a possibility?*

$$\mathcal{M}, s \models \langle (► \cup ▼ \cup ◄ \cup ▲)^* \rangle (K_b water \wedge \hat{K}_a patrolling)$$

In planning (and also in epistemic planning), we may ask for the existence of a plan of bounded length, e.g., less than 4 actions. The Star-FreeExistential fragment is sufficiently expressive to tackle the so-called *bounded* epistemic planning.

Example 13 (bounded epistemic planning, Star-Free Existential fragment)**.** *Does there exist a sequence of at most 4 moves such that agent b would know that the drone is searching for water while agent a would still consider patrolling a possibility?*

$$\mathcal{M}, s \models \langle (\blacktriangleright \cup \blacktriangledown \cup \blacktriangleleft \cup \blacktriangle \cup \epsilon)^4 \rangle (K_b water \wedge \hat{K}_a patrolling)$$

Interestingly, the Star-Free fragment and the full language are able to express properties, mixing existence and non-existence of plans, in respectively the bounded and unbounded cases.

Example 14 (Star-Free fragment)**.** *Agent a would not gain the knowledge that the drone will search for water with less than or equal to 2 movements but it is possible with 3 movements:*

$$\begin{aligned}\mathcal{M}, s \models & [(\blacktriangleright \cup \blacktriangledown \cup \blacktriangleleft \cup \blacktriangle)^2] \neg K_a water] \wedge \\ & \langle (\blacktriangleright \cup \blacktriangledown \cup \blacktriangleleft \cup \blacktriangle)^3 \rangle K_a water\end{aligned}$$

Example 15 (full language)**.** *It is impossible for the agent a to know that the drone is searching for water with only down and left movements but there is a plan if all movements are allowed:*

$$\begin{aligned}\mathcal{M}, s \models & [(\blacktriangledown \cup \blacktriangleleft)^*] \neg K_a water] \wedge \\ & \langle (\blacktriangleright \cup \blacktriangledown \cup \blacktriangleleft \cup \blacktriangle)^* \rangle K_a water\end{aligned}$$

4.3.1 On implementation.

We have implemented in this work, the Star-Free – Existential fragment and the Word fragment of POL.

Star-FreeExistential **Implementation**

The model checking for the Star-FreeExistential fragment can be implemented via a reduction to SAT. In this section, we explain how to provide an efficient implementation for the Star-Free – Existential fragment of POL model checking by providing a reduction to SAT. We explain how to check that $\mathcal{M}, w \models \langle \pi \rangle \varphi$, where π is star-free and φ is an epistemic formula. In other words, we aim at checking whether there is a guessed word that belongs to

the language of π such that w survives the announcement of that word and φ holds in the updated model. As π is star-free, the guessed word is bounded by the size of π. W.l.o.g. we suppose that the guessed word is of length k. We also suppose that all the automata are deterministic. This costs an exponential time in principle, but we can rely on an efficient minimization algorithm to obtain small deterministic automata in practical cases.

We now explain how to define a Boolean formula $tr(M, w, \langle\pi\rangle\varphi)$ such that $\mathcal{M}, w \models \langle\pi\rangle\varphi$ iff $tr(M, w, \langle\pi\rangle\varphi)$ is a satisfiable Boolean formula. To this end, we introduce several propositional variables:

- $p_{t,a}$: the t-th letter of the guessed word is a
- $t_{u,\psi}$: subformula ψ (of φ) holds in world u in the updated model
- $a_{A,t,q}$: in the automaton A, the state after having read t letters in the guessed word is q. Note that A can denote either an automaton for π or that for any prefix language $Exp(u)$.

Given a finite set P of propositional variables, we write $\#(P) = 1$ for a Boolean formula saying that exactly one propositional variable in P is true. We construct the following Boolean formula

$$t_{w,\varphi} \wedge guessedWord \wedge good \wedge surv \wedge rules$$

which is satisfiable iff $\mathcal{M}, w \models \langle\pi\rangle\varphi$. The first part $t_{w,\varphi}$ enforces that the formula φ is true in w after the announcement of the guessed word.

The second part $guessedWord$ says that the guessed word is uniquely determined by the propositions $p_{t,a}$. More precisely, that part is: $\bigwedge_{t=1..k} \#(\{p_{t,a} \mid a \in \Sigma\}) = 1$.

In other words, it means that there is a path in the non-deterministic automaton A corresponding to π, starting from the initial state q_0, following the guessed word, and leading to a final state. Given A, the automaton for π, the formula $good$ is the conjunction of:

- $a_{A,0,q_0}$;
- $\bigwedge_{t=0..k} \#(\{a_{A,t,q} \mid q \in Q\}) = 1$;
- $\bigwedge_{t=0..k-1,q,a} a_{A,t,q} \wedge p_{t,a} \rightarrow p_{A,t+1}\delta(q,a)$
- $\bigvee_{q\in F} a_{A,k,q}$.

The part $surv$ says that the guessed word belongs to the prefix language of $Exp(s)$. That formula is similar to $good$ but for the automaton corresponding to the prefix language of $Exp(s)$. Finally, $rules$ is a formula that mimics the semantics of $t_{w,\varphi}$ in the same spirit as Tseitin transformation. Formula $rules$ is the conjunction of:

- $t_{u,p} \leftrightarrow \top$ if $u \models p$;
- $t_{u,p} \leftrightarrow \bot$ if $u \not\models p$;
- $t_{u,\neg\psi} \leftrightarrow \neg t_{u,\psi}$;
- $t_{u,\psi_1 \wedge \psi_2} \leftrightarrow (t_{u,\psi_1} \wedge t_{u,\psi_2})$;
- $t_{u,K_i\psi} \leftrightarrow \bigwedge_{v|u \to_i v} surv(v) \to t_{u,\psi}$

where $surv(v)$ is a propositional variable saying that world v survives the announcement of the guessed word. The variable $surv(v)$ alone is not sufficient. It is accompanied by a collection of clauses in the same spirit as $surv$ but with the automaton of the prefixes of $Exp(v)$.

We have $\mathcal{M}, w \models \langle\pi\rangle\varphi$ iff $tr(M, w, \langle\pi\rangle\varphi)$ is a satisfiable Boolean formula. Moreover, the truth values of propositions $p_{t,a}$ in a valuation satisfying $tr(M, w, \langle\pi\rangle\varphi)$ gives a plan, i.e. the guessed word in the language of π such that φ holds after executing that word from $\mathcal{M}, w$. The full case of Star-Free – Existential fragment follows the same idea but is cumbersome.

The reduction to SAT has been implemented in `Python3` using the library `pySAT` (with the SAT solver `Glucose`) and the library `automata-lib`. To get an idea, the running time for checking Example 3 is around 10ms.[1]

Word **Implementation**

The model checking for the Word fragment can be implemented in poly-time with a bottom-up traversal of the parse tree of the formula, as for CTL [7, Section 6.4].
The input POL formula is a string that follows the following syntax. Let $tr(\varphi)$ be the string corresponding to the POL formula φ which is defined as :

- When $\varphi = p$, where p is a propositional formula, $tr(p) =' p'$.
- When $\varphi = \psi \vee \chi$, then $tr(\varphi) =' OR(tr(\psi), tr(\chi))'$.
- When $\varphi = \psi \wedge \chi$, then $tr(\varphi) =' AND(tr(\psi), tr(\chi))'$.
- When $\varphi = \neg\psi$, then $tr(\varphi) =' NOT(tr(\psi))'$.
- When $\varphi = \hat{K}_a\psi$, then $tr(\varphi) =' KP(a, tr(\psi))'$, where a is an agent.
- When $\varphi = \langle w\rangle\psi$, then $tr(\varphi) =' DIM(w, tr(\chi))'$, where w is a word.

[1] The accompanying codes can be found in the following link: https://github.com/francoisschwarzentruber/polmc

The POL model is a dictionary instance with the following index names:

- 'worlds' is a set of worlds in the model.
- 'agents' is a set of agents in the model.
- 'propositions' is the set of propositions valuations of the worlds are on.
- 'relation' is a dictionary indexed by agents. Each agent index 'a' has a dictionary indexed by the worlds, each world index 'u' is a set of worlds 'a' cannot distinguish from 'u'.
- 'valuation' is a dictionary indexed by worlds, each world 'u' has a set of propositions valuated TRUE in 'u'.
- 'expectation' is a dictionary indexed by worlds, each world 'u' has an NFA which is the expectation in the world.

Note that, we are using NFA is expectation instead of regular expressions, although it does not add much to the complexity, since the conversion overhead between them is polynomial with respect to the size of NFA or regular expression.
The function "wordMC" in the program returns the set of worlds in the input model where the input formula stands TRUE. The function is a recursive function on the formula. The interesting case is of the diamond operator, where it calculates the residue model letter by letter of the word in the diamond iteratively. It uses the function "residue", which given an NFA and a letter, returns an NFA which is the residue of input NFA with the letter. It returns "None" (Python Datatype) if the residue is empty (δ).
The code has been implemented using `Python3` and uses the library `automathon`. The example of drone as well as traffic signal has been shown in the code.[2]

4.4 Related Work

Dynamic epistemic reasoning The model checking of standard epistemic logic (EL) is PTIME-complete [53]. Public Observation Logic (POL) is quite similar to Public announcement logic (PAL) [48]. When public announcements are performed, the number of possible worlds reduces, making the model checking of PAL still in PTIME [56] as for standard epistemic logic. When actions can be private, the model checking becomes PSPACE-complete for DEL with action models [6].

[2]The accompanying codes can be cloned or found here: https://github.com/itsmeavi/POLmc.git

In PAL, a possible world is equipped with a valuation, while in POL it is also equipped with a regular expression denoting the expectation in that world. In PAL, the public announcement is fully specified and its effect is deterministic. In POL, we may reason on sets of possible observations represented by regular expressions π. When these sets are singletons, we again obtain a PTIME upper bound (Theorem 34). In this sense, POL is close to Arbitrary PAL (APAL) [29] whose model checking is also PSPACE-complete [1]. In APAL, any epistemic formula can be announced: there are no expectations. However, in POL, we have to reason about the constraints between the possible expectations, and the set of observations (given by π). Our contribution can be reformulated as follows: we prove that (i) reasoning about these constraints can still be done in PSPACE, and, (ii) this reasoning is sufficiently involved for the model checking to be PSPACE-hard.

In POL, regular expressions are used to represent sets of observations, while van Benthem et al. [57] used regular expressions (actually, programs of Propositional dynamic logic (PDL) [28]) to denote epistemic relations. Charrier et al. [22] considered a logic for reasoning about protocols where actions are public announcements and not abstract observations as in POL: in this sense, POL is more general.

Epistemic temporal reasoning It is natural to describe computational behaviours with regular expressions. Finite-state controllers, i.e., automata are used to describe policies in planning [16]. Interestingly, Lomuscio and Michaliszyn [42] studied an epistemic logic where formulas are evaluated on intervals and the language provides Allen's operators on intervals: in their setting, the model is an interpreted system, and a propositional variable p is true in an interval I if the trace of I matches a given regular expression associated to p. In contrast, POL is not based on an already set-up model but relies on updates in a model. Bozzelli et al. [17] studied the complexity of the model checking of that logic depending on the restrictions on the allowed set of Allen's operators. Their framework is similar to ours because it relies on regular expressions but the approach is orthogonal to model updates and hence, to epistemic planning.

Epistemic planning As far as we know, epistemic planning frameworks (based on DEL [15], or the so-called MEP for Multi-agent Epistemic Planning [46]) all provide a mechanism for reasoning about preconditions and effects of actions. Expectations about others or about the world are not dealt with. However, Saffidine et al. [51] propose a collaborative setup for epistemic planning where each agent executes its own knowledge-based policy/program (KBP) while agents commonly know all the KBPs that are being executed, meaning that agents expect that the other agents follow their own KBP. On the contrary, in POL, observations

are public but expectations are in general not commonly known. Reasoning about some epistemic properties that are true after the execution of any kind of KBPs is undecidable, but is PSPACE-complete for star-free KBPs. The complexity is high for different reasons: the initial model is represented symbolically; observations are not already public, and KBPs may contain tests.

Strategic reasoning Usually in logics for strategic reasoning (e.g., alternating-time temporal logic [2], and strategy logic [23]), agents do not have expectations: an agent may consider all possible strategies for the others. Recently, Belnardinelli et al. [10] propose a variant of strategy logic (SL) where a player may know completely the strategy of another player. In contrast, in POL agents may have partial information about the expectations. In POL, agents also have higher-order knowledge about these expectations. In SL, strategies are abstract objects in the logical language whereas in POL, observations are represented as composite structures that the agents can reason about, similar to the work on games and strategies presented in [31]. In this sense, POL can be seen as EL extended with PDL operators.

4.5 Conclusion

In this chapter and the last, we showed that the model checking for POL is PSPACE-complete. Such complexity studies were left open in [62]. We also identified more tractable fragments (see Figure 4.1) of POL. Finally, we discussed the applicability of our study in verifying various features of interactive systems related to epistemic planning. A discussion on implementation is also provided.

Many interesting features of such interactive systems remain to be investigated : private observations, like in DEL with action models [60]; dynamic aspects (e.g., changing expectations); richer languages of expectations (e.g., context-free grammars for expectations), among others. Symbolic model checking can be considered as well following the trends of [58] and [22].

This paper also opens up a research avenue for developing variants and extensions for reasoning about expectations and observations that can be expressive enough with reasonable complexities for the model checking problem.

To sum up, POL mixes epistemic logic and language theory for modelling mechanisms of social intelligent agents, and the current investigations on model checking set it up as a useful tool in building social software for AI.

We investigate model checking for EPL, an extension of POL, also proposed in [62], in a later chapter. Next, we move on to the satisfiability problem for POL.

Chapter 5

Star-Free Satisfiability

5.1 Introduction

In this chapter, and the next, we look into the satisfiability problem for POL. Earlier, we were looking into verifying properties with respect to a specific model. Unlike model-checking, satisfiability is a tool to check properties in a class of models. Hence, even the applications of epistemic planning can be extended here as well, for example, whether an epistemic goal can be achieved in a class of scenarios that satisfy certain conditions.

Recall the example 4 introduced in chapter 2 (Figure 5.1) involving the cleaner bot moving for power supply in one corner and debris on another. The example concerns certain rules that we follow in our daily life, they deal with situations where agents expect certain observations at certain states based on some pre-defined *protocols*, viz. the bot mechanism in the example. They get to know about the actual situation by observing certain actions which agree with their expectations corresponding to that situation. POL does not deal with the protocols themselves, but the effect those protocols have in our understanding of the world around us in terms of our expectations and observations. In [18] as well as chapters 3 and 4, we have investigated the computational complexity of the model-checking problem of different fragments of POL, and in this chapter, we will deal with the computational complexity of the satisfaction problem of various proper fragments of POL (cf. Figure 5.2). We will show how certain simple fragments of POL give rise to high complexity with respect to their computational behaviour.

Recall that the logic POL^- is the Star-Free **fragment of** POL, that is, it is the set of formulas in which the π's do not contain any Kleene star $*$. A more restricted version is the Word **fragment of** POL^-, where π's are words. We consider both the **single-agent word fragment of** POL^-, and **multi-agent word fragment of** POL^-. Furthermore, we consider

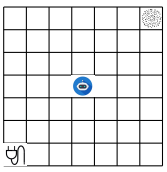

Fig. 5.1 A robotic vacuum cleaner on the floor (in the middle of the grid). The power source is at bottom left, whereas the debris-disposal area is at top right.

	Single-agent	Multi-agent
Word POL^-	NP-complete	PSPACE-complete
POL^-	PSPACE-Hard	NEXPTIME-complete

Fig. 5.2 Complexity results of satisfiability of various fragments of POL^-.

single-agent POL^-, and **multi-agent** POL^- (full POL^-). The main theorem we prove in this chapter is:

Theorem 36. *Satisfiability problem of* POL^- *is* NEXPTIME*-complete.*

Furthermore, we prove similar theorems as well for fragments. The list of results in this chapter is listed in Figure 5.2. To prove the complexity results of some of these fragment(s) of POL we use a translation to Public announcement logic (PAL) [48], whereas, for other fragment(s), a tableau method is utilized where the tableau rules provide a mix of modal logic reasoning and computations of language theory residuals.

Outline. In Section 5.2, we describe an application of the satisfiability problem of POL^-. In Section 5.3 we present a NEXPTIME algorithm for POL^- using the tableau method. In Section 5.4, we prove that POL^- is in NEXPTIME-Hard. In section 5.5, we present the complexity results for various fragments of POL^-. Section 5.6 discusses related work, and Section 5.7 concludes the chapter.

5.2 An application

Let us now consider a scenario which can be aptly described using the satisfiability problem of POL^-. We go back to the cleaning bot example (Figure 5.1) introduced earlier (Example 4). Let Alice be agent a and Bob be agent b. Figure 5.3 shows an epistemic expectation model of this Example 4. Unlike model-checking, which verifies a property with respect to a

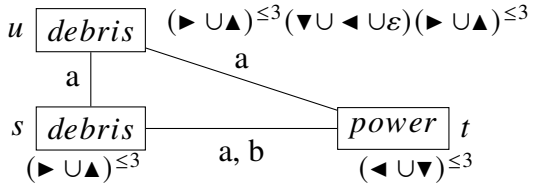

Fig. 5.3 Model describing the initial knowledge of the two agents Alice and Bob about the expectation of the *vbot*.

specific model, one of many utilities of satisfiability is checking whether a property holds in every models from a class of models represented by some formula.

Suppose the *vbot* is moving towards the power source without making any error. Evidently, the possibilities considered by the agents, based on the information available to them are given as follows:

- Possibilities considered by Alice who has the information about the glitch in the bot:

$$\hat{K}_a debris \wedge \hat{K}_a \langle \blacktriangleleft + \blacktriangledown \rangle debris \wedge \hat{K}_a power$$

- Possibilities considered by Bob who is not aware of the glitch in the bot:

$$\hat{K}_b debris \wedge \hat{K}_b power$$

Now, we model the *expectations* as follows: Consider the expression, $\pi_n^p = (\blacktriangledown + \blacktriangleleft)^n$ that represents a sequence of moves of length n the bot can make to get to to the power source without any error. We use a formula P_n to express the following: As long as the bot is observed to make n many moves towards the power source, reaching it is still a possibility.

$$\begin{aligned} P_n = &(\langle \blacktriangleleft \rangle \top \wedge \langle \blacktriangledown \rangle \top) \\ &\wedge [\pi_1^p](\langle \blacktriangleleft \rangle \top \wedge \langle \blacktriangledown \rangle \top) \\ &\wedge [\pi_2^p](\langle \blacktriangleleft \rangle \top \wedge \langle \blacktriangledown \rangle \top) \ldots \\ &\wedge [\pi_n^p](\langle \blacktriangleleft \rangle \top \wedge \langle \blacktriangledown \rangle \top) \end{aligned}$$

The first conjunct of P_n translates to move towards the power source, a move towards down or left can be observed. The second conjunct translates to the following: after the observation of a single left or down movement, another left or down movement can be observed. The other conjuncts can be described similarly.

For the scenario described in the introduction, we can consider P_n to create a formula where n is at most 3, without an error. Let us denote such a formula by ψ_p. Similarly, a formula can express the movement towards debris-disposal with at most one error and with no error as ψ_{de} and ψ_d, respectively. A situation where the bot is moving towards the power source without any error, but a considers the possibility of moving towards debris-disposal with an error can be expressed as $\hat{K}_a\psi_{de} \wedge \psi_p$. Similarly, a formula can be considered for modelling the expected observation when both the agents consider the possibility of the bot moving towards debris-disposal area without an error: $\hat{K}_a\psi_d \wedge \hat{K}_b\psi_d$. We call the (finite) set of all such formulas, Γ_p. Similarly, we can construct a set Γ_{de} of formulas, when the bot can make an error while going towards debris-disposal area or Γ_d when it is moving towards the debris-disposal without any error.

Suppose we want to conclude the following in the current scenario: After one wrong move, b knows that the bot is not moving towards debris-disposal, but a still considers the possibility. The formula, $INFO_{ab}$, say, turns out to be

$$\langle \blacktriangledown + \blacktriangleleft \rangle (K_b power \wedge \hat{K}_a debris)$$

The actual scenario is that the bot is indeed moving towards $power$. Hence, to check whether $INFO_{ab}$ can be concluded in this scenario, a satisfiability solver for POL^- can check the (un)satisfiability of the formula

$$\neg((\bigwedge_{\psi \in \Gamma_p} \psi) \rightarrow INFO_{ab})$$

Note that, checking whether the above formula is valid implies $INFO_{ab}$ is valid property that holds in every model that satisfies $\bigwedge_{\psi \in \Gamma_p} \psi$. Moreover, the model in Figure 5.3 is one such model where the formula holds from the possibility t.

5.3 Algorithm for the Satisfiability Problem of POL^-

In this section, we design a proof system using tableau method to prove satisfiability of POL^-.

A term in a tableau proof is of the form $(\sigma \ w \ \psi) \mid (\sigma \ w \ \checkmark) \mid (\sigma, \sigma')_i$, where $i \in Agt$. The σ is called a state label that represents a state in the model, $w \in \Sigma^*$ is a word over a finite alphabet and ψ is a formula in POL^-.

The term $(\sigma \ w \ \psi)$ represents the fact that the state labelled by σ survives after the model is projected on the word w, and after projecting on w, ψ holds true in the state corresponding to σ.

The term $(\sigma \;\; w \;\; \checkmark)$ represents the fact that the state labelled by σ survives after the model is projected on word w.

The term $(\sigma_1, \sigma_2)_i$ represents in the model, the states represented by σ_1 and σ_2 should be indistinguishable for the agent $i \in Agt$, where Agt is a finite set of agents.

For space reasons, the term $(\sigma_1, \sigma_2)_{i \in Agt}$ stands for the set of terms $\{(\sigma_1, \sigma_2)_i \mid i \in Agt\}$.

Without loss of generality, the formula φ is assumed to be in Negative Normal form, the syntax of which is as follows:

$$\varphi := \top \mid p \mid \neg p \mid \psi \vee \chi \mid \psi \wedge \chi \mid$$
$$\hat{K}_i\psi \mid K_i\psi \mid \langle\pi\rangle\psi \mid [\pi]\psi$$

For the satisfiability results, in this chapter and in the next, we need a vital notion of *Fischer-Ladner closure*.

Definition 44. ***Fischer-Ladner Closure*** *The Fisher-Ladner closure of a formula φ , denoted $FL(\varphi)$, is the smallest set containing φ and satisfying*

- *if $\psi \in FL(\varphi)$ and ψ not starting with $\neg$ then $\neg\psi \in FL(\varphi)$*
- *if $\neg\psi, K_i\psi, \hat{K}_i\psi, [\pi]\psi$ or $\langle\pi\rangle\psi$ in $FL(\varphi)$ then $\psi \in FL(\varphi)$*
- *if $\psi \wedge \chi$ or $\psi \vee \chi$ are in $FL(\varphi)$ then $\psi, \chi \in FL(\varphi)$*
- *if $\langle\pi_1;\pi_2\rangle\psi \in FL(\varphi)$ then $\langle\pi_1\rangle\langle\pi_2\rangle\psi \in FL(\varphi)$*
- *if $\langle\pi_1 + \pi_2\rangle\psi \in FL(\varphi)$ then $\langle\pi_1\rangle\psi, \langle\pi_2\rangle\psi \in FL(\varphi)$*
- *if $\langle\pi^\star\rangle\psi \in FL(\varphi)$ then $\langle\pi\rangle\langle\pi^\star\rangle\psi \in FL(\varphi)$*
- *if $[\pi_1;\pi_2]\psi \in FL(\varphi)$ then $[\pi_1][\pi_2]\psi \in FL(\varphi)$*
- *if $[\pi_1 + \pi_2]\psi \in FL(\varphi)$ then $[\pi_1]\psi, [\pi_2]\psi \in FL(\varphi)$*
- *if $[\pi^\star]\psi \in FL(\varphi)$ then $[\pi][\pi^\star]\psi \in FL(\varphi)$*

In simple words, Fischer Ladner closure of a formula talks of all the subformulas that are to be considered in a satisiability argument in order to satisfy the original formula. Note that if $[\pi]\psi \in FL(\varphi)$, we have $[\pi\backslash a]\psi \in FL(\varphi)$. For example, suppose $[b^\star ab]\psi \in FL(\varphi)$. Note that $b^\star ab\backslash a = b$. By definition **??**, $[b^\star][ab]\psi \in FL(\varphi)$ implies $[ab]\psi, [b][b^\star][a]\psi \in FL(\varphi)$. Since $[ab]\psi \in FL(\varphi)$, therefore $[a][b]\psi \in FL(\varphi)$, which finally implies $[b]\psi \in FL(\varphi)$. The same is also true for the diamond formulas ($\langle\pi\rangle\psi$) as well.

Example 16. *Consider* $\varphi = \langle(\blacktriangleright + \blacktriangleleft)^\star\rangle power$. *Hence* $FL(\varphi) = \{\varphi, \neg\varphi, power, \langle\blacktriangleright + \blacktriangleleft\rangle\langle(\blacktriangleright + \blacktriangleleft)^\star\rangle power, \ldots\}$.

Observation 37. *[35] Given* φ, $|FL(\varphi)| \leq O(|\varphi|)$.

Given a formula we denote by φ, $FL(\varphi)$ the Fischer-Ladner Closure of φ, (for more details, see [35]).

5.3.1 The Tableau Rules

The tableau rules for this fragment have been shown in Figure 5.4. Here an inference rule looks like this: $\frac{A}{C_1|C_2|\ldots|C_n}$.

Here each C_i and A is a set of tableau terms. The C_is are called consequences, A is the antecedent. Intuitively the rule is interpreted as "If all the terms in A are true, then all the terms in at least one of C_i's are true".

In Figure 5.4, the left column is the rule name and the right column is the rule. For example, the Box Project rule states that "The state labelled by σ survives after projection on word w and it satisfies $[\pi]\psi$ (the term $(\sigma \;\; w \;\; [\pi]\psi)$ in the numerator) and σ still survives a further projection on letter a(the second term $(\sigma \;\; wa \;\; \checkmark)$ in the numerator) then after further projection on a, $[\pi\backslash a]\psi$ should hold true in the state labelled by σ (the term $(\sigma \;\; wa \;\; [\pi\backslash a]\psi)$ in the denominator).". Recall $\pi\backslash a$ denotes the residual of π by a (see chapter 2).

Similarly, the Diamond Project rule says that if a certain state σ, under some word projection w has to satisfy $\langle a\rangle\psi$, then that state σ has to survive projection on wa and also satisfy ψ under the same projection.

A tableau proof can be assumed a tree $\mathcal{T}$. Each node of the tree is a set of tableau terms Γ. An inference rule can be applied in the following way:

If $A \subseteq \Gamma$ and C_i's are not in Γ, the children of Γ are $\Gamma \cup C_i$ for each $i \in [n]$.

When no rules can be applied on a Γ, we say Γ is saturated (leaf node in the proof tree).

If $\bot \in \Gamma$, we say that branch is **closed**. If all the branches of the proof tree is **closed**, we say the **tableau is closed**, else is open.

Given a POL^- formula φ, we start with $\Gamma = \{(\sigma \;\; \epsilon \;\; \varphi), (\sigma \;\; \epsilon \;\; \checkmark)\} \cup \{(\sigma,\sigma)_i, i \in Agt\}$.

Example 17. *Suppose we aim at deciding whether*

$$\varphi := \hat{K}_i\langle a\rangle p \wedge \langle a\rangle K_i \neg p$$

is satisfiable or not. For simplicity we suppose there is a single agent i. Here are the terms added to the set of terms:

Propositional Rules

Clash rule $\dfrac{(\sigma \quad w \quad p),\ (\sigma \quad w \quad \neg p)}{\bot}$

AND rule $\dfrac{(\sigma \quad w \quad \psi \wedge \chi)}{(\sigma \ w \ \psi),(\sigma \ w \ \chi)}$

OR rule $\dfrac{(\sigma \quad w \quad \psi \vee \chi)}{(\sigma \ w \ \psi) \ | \ (\sigma \ w \ \chi)}$

Knowledge Rules

Knowledge $\dfrac{(\sigma \quad w \quad K_i\psi),\ (\sigma' \quad w \quad \checkmark),\ (\sigma,\sigma')_i}{(\sigma' \ w \ \psi)}$

Possibility $\dfrac{(\sigma \quad w \quad \hat{K}_i\psi)}{(\sigma,\sigma_n)_i,(\sigma_n \ w \ \checkmark),(\sigma_n \ w \ \psi),(\sigma_n,\sigma_n)_{i\in Agt}}$

Transitivity $\dfrac{(\sigma,\sigma'')_i,\ (\sigma'',\sigma')_i}{(\sigma,\sigma')_i}$

Symmetry $\dfrac{(\sigma',\sigma)_i}{(\sigma,\sigma')_i,\ i \in Agt}$

Diamond and Box Rules

Diamond Decompose $\dfrac{(\sigma \quad w \quad \langle\pi\pi'\rangle\psi)}{(\sigma \ w \ \langle\pi\rangle\langle\pi'\rangle\psi)}$

Diamond ND Decompose $\dfrac{(\sigma \quad w \quad \langle\pi_1+\pi_2\rangle\psi)}{(\sigma \ w \ \langle\pi_1\rangle\psi) \ | \ (\sigma \ w \ \langle\pi_2\rangle\psi)}$

Diamond Project $\dfrac{(\sigma \quad w \quad \langle a\rangle\psi)}{(\sigma \ wa \ \checkmark),(\sigma \ wa \ \psi)}$

Box Project $\dfrac{(\sigma \quad w \quad [\pi]\psi),\ (\sigma \quad wa \quad \checkmark)}{(\sigma \ wa \ [\pi\backslash a]\psi)}$

Empty Box $\dfrac{(\sigma \quad w \quad [\epsilon]\psi)}{(\sigma \ w \ \psi)}$

Survival Rules

Constant Valuation Up $\dfrac{(\sigma \quad w \quad p)}{(\sigma \ \epsilon \ p)} \quad \dfrac{(\sigma \quad w \quad \neg p)}{(\sigma \ \epsilon \ \neg p)}$

Survival Chain $\dfrac{(\sigma \quad wa \quad \checkmark)}{(\sigma \ w \ \checkmark)}$

Fig. 5.4 Tableau rules. σ is any state symbol, w is any word, p is any propositional variable, i is any agent, π is any regular expression, a is any letter.

1. $(\sigma \ \epsilon \ \varphi),\ (\sigma \ \epsilon \ \checkmark),\ (\sigma,\sigma)_i$ *(initialization)*
2. $(\sigma \ \epsilon \ \hat{K}_i\langle a\rangle p),\ (\sigma \ \epsilon \ \langle a\rangle K_i\neg p)$ *by AND rule*
3. $(\sigma' \ \epsilon \ \langle a\rangle p),(\sigma' \ \epsilon \ \checkmark),(\sigma,\sigma')_i,(\sigma',\sigma')_i$ *by Possibility rule*
4. $(\sigma',\sigma)_i$ *by Symmetry rule*

5. $(\sigma' \ a \ p), (\sigma' \ a \ \checkmark)$ *by Diamond Project on 2*

6. $(\sigma \ a \ \checkmark), (\sigma \ a \ K_i \neg p)$ *by Diamond Project on 2*

7. $(\sigma' \ a \ \neg p)$ *by Knowledge rule on 3, 5, 6*

8. $\bot$ *by Clash rule on 5,7*

As we obtain $\bot$, the formula φ is not satisfiable (by the upcoming Theorem 38).

5.3.2 Soundness and Completeness of the Tableau Rules

In this section, we provide the soundness and completeness proof of the Tableau method for the satisfiability of POL^-

Theorem 38. *Given a formula φ, if φ is satisfiable, then the tableau for $\Gamma = \{(\sigma \in \varphi), (\sigma \in \checkmark), (\sigma, \sigma)_{i \in Agt}\}$ is open.*

Proof. We prove that if φ is satisfiable then there exists a subtree rooted at some child Γ_c of the root Γ in $\mathcal{T}$ which is open using induction on the depth of the tableau tree $\mathcal{T}$. Let the pointed epistemic model that satisfy φ be $\mathcal{M}, s$.

Base Case. Let the base case be $|\Gamma| = 2 + |Agt|$. Since we start from $\{(\sigma \in \varphi), (\sigma, \sigma)_{i \in Agt}, (\sigma \in \checkmark)\}$. Hence this implies $\varphi = l$ or $\varphi = [a]\psi$, where l is a literal (a positive propositional letter or a negation of it). Hence the tableau remains open since no $\bot \in \Gamma$.

Induction Hypothesis. Let the statement be true for any tableau tree $\mathcal{T}$ of depth at most n.

Inductive Step. Consider the tableau tree $\mathcal{T}$ rooted at $\Gamma = \{(\sigma \in \varphi), (\sigma, \sigma), (\sigma \in \checkmark)\}$ of depth at most $n+1$. Now we go case by case with φ:

- $\varphi = \psi \wedge \chi$: We apply AND rule, and hence $\Gamma' = \Gamma \cup \{(\sigma \in \psi), (\sigma \in \chi)\}$, which is a child rooted at Γ. Since $\mathcal{M}, s \vDash \varphi$, by definition $\mathcal{M}, s \vDash \psi$ and $\mathcal{M}, s \vDash \chi$. By IH, tableau tree rooted at Γ' is open, which suggestes, $\mathcal{T}$ is open.

- $\varphi = \psi \vee \chi$: We apply OR rule and hence we get two children, $\Gamma_1 = \Gamma \cup \{(\sigma \in \psi)\}$ and $\Gamma_2 = \Gamma \cup \{(\sigma \in \chi)\}$. Since $\mathcal{M}, s \vDash \psi \vee \chi$, hence $\mathcal{M}, s \vDash \psi$ or $\mathcal{M}, s \vDash \chi$. By IH, one of the sub tableau tree rooted at Γ_1 or Γ_2 will be open, hence implying $\mathcal{T}$ to be open.

- $\varphi = K_i\psi$: By applying the Knowledge rule, $\Gamma' = \Gamma \cup \{(\sigma' \in \psi) \mid \{(\sigma, \sigma')_i, (\sigma' \in \checkmark)\} \subseteq \Gamma\}$. Since $\mathcal{M}, s \vDash K_i\psi$, hence for every $s' \in \mathcal{M}$ such that $s \sim_i s'$, $\mathcal{M}, s' \vDash \psi$. By IH, the tableau subtree rooted at Γ' is open.

- $\varphi = \hat{K}_i\chi$: By applying the Possibility rule, $\Gamma' = \Gamma \cup \{(\sigma_n, \sigma_n)_{i\in Agt}, (\sigma_n \ \epsilon \ \checkmark), (\sigma_n \ \epsilon \ \chi)\} \cup \{(\sigma, \sigma_n)_i, (\sigma_n, \sigma)_i\} \cup \{(\sigma', \sigma_n)_i, (\sigma_n, \sigma')_i \mid \{(\sigma' \ \epsilon \ \checkmark), (\sigma, \sigma')_i\} \subseteq \Gamma_l\}$. Since, $\mathcal{M}, s \vDash \hat{K}_i\chi$, hence there is an $s_n \in \mathcal{M}$ such that $\mathcal{M}, s_n \vDash \chi$, also $s_n \sim s'$ for every $s' \in \mathcal{M}$ since $\sim$ is an equivalence relation. Hence by IH, the sub tableau tree rooted at Γ' is open.

- $\varphi = \langle\pi\pi'\rangle\psi$: Hence by the Diamond Decomposition rule $\Gamma' = \Gamma \cup \{(\sigma \ \epsilon \ \langle\pi\rangle\langle\pi'\rangle\psi)\}$. Since $\mathcal{M}, s \vDash \langle\pi\pi'\rangle\psi$, hence $\mathcal{M}|_{w_*}, s \vDash \psi$, for some $w_* \in \mathcal{L}(\pi\pi')$ which also suggests, there is a $w \in \mathcal{L}(\pi)$ and a $w' \in \mathcal{L}(\pi')$ such that $w_* = ww'$. Hence $\mathcal{M}|_w, s \vDash \langle\pi'\rangle\psi$, which implies $\mathcal{M}, s \vDash \langle\pi\rangle\langle\pi'\rangle\psi$. By IH, the tableau for Γ' is open.

- $\varphi = \langle a\rangle\psi$. Hence now by Projection rule, $\Gamma' = \Gamma \cup \{(\sigma \ a \ \checkmark), (\sigma \ a \ \psi)\}$. Since $\mathcal{M}, s \vDash \langle a\rangle\psi$, therefore $s \in \mathcal{M}|_a$ and $\mathcal{M}|_a, s \vDash \psi$. And hence by IH, tableau tree rooted at Γ' is open.

- $\varphi = [\pi]\psi$. Say $(\sigma \ a \ \checkmark)$ is added for some diamond formula term, say of the form $(\sigma \ \epsilon \ \langle a\rangle\psi')$, hence the proof has added $(\sigma \ a \ [\pi\backslash]\psi)$. By assumption $\mathcal{M}, s \vDash \langle a\rangle\psi'$, and hence $s \in \mathcal{M}|_a$. Therefore $\mathcal{M}|_a, s \vDash [\pi\backslash a]\psi$, hence the proof starting from terms $\{(\sigma, \sigma), (\sigma \ a \ \checkmark), (\sigma \ a \ [\pi\backslash a]\psi)\}$ will remain open.

  For the other case $(\sigma \ a \ \checkmark)$ is not added, hence $s \notin \mathcal{M}|_a$, therefore the tableau remains open.

□

Theorem 39. *Given a formula φ, if the tableau for $\Gamma = \{(\sigma \ \epsilon \ \varphi), (\sigma \ \epsilon \ \checkmark), (\sigma, \sigma)_{i\in Agt}\}$ is open, then φ is satisfiable.*

Proof. Since by assumption, the tableau for $\Gamma = \{(\sigma \ \epsilon \ \varphi), (\sigma \ \epsilon \ \checkmark), (\sigma, \sigma)_{i\in Agt}\}$ is open, there exists a branch in the tableau tree where in the leaf node there is a set of terms Γ_l such that it is saturated and $\bot \notin \Gamma_l$.

For the purpose of this proof, let us define a relation over the words $\bar{w}$ that appears in Γ_l. For any two word $\bar{w}_1$ and $\bar{w}_2$ that appears in Γ_l, $\bar{w}_1 \leq_{pre} \bar{w}_2$ if and only if $\bar{w}_1 \in \mathcal{L}(Pre(\bar{w}_2))$. Now, this relation is reflexive ($\bar{w}_1 \in \mathcal{L}(Pre(\bar{w}_1))$), asymmetric (if $\bar{w}_1 \in \mathcal{L}(Pre(\bar{w}_2))$ and $\bar{w}_2 \in \mathcal{L}(Pre(\bar{w}_1))$ then $\bar{w}_1 = \bar{w}_2$) and transitive (if $\bar{w}_1 \in \mathcal{L}(Pre(\bar{w}_2))$ and $\bar{w}_2 \in \mathcal{L}(Pre(\bar{w}_3))$ then $\bar{w}_1 \in \mathcal{L}(Pre(\bar{w}_3))$). Hence this relation creates a partial order among all the words occurring in Γ_l. We also denote $\bar{w_1} <_{pre} \bar{w_2}$ to interpret the fact that $\bar{w_1} \leq_{pre} \bar{w_2}$ and $\bar{w_1} \neq \bar{w_2}$.

Now we create a model $\mathcal{M} = \langle W, \{R_i\}_{i\in Agt}, V, Exp\rangle$ out of Γ_l and prove that φ is satisfied by some state in the model.

- $W = \{s_\sigma \mid \sigma$ is a distinct label in the terms occuring in $\Gamma_l\}$

- $R_i = \{\{s_{\sigma_1}, s_{\sigma_2}\} \mid (\sigma_1, \sigma_2)_i \in \Gamma_l\}$
- $V(s_\sigma) = \{p \mid (\sigma \ \epsilon \ p) \in \Gamma_l\}$
- $Exp(s_\sigma) = \sum_{w \in \Lambda_\sigma} w$, where $\Lambda_\sigma = \{w \mid (\sigma \ \ w \ \ \checkmark) \in \Gamma_l$ and $\nexists w' : ((\sigma \ \ w' \ \ \checkmark) \in \Gamma_l$ and $w <_{pre} w')\}$

Note that, the new state label σ_n is only created in the possibility rule, with a reflexive relation on itself. Now consider the set $R' = \{(\sigma, \sigma') \mid \{(sigma \ \ w \ \ \checkmark), (\sigma' \ \ w' \ \ \checkmark)\} \subseteq \Gamma_l\}$. Hence this can be considered a binary relation over the set of all distinct σ that occurs in Γ_l. When a σ' is created by the possibility rule, it is reflexive. Also by the relation rules, they are made symmetrically and transitively related to every other label that has been previously there. Hence R' is an equivalence relation, hence making R_i in the model an equivalence relation.

Now, Theorem 39 follows from the following two claims, the proofs of which we present later.

Claim 40. *If* $(\sigma \ \ w \ \ \checkmark) \in \Gamma_l$ *then* s_σ *survives in* $\mathcal{M}|_w$.

Claim 41. *For any word* w *that occurs in* Γ_l*, any label* σ *and any formula* ψ*, if* $(\sigma \ \ w \ \ \psi) \in \Gamma_l$ *and* $(\sigma \ \ w \ \ \checkmark) \in \Gamma_l$ *then* s_σ *survives in* $\mathcal{M}|_w$ *and* $\mathcal{M}|_w, s_\sigma \vDash \psi$.

Proof of Claim 40. We induct on the size of $|w|$.

Base Case. Let $|w| \leq 1$. Hence $w \in \{\epsilon\} \cup \Sigma$. Since $\Gamma \subseteq \Gamma_l$ and $(\sigma \ \ \epsilon \ \ \checkmark)$, and s_σ is in $\mathcal{M}|_\epsilon = \mathcal{M}$.

For the case $w = a$, where $a \in \Sigma$: There exists a word w' that occurs in a term in Γ_l labelled by σ such that $w \in \mathcal{L}(Pre(w'))$ and there is no other word bigger than w' such that w' is in its prefix, since the proof is on finite words and formula, the proof terminates. Hence by definition of $Exp(s_\sigma)$, s_σ survives in $\mathcal{M}|_a$.

Induction Hypothesis. Assume the statement to be true for $|w| = n$.

Inductive Step. Consider the case where $|w| = n + 1$.

By assumption, $(\sigma \ \ w \ \ \checkmark) \in \Gamma_l$. Hence by the fact that Γ_l is saturation and by the rule "Survival Chain", there is $(\sigma \ \ w' \ \ \checkmark) \in \Gamma_l$, where $w = w'a$ for some $a \in \Sigma$. Hence by IH, the result follows that s_σ survives in $\mathcal{M}|_{w'}$.

Now, by termination, there are finite many unique words occurring in Γ_l. Clearly, $w' \leq_{pre} w$. Since there are finite many words, there is a w_*, which is of maximum size such that $w \leq_{pre} w_*$ and $(\sigma \ \ w_* \ \ \checkmark) \in \Gamma_l$. Hence $w_* \in \Lambda_\sigma$ in the definition of Exp of the model. Therefore $w_* \in \mathcal{L}(Exp(s_\sigma))$ and since $w' \leq_{pre} w \leq_{pre} w_*$, s_σ survives in $\mathcal{M}|_{w'}$, hence s_σ shall survive in $\mathcal{M}|_w$. □

Proof of Claim 41. Naturally, we shall induct upon the size of ψ.

Base Case. Let ψ is of the form p or $\neg p$. By the definition of the function V for the model and the previous proof, the statement stands true.

Induction Hypothesis. Let us consider the statement is true for any ψ such that $|\psi| < n'$ for some n'.

Inductive Step. We prove for $|\psi| = n'$. Again, we go case by case on the syntax of ψ.

- $\psi = \hat{K}_i\chi$. Since Γ_l is saturated, by the rule of possibility, $\{(\sigma' \ w \ \chi), (\sigma, \sigma')_i, (\sigma' \ w \ \checkmark)\} \subseteq \Gamma_l$. By IH on the subformula χ, the definition of the model, the proof of the previous statement, and the rule "survival chain", $s_{\sigma'}$ survives in $\mathcal{M}|_w$ and $\mathcal{M}|_w, s_{\sigma'} \vDash \chi$. Also by definition, $\{s_\sigma, s_{\sigma'}\} \in R_i$, hence proving $\mathcal{M}|_w, s_\sigma \vDash \hat{K}_i\chi$.

- $\psi = K_i\chi$. Since Γ_l is saturated, and by previous statement $s_{\sigma'}$ is surviving for every $(\sigma' \ w \ \checkmark)$, by the rule of knowledge $(\sigma' \ w \ \chi) \in \Gamma_l$ for every $(\sigma, \sigma')_i$. Hence by IH on subformula, $\mathcal{M}|_w, s_{\sigma'} \vDash \chi$ for every σ' such that $\{\sigma, \sigma'\} \in R_i$.

- $\psi = \langle \pi + \pi' \rangle \chi$. Since Γ_l is saturated, hence by the ND Decomposition, either the term $(\sigma \ w \ \langle \pi \rangle \chi) \in \Gamma_l$ or $(\sigma \ w \ \langle \pi' \rangle \chi) \in \Gamma_l$. By IH, $\mathcal{M}|_w, s_\sigma \vDash \langle \pi \rangle \chi$ or $\mathcal{M}|_w, s_\sigma \vDash \langle \pi' \rangle \chi$ and hence $\mathcal{M}|_w, s_\sigma \vDash \langle \pi + \pi' \rangle \chi$.

- $\psi = \langle \pi\pi' \rangle \chi$. Since Γ_l is saturated, hence $(\sigma \ w \ \langle \pi \rangle \langle \pi' \rangle \chi) \in \Gamma_l$. By IH, since $\langle \pi \rangle \langle \pi' \rangle \chi \in FL(\psi)$, hence $\mathcal{M}|_w, s_\sigma \vDash \psi$.

- $\psi = \langle a \rangle \chi$. Note that we don't consider a general word w' in the diamond as given $w' = aw''$, a formula $\langle w' \rangle \chi$ is satisfiable if and only if $\langle a \rangle \langle w'' \rangle \chi$ is satisfiable.

- $\psi = [\pi]\chi$. Let us consider $(\sigma \ wa \ \checkmark) \in \Gamma_l$ for some $a \in \Sigma$. Hence by the proof of the first statement, $s_\sigma \in \mathcal{M}|_{wa}$. Also $|\mathcal{L}(\pi)| < |\mathcal{L}(\pi \backslash a)|$. Hence by induction on the size of formula $\mathcal{M}|_{wa}, s\sigma \vDash [\pi \backslash a]\chi$ which implies $\mathcal{M}|_w, s_\sigma \vDash [\pi]\chi$. □

This completes the proof of Theorem 39 □

5.3.3 A NEXPTIME Upper Bound

Now we design an algorithm based on tableau and prove existence of an algorithm that takes non-deterministically exponential steps with respect to the size of φ. Now given a φ, we now create a tree of nodes that contains terms of the form (w, ψ) and $(w, \checkmark)$, where $w \in \Sigma^*$ is a word that is occuring in tableau, and ψ is a formula in $FL(\varphi)$. Each node T_σ refers to a state label σ in tableau, a term of the $(w, \psi) \in T_\sigma$ intuitively translates to in the state corresponding to σ, after projecting model on w, the state survives and there ψ is satisfied, and hence refers

to the term $(\sigma \;\; w \;\; \psi)$ in tableau. Similarly, $(w, \checkmark) \in T_\sigma$ means state corresponding to σ survives after projection on w, and hence refers to the term $(\sigma \;\; w \;\; \checkmark)$ in the tableau. The tableau tree created, we call it $\mathcal{T}_\mathsf{P}$

For this algorithm, we change the definition of saturation and unsaturation a bit from the earlier definition. We say T_σ is **unsaturated** against a rule R iff there is a term in $(w, \psi) \in T_\sigma$ or $(w, \checkmark) \in T_\sigma$, such that $(\sigma \;\; w \;\; \psi)$ or $(\sigma \;\; w \;\; \checkmark)$ lies in the numerator of R but there is no denominator $(\sigma \;\; w \;\; \psi')$ of R such that (w, ψ') is in T_σ, similar for terms like $(w, \checkmark)$. We call the term (w, ψ) or $(w, \checkmark)$ here to be the **reason for unsaturation**.

We saturate the rules carefully such that each node in the tree corresponds to a single state in the model. This technique is well studied in [32].

Algorithm 14 StarFree-SAT

procedure StarFree-SAT(φ)
  $T_{\sigma_0} \leftarrow \{(\epsilon, \varphi), (\epsilon, \checkmark)\}$
  T_{σ_0} is the root of tree $\mathcal{T}_P$.
  while there is a leaf of $\mathcal{T}_P$ that satisfies one of the following conditions **do**
    if $\bot \notin T_\sigma$ and T_σ is unsaturated against Propositional and Survival Rules **then**
      Let (w, ψ) or $(w, \checkmark)$ be the reason for unsaturation against the above rules.
      if $\psi = \psi_1 \wedge \psi_2$ **then** $T_\sigma = T_\sigma \cup \{(w, \psi_1), (w, \psi_2)\}$
      else if $\psi = \psi_1 \vee \psi_2$ **then** non deterministically choose ψ_1 or ψ_2 and
        $T_\sigma = T_\sigma \cup \{(w, \psi_1)\}$ or $T_\sigma = T_\sigma \cup \{(w, \psi_2)\}$ as per choice.
      else if $\psi = l$ where l is a literal **then** $T_\sigma = T_\sigma \cup \{(w, [a]l), (\epsilon, l)\}$.
      else if $\{(w, \psi), (w, \neg\psi)\} \subseteq T_\sigma$ **then** $T_\sigma = T_\sigma \cup \{\bot\}$.
      else if $(w, \checkmark) \in T_\sigma$ **then** $T_\sigma = T_\sigma \cup \{(w', \checkmark) \mid w' \in Pre(\mathcal{L}(w))\}$.
    else if $\bot \notin T_\sigma$ and T_σ is propositionally saturated but unsaturated against
      Box and Diamond Rules **then**
      Let the reason for unsaturation be (w, ψ).
      If $\psi = \langle \pi_1 \pi_2 \rangle \psi'$ then $T_\sigma = T_\sigma \cup \{(w, \langle \pi_1 \rangle \langle \pi_2 \rangle \psi')\}$.
      If $\psi = \langle \pi_1 + \pi_2 \rangle \psi'$ then nondeterministically choose a π_i, where $i \in \{1, 2\}$
      and make $T_\sigma = T_\sigma \cup \{(w, \langle \pi_i \rangle \psi')\}$
      If $\psi = \langle a \rangle \psi'$ then $T_\sigma = T_\sigma \cup \{(wa, \checkmark), (wa, \psi'), (w, [a]\psi')\}$.
      If $\psi = [\pi]\psi'$ and $(wa, \checkmark) \in T_\sigma$ then $T_\sigma = T_\sigma \cup \{(wa, [\pi \backslash a]\psi')\}$
    else if $\bot \notin T_\sigma$ and it saturated against Propositional, Survival and
      Box, Diamond Rules but for all formula ψ that occurs in terms $(w, \psi) \in T_\sigma$,
      there is a $\psi' \in FL(\psi)$ such that neither ψ' nor $\neg\psi'$ occurs in T_σ **then**
      Non-deterministically choose ψ' or $\neg\psi'$ and
      $T_\sigma = T_\sigma \cup \{(w, \psi')\}$ or $T_\sigma = T_\sigma \cup \{(w, \neg\psi')\}$ as per choice,

where $\neg\psi'$ is in Negation normal form.

else if $\perp \notin T_\sigma$ and T_σ is saturated against Propositional rules, Survival Rules and Diamond, Box rules and there is no formula ψ that occurs as $(w,\psi) \in T_\sigma$, there is a $\psi' \in FL(\psi)$ such that neither ψ' nor $\neg\psi'$ occurs in T_σ **then**

for each $\hat{K}_i\psi$ that occurs in T_σ **do**

$T = \{(w,\hat{K}_i\psi') \mid (w,\hat{K}_i\psi') \in T_\sigma\} \cup \{(w,K_i\psi') \mid (w,K_i\psi') \in T_\sigma\}$ $\cup\{(w,\psi) \mid (w,\hat{K}_i\psi) \in T_\sigma\}$

If there is no i-ancestor $T_{\sigma''}$ of T_σ such that $T \subseteq T_{\sigma''}$ then add a i-child $T_{\sigma'} = T$

while T_{σ_0} is not marked **do**

if there is an unmarked leaf node T_σ of $\mathcal{T}_P$ **then**

if $\perp \in T_\sigma$ or $\{(w,K_i\psi),(w,\neg\psi)\} \subseteq T_\sigma$ **then**

Mark T_{σ_0} "UNSAT"

else

Mark T_{σ_0} "SAT"

else

T_σ is an unmarked internal node whose all children are marked.

if all children of T_σ is marked "SAT" **then**

Mark T_σ "SAT"

else

Mark T_σ "UNSAT"

if T_{σ_0} is marked "SAT" **then**

Return SAT

else

Return UNSAT

We saturate the rules carefully such that each node in the tree corresponds to a single state in the model. This technique is well studied in [32].

Theorem 42. *The satisfiability of* POL^- *is in* NEXPTIME.

Proof. Given the tree $\mathcal{T}_\mathsf{P}$ we create in the procedure, a node T_σ is marked satisfiable iff it does not have $\perp$, $\{(\sigma\ w\ K_i\psi),(\sigma\ w\ \neg\psi)\} \nsubseteq T_\sigma$ and all its successors are marked satisfiable. We prove three statements:

- **Statement 1:** Each node is of at most exponential size, that is, has at most exponential many terms.

- **Statement 2:** Maximum children a node can have is polynomial.
- **Statement 3:** The height of the tree is polynomial.

The above three statements prove the proof tree $\mathcal{T}_{\mathsf{P}}$ is at most of exponential size and hence the non-deterministic algorithm can just guess such a tree and verify the correctness of it in polynomial time with respect to the size of $\mathcal{T}_{\mathsf{P}}$.

Proof of Statement 1. Since a term in a node T_σ is of the form $(\sigma \;\; w \;\; \psi)$, where w is a word over some finite alphabet Σ and ψ is a formula of POL^-.

According to the shape of the rules, a formula that can be derived is always in $FL(\varphi)$. Since $|FL(\varphi)| \leq O(|\varphi|)$[35], hence there can be at most $O(|\varphi|)$ many formulas.

Also, since a regular expression π occuring in a modality is star-free (that is does not contain the Kleene star), hence a word $w \in \mathcal{L}(\pi)$ is of length at most $|\pi|$ which is again of length at most φ. Also there are at most $|FL(\varphi)|$ many regular expressions. Hence there are at most $|\Sigma|^{O(p(|\varphi|))}$, where $p(X)$ is some polynomial on X, many unique words possible. Hence therefore, there can be at most exponential many terms in a single node.

Proof of Statement 2. From a node T_σ, a child is created for every unique triplet of $(\sigma \;\; w \;\; \hat{K}\psi)$ in T_σ. Number of such triplets possible is, as proved is at most polynomial with respect to $|\varphi|$.

Proof of Statement 3. For proving this, we use $md(\Gamma)$, given a set of formulas Γ, is the maximum modal depth over all formulas in Γ. Finally we define $F(T_\sigma)$ as the set of formulas occuring in the node T_σ.

Consider T_σ, the node $T^i_{\sigma'}$ is i- successor of T_σ and $T^j_{\sigma''}$ be the j successor of $T^i_{\sigma'}$ $(i \neq j)$. Note that all the formulas in $F(T^j_{\sigma''})$ are from FL closure of all the K_j and $\hat{K}_j$ formulas from $F(T^i_{\sigma'})$.

Also all the formulas in $F(T^i_{\sigma'})$ are in the FL closure of the K_i and $\hat{K}_i$ formulas occusring in T_σ. Hence $md(T^j_{\sigma''}) \leq md(F(T^i_{\sigma'}))$. Therefore, there can be at most $O(|\varphi|^c)$ such agent alterations in one path of $\mathcal{T}_P$ (not linear because there can be polynomial many words paired with each formula).

Now let us consider how many consecutive i succesors can happen in a path. Suppose a T_σ has a new i-successor node $T_{\sigma'}$ for the term $(\sigma \;\; w \;\; \hat{K}_i\psi)$. Due to the fact that the indistinguishability relation is equivalence for each agent due to the Transitivity, Symmetry rule and the reflexivity that infers in the possibility rule, hence all the possibility and the knowledge formula terms of the form $(\sigma \;\; w' \;\; \hat{K}_i\xi)$ or $(\sigma \;\; w' \;\; K_i\xi)$ of agent i are in the successor node $T_{\sigma'}$ in the form $(\sigma' \;\; w' \;\; \hat{K}_i\xi)$ or $(\sigma' \;\; w' \;\; \hat{K}_i\xi)$ respectively, along with the term $(\sigma' \;\; w \;\; \psi)$. Hence the number of such unique combination of terms will be at most polynomial to the size of $|FL(\varphi)|$.

Therefore, the height of $\mathcal{T}_{\mathsf{P}}$ is polynomial with respect to the $|\varphi|$. □

Fig. 5.5 A set of tile types and an empty square, and a solution.

5.4 Hardness of Satisfiability in POL⁻

In this section, we give a lower bound to the Satisfiability problem of POL⁻. We reduce the well-known NEXPTIME-complete Tiling problem to come up with a formula in the POL⁻ fragment that only has 2 agents.

Theorem 43. POL⁻ *satisfiability problem is* NEXPTIME*-Hard.*

Proof. We reduce the NEXPTIME-complete tiling problem of a square whose size is 2^n where n is encoded in unary [63] (see Figure 5.5). The instance of the tiling problem is (T, t_0, n) where T is a set of tile types (e.g ⊠), t_0 is a specific tile that should be at position $(0,0)$, and n is an integer given in unary. Note that the size of the square is exponential in n. We require the colours of the tiles to match horizontally and vertically.

The idea of the reduction works as follows. We consider two tilings A and B. We will construct a formula $tr(T, t_0, n)$ expressing that the two tilings are equal, contains t_0 at (0, 0), and respect the horizontal and vertical constraints.

With the help of two epistemic modalities K_i and K_j we can simulate a standard K modal logic $\Box$. For the rest of the proof, we consider such a $\Box$ modality and its dual $\Diamond$. We encode a binary tree whose leaves are pairs of positions (one position in tiling A and one in tiling B). Such a tree is of depth $4n$: n bits to encode the x-coordinate in tiling A, n bits to encode the x-coordinate in tiling B, n bits to encode the y-coordinate in tiling A, n bits to encode the y-coordinate in tiling B. A pair of positions is encoded with the $4n$ propositional variables: $p_0, \ldots, p_{4n-1}$. The first $p_0, \ldots, p_{2n-1}$ encodes the position in tiling A while the later $p_{2n}, \ldots, p_{4n-1}$ encodes the position in tiling B. At each leaf, we also use propositional variables q_t^A (resp. q_t^B) to say there is tile t at the corresponding position in tiling A (resp. tiling B). The following formula enforces the existence of that binary tree $\mathcal{T}$ by branching over the truth value of proposition p_ℓ at depth ℓ:

$$\bigwedge_{\ell<4n} \Box^\ell \left(\Diamond p_\ell \wedge \Diamond \neg p_\ell \wedge \bigwedge_{i<\ell} (p_i \rightarrow \Box p_i) \wedge (\neg p_i \rightarrow \Box \neg p_i) \right) \tag{5.1}$$

Now, by using of specific Boolean formulas over $p_0, \ldots, p_{4n-1}$, it is easy to express equality, presence of t_0 at $(0,0)$ and horizontal and vertical constraints:

$$\Box^{4n}\left(\bigvee_t q_t^A \ \wedge \ \bigwedge_{t\neq t'}(\neg q_t^A \vee \neg q_{t'}^A)\right) \tag{5.2}$$

$$\Box^{4n}\left(\bigvee_t q_t^B \ \wedge \ \bigwedge_{t\neq t'}(\neg q_t^B \vee \neg q_{t'}^B)\right) \tag{5.3}$$

$$\Box^{4n}(\text{position in tiling } A = 0) \rightarrow q_{t_0}^A \tag{5.4}$$

$$\Box^{4n}\left(\begin{array}{l} x\text{-coordinate of position in } A \\ = 1 + x\text{-coordinate of position in } B \end{array}\right) \tag{5.5}$$

$$\rightarrow \bigvee_{t,t' \mid t \text{ matches } t' \text{ horizontally}} (q_t^A \wedge q_{t'}^B) \tag{5.6}$$

$$\Box^{4n}\left(\begin{array}{l} y\text{-coordinate of position in } A \\ = 1 + y\text{-coordinate of position in } B \end{array}\right) \tag{5.7}$$

$$\rightarrow \bigvee_{t,t' \mid t \text{ matches } t' \text{ vertically}} (q_t^A \wedge q_{t'}^B) \tag{5.8}$$

The main difficulty is to be sure that all pairs of positions with the same position for - let's say - tiling A indicates the same tile for the tiling A (i.e. the same variable q_t^A is true). To this aim, we will write a formula of the following form

$$[\pi_{\text{ any position in A }}] \bigvee_t \Box^{4n} q_t^A \ \wedge \ [\pi_{\text{any position in B}}] \bigvee_t \Box^{4n} q_t^B.$$

To be able to perform observations to select any position in tiling A (resp. B) whatever the position in tiling B (resp. A) is, we introduce the alphabet $\Sigma = \{A, \bar{A}, B, \bar{B}\}$. We write these two formulas that make a correspondence between valuations on the leaves and observations:

$$\Box^{4n} \bigwedge_{i=0..2n-1} [A + \bar{A}]^i \left(\begin{array}{l} (p_i \rightarrow \langle A\rangle \top \wedge [\bar{A}]\bot) \wedge \\ (\neg p_i \rightarrow \langle \bar{A}\rangle \top \wedge [A]\bot) \end{array}\right) \tag{5.9}$$

$$\Box^{4n} \bigwedge_{i=2n..4n-1} [B + \bar{B}]^{i-2n} \left(\begin{array}{l} (p_i \rightarrow \langle B\rangle \top \wedge [\bar{B}]\bot) \wedge \\ (\neg p_i \rightarrow \langle \bar{B}\rangle \top \wedge [B]\bot) \end{array}\right) \tag{5.10}$$

The idea is that a $2n$-length word on alphabet $\{A, \bar{A}\}$ corresponds to a valuation over $p_1, \ldots, p_{2n-1}$, and thus a position in tiling A and only that $2n$-length word on alphabet $\{A, \bar{A}\}$

is observable. In the same way, a word on alphabet $\{B, \bar{B}\}$ corresponds to a valuation over $p_{2n}, \ldots, p_{4n-1}$, thus a position in tiling B.

We also say that the inner node of the binary tree is never pruned by observations (all $2n$-length words over $\{A, \bar{A}, B, \bar{B}\}$ are observable):

$$\Box^{\leq 4n} \bigwedge_{i=0..2n-1} [\Sigma]^i (\langle A \rangle \top \wedge \langle \bar{A} \rangle \top \wedge \langle B \rangle \top \wedge \langle \bar{B} \rangle \top) \tag{5.11}$$

The formula for ensuring the uniqueness of q_t^A whatever the position in tiling B, and the other way around are then:

$$[(A + \bar{A})^{2n}] \bigvee_t \Box^{4n} q_t^A \wedge [(B + \bar{B})^{2n}] \bigvee_t \Box^{4n} q_t^B \tag{5.12}$$

The intuition works as follows. When evaluating $[(A + \bar{A})^{2n}]\Box^{4n} q_t^A$, we consider all worlds w in $\mathcal{L}((A + \bar{A})^{2n})$ and we consider any pruning $\mathcal{M}|_w$ of the model $\mathcal{M}$ which contains the binary tree $\mathcal{T}$. In $\mathcal{M}|_w$, only the leaves where the valuation on $p_0, \ldots, p_{2n-1}$ that corresponds to w stays. With $\bigvee_t$, we choose a tile type t in T. The modality $\Box^{4n}$ then reaches all the leaves and imposes that q_t^A holds.

The reduction consists of computing from an instance (T, t_0, n) of the tiling problem the POL^- formula $tr(T, t_0, n)$ which is the conjunction of (1-12), which is computable in poly-time in the size of (T, t_0, n) (recall n is in unary). Furthermore, one can check that (T, t_0, n) is a positive instance of the tiling problem iff $tr(T, t_0, n)$ is satisfiable. □

5.5 Complexity results of Fragments of POL^-

In this section, we consider a few fragments of POL^- and we give complexity results for them. First, we consider the single agent fragment of POL^-, and then we prove complexity results for the word fragment of POL^- (both single and multi-agent) using reductions to PAL.

5.5.1 Single agent fragment of POL^-

While we have shown (in Theorem 42) that the satisfiability problem of the POL^- is $\mathsf{NEXPTIME}$-hard, the hardness proof holds only for the case when the number of agents is at least 2. However, we prove that satisfiability problem in the single Agent fragment of POL^- is PSPACE-hard, although single-agent epistemic logic $S5$ is NP-complete.

We prove it by reducing TQBF into our problem. The TQBF problem is: given a formula φ of the form $Q_1x_1Q_2x_2\dots Q_nx_n\xi(x_1,x_2,\dots,x_n)$ where $Q_i \in \{\forall,\exists\}$ and $\xi(x_1,x_2,\dots,x_n)$ is a Boolean formula in CNF over variables $x_1,\dots,x_n$, decide whether the formula φ is true.

Theorem 44. *The satisfiability problem for single agent fragment of* POL^- *is* PSPACE*-hard.*

Proof. Given a QBF formula $\varphi = \exists x_1 \forall x_2 \dots Q_n x_n \xi(x_1,\dots,x_n)$, where Q_i is $\exists$ if i odd, and is $\forall$ is even, and $\xi(x_1,\dots,x_n)$ is a propositional formula in CNF over variables $x_1,\dots,x_n$. Without loss of generality, we suppose we have m clauses with at most 3 literals in each. The objective is to define a POL^--formula $\tau(\varphi)$, computable in poly-time in $|\varphi|$, such that φ is QBF-true iff $\tau(\varphi)$ is POL^--satisfiable. To save space, we denote $\neg x_i$ as $\overline{x_i}$, where x_i is a variable in the TQBF. We also write $\overline{\overline{x_i}}$ for x_i.

Definition of $\tau(\varphi)$. We encode valuations over $x_1,\dots,x_n$ by words on the alphabet $\{a_{x_i}, a_{\overline{x_i}} \mid i = 1..n\}$. We say that a literal ℓ_h is consistent with a word w if a_{ℓ_h} appears in w. For instance the word $a_{x_1}a_{\overline{x_2}}a_{\overline{x_3}}$ encodes the valuation in which x_1 is true and both x_2 and x_3 are false. Set of valuations are represented by languages. For $1 \le u \le v \le n$, $B^u_v := (a_{x_{u+1}} + a_{\overline{x_{u+1}}})(a_{x_{u+2}} + a_{\overline{x_{u+2}}}) \dots (a_{x_v} + a_{\overline{x_v}})$, (by convention $B^u_v = \epsilon$ when $u = v$). Intuitively, B^u_v is the language encoding the set of all possible valuations over propositions $x_{u+1},\dots,x_v$.

- We first define several formulas to express constraints on expectations:
  - The formula $T_i := (\langle a_{x_i}\rangle\top \wedge \langle a_{\overline{x_i}}\rangle\top)$ imposes that the current state survives after observing both a_{x_i} as well as $a_{\overline{x_i}}$.
  - For each literal ℓ_h being x_h or $\overline{x_h}$, we build a formula L_h that enforces the expectation at the current state contains all words encoding valuations over propositions $x_1,\dots,x_n$ in which ℓ_h is true:

$$\begin{aligned} L_h := & \bigwedge_{i=1}^{h-1} ([B^0_{i-1}]T_i \\ & \wedge [B^0_{h-1}](\langle a_{\ell_h}\rangle\top \wedge [a_{\overline{\ell_h}}]\bot) \\ & \wedge \bigwedge_{i=h+1}^{n} [B^0_{h-1}a_{\ell_h}B^h_{i-1}]T_i \end{aligned}$$

  - Finally for each clause $C_j = (\ell_h \vee \ell_r \vee \ell_k)$, we define the formula $\mathsf{tr}(C_j) := K(p_j \to (L_h \vee L_r \vee L_k)) \wedge \hat{K}p_j$. The subformula $\hat{K}p_j$ enforces the existence of a p_j-state. And the subformula $K(p_j \to (L_h \vee L_r \vee L_k))$ enforces that any p_j-state survives on all the words from $w \in B^0_n$ which are consistent with either l_h, or l_r or l_k.

- $S := \mathsf{tr}(Q_1x_1)\mathsf{tr}(Q_2x_2)\ldots\mathsf{tr}(Q_nx_n)\bigwedge_{j=1}^{m}\hat{K}p_j$ where $\mathsf{tr}(\forall x_i) = [a_{x_i} + a_{\overline{x_i}}]$, $\mathsf{tr}(\exists x_i) = \langle a_{x_i} + a_{\overline{x_i}}\rangle$ for any $i \in [n]$. Intuitively, the choice of a valuation over $x_1,\ldots,x_n$ by the two players $\exists$ and $\forall$ in the QBF-prefix $Q_1x_1\ldots Q_nx_n$ is simulated by the choice of a word in B_n^0 by the two players $\langle.\rangle$ and $[.]$ so that all clauses are true (i.e. all p_j-state survives the observation of w: $\bigwedge_{j=1}^{m}\hat{K}p_j$).

The POL^{-}-formula $\tau(\varphi)$ is defined by

$$\tau(\varphi) = \bigwedge_{C_j\in\varphi}\mathsf{tr}(C_j) \wedge S \wedge (\bigvee_{j=1}^{m} p_j)$$

Note that $\tau(\varphi)$ can be computed in poly-time in $|\varphi|$. Let us prove that φ is a QBF-true iff $\tau(\varphi)$ is POL^{-}-satisfiable.

$\boxed{\Rightarrow}$ First assume φ is QBF-true. Hence there is a Quantifier tree that is certifying the truth. We create a model:

- $W = \{1,2,\ldots,m\}$
- $R = W \times W$
- $V(j) = \{p_j\}$
- $Exp(j) = \sum_{l_i\in C_j}(\Pi_{k=1}^{i-1}(a_k + \bar{a_k})\mathsf{tr}_c(l_i)\Pi_{k=i+1}^{n}(a_k + \bar{a_k}))$

First we prove $\mathcal{M}_\varphi, 1 \vDash \mathsf{tr}(C_j)$ for every $j \in [m]$.

Consider for any state j, since $V(j) = \{p_j\}$, hence $\hat{K}p_j$ stands true from any state. Now we prove, since the formula $K(p_j \rightarrow (\mathsf{tr}_m(l_h) \vee \mathsf{tr}_m(l_i) \vee \mathsf{tr}_m(l_k)))$ insists, that $\mathcal{M}_\varphi, j \vDash (\mathsf{tr}_m(l_h) \vee \mathsf{tr}_m(l_i) \vee \mathsf{tr}_m(l_k))$

Given $C_j = (l_h \vee l_k \vee l_r)$, since φ is true, there is a path (among many) is the quantifier tree where at the end of the path (leaf node), it is true that in every clause at least one literal is assigned true. Let us consider in C_j, in this path, l_h was assigned true.

By induction on $1 \leq i < h$, it can be proved that $\mathcal{M}_\varphi|_w, j \vDash T_i$ for every $w \in \mathcal{L}(B_{i-1}^1)$.

By the definition of the model and by the fact that for every $w \in \mathcal{L}(B_{h-2}^1)$, $\mathcal{M}_\varphi|_w, j \vDash T_{h-1}$, it can be seen that $\mathcal{M}_\varphi|_w, j \vDash ([a_{\overline{l_h}}]\bot \wedge \langle a_{l_h}\rangle\top)$ for every $w \in \mathcal{L}(B_{h-1}^1)$

Again, by induction on $h < i \leq n$, it can be proved that, $\mathcal{M}_\varphi|_w, j \vDash T_i$ for every $w \in \mathcal{L}(B_{h-1}^0 a_{l_h} B_{i-1}^h)$.

Hence $\mathcal{M}_\varphi, 1 \vDash \mathsf{tr}(C_j)$.

Now we prove If φ is true then $\mathcal{M}_\varphi, 1 \vDash \mathsf{tr}_c(Q_1x_1)\mathsf{tr}_c(Q_2x_2)\ldots\mathsf{tr}_c(Q_nx_n)\bigwedge_{j=1}^{m}\hat{K}p_j$.

We prove for any $i \in \{0,\ldots,n-1\}$, $\mathcal{M}_\varphi|_{\mathsf{tr}(l_1)\ldots\mathsf{tr}(l_{n-i})}, 1 \vDash \mathsf{tr}(Q_{n-i+1}x_{n-i+1})\ldots\mathsf{tr}(Q_nx_n)\bigwedge_{j=1}^m \hat{K}p_j$, where $(l_1,\ldots,l_{n-i})$ represent any assignment respect to the true paths in the quantified boolean tree upto level $n-i$.

Base Case Consider the case for $i=1$. Since by assumption n is even, $\mathsf{tr}(Q_nx_n) = [a_{x_n} + a_{\overline{x_n}}]\bigwedge_{j=1}^m \hat{K}p_j$. Since by assumption, $(l_1,\ldots,x_n)$ and $(l_1,\ldots,\bar{x_n})$ are a satisfying assignment for ξ, hence $\mathcal{M}_\varphi|_{\mathsf{tr}(l_1)\ldots a_{x_n}}, 1 \vDash \bigwedge_{j=1}^m \hat{K}p_j$ as well as $\mathcal{M}_\varphi|_{\mathsf{tr}(l_1)\ldots a_{\overline{x_n}}}, 1 \vDash \bigwedge_{j=1}^m \hat{K}p_j$.

Inductive Step Consider $i = k+1$. Consider the case where i is even. Hence $\mathsf{tr}(Q_{n-i+1}x_{n-i+1}) = \langle a_{x_{n-i+1}} + a_{\overline{x_{n-i+1}}}\rangle$. Therefor by assumption $(l_1,\ldots,x_{n-i+1})$ or $(l_1,\ldots,\bar{x}_{n-i+1})$ is an assignment that is making ξ true. Hence by IH , $\mathcal{M}_\varphi|_{\mathsf{tr}(l_1)\ldots\mathsf{tr}(l_{n-i})a_{x_{n-i+1}}}, 1 \vDash \mathsf{tr}(Q_{n-i+2}x_{n-i+2})\ldots\mathsf{tr}(Q_nx_n)\bigwedge_{j=1}^m$ or $\mathcal{M}_\varphi|_{\mathsf{tr}(l_1)\ldots\mathsf{tr}(l_{n-i})a_{\overline{x_{n-i+1}}}}, 1 \vDash \mathsf{tr}(Q_{n-i+2}x_{n-i+2})\ldots\mathsf{tr}(Q_nx_n)\bigwedge_{j=1}^m \hat{K}p_j$, which implies $\mathcal{M}_\varphi|_{\mathsf{tr}(l_1)\ldots\mathsf{tr}(l_{n-i})}, 1 \vDash \langle a_{x_{n-i+1}} + a_{\overline{x_{n-i+1}}}\rangle\mathsf{tr}(Q_{n-i+2}x_{n-i+2})\ldots\mathsf{tr}(Q_nx_n)\bigwedge_{j=1}^m \hat{K}p_j$.

$\boxed{\Leftarrow}$ Now assume $\tau(\varphi)$ has a model $\mathcal{M}$ such that $\mathcal{M}, s \vDash \tau(\varphi)$. Now we derive the quantifier tree certifying φ to be true.

We prove the following:

Proposition 45. *If $t \in \mathcal{M}|_{\mathsf{tr}(l_1)\ldots\mathsf{tr}(l_n)}$ and $M, t \vDash p_j$ then $(l_1,\ldots,l_n)$ is a satisfying assignment for clause C_j.*

Proof. Without loss of generality, let us consider $C_j = (l'_h \vee l'_r \vee l'_k)$. Hence to $\sigma = (l_1,\ldots,l_n)$ to be a satisfying assignment for C_j, at least one of l'_h, l'_r or l'_k should be in σ.

Suppose $(l_1,\ldots,l_n)$ is not a satisfying assignment. Hence $l_h = \bar{l'_h}, l_r = \bar{l'_r}$ and $l_k = \bar{l'_k}$. Also by assumption, p_j is true in t. Therefore either L'_h or L'_r or L'_k is true here. Consider the term L'_h. By definition the term $[B^0_{h-1}](\langle a_{l'_h}\rangle\top \wedge [a_{\overline{l'_h}}]\bot)$ is ANDed and hence is true, but this cannot be true since after projecting on $\mathsf{tr}(l_1)\ldots\mathsf{tr}(l_{h-1})$, $B^0_{h-1}\backslash\mathsf{tr}(l_1)\ldots\mathsf{tr}(l_{h-1})$ is non-empty and hence $\mathcal{M}|_{\mathsf{tr}(l_1)\ldots\mathsf{tr}(l_{h-1})}, t \vDash (\langle a_{l'_h}\rangle\top \wedge [a_{\overline{l'_h}}]\bot)$. But this is a contradiction since $\mathsf{tr}(l_h) = \mathsf{tr}(\bar{l'_h}) = a_{\overline{l'_h}}$. □

Proposition 46. *For any $1 \le i \le n$, If $s \in \mathcal{M}|_{\mathsf{tr}(l_1)\ldots\mathsf{tr}(l_{n-i})}$ and $\mathcal{M}|_{\mathsf{tr}(l_1)\ldots\mathsf{tr}(l_{n-i})}, s \vDash \mathsf{tr}(Q_{n-i+1}x_{n-i+1})\ldots\mathsf{tr}(Q_nx_n)\bigwedge_{j=}^m$ then $Q_{n-i+1}x_{n-i+1}\ldots Q_nx_n\xi|_{(l_1,\ldots,l_{n-i})}$ is true.*

Proof. We state that the statement as the Induction Hypothesis. Now we prove the Base Case for it, that is $i = 1$.

Base Case. By assumption s survives in the projection and $\mathcal{M}|_{\mathsf{tr}(l_1)\ldots\mathsf{tr}(l_{n-1})}, s \vDash [a_{\overline{x_n}} + a_{x_n}]\bigwedge_{j=1}^m \hat{K}p_j$. By proposition 1, since at least one state with p_j for each j is surviving, hence all the clause is still surviving in $\xi|_{(l_1,\ldots,l_{n-1})}$. Since s has at least one of p_j true here and because of the K formula, s survives on both projection on a_{x_n} as well as $a_{\overline{x_n}}$, and hence after that at least one state where p_j is true surviving for each j. Hence $\forall x_n\xi_{l_1,\ldots,l_{n-1}}$ is true.

Inductive Step. The Inductive Step is similarly proven as in base case. □

□

5.5.2 Word fragment of POL^-

To investigate the complexity of the satisfaction problem of the word fragment of POL^-, we use a translation of POL^- to PAL. Before going forward, let us give a very brief overview of the syntax and semantics of PAL.

Public announcement logic (PAL)

To reason about announcements of agents and their effects on agent knowledge, PAL [48] was proposed. The underlying model that is dealt with in PAL is epistemic, $\langle S, \sim, V\rangle$ where S is a non-empty set of states, $\sim$ assigns to each agent in $\mathbf{I}$ an equivalence relation $\sim_i \subseteq S \times S$, and $V : S \to 2^{\mathbf{P}}$ is a valuation function. The language is given as follows:

Definition 45 (PAL syntax)**.** *Given a countable set of* propositional variables $\mathbf{P}$*, and a finite set of* agents $\mathbf{I}$*, a formula φ in Public Announcement Logic (*PAL*) can be defined recursively as:*

$$\varphi := \top \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid K_i\varphi \mid [\varphi!]\varphi$$

where $p \in \mathbf{P}$, and $i \in \mathbf{I}$.

Typically, $[\varphi!]\psi$ says that 'if φ is true, then ψ holds after having publicly announced φ'. Similarly, as in POL^- syntax, the respective dual formulas are defined as,

$$\hat{K}_i\psi = \neg K_i \neg\psi$$
$$\langle\varphi!\rangle\psi = \neg[\varphi!]\neg\psi$$

Formula $\langle\varphi!\rangle\psi$ says that φ is true, and ψ holds after announcing φ. Before going into the truth definitions of the formulas in PAL, let us first define the notion of model update.

Definition 46 (Model Update by Announcement)**.** *Given an epistemic model, $\mathcal{M} = \langle S, \sim, V\rangle$, $s \in S$, and a* PAL *formula φ, the model $\mathcal{M}|_\varphi = \langle S', \sim', V'\rangle$ is defined as:*

- $S' = \{s \in S \mid \mathcal{M}, s \vDash \varphi\}$
- $\sim_i' = \sim_i|_{S' \times S'}$,
- $V'(s) = V(s)$ *for any* $s \in S'$.

Now we are all set to give the truth definitions of the formulas in PAL with respect to pointed epistemic models:

Definition 47 (Truth of a PAL formula)**.** *Given an epistemic model $\mathcal{M} = \langle S, \sim, V\rangle$ and an $s \in S$, a* PAL *formula φ is said to hold at s if the following holds:*

- $\mathcal{M}, s \vDash p$ *iff* $p \in V(s)$*, where* $p \in \mathbf{P}$.
- $\mathcal{M}, s \vDash \neg\varphi$ *iff* $\mathcal{M}, s \nvDash \varphi$.
- $\mathcal{M}, s \vDash \varphi \wedge \psi$ *iff* $\mathcal{M}, s \vDash \varphi$ *or* $\mathcal{M}, s \vDash \psi$.
- $\mathcal{M}, s \vDash K_i\varphi$ *iff for all* $t \in S$ *with* $s \sim_i t$, $\mathcal{M}, t \vDash \varphi$.
- $\mathcal{M}, s \vDash [\psi!]\varphi$ *iff* $\mathcal{M}, s \vDash \psi$ *implies* $\mathcal{M}|_\psi, s \vDash \varphi$.

On complexity

To study the satisfiability problem for the word fragment of POL^-, we transfer the following result from PAL to POL^-:

Theorem 47. *[43] The satisfiability problem of* PAL *is* NP*-complete for the single-agent case and* PSPACE*-complete for the multi-agent case.*

PAL is the extension of epistemic logic with dynamic modal constructions of the form $[\varphi!]\psi$ that expresses 'if φ holds, then ψ holds after having announced φ publicly'. The dynamic operator $\langle\pi\rangle$ in the word fragment of POL^- consists in announcing publicly a sequence of observations. W.l.o.g. as π is a word $a_1 \dots a_k$, $\langle\pi\rangle$ can be rewritten as $\langle a_1\rangle \dots \langle a_k\rangle$. In other words, we suppose that the POL^- dynamic operators only contain a single letter. The mechanism of POL^- is close to Public announcement logic (PAL). Observing a consists in announcing publicly that wa occurred where w is the observations already seen so far.

We introduce fresh atomic propositions p_{wa} to say that letter a is compatible with the current state given that the sequence w was already observed.

For all words $w \in \Sigma^*$, we then define tr_w that translates a POL^- formula into a PAL formula given that w is the already seen observations seen so far:

$$\begin{aligned} tr_w(p) &= p \\ tr_w(\neg\varphi) &= \neg tr_w(\varphi) \\ tr_w(\varphi \wedge \psi) &= tr_w(\varphi) \wedge tr_w(\psi) \\ tr_w(K_i\varphi) &= K_i tr_w(\varphi) \\ tr_w(\langle a\rangle\varphi) &= \langle p_{wa}!\rangle tr_{wa}(\varphi) \end{aligned}$$

We finally transform any POL^- formula φ into $tr(\varphi) := tr_\epsilon(\varphi)$.

Example 18. *Consider the* POL^- *formula* $\varphi := [a]\bot \wedge \langle a\rangle\langle a\rangle\top$. $tr(\varphi)$ *is* $[p_a!]\bot \wedge \langle p_a!\rangle\langle p_{aa}!\rangle\top$. *Note that if* p_a *is false, the truth value of* p_{aa} *is irrelevant.*

Proposition 48. *φ is satisfiable in the word fragment of* POL^- *iff* $tr(\varphi)$ *is satisfiable in* PAL.

Proof. (sketch) $\boxed{\Rightarrow}$ Suppose there is a pointed POL^- model $\mathcal{M}, s_0$ such that $\mathcal{M}, s_0 \models \varphi$. We define $\mathcal{M}'$ to be like $\mathcal{M}$ except that for all states s in $\mathcal{M}$, for all $w \in \Sigma^*$, we say that p_w is true at $\mathcal{M}', s$ iff $Exp(s)\backslash w \neq \emptyset$. It remains to prove that $\mathcal{M}', s_0 \models tr(\varphi)$. We prove by induction on φ that for all $w \in words(\varphi)$, if $Exp(s)\backslash w \neq \emptyset$ then $\mathcal{M}|_w, s \models \varphi$ iff $\mathcal{M}', s \models tr_w(\varphi)$.

We only show the interesting case of $\varphi = \langle a\rangle\psi$. Here the $\mathsf{tr}_w(\langle a\rangle\psi) = \langle p_{wa}!\rangle\mathsf{tr}_{wa}(\psi)$. By assumption, $\mathcal{M}|_w, s \vDash \langle a\rangle\psi$. Hence $\mathcal{M}|_{wa}, s \vDash \psi$. Therefore $Exp(s)\backslash wa \neq \emptyset$. By definition of $\mathcal{M}'$, p_{wa} is true in s. Therefore by IH $\mathcal{M}', s \vDash \mathsf{tr}_{wa}(\psi)$. And since p_{wa} is true, hence $\mathcal{M}', s \vDash \langle p_{wa}!\rangle\mathsf{tr}_{wa}\psi$. Conversely, assuming $\mathcal{M}', s \vDash \langle p_{wa}!\rangle tr_{wa}(\psi)$. Hence p_{wa} is true in s. By definition, p_{wa} is true iff $Exp(s)\backslash wa \neq \emptyset$. Also by IH, $\mathcal{M}|_{wa}, s \vDash \psi$. Hence $\mathcal{M}|_w, s \vDash \langle a\rangle\psi$.

$\boxed{\Leftarrow}$ Suppose there is a pointed epistemic model $\mathcal{M}', s_0$ such that $\mathcal{M}', s_0 \models tr(\varphi)$. We define a POL^- model $\mathcal{M}$ like $\mathcal{M}'$ except that for all states s, $Exp(s) = \{w \in \Sigma^* \mid \mathcal{M}, s \models p_w\}$. It remains to prove that $\mathcal{M}, s_0 \models \varphi$. For the rest of the proof, we prove by induction on φ that for all $w \in \Sigma^*$, if $Exp(s)\backslash w \neq \emptyset$ then $\mathcal{M}|_w, s \models \varphi$ iff $\mathcal{M}', s \models tr_w(\varphi)$. The proof goes similarly as earlier. □

Note that the single-agent and multi-agent word fragment of POL^- is a syntactic extension of propositional logic and the multi-agent epistemic logic respectively, which are NP-Hard and PSPACE-Hard respectively. From the fact that the satisfiability problem of single agent and the multi-agent fragments of PAL is in NP and PSPACE respectively, we have the following corollaries of Proposition 48.

Corollary 49. *The satisfiability problem of the single-agent word fragment of* POL^- *is* NP*-complete.*

Corollary 50. *The satisfiability problem of the multi-agent word fragment of* POL^- *is* PSPACE*-complete.*

5.6 Related work

The complexity of Dynamic Epistemic Logic with action models and non-deterministic choice of actions is $\mathsf{NEXPTIME}$-complete too [6] and their proof is similar to the one of Theorem 43.

The tableau method described for POL^- uses a general technique where terms contain the observations/announcements/actions played so far. This technique was already used for PAL [8], DEL [6], and for a non-normal variant of PAL [44].

Decidability of (single-agent) epistemic propositional dynamic logic (EPDL) with Perfect Recall (PR) and No Miracles (NM) is addressed in [41]. Although PR and NM are validities in POL^-, there are differences to consider even in single agent. Firstly, in an EPDL model, a possible state can execute a program a and can non-deterministically transition to a state among multiple states, whereas in POL^-, if a state survives after observation a, it gives rise to the same state except the *Exp* function gets residued. Also, in EPDL, after execution of a program, the state changes hence the propositional valuation in the state changes, whereas in POL^-, the state *survives* after a certain observation and hence the propositional valuation remains the same.

Whereas in POL^-, observations update the model, there are other lines of work in which specifying what agents observe define the epistemic relations in the underlying Kripke model [21] (typically, two states are equivalent for some agent i if agent i observes the same facts in the two states).

5.7 Perspectives

This work paves the way to an interesting technical open question in modal logic: the connection between POL^- and product modal logics. Single-agent POL^- is close to the product modal logic $S5 \times K$, the logic where models are Cartesian products of an S5-model and a K-model. Indeed, the first component corresponds to the epistemic modality $\hat{K}_i$ while the second component corresponds to observation modalities $\langle \pi \rangle$. There are however two important differences. First, in POL^-, valuations do not change when observations are made. Second, the modality $\langle \pi \rangle$ is of branching at most exponential in π while modalities in K-models do not have branching limitations. We conjecture that the two limitations can be circumvented but it requires some care when applying the finite model property of product modal logic $S5 \times K$. If this connection works, it would be a way to prove $\mathsf{NEXPTIME}$-completeness of star-free single-agent POL^-.

Recall that POL^- is close to PAL with propositional announcements only (see Proposition 48). We conjecture some connections between POL^- and arbitrary PAL [29], and more precisely with Boolean arbitrary public announcement logic [59]. Indeed, the non-deterministic choice + enables to check the existence of some observation to make (for instance, $\langle (a+b)^{10} \rangle \varphi$ checks for the existence of a 10-length word to observe), which is similar to checking the existence of some Boolean announcement.

The next perspective is also to tackle POL with Kleene-star in the language. This study rely on techniques used in epistemic temporal logics and also automata. PAL with Kleene-star is undecidable [45]. Again, the undecidability proof relies on modal announcements. But in the next chapter, we prove that POL is decidable. The idea would be to exploit the link between dynamic epistemic logics and temporal logics [61], and rely on techniques developed for tackling the satisfiability problem in epistemic temporal logics [33].

Chapter 6

POL Decidability

6.1 Introduction

In this chapter, we delve into studying the satisfiability problem for the full POL language. In the last chapter, we dealt only with Star-free fragment which did not allow Kleene star in the regular expressions in the formula. These formulas have an advantage of getting reduced in size with each induction step. To be precise, given a star-free formula, say $\langle\pi\rangle\varphi$, assume there is a pointed model $\mathcal{M}, s$ such that $\mathcal{M}, s \vDash \langle\pi\rangle\varphi$. Let us look what happens with each syntactic form of π:

- For $\pi = \pi_1 + \pi_2$, either $\mathcal{M}, s \vDash \langle\pi_1\rangle\varphi$ or $\mathcal{M}, s \vDash \langle\pi_2\rangle\varphi$.
- For $\pi = \pi_1\pi_2$, $\mathcal{M}, s \vDash \langle\pi_1\rangle\langle\pi_2\rangle\varphi$ which in turn implies there is a $w \in \mathcal{L}(\pi_1)$ such that $\mathcal{M}|_w, s \vDash \langle\pi_2\rangle\varphi$

Note that in both of the above cases, the formula to be checked in order to satisfy the input formula is of size strictly less than $\langle\pi\rangle\varphi$. This is not the case if Kleene star is allowed in π. Suppose there is a pointed model $\mathcal{M}, s$ such that $\mathcal{M}, s \vDash \langle\pi^\star\rangle\psi$. This implies either $\mathcal{M}, s \vDash \psi$, in which case we take $\epsilon \in \mathcal{L}(\pi^\star)$ residuing the model. If not so, then $\mathcal{M}, s \vDash \langle\pi\rangle\langle\pi^\star\rangle\psi$. And this goes on until we can satisfy. The problem here is when to stop checking? What if the formula is unsatisfiable (for example, $p \wedge \langle a^\star\rangle\neg p$)?

Reasoning on similar kind of interpretations are Epistemic Temporal Logics (LTL_k) [34]. Here the proof goes by reducing the problem of existence of such models to non-emptiness of a certain automata. We take a very similar approach in this chapter to prove decidability of POL. Along with that, we also explore, in which sense LTL_k and POL are similar and how they are different. The main result we prove in this chapter is:

Theorem 51. *Satisfiability of* POL *is in* $2-$EXPSPACE.

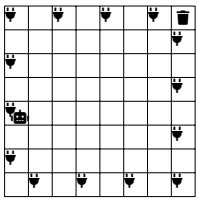

Fig. 6.1 A robot, some power sources, and the debris-disposal area.

Along similar lines, let us recall the example scenario 5, akin to the setting of chapter 5, yet with subtle differences, especially considering the full POL language we focus on in this work, in comparison to the simpler star-free fragment in chapter 5.

Recalling the example 5 (Figure 6.1), just like examples 3 and 4, this scenario had three possibilities, each having a certain expectation.

1. For the possible goal 'moving towards a charging point', the expected moves for the robot is a finite *even* iteration of horizontal or vertical moves, that is, $(\Delta\Delta)^\star$, where, $\Delta = \{\blacktriangleright, \blacktriangle, \blacktriangledown, \blacktriangleleft\}$.

2. For the possible goal 'towards the debris', the expected number of such moves is *odd* without any glitch, expressed by $\Delta(\Delta\Delta)^\star$.

3. Lastly, for the possible goal where the goal is the debris but with a glitch is given by $(\Delta\Delta)^\star\Delta(\Delta\Delta)^\star + \Delta(\Delta\Delta)^\star)$.

As discussed in the last chapter, in many situations, we do not have a specific model, but just a set of properties (Γ, say) based on which we either need to synthesize a model, or need to figure out some other inferred condition (φ, say). These tasks basically reduce to the satisfiability problem of the relevant logic, POL in this case: Checking whether $\Gamma \cup \{\neg\varphi\}$ is satisfiable.

Contribution. Our main contribution is to show the decidability of the satisfiability problem for full POL (with Kleene star). The argument is quite involved and goes on as follows: We start with the usual filtration argument [12] - if a formula is satisfiable then it is satisfiable in a model with exponential many states/possibilities. Although filtration is usually sufficient to prove decidability, for POL it is not the case. This is because of the fact that the expectated observations associated with each world in a model may be arbitrary. We provide an algorithm that guesses a finite syntactic structure defined by means of Hintikka's sets [55]. The filtration ensures that each node of the structure, called a bubble, is of exponential size at most. The connection with POL-satisfiability is then established by carefully applying Arden's lemma [3]. To put POL in perspective with respect to the other

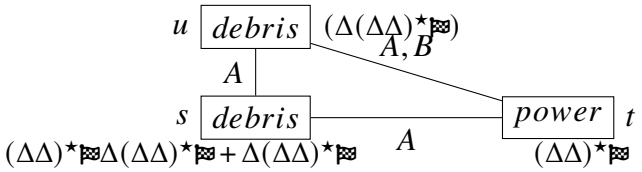

Fig. 6.2 Model $\mathcal{M}$ describing the initial knowledge of the two agents A and B about the moves of the robot. Here $\Delta = \{\blacktriangleright, \blacktriangle, \blacktriangleleft, \blacktriangledown\}$. The model is made up of 3 states s, t, u.

well-known computatation logics of knowledge, we distinguish between the expressive powers of POL and linear temporal logic with knowledge (LTL_k) [34], vis-a-vis, their computational behaviors.

Outline. In Section 6.2, we discuss the satisfiability problem of POL. In Section 6.3, we show the finite (exponential size) model property. In Section 6.4, we give the proof for decidability of POL. In Section 6.5, we compare a fragment of POL with LTL_k. We conclude by discussing other related works, to provide a broad perspective of the work on POL that we present here.

6.2 The Satisfiability Problem

We start with defining POL satisfiability as follows.

Definition 48 (satisfiability problem)**.** *Given a* POL *formula φ, does there exist a* POL *model $\mathcal{M}$ and a state s in it such that $\mathcal{M}, s \vDash \varphi$?*

Before delving into the decidability proof of the satisfiability problem of POL, let us explore a utility of such an algorithm or procedure that decides the satisfiability problem. In line with an application of the star-free POL satisfiability problem given in [19], we consider the following one in full POL. The basic motivation is that we want to verify whether a scenario has a certain property, similar to the model checking problem. However, in this case, we deal with a class of models satisfying certain properties, that can be expressed by POL formulas.

Suppose the robot (cf. Example 5) is moving towards a charging station, represented by the state t in Figure 6.2. We use formulas to describe some relevant properties of the model. Let us call such a formula $\varphi_{\mathcal{M},t}$. It involves three major parts:

1. Assigning propositional truths to each possibility
2. Assigning expectation to each possibility.
3. Interpreting indistinguishability among agents

Now, consider the following formula:

$$\varphi^{t}_{Exp} = \langle\Delta\rangle(\langle\Delta\rangle\top \wedge [⛋]\bot) \wedge [⛋]\bot \\ \wedge [(\Delta\Delta)^{\star}](\langle\Delta\rangle\langle\Delta\rangle\top) \wedge [\Delta\Delta)^{\star}](\langle⛋\rangle\top \rightarrow [⛋\Sigma^{\star}]\bot)$$

The first two conjuncts impose there should exist an observation sequence of the form $(\Delta\Delta)^{\star}$. The third conjunct insists that when such an observation sequence ends with ⛋, no more observation is to be expected.

Similarly φ^{s}_{Exp} and φ^{u}_{Exp} can be created such that the respective observation expressions are expected in s and u, respectively. Then, the formula expressing the properties of the scenario takes the form:

$$\varphi_{\mathcal{M},t} = (\varphi^{t}_{Exp} \wedge power \wedge \neg debris) \\ \wedge \hat{K}_A(\varphi^{s}_{Exp} \wedge debris) \\ \wedge \hat{K}_B(\varphi^{s}_{Exp} \wedge debris \wedge \varphi^{u}_{Exp} \wedge \neg power)$$

Consider $\varphi_P := \langle(\Delta\Delta)^{\star}⛋\rangle(K_A power \wedge \hat{K}_B debris)$. In order to check whether φ_P holds in every model of $\varphi_{\mathcal{M},t}$, we check the (un)satisfiability of $\neg(\varphi_{\mathcal{M},t} \rightarrow \varphi_P)$. We now move on to the proof of decidability for the POL satisfiability problem.

6.3 Finite model property

We now take the first step and prove the following theorem:

Theorem 52 (Finite model property)**.** *If φ is satisfiable then φ is satisfied in a model where the number of states is $2^{O(|\varphi|)}$.*

6.3.1 Filtration

One standard approach to satisfiability is the filtration technique [12], which goes by proving the small model property. That is, for any satisfiable formula in a logical framework, there will always exist a small model that satisfies it whose size can be bounded with respect to the input formula. We prove a similar argument using the following construction of the *small model*. Before the definition, we define an equivalence relation among the states of any model $\mathcal{M} = \langle S, \{R_i\}_{i\in\mathbf{I}}, V, Exp\rangle$ (call it $\sim \subseteq S \times S$) with respect to a formula φ as:

$$s \sim s' \text{ iff for all } \psi \in FL(\varphi), (\mathcal{M}, s \models \psi \text{ iff } \mathcal{M}, s' \models \psi)$$

Note that, the relation $\sim$ is reflexive, transitive and symmetric; thus it is indeed an equivalence relation over S. For any $s \in S$, we denote $[s]$ to be the equivalence class of $\sim$ containing s. Next we give the small model construction.

Definition 49 (Small Model of a Formula)**.** *Given a model $\mathcal{M} = \langle S, \{R_i\}_{i\in\mathbf{I}}, V, Exp\rangle$ and a formula φ, a small model $\mathcal{M}^{\sim} = \langle S^{\sim}, \{R_i^{\sim}\}_{i\in\mathbf{I}}, V^{\sim}, Exp^{\sim}\rangle$ is such that:*

- $S^{\sim} = \{[s] \mid s \in S\}$
- $([s],[s']) \in R_i^{\sim}$ *if these conditions hold:*
  1. *there exists $s_1 \in [s]$ and $s_2 \in [s']$ s. t. $(s_1, s_2) \in R_i$.*
  2. *for all $\hat{K}_i\psi \in FL(\varphi)$, if $\mathcal{M}, s' \vDash \psi \vee \hat{K}_i\psi$ then $\mathcal{M}, s \vDash \hat{K}_i\psi$.*
- $V^{\sim}([s]) = V(s)$
- $Exp^{\sim}([s]) = Exp(s^c)$ *for some $s^c \in [s]$.*

Note that, the above relation $R_i^{\sim} \subseteq S^{\sim} \times S^{\sim}$ is an equivalence relation. The trickier part is the transitivity and symmetry. Transitivity can be proved as in [12], and condition 1 imposes symmetricity. Before proving the small model property, we prove the following lemma, that show the choice of s^c in Definition 49 is not important. The main theorem follows.

Lemma 53. *For any formula of the form $\psi' = \langle\pi\rangle\psi \in FL(\varphi)$, if $s \sim s'$, then there exists a $w \in \mathcal{L}(\pi)$ such that ($\mathcal{M}, s \vDash \psi'$, s survives in $\mathcal{M}|_w$ and $\mathcal{M}|_w, s \vDash \psi$ iff $\mathcal{M}, s' \vDash \psi'$, s' survives in $\mathcal{M}|_w$ and $\mathcal{M}|_w, s' \vDash \psi$).*

Proof. We prove that using induction on π.
Base Case $\pi = a$. Consider $\mathcal{M}, s \vDash \langle a\rangle\psi$. Hence s is in $\mathcal{M}|_a, s \vDash \psi$. Since $s \sim s'$, $\mathcal{M}, s' \vDash \langle a\rangle\psi$ and hence follows.
Inductive case.

- $\pi = \pi_1 + \pi_2$. $\mathcal{M}, s \vDash \langle\pi_1 + \pi_2\rangle\psi$ hence there is a $w \in \mathcal{L}(\pi + \pi_2)$ and $\mathcal{M}|_w, s \vDash \psi$. Now since $\mathcal{M}, s \vDash \langle\pi_1 + \pi_2\rangle\psi$, hence $\langle\pi_1\rangle\psi$ or $\langle\pi_2\rangle\psi$ is satisfied in s. Wlog, suppose $\mathcal{M}, s \vDash \langle\pi_1\rangle\psi$, hence $\mathcal{M}, s' \vDash \langle\pi_1\rangle\psi$ by definition of $s \sim s'$. And hence by IH, our claim holds.
- $\pi = \pi_1\pi_2$. $\mathcal{M}, s \vDash \langle\pi_1\pi_2\rangle\psi$, hence $\mathcal{M}, s \vDash \langle\pi_1\rangle\langle\pi_2\rangle\psi$. hence there is a $w \in \mathcal{L}(\pi_1)$ such that s survives in $\mathcal{M}|_w$ and $\mathcal{M}|_w, s \vDash \langle\pi_2\rangle\psi$. Therefore, by IH $\mathcal{M}|_w, s' \vDash \langle\pi_2\rangle\psi$ and s' survives in $\mathcal{M}|_w$.

- $\pi = \pi_1^\star$. $\mathcal{M}, s \vDash \langle \pi_1^\star \rangle \psi$. This can only happen iff there is a word $w \in \mathcal{L}(\pi_1^k)$ for some $k \geq 0$ such that $\mathcal{M}|_w, s \vDash \psi$. Assume the $w = a_1 a_2 \ldots a_m$. Hence, $\mathcal{M}, s \vDash \langle a_1 \rangle \langle a_2 \rangle \ldots \langle a_m \rangle \psi$ and $\langle a_1 \rangle \langle a_2 \rangle \ldots \langle a_m \rangle \psi \in FL(\varphi)$, which means $\mathcal{M}, s' \vDash \langle a_1 \rangle \langle a_2 \rangle \ldots \langle a_m \rangle \psi$ which gives the result.

This completes the proof. □

Theorem 54. *Given a model $\mathcal{M} = \langle S, \{R_i\}_{i \in \mathbf{I}}, V, Exp \rangle$ and a formula φ, for any $\psi \in FL(\varphi)$ and $w \in \Sigma^\star$,*

$$\mathcal{M}|_w, s \vDash \psi \text{ iff } \mathcal{M}^\sim|_w, [s] \vDash \psi.$$

Although we have a small model property for POL, guessing such a model is not that straightforward. Unlike popular filtration approaches [12], we not only have valuation at each state of the model, but also a regular expression assigned to it. Given a formula, we can guess the number of states from the Fischer-Ladner closure, number of valuations from the propositions used as they can be bounded. However, we cannot guess the number of regular expressions from such straightforward approach, as given an alphabet, number of such regular expressions cannot be bounded. For example, a formula $\langle a^\star \rangle p$ is satisfied with expectation $a^\star$ as well as $(aa)^\star$ as well as $(aaa)^\star$ and so on. A notion of minimal or maximal could be defined, which is not straightforward.

6.4 Decidability of Satisfiability

The goal of this section is to prove the decidability of the POL satisfiability problem:

Theorem 55. *The satisfiability problem is decidable.*

The rest of the section is devoted to the proof of Theorem 55. The general idea is to build a so-called finite bubble transition structure (FBTS) that encodes a satisfying model whose existence is equivalent to the satisfiability of the formula. The algorithm checks whether an FBTS exists.

6.4.1 The Finite Transition Model

Hintikka sets [12] list constraints the set of true formulas in a state should satisfy.

Definition 50 (Hintikka set of Formulas)**.** *A Hintikka set H is a set of formulas such that it satisfy following conditions:*

1. *If ψ does not start with negation, $\psi \in H$ iff $\neg\psi \notin H$.*
2. *$\psi_1 \wedge \psi_2 \in H$ iff $\{\psi_1, \psi_2\} \subseteq H$.*
3. *$\psi_1 \vee \psi_2 \in H$ iff $\psi_1 \in H$ or $\psi_2 \in H$.*
4. *If $K_i\psi \in H$ then $\psi \in H$.*
5. *If $\langle \pi_1 + \pi_2 \rangle \psi \in H$ then $\langle \pi_1 \rangle \psi \in H$ or $\langle \pi_2 \rangle \psi \in H$.*
6. *If $\langle \pi_1 \pi_2 \rangle \psi \in H$ then $\langle \pi_1 \rangle \langle \pi_2 \rangle \psi \in H$.*
7. *If $\langle \pi^\star \rangle \psi \in H$ then either $\psi \in s$ or $\langle \pi \rangle \langle \pi^\star \rangle \psi \in H$.*
8. *If $[\pi_1 + \pi_2]\psi \in H$ then $\{[\pi_1]\psi, [\pi_2]\psi\} \subseteq H$*
9. *If $[\pi_1 \pi_2]\psi \in H$ then $[\pi_1][\pi_2]\psi \in H$*
10. *If $[\pi^*]\psi \in H$ then $\{\psi, [\pi][\pi^*]\psi\} \subseteq H$*

Example 19. *Consider $\varphi := K_i(\langle a \rangle (p \vee q) \wedge [a^\star]\langle a \rangle (p \vee q))$. Here is a Hintikka set containing φ:*

$$\begin{aligned} H = \{&\varphi, \langle a \rangle (p \vee q) \wedge [a^\star]\langle a \rangle (p \vee q), \\ &\langle a \rangle (p \vee q), [a^\star]\langle a \rangle (p \vee q), [a][a^\star]\langle a \rangle (p \vee q)\} \end{aligned}$$

The 2nd formula in H is due to Point 4, the 3rd and 4th ones are due to Point 2, and the last one comes from Point 10.

Let $\mathcal{K}_i(H) = \{K_i\psi \mid K_i\psi \in H\}$ be the set of knowledge formulas in H. We now define *epistemic bubbles* or bubbles that are relational structures on Hintikka sets. This encodes the epistemic skeleton of a POL model, where each set corresponds to a state in the model, with the formulas in a set encoding the formulas true in the corresponding state in the model. The relation among the sets encodes the indistinguishability relation.

Definition 51 (epistemic bubble)**.** *An* epistemic bubble *or a* bubble *is a labelled relational structure $\langle S, \{R_i\}_{i \in \mathbf{I}}, L \rangle$ such that:*

1. *S is a set of (abstract) states such that $0 < |S| \leq 2^{|FL(\varphi)|}$*
2. *$L : S \to 2^{FL(\varphi)}$ is a labelling function such that for every $s \in S$, $L(s)$ is a Hintikka set.*
3. *$R_i \subseteq S \times S$, is a binary equivalence relation among the states satisfying following conditions:*

(a) For any $s \in S$, any formula $\hat{K}_i\psi \in L(s)$, there exists an $s' \in S$ such that $\psi \in L(s')$ and $(s, s') \in R_i$.

(b) For all $s', s'' \in [s]_i$, the equivalence class of s under R_i, $\mathcal{K}_i(L(s')) = \mathcal{K}_i(L(s''))$.

Informally, the first condition is because of the small model property. Note that the equivalence classes $[s]$ in the small model can be characterised using the number of satisfied formulas from $FL(\varphi)$. Hence there can be at most $2^{FL(\varphi)}$ many states in the small model.

Condition 3 describes the properties of the indistinguishability relation. If a state satisfies $\hat{K}_i\psi$, then there has to be an i-indistinguishable state satisfying ψ (3a). Also, knowledge of agent i (formulas of the form $K_i\psi$) should be same in all the i-indistinguishable states (3b).

We now introduce *observation successor*. Considering a bubble B being an epistemic skeleton of a POL model $\mathcal{M}$, an observation successor of that bubble is another bubble B' which represents the epistemic skeleton of another POL model which is a result of the projection of a letter on $\mathcal{M}$, that is $\mathcal{M}|_a$ for some $a \in \Sigma$. For the next definition, given a Hintikka set H, consider $\mathcal{P}(H)$ to be the set of all atomic propositions in H.

Definition 52 (observation successor)**.** *Let $B = \langle S, \{R_i\}_{i\in\mathbf{I}}, L\rangle$ and $B' = \langle S', \{R'_i\}_{i\in\mathbf{I}}, L'\rangle$ be two bubbles. Let $a \in \Sigma$ be a letter. We say that B' is an a-*observation successor *of B if it satisfies the following conditions:*

1. *$S' \subseteq S$.*
2. *For all $s \in S'$, $\mathcal{P}(L(s)) = \mathcal{P}(L'(s))$.*
3. *If $\pi\backslash a \neq \emptyset$ then for all $s \in S$,*

$$(\langle\pi\rangle\psi \in L(s) \text{ iff } (s \in S' \text{ and } \langle\pi\backslash a\rangle\psi \in L'(s))).$$

4. *For all $s \in S'$, $[\pi]\psi \in L(s)$ iff $[\pi\backslash a]\psi \in L'(s)$.*
5. *(Perfect Recall): For all $s \in S$, if there is an $s' \in S$ such that $(s, s') \in R'_i$, then $(s, s') \in R_i$.*

Point 1 explains that a model projected on a letter can retain *at most* all its states. Point 2 says that the surviving states have their valuation unchanged. Point 3 says that any state s satisfying a diamond formula $\langle\pi\rangle\psi$ must survive after observing a, and then, must satisfy $\langle\pi\backslash a\rangle\psi$. Point 4 states that if a state survives, same rule should apply for box formulas as well. Finally, the last point suggests that if an agent considers a state possible from the state s in the projected (residuated) model, they should consider it possible before projection ($\langle a\rangle\hat{K}_i\psi \to \hat{K}_i\langle a\rangle\psi$).

Now we introduce the *finite bubble transition structure* of a formula φ. This is an automata-like labelled transition structure. Here, each node is a bubble, with its child being an observation successor of the letter that the transition is labelled with.

Definition 53 (FBTS)**.** *Given a* POL *formula φ, a* finite bubble transition structure *(FBTS) of φ is a structure $\mathbb{B}_\varphi = \langle \mathcal{B}, \delta \rangle$ where $\mathcal{B}$ is the (finite) set of all bubbles and $\delta : \mathcal{B} \times \Sigma \to 2^{\mathcal{B}}$ is the labelled transition function satisfying the following conditions:*

1. *Each node $B = \langle S, \{R_i\}_{i \in \mathbf{I}}, L \rangle \in \mathcal{B}$ of the structure is a bubble. For all $s \in S$, for all $\psi \in L(s)$, $\psi \in FL(\varphi)$.*

2. *There is a bubble $B^* = \langle S^*, \{R_i^*\}_{i \in \mathbf{I}}, L^* \rangle$ such that there is an $s \in S^*$, with $\varphi \in L^*(s)$.*

3. *For each node $B = \langle S, \{R_i\}_{i \in \mathbf{I}}, L \rangle \in \mathcal{B}$, either $\delta(B, a) = \emptyset$, or, $\delta(B, a) = \{B^a\}$ where B^a is an a-observation successor of B.*

4. *For every $\langle \pi \rangle \psi \in L(s)$ for any $s \in S$ of any node $B = \langle S, \{R_i\}_{i \in \mathbf{I}}, L \rangle \in \mathcal{B}$ in $\mathbb{B}_\varphi$, there is a word $w = a_1 a_2 \dots a_k \in \Sigma^*$, a sequence of nodes (bubbles) $B^0 = B, B^1, \dots, B^k$, where each $B^j \in \mathcal{B}$, such that:*

   (a) $w \in \mathcal{L}(\pi)$.

   (b) $B^j \in \delta(B^{j-1}, a_j)$, *for all* $1 \leq j \leq k$.

   (c) $s \in S^k$ *and* $\psi \in L^k(s)$, *where* $B_k = \langle S^k, \{R_i^k\}_{i \in \mathbf{I}}, L^k \rangle$.

In other words, the FBTS can be thought of where each node represents some residue of the model represented by the bubble $B^\star$ in point 2, which represents the model accepting φ in some state. Point 3 interprets that a model can have at most one residue structure for every letter $a \in \Sigma$. Point 4 interprets the fact that if a state in some model satisfies a formula of the form $\langle \pi \rangle \psi$, then there should exist a model which is residued on a $w \in \mathcal{L}(\pi)$ and the same state in that residued model should satisfy ψ.

Note that a formula may have zero or several FBTS s.

Example 20. *Consider the example in Figure 6.3 of an FBTS $\mathbb{B}_\varphi$ where $\varphi := [a]\bot \wedge \hat{K}_i(\langle a \rangle (p \vee q) \wedge [a^\star]\langle a \rangle (p \vee q))$.*

- *Point 1 of Definition 53 is satisfied: all formulas appearing in the labels are in $FL(\varphi)$*
- *Point 2 of Definition 53 is satisfied since φ appears in the label of s in B^*.*
- *Point 3 of Definition 53 is satisfied. The bubble B is an a-observation successor of B^*. Point 1 of Definition 52 $S' \subseteq S$ is $\{t\} \subseteq \{s, t\}$ in our case. Point 2 of Definition 52 since p appears both in t in B^* and in t in B.*

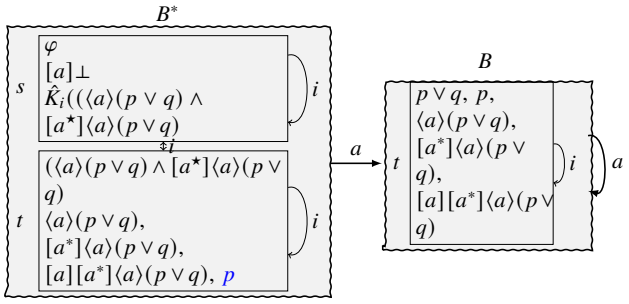

Fig. 6.3 FBTS for $\varphi := [a]\bot \wedge \hat{K}_i(\langle a\rangle(p \vee q) \wedge [a^\star]\langle a\rangle(p \vee q))$. There are two bubbles: B^* and B. There are two abstract states: s and t. If a state appears in a bubble, it is labelled by a Hintikka set: for instance, $L^*(s) = \{\varphi, [a]\bot, \dots\}$.

Algorithm 15 POL-SAT

procedure POL-SAT(φ)
  $\mathcal{S} \leftarrow \{1, \dots 2^{|FL(\varphi)|}\}$
  $\mathcal{H} \leftarrow \{h \subseteq FL(\varphi) \mid h$ satisfies Definition 50$\}$.
  $\mathcal{B} \leftarrow \{B = \langle S \in 2^{\mathcal{S}}, \{R_i \subseteq S \times S\}_{i\in\mathbf{I}}, L\rangle \mid B$ satisfies Definition 51$\}$.
  for each possible $\delta \subseteq \mathcal{B} \times \Sigma \times 2^{\mathcal{B}}$ **do**
    if $\langle \mathcal{B}, \delta\rangle$ satisfies Definition 53 **then**
      Return True
  Return False

By the definition of Hintikka set, since $[a^\star]\langle a\rangle(p \vee q) \in L^*(t)$*, hence* $[a][a^*]\langle a\rangle(p \vee q) \in L^*(t)$*. Now by point 4 of definition 53, since* $\langle a\rangle(p \vee q) \in L^*(t)$*, we have* $(p \vee q) \in L(t)$*. We also have* $[a^\star]\langle a\rangle(p \vee q) \in L^*(t)$*, which again, by definition of Hintikka set gives rise to* $\{\langle a\rangle(p \vee q), [a][a^\star]\langle a\rangle(p \vee q)\} \subseteq L^*(t)$.

6.4.2 Completeness

Theorem 56. *Let* φ *be a formula. Formula* φ *is satisfiable implies there is a finite bubble transition structure of* φ.

Proof. $\boxed{\Rightarrow}$ Suppose φ is satisfiable. Hence there exists a model $\mathcal{M} = \langle S, \{R_i\}_{i\in\mathbf{I}}, V, Exp\rangle$ such that $\mathcal{M}, s \vDash \varphi$ for some $s \in S$ and also by Theorem 54, $|S| \leq 2^{FL(\varphi)}$. Note that, in the context of this proof, $\mathcal{M}|_w = \langle S^w, \{R_i^w\}_{i\in\mathbf{I}}, V^w, Exp^w\rangle$ for any $w \in \Sigma^\star$.

We now create a transition system $\mathbb{B}_\varphi = \langle \mathcal{B}, \delta\rangle$ and prove it to be the finite transition system for φ.

- For all $w \in \Sigma^\star$, we set $B^w := \langle S^w, \{R_i^w\}_{i\in\mathbf{I}}, L^w\rangle$ such that for any $s \in S^w$, $L^w(s) = \{\psi \in FL(\varphi) \mid \mathcal{M}|_w, s \vDash \psi\}$.

- $\mathcal{B} = \{B^w \mid w \in \Sigma^\star\}$
- $\delta(B^w, a) = \{B^{wa}\}$

Now we prove that the transition structure satisfies every point in Definition 53.

1. **Each $B^w \in \mathcal{B}$ is a bubble.** $B^w = \{S^w, \{R_i^w\}_{i\in\mathbf{I}}, L^w\}$.
   - Consider any $s \in S^w$.We show proof when $K_i\psi \in L^w(s)$ and $\langle\pi^\star\rangle\psi \in L^w(s)$. Since $\mathcal{M}|_w, s \vDash K_i\psi$ and the indistinguishability relations are also reflexive, hence $\mathcal{M}|_w, s \vDash \psi$, hence $\psi \in L^w(s)$. Now consider $\mathcal{M}|_w, s \vDash \langle\pi^\star\rangle\psi$, hence by truth condition $\mathcal{M}|_w, s \vDash \langle\pi\rangle\langle\pi^\star\rangle\psi$. Hence $\langle\pi\rangle\langle\pi^\star\rangle\psi \in L(s)$. With similar deductions it can be proved for each $s \in S^w$, $L^w(s)$ is a Hintikka set.
   - Since it is proved each $s \in S^w$ is such that $L^w(s)$ is a Hintikka set, we now prove $B^w = \langle S^w, \{R_i^w\}_{i\in\mathbf{I}}, L^w\rangle$ is a bubble. Consider a formula $\hat{K}_i\psi$ such that $\mathcal{M}|_w, s \vDash \hat{K}_i\psi$. Hence there is another s' such that $sR_i^w s'$ and $\mathcal{M}|_w, s' \vDash \psi$. Hence there a $s' \in S^w$ such that $sR_i^w s'$ and $\psi \in L^w(s')$. The knowledge condition can also be proved similarly using the fact that R_i is an equivalence relation.
2. Since $\mathcal{M}$ is such that $\mathcal{M}, s \vDash \varphi$, we have our second condition of definition 53.
3. Consider B^w and by construction $\delta(B^w, a) = \{B^{wa}\}$. Since valuation in a state of the model remains consistent in updates, condition 2 of definition 52 is satisfied. $\mathcal{M}|_w, s \vDash \langle\pi\rangle\psi$ iff $\mathcal{M}|_{wa}, s \vDash \langle\pi\backslash a\rangle\psi$ if $\pi\backslash a \neq \emptyset$, hence satisfying condition 3. Since distinguishability relation disappears whenever a state disappears and a non-surviving state can never reappear in further updates, condition 5 is satisfied.
4. Let $\langle\pi\rangle\psi \in L^w(s)$. Since $\mathcal{M}|_w, s \vDash \langle\pi\rangle\psi$ implies there exists a $w' \in \mathcal{L}(\pi)$ such that s survives in $\mathcal{M}|_{ww'}$ and $\mathcal{M}|_{ww'}, s \vDash \psi$, hence there exists a series of δ transitions labelled by w' after which $\psi \in L^{ww'}(s)$.

□

6.4.3 Soundness

The most difficult part in creating model out of FBTS, as denoted earlier, is assigning expectations to each state we create in the satisfying model. Here we use a very similar tactic as used in Arden's Lemma while extracting regular expression from finite automata. But unlike such a method where the regular expression gets *collected* in the final state of the

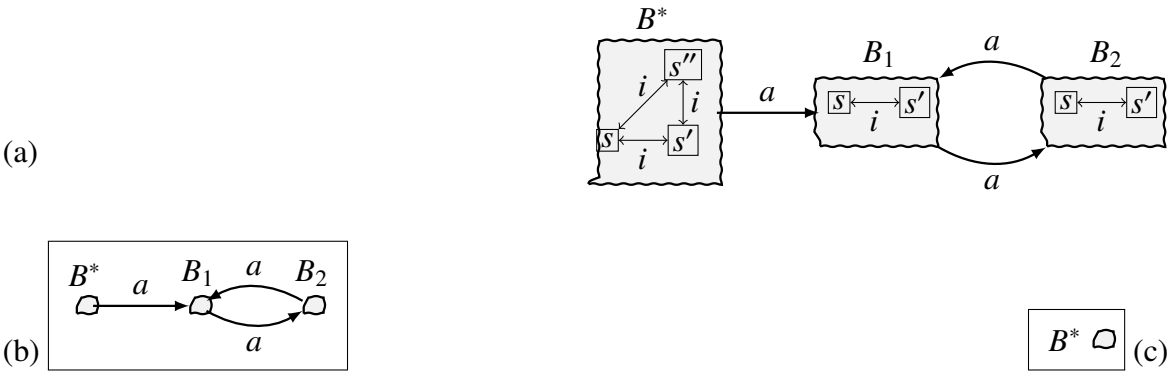

Fig. 6.4 p-FTS for s, s' (b), and s'' (c) extracted from a FBTS (a).

automata from the start state, in our case the letters are residued from the beginning of the regular expressions.

To do that, we define a *pointed finite transition structure* (p-FTS), which essentially extracts the FBTS induced on a given state.

Definition 54 (p-FTS)**.** *Given an FBTS* $\mathbb{B}_\varphi = \langle \mathcal{B}, \delta \rangle$*, the bubble* $B^0 = \langle S^0, \{R_i^0\}_{i \in \mathbf{I}}, L^0 \rangle$ *which is the start node and an* $s \in S^0$*, a* pointed finite transition structure*(p-FTS)* $\langle \mathcal{B}_s, \delta_s \rangle$ *is defined by*

- $\mathcal{B}_s = \{B = \langle S, \{R_i\}_{i \in \mathbf{I}}, L \rangle \in \mathcal{B} \mid s \in S\}$
- $\delta_s(B, a) = \{B' \in \mathcal{B}_s \mid B' \in \delta(B, a)\}$

Basically, given $\mathbb{B}_\varphi$ and s, which has a Hintikka set labelled in start node, a p-FTS with respect to s is an induced structure where the nodes are only the labels that survive in $\mathbb{B}_\varphi$ and the transition in the FBTS are induced on nodes where the states survive.

For example in Figure 6.4, the bubble transition $\mathbb{B}_\varphi$ has three bubbles with $B^\star$ being the start node. In $B^\star$ there are three states s, s', s''. In its successor s'' does not survive. B_1 and B_2 both has s and s'. Hence if this bubble transition is pointed on the state s or s', the resultant pointed transition has all the transitions induced (6.4b), yet if pointed on s'', only the node s'' remains(6.4c).

Theorem 57. *Let* φ *be a formula. There is a finite bubble transition structure of* φ *implies* φ *is satisfiable.*

Proof. Assume an FBTS of φ of the form $\mathbb{B}_\varphi = \langle \mathcal{B}, \delta \rangle$,

Now we create a POL model $\mathcal{M}_\varphi$ out of this. For the time being, we provide idea of how the states of the model and the indistinguishable relations will be, which will essentially be

the bubble, we call it $B^0 = \langle S^0, \{R_i^0\}_{i\in\mathbf{I}}, L^0\rangle \in \mathcal{B}$, that has φ in it as per point 2 in definition 53. We call B^0 to be the *starting bubble* or *start node*.

To assign expectations, we use Arden's Lemma which uses an equational system to find regular expressions from finite automata.

Lemma 58. *[Arden's lemma] Let π_1, π_2 and E be regular expressions over some alphabet, then the equation $E = \pi_2 + \pi_1 E$ has a unique solution for E, which is $E = \pi_1^\star \pi_2$.*

Proof. First we prove whether $E = \pi_1^\star \pi_2$ is a solution.

$$\begin{aligned} E &= \pi_2 + \pi_1(\pi_1^\star \pi_2) \\ &= \pi_2 + \pi_1^+ \pi_2, \text{where } \mathcal{L}(\pi_1^+) = \mathcal{L}(\pi_1^\star)\backslash\{\epsilon\} \\ &= (\epsilon + \pi_1^+)\pi_2 \\ &= \pi_1^\star \pi_2 \end{aligned}$$

Now we prove whether this is a unique solution

$$\begin{aligned} E &= \pi_2 + \pi_1 E \\ &= \pi_2 + \pi_1(\pi_2 + \pi_1 E), \text{unfolding } E \\ &= \pi_2 + \pi_1\pi_2 + \pi_1^2 E \\ &= \pi_2 + \pi_1\pi_2 + \pi_1^2\pi_2 + \pi_1^3 E \\ &= \vdots \\ &= (\epsilon + \sum_{i=1}^{\infty} \pi_1^i)\pi_2 \\ &= \pi_1^\star \pi_2 \end{aligned}$$

□

Now to assign expectation, given the finite bubble transition structure $\mathbb{B}_\varphi$, we create a *pointed finite transition structure* with respect to a Hintikka set labelled (using L^0 function) at $s \in S^0$.

Assigning Expectations Now we start assigning regular expressions to the states. We use the following method, given $\mathbb{B}_\varphi$ and the initial bubble B^0:

- Iterate the following steps for each $s \in S^0$ of B^0:

  1. Create the p-FTS $\langle \mathcal{B}_s, \delta_s \rangle$.

2. For each $B \in \mathcal{B}_s$ do form the equation $E_B = \sum_{B' | B' \in \delta_s(B,a), a \in \Sigma} aE_{B'}$, where E_B and $E_{B'}$'s are variables representing regular expressions.
3. Solve the obtained system of $|\mathcal{B}_s|$ of equations.
4. Assign E_B as the regular expression of s.

We have the following lemma which is self-explanatory.

Lemma 59. *The system of equations created from step (a)-(d) will always have a unique solution.*

Now we need to prove that the residuation of each expression will always accept the expression of the successor.

Before we move on to proving the fact the model created is indeed satisfying φ, for ease of understading we define the notion of *w-survivor* for any $w \in \Sigma^\star$.

For any word $w = a_1 a_2 \dots a_k \in \Sigma^\star$, we call an $s \in S$, where bubble $B = \langle S, \{R_i\}_{i \in \mathbf{I}}, L \rangle$ is in the transition structure $\mathbb{B}_\varphi$ to be a w**-survivor** if $s \in S^w$, where $B^w = \langle S^w, \{R_i^w\}_{i \in \mathbf{I}}, L^w \rangle$ is a bubble that can be reached from B by following a w path in $\mathbb{B}_\varphi$.

Lemma 60. *For any state s in B^0, let B be any node in p-FTS $\langle \mathcal{B}_s, \delta_s \rangle$ such that B can be reached from B^0 using a w labelled path and s is a state in B, where $|w| = k \in \mathbb{N}$, then $\mathcal{L}(E_B) \subseteq \mathcal{L}(E_{B^0} \backslash w)$ and $\mathcal{L}(E_B) \neq \emptyset$.*

Proof. Since s is a w-survivor, hence in every node in the path from B^0 to B, s survives, which means in the p-FTS of the FBTS pointed on s, this path remains. Let $w = a_1 a_2 \dots a_k$

Now when forming the equation for B^0 in the p-FTS on s, $a_1 E_{B^1}$ is a term for E_{B^0}, where B^1 is the a_1 successor of B^0. Also E_{B^1} has a term $a_2 E_{B^2}$ and so on until $E_{B^{k-1}}$ has a term $a_k E_B$. Hence after successive substitution, in the expression of E_{B^0}, the term $a_1 a_2 \dots a_k E_B$ will turn up as +ed term. Hence solving those set of equations will lead to $\mathcal{L}(E_{B^0} \backslash a_1 a_2 \dots a_k) \subseteq \mathcal{L}(E_B)$. □

Now we move on to create the model $\mathcal{M}^\varphi = \langle S^\varphi = S^0, \{R_i^\varphi = R_i^0\}_{i \in \mathbf{I}}, V^\varphi, Exp^\varphi \rangle$, where $V^\varphi(s) = \mathcal{P}(\varphi) \cap L^0(s)$, where $\mathcal{P}(\varphi)$ are propositions in φ and $Exp^\varphi(s) = E_{B^0}$, where E_{B^0} is the expression evaluated using Arden equations on p-FTS on s. We have the following lemma:

Lemma 61. *For all words $w \in \Sigma^\star$, for all $s \in S^0$, if $\psi \in L^w(s)$ where s is a w-survivor, then s survives in $\mathcal{M}^\varphi|_w$ and $\mathcal{M}^\varphi|_w, s \vDash \psi$.*

Proof. We induct on the size of ψ. We consider the property

$$\begin{aligned}\mathcal{P}(\psi)\text{: For all words } w \in \Sigma^\star, \text{ for all } s \in S^0,\\ \text{if } \psi \in L^w(s) \text{ such that } s \text{ is the } w-\text{survivor then}\\ s \text{ survives in } \mathcal{M}^\varphi|_w \text{ and } \mathcal{M}^\varphi|_w, s \vDash \psi.\end{aligned}$$

We only consider the interesting **Case:** $\psi = \langle\pi\rangle\chi$ Since $\psi \in L^w(s)$ such that s is the w-survivor, hence in the structure $(\mathbb{B}_\varphi, s)$, by Lemma 60, $\mathcal{L}(E_B) \subseteq \mathcal{L}(E_{B^0}\backslash w)$ and is not empty (E_B and E_{B^0} are assigned to s in the respective bubbles). Hence s survives in $\mathcal{M}^\varphi|_w$.

Since $\langle\pi\rangle\chi \in L^w(s)$, hence by Definition 53, there is a word $w' = a_1 \dots a_k$ labelled path, where $w' \in \Sigma^*$, a sequence of nodes (bubbles) $B, B^{a_1}, B^{a_2}, \dots, B^{a_k}$, where each $B^{a_i} \in \mathcal{B}$, such that:

1. $w' \in \mathcal{L}(\pi)$.
2. $B^{a_j} \in \delta(B^{a_{j-1}}, a_j)$.
3. $\chi \in L^{ww'}(s) \in S^k$, where $B^k = \langle S^k, \{R_i^k\}_{i\in\mathbf{I}}, L^{ww'}\rangle$

Hence s is the ww'-survivor which proves by IH $\mathcal{M}^\varphi|_{ww'}, s \vDash \chi$. Therefore $\mathcal{M}^\varphi|_w, s \vDash \langle\pi\rangle\chi$. □

This completes the proof of the main theorem. □

Theorem 57 and 56 provides equivalence between existence of an FBTS and satisfiability.

Note that, by observation 37, in the algorithm $\mathsf{POL-SAT}(\varphi)$, the loop 5 and the entire algorithm is guaranteed to terminate after finitely many steps. Therefore we have the following correctness:

Theorem 62. *POL-SAT(φ) returns* True *iff φ is satisfiable.*

Proof. $\boxed{\Rightarrow}$ Assuming POL-SAT(φ) returns True. Since Step 6 verifies whether among all the possible transitions the loop 5 creates, there exists one such transition that satisfies Definition 53, and by Theorem 57, there is a model satisfying φ.

$\boxed{\Leftarrow}$ Conversely, assuming the algorithm returns False. Hence the run of the algorithm could not reach line 7, implying either bubbles could not be created in line 4 or all the transitions created failed to satisfy the condition in line 6. This implies there is no such possible transitions and by Theorem 56, there is no satisfying model for φ. □

6.4.4 Complexity of POL – SAT

Theorem 63. POL – SAT *is in* 2 – EXPSPACE.

Proof. Given a formula φ, the Fischer-Ladner closure of φ is such that $|FL(\varphi)| \leq poly(|\varphi|)$ [35], where $poly(n)$ for an integer n represents a polynomial over n.

A Hintikka set is just a subset of $FL(\varphi)$, hence, any Hintikka set will be of space at most $poly(|\varphi|)$. A bubble is a relation structure, where there are $|\mathbf{I}|$ many distinct relations, over Hintikka sets. Hence a bubble is of size at most $2^{poly|\varphi|}$. The set $\mathcal{B}$ created in line 4 is such that $|\mathcal{B}| \leq 2^{2^{poly(|\varphi|)}}$. Hence, the size of the entire transition structure will of size at most $poly(2^{2^{poly(|\varphi|)}})$.

At each step, the algorithm non-deterministically guesses. Hence the algorithm is Non-deterministically double exponential space. By Savitch's theorem, the algorithm runs in Double Exponential Space. □

Note that, the assignment of the expectation function to each state of the model, especially the size of it is not of concern as far as the complexity of the algorithm is concerned. The expectation assignment is a concern for the proof of correctness, what the algorithm concerns itself is the existence of a finite bubble transition structure. Although, one can carefully form a recursive relation out of the equations that are used in the correctness proof and come up with a bound on the size of expectations.

Note that, the recursive relations are not straightforward, since the transition structure is not necessarily a tree. The loops (which only arise because of Kleene-star) needs to be handled.

The computation of the expectation function is however needed if one needs a model for a formula φ, and not only the yes/no answer to the POL satisfiability decision problem.

6.5 Relation with Knowledge and Time

In this section, we compare a fragment of POL with linear temporal logic with knowledge operator under the perfect recall assumption. We call this logic $\mathsf{LTL_K}$ [34].

First we consider a fragment of POL, named POL^1, that contains regular expressions over single letter a and no concatenation is allowed under the scope of the Kleene Star. The syntax of the regular expressions in POL^1 is given by:

$$\pi := a \mid \pi + \pi \mid \pi \cdot \pi \mid a^*$$

We define a translation of formulas from POL^1 to $\mathsf{LTL_K}$ as follows. Recall the modal constructions of $\mathsf{LTL_K}$: $X\varphi$ (φ is true in the next state, $\varphi U\psi$ (φ remains true until ψ holds), and $\hat{K}_i\varphi$ (φ is epistemically possible for agent i).

Definition 55. *The translation function* $\mathsf{tr} : \mathsf{POL}^1 \to \mathsf{LTL_K}$ *is defined as:*

- $\mathsf{tr}(p) = p$; $\mathsf{tr}(\neg\varphi) = \neg\mathsf{tr}(\varphi)$
- $\mathsf{tr}(\varphi_1 \vee \varphi_2) = (\mathsf{tr}(\varphi_1) \vee \mathsf{tr}(\varphi_2))$
- $\mathsf{tr}(\hat{K}_i\varphi) = \hat{K}_i(\mathsf{tr}(\varphi) \wedge \mathsf{pres}(\varphi))$
- $\mathsf{tr}(\langle a\rangle\varphi) = X(\mathsf{tr}(\varphi))$
- $\mathsf{tr}(\langle \pi_1\pi_2\rangle\varphi) = \mathsf{tr}(\langle \pi_1\rangle\langle \pi_2\rangle\varphi)$
- $\mathsf{tr}(\langle \pi_1 + \pi_2\rangle\varphi) = \mathsf{tr}(\langle \pi_1\rangle\varphi) \vee \mathsf{tr}(\langle \pi_2\rangle\varphi)$
- $\mathsf{tr}(\langle a^*\rangle\varphi) = (\mathsf{tr}(\langle a\rangle\neg\varphi)U\mathsf{tr}(\langle a\rangle\varphi))$

where, $\mathsf{pres}(\varphi) = \bigwedge_{p\in\mathcal{P}}(Gp \vee G\neg p)$. $\mathcal{P}(\varphi)$ *is the set of all propositional letters in* φ.

Note that, the formula $\mathsf{pres}(\varphi)$ is crucial as it describes a point of distinction between POL and $\mathsf{LTL_K}$ with perfect recall and synchronicity: consistency in the state evaluation. Formally, a surviving state should retain its propositional valuation. The only time $\mathsf{pres}(\varphi)$ is used in the translation is when translating $\hat{K}_i$ formulas since that requires new possibilities. The formula $\mathsf{pres}(\varphi)$ maintains the truth values of the propositional letters across all those possibilities

Theorem 64. *Given a formula* $\varphi \in \mathsf{POL}^1$, φ *is* POL^1*-satisfiable iff* $\mathsf{tr}(\varphi) \wedge \mathsf{pres}(\varphi)$ *is* $\mathsf{LTL_K}$*-satisfiable.*

Proof. By induction on the structure of φ.

Consider the case for $\langle a^\star\rangle\psi$. A model $\mathcal{M}, s \vDash \langle a^\star\rangle\psi$ iff there there is a word a^k such that $\mathcal{M}|_{a^k}, s \vDash \psi$. Hence this is equivalent of having a sequence of $\mathcal{M}, \mathcal{M}|_a, \mathcal{M}|_{a^2} \ldots \mathcal{M}|_{a^k}$ such that for every $\mathcal{M}|_{a^i}$ where $i \in [0, k-2]$, $\mathcal{M}|_{a^i}, s\langle a\rangle\neg\psi$ and $\mathcal{M}|_{a^{k-1}}, s \vDash \langle a\rangle\psi$. Also pres expressions keeps the valuation same by the global assignments.

Also consider the formula $\hat{K}_i\psi$. A model $\mathcal{M}, s \vDash \hat{K}_i\psi$ iff there is an s' such that sR_is' and $\mathcal{M}, s' \vDash \psi$. Hence this is equivalent of having a set of i-indistinguishable points in the run where there is a point which satisfies $\mathsf{tr}(\psi)$. Also, $\mathsf{pres}(\psi)$ again maintains the consistency of valuation in such a point in further temporal transition. □

Note that this particular fragment of LTL_K satisfies perfect recall and synchronous knowledge, since POL satisfies these properties by default. This fragment is decidable [34], which goes in line with our result. On the other hand, the assumption on the regular expressions that we have for POL^1 is *no concatenation under Kleene Star*. This is because of the fact that regular expressions are more powerful than LTL, for example, LTL cannot count the number of transitions, that is, LTL cannot express $\langle (aa)^\star \rangle \psi$ [64].

6.6 Related work

In this chapter, we have shown that the Public Observation Logic is decidable. In doing so, we have came across connections and similarities regarding the structures and the expressivity of other logical systems. Each state in a POL model is equipped with a regular expression. If we look at the proof of Theorem 62, we prove equivalence between a POL model and a finite transition structure, whose nodes intuitively correspond to the epistemic skeleton of some POL models, $\mathcal{M}|_w$, where $w \in \Sigma^*$. It can be seen that such a structure would have, for each node, at most one transition labelled for each letter in Σ. This gives rise to a natural connection with a Deterministic Propositional Dynamic Logic (DPDL) [11] model structure. But, there is a significant difference: each node in our constructed transition structure is an epistemic model, whereas in DPDL, it is a propositional valuation. To the best of our knowledge, no such deterministic structures involving epistemic constructions have been studied beforehand.

A more general setting has been studied in the form of Epistemic Propositional Dynamic Logic (EPDL) [41] without any reference to deterministic transitions. Each node of the model is an epistemic structure, with possible nondeterministic transitions. [41] assumes *perfect recall*, as in POL. The most important difference here constitutes the notion of valuation and overall epistemic structure. To be precise, in our case, when a model is residuated or projected, the surviving nodes remain the same except for the expectation expressions getting revised, thus keeping the propositional valuations intact. However, in EPDL, the epsitemic model that is transitioned to, can be arbitrary, satisfying only some transition conditions.

Another study on a PDL-like logic with epistemic operators [36] concerns single agent knowledge formulas, where, verification of *only* knowledge formulas in the regular expressions is considered. Although, they have given a hardness result (EXPTIME-hard) that essentially follows from that for PDL, they do not have any matching upper bound.

Since we are dealing with sequences of observations and how the expectation expressions get residuated, there is a subtle temporal aspect to POL. We explored such a connection with LTL_K [34] in the previous section. In LTL_K [34], the interaction between time and

knowledge assumes *perfect recall* and *synchronous* rules, which also hold in POL. But, there are some differences in the *linearity* aspects. More precisely, since POL deals with multiple letters in the transition alphabet, knowledge does not change linearly, in the sense that, knowledge can change depending on the observations that occur. At a single point of occurrence this can be any one of the letters in the alphabet. We re-iterate here that a valuation in a state in the next temporal transition in $\mathsf{LTL_K}$ can change, whereas such a valuation remains consistent in POL. Thus, $\neg p \wedge Xp$ is satisfiable in $\mathsf{LTL_K}$, whereas $\neg p \wedge \langle a \rangle p$ is not satisfiable in POL, which leads to a difference in expressive power. From the complexity viewpoint, the satisfiability problem of $\mathsf{LTL_K}$ is non-elementary, whereas our POL – SAT algorithm is in 2 – EXPSPACE. We end this discussion noting the fact that both $\mathsf{LTL_K}$ and POL can express that *after some finite sequence of observations, φ holds*, but only POL can express that *after some even occurrences of observations, φ holds* (cf. Section 6). Evidently, resolving the intricate relationship between $\mathsf{LTL_K}$ and POL needs further study.

Since knowledge updates *branch out* in POL depending upon the action observed, Computational Tree Logic with epistemic operators (CTLK) [26] forms a close neighbour, whose satisfiability problem turns out to be undecidable. We note here that unravelled POL models create indistinguishability within the nodes (distinct POL models) themselves, whereas, indistinguishability in a CTLK model occurs among nodes across the tree.

POL is quite similar to Public Announcement Logic (PAL) as well as discussed in chapter 5. PAL with Kleene-star (operators of the form $[\varphi^*]\psi$: after announcing φ publicly a finite number of times, ψ holds) has been shown to be undecidable [45]. The main difference with the decidability of POL resides in the fact that announcements in POL are not modal (agents observe atomic actions but not modal properties like 'agent i knows p').

6.7 Conclusion

We have proved the decidability of the satisfiability problem for the Public Observation Logic, providing a 2 – EXPSPACE upper bound to the problem. We have also analysed the connection of POL with some restrictions on the Kleene-star with a certain fragment of $\mathsf{LTL_K}$.

We leave the hardness result for the future, providing some intuitive ideas in the following. The first and foremost difficulty in simulating POL-like logics is its notion of valuation consistency. Each state, if it survives under a transition, retains its valuation. This aspect is absent in both EPDL and $\mathsf{LTL_K}$. The problem of identifying whether two regular expressions represent the same language, where the expressions have an additional operator of 'taking squares', is EXPSPACE-complete. This is another potential problem that appears very

intuitive to reduce from – two regular expressions are equal iff for any word over the alphabet, the residue of those expressions over that word are equal. But, it is tricky to find a *small* way to express this using POL operators such that the satisfying models get the respective regular expressions assigned to the respective states. Not only we need to express the words that lie in the prefix, but also the words that do not. And the complement expressions blow up exponentially in a straightforward approach.

Except for the hardness result, this work marks a final step to the complexity study of POL, starting from model checking POL [18]. But even then, there is a dynamic extension of POL in [62] called the Epistemic Protocol Logic (EPL), that has not been studied from the computational viewpoint. This logic is very similar to DEL in the sense that it comes with operators that have pointed event or action model like DEL, which *assigns* expectations to the POL models. Such action model also deals with PDL-like regular expressions with Boolean verifiers. We study its model-checking in the next chapter.

Chapter 7

EPL Model Checking

7.1 Introduction

This chapter studies the model-checking of EPL or Epistemic Protocol Logic. As introduced in chapter 2, EPL is a DEL-like extension of POL. It is extended by the introduction of events with protocols that assigns expectations to new possibilities. It is to be noted that, we are not considering EPL in its most general form as introduced in [62]. To be precise, the actions we are considering do not change the propositional facts. A detailed discussion can be found in Chapter 2.

We prove the decidability of the model-checking problem of EPL. The main result we prove here is this:

Theorem 65. *Model-checking problem of* EPL *without any fact-changing action is in* 2 – EXPTIME.

The model-checking of EPL finds an interesting utility in the case of plan-existence problem where there can emerge new possibilities and one wants to decide whether there is a plan in this updated scenario. For example, take the message passing example introduced in chapter 2. One can ask whether there is a plan to let the reciever know of the decision d without letting attacker know it after installing some protocol, model checking of EPL can correctly decide this. For more formal discussion on this, please refer the discussion after definition 21 and the subsection 2.4.6 in chapter 2.

7.2 The Model Checking Procedure

In this section, we provide an algorithm for model-checking of EPL (see Algorithm 16). The algorithm extends the algorithm mcPOL (Algorithm 6) with the the case for formulas of the

form $\langle \mathcal{A}, e \rangle \psi$. The overall idea of the decidability is since the formula is finite, a protocol model $\mathcal{A}$ is also finite, hence the decidability follows.

The most interesting case here is of $\langle \mathcal{A}, e \rangle \psi$ in the model checking algorithm mcEPL. The rest of the cases stay the same as mcPOL.

Algorithm 16 mcEPL

Input: $\mathcal{M} = \langle W, R, V, Exp \rangle, s \in W, \varphi$.
Output: Returns True iff $\mathcal{M}, s \vDash \varphi$

1: **if** $\varphi = \langle \mathcal{A}, e \rangle \psi$ **then**
2: **if** $f_{V(s)}(Prot(e)) = \emptyset$ **then**
3: Return False
4: Return mcEPL$(\mathcal{A} \otimes \mathcal{M}, (e, s), \psi)$
5: **if** $\varphi = p$ is a propositional variable **then**
6: return True if $p \in V(s)$; False otherwise
7: **if** $\varphi = \neg\psi$ **then**
8: return not mcEPL$(\mathcal{M}, s, \psi)$
9: **if** $\varphi = \psi' \vee \psi$ **then**
10: return mcEPL$(\mathcal{M}, s, \psi)$ or mcEPL$(\mathcal{M}, s, \psi')$
11: **if** $\varphi = \langle \pi \rangle \psi$ **then**
12: **for** all models $\mathcal{M}'$ in $\mathcal{M}_{\Sigma^\star}$ **do**
13: **if** $s \in \mathcal{M}'$ and the oracle ResidueModelExistence$(\mathcal{M}, \mathcal{M}', \pi)$ accepts **then**
14: return mcEPL$(\mathcal{M}', s, \psi)$
15: return False
16: **if** $\varphi = \hat{K}_i \psi$ **then**
17: **if** $\exists t \in S$ such that $t \sim_i s$ and mcEPL$(\mathcal{M}, t, \psi)$ **then**
18: return True
19: **else**
20: return False

We now provide the correctness of the algorithm mcEPL.

Theorem 66. mcEPL$(\mathcal{M}, s, \varphi)$ *returns* True *iff* $\mathcal{M}, s \vDash \varphi$.

Proof. The proof is by induction on the structure of φ, similar to the proof of Lemma 24, which is the proof of correctness for mcPOL (Algorithm 6). We extend the correction in

Lemma 24 with the new case $\varphi = \langle \mathcal{A}, e \rangle \psi$.

$$\begin{aligned}
\mathcal{M}, s \vDash \langle \mathcal{A}, e \rangle \psi \text{ iff } & \mathcal{L}(f_{V(s)}(Prot(e))) \neq \emptyset \text{ and } \mathcal{A} \otimes \mathcal{M}, (e, s) \vDash \psi \\
& \text{iff condition in Line 2 fails and} \\
& \mathcal{M} \otimes \mathcal{A}, (e, s) \vDash \psi \\
& \text{iff condition in Line 2 fails and} \\
& \mathsf{mcEPL}(\mathcal{M} \otimes \mathcal{A}, (e, s), \psi) = \mathsf{True}, \text{ by IH} \\
& \text{iff } \mathsf{mcEPL}(\mathcal{M}, s, \langle \mathcal{A}, e \rangle \psi)
\end{aligned}$$

□

Moreover, what is left to prove is its decidability, that is, even though by theorem 66 the algorithm outputs correctly, is the algorithm deciding it in finite steps?

Theorem 67. EPL *model-checking is in* 2 – EXPTIME.

Proof. The Algorithm 16 is correct by Theorem 66. It remains to shows that it runs in double exponential time. Algorithm 16 is recursive. Considering m to be size of the model and n be the size of formula.

Consider the oracle call in line 13 (ResidueModelExistence). As per lemma 12, given a model $\mathcal{M}$ of size m, there are at most $O(2^m)$ many elements in $\mathcal{M}_{\Sigma^\star}$. Hence given a model of size m, the oracle takes deterministically at most $O(2^m)$ steps since it traverses along $|\mathcal{M}_{\Sigma^\star}|$ many models in worst case.

Note that, as recursion goes deep, the size of the model increases and the size of the formula decreases. Assuming $|\mathcal{A}| = k$, for some positive integer k, the input size of the recursive call inside this case will have model size to be mk and formula size $n - k$.

Since k can be of any size from 1 to $n - 1$, we take the largest of all protocol models occurring in φ, say the size be a. In the following recurrence equation, $T(m, n)$ denotes the steps required for the algorithm on input of model of size m and formula of size n. The extra $O(2^m)$ are the steps required by the oracle ResidueModelExistence.:

$$T(m, n) \leq T(ma, n - a) + O(2^m) \tag{7.1}$$

By induction, it can be proven that $T(m, n) \leq O(2^{ma^{poly(mn)}})$, where $poly(mn)$ is a polynomial over mn. □

7.3 Conclusion and Future Work

One of the central themes of this thesis was trying to bring out a fragment of Epistemic Planning which is computationally much easier. We posed two main questions in chapter 1 that this thesis has tried to answer.

The first question was about finding a decidable fragment of Epistemic planning. The approach was to bringing in something similar to public actions rather than more general private ones that DEL deals with.

In chapters 3, 4 and furthermore in chapters 5 and 6, we have seen that model-checking and satisfiability of POL serves this purpose. Essentially, each “public" observation can be thought of an atomic action model. To be precise, each observation can be considered as an action or event model, which has only one action point, hence signifying it is public, since it mean all the agents can actually distinguish which action is being executed. Note that, in DEL as well, updating with public action models can never increase the number of possibilities in the epistemic model, an occurence that is very similar to POL

The main advantage here that we get is the use of Kleene star. What separates the PSPACE-completeness of DEL model-checking [6] and the undecidability of Epistemic planning [14] is the fact that the syntax in the DEL formula is extended by composing actions with Kleene-star in addition to choice and concatenation. Hence, one can talk of properties resulting after some finite iterations of actions. In POL model-checking, even though we take away general actions, reasoning of finite iterations of such public actions, that is composition with Kleene-star, stays intact.

Hence a natural extension of this study will be answering the question as to what extent can the action models be generalised, so that one can get to keep Kleene-star as well as keep the planning instance, at least in the decidable region, if not easier complexity classes?

Coming to the second question. The second question that was presented in the first chapter was about epistemic extension of linear temporal logic (LTL_k). The connection between POL and LTL_k has been shown in chapter 6. As it turns out, although POL gains some reasoning power against LTL_k in terms of counting even or odd temporal points in the future, it is not entirely as powerful as LTL_k. This is because, in LTL_k, as time proceeds, the states can change its valuation, but in POL it remains constant, even though the expectations residue. Another very related framework would be EPDL that is necessary to be looked into [41] for studies in this case and how close the framework comes to LTL_k.

In addition to this, we have also looked into model-checking of EPL in this chapter, where we consider actions that do not change propositions. The action models in DEL can also have a post-condition attached to each action, as defined in [20]. Post-conditions give new valuations in possibilities in the updated model. If public observations correspond to

public actions, expectations in possibilities are quite similar to the notion of pre-conditions in the actions from point of view of whether one “can observe". From this point, reassigning expectations, as EPL and its general form defined in [62] does, can correspond to a similar notion of what post-condition does. Hence exploring the more general Epistemic Protocol Logic and its connection with DEL needs to be studied further.

List of publications on which this thesis is based

Published

- **On Verifying Expectations and Observations of Intelligent Agents.**

  with Sourav Chakraborty, Sujata Ghosh and François Schwarzentruber.

  Published in *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Pages: 2568–2574*

  DOI: 10.24963/IJCAI.2022/356

- **On Simple Expectations and Observations of Intelligent Agents: A Complexity Study**

  with Sourav Chakraborty, Sujata Ghosh and François Schwarzentruber.

  Published in *Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning, KR 2023, Pages: 136–145*

  DOI: 10.24963/KR.2023/14

Accepted

- **Reasoning About Knowledge on Regular Expressions is 2EXPTIME-complete**

  with Sujata Ghosh and François Schwarzentruber.

  Accepted in *International Conference on Principles of Knowledge Representation and Reasoning, KR 2025*.

  This paper contains an extended version of the results presented in Chapter 6 of this thesis.

References

[1] Thomas Ågotnes, Philippe Balbiani, Hans van Ditmarsch, and Pablo Seban. Group announcement logic. *J. Appl. Log.*, 8(1):62–81, 2010. doi: 10.1016/j.jal.2008.12.002. URL https://doi.org/10.1016/j.jal.2008.12.002.

[2] Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. *JACM*, 49(5):672–713, 2002. doi: 10.1145/585265.585270. URL https://doi.org/10.1145/585265.585270.

[3] Dean N. Arden. Delayed-logic and finite-state machines. In *2nd Annual Symposium on Switching Circuit Theory and Logical Design, Detroit, Michigan, USA, October 17-20, 1961*, pages 133–151. IEEE Computer Society, 1961. doi: 10.1109/FOCS.1961.13. URL https://doi.org/10.1109/FOCS.1961.13.

[4] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009. ISBN 978-0-521-42426-4. URL http://www.cambridge.org/catalogue/catalogue.asp?isbn=9780521424264.

[5] Guillaume Aucher and Thomas Bolander. Undecidability in epistemic planning. In Francesca Rossi, editor, *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, pages 27–33. IJCAI/AAAI, 2013. URL http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6903.

[6] Guillaume Aucher and François Schwarzentruber. On the complexity of dynamic epistemic logic. *TARK*, 2013.

[7] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008. ISBN 978-0-262-02649-9.

[8] Philippe Balbiani, Hans van Ditmarsch, Andreas Herzig, and Tiago de Lima. Tableaux for public announcement logic. *J. Log. Comput.*, 20(1):55–76, 2010. doi: 10.1093/logcom/exn060. URL https://doi.org/10.1093/logcom/exn060.

[9] José L. Balcázar, Josep Díaz, and Ricard Gavaldà. Algorithms for learning finite automata from queries: A unified view. *Advances in Algorithms, Languages, and Complexity*, pages 53–72, 1997.

[10] Francesco Belardinelli, Sophia Knight, Alessio Lomuscio, Bastien Maubert, Aniello Murano, and Sasha Rubin. Reasoning about agents that may know other agents' strategies. *IJCAI*, pages 1787–1793, 2021. doi: 10.24963/ijcai.2021/246. URL https://doi.org/10.24963/ijcai.2021/246.

[11] Mordechai Ben-Ari, Joseph Y. Halpern, and Amir Pnueli. Deterministic propositional dynamic logic: Finite models, complexity, and completeness. *J. Comput. Syst. Sci.*, 25(3):402–417, 1982. doi: 10.1016/0022-0000(82)90018-6. URL https://doi.org/10.1016/0022-0000(82)90018-6.

[12] Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic*, volume 53 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2001. ISBN 978-1-10705088-4. doi: 10.1017/CBO9781107050884. URL https://doi.org/10.1017/CBO9781107050884.

[13] Thomas Bolander and Mikkel Birkegaard Andersen. Epistemic planning for single and multi-agent systems. *J. Appl. Non Class. Logics*, 21(1):9–34, 2011. doi: 10.3166/JANCL.21.9-34. URL https://doi.org/10.3166/jancl.21.9-34.

[14] Thomas Bolander, Martin Holm Jensen, and François Schwarzentruber. Complexity results in epistemic planning. *IJCAI*, pages 2791–2797, 2015. URL http://ijcai.org/Abstract/15/395.

[15] Thomas Bolander, Tristan Charrier, Sophie Pinchinat, and François Schwarzentruber. DEL-based epistemic planning: Decidability and complexity. *Artificial Intelligence*, 287, 2020. doi: 10.1016/j.artint.2020.103304. URL https://doi.org/10.1016/j.artint.2020.103304.

[16] Blai Bonet, Héctor Palacios, and Hector Geffner. Automatic derivation of memoryless policies and finite-state controllers using classical planners. *ICAPS*, 2009. URL http://aaai.org/ocs/index.php/ICAPS/ICAPS09/paper/view/752.

[17] Laura Bozzelli, Alberto Molinari, Angelo Montanari, and Adriano Peron. An in-depth investigation of interval temporal logic model checking with regular expressions. *SEFM*, 10469:104–119, 2017.

[18] Sourav Chakraborty, Avijeet Ghosh, Sujata Ghosh, and François Schwarzentruber. On verifying expectations and observations of intelligent agents. In Luc De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pages 2568–2574. ijcai.org, 2022. doi: 10.24963/ijcai.2022/356. URL https://doi.org/10.24963/ijcai.2022/356.

[19] Sourav Chakraborty, Avijeet Ghosh, Sujata Ghosh, and François Schwarzentruber. On simple expectations and observations of intelligent agents: A complexity study. In Pierre Marquis, Tran Cao Son, and Gabriele Kern-Isberner, editors, *Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning, KR 2023, Rhodes, Greece, September 2-8, 2023*, pages 136–145, 2023. doi: 10.24963/KR.2023/14. URL https://doi.org/10.24963/kr.2023/14.

[20] Tristan Charrier and François Schwarzentruber. A succinct language for dynamic epistemic logic. In Kate Larson, Michael Winikoff, Sanmay Das, and Edmund H. Durfee, editors, *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2017, São Paulo, Brazil, May 8-12, 2017*, pages 123–131. ACM, 2017. URL http://dl.acm.org/citation.cfm?id=3091148.

[21] Tristan Charrier, Andreas Herzig, Emiliano Lorini, Faustine Maffre, and François Schwarzentruber. Building epistemic logic from observations and public announcements. In Chitta Baral, James P. Delgrande, and Frank Wolter, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifteenth International Conference, KR 2016, Cape Town, South Africa, April 25-29, 2016*, pages 268–277. AAAI Press, 2016. URL http://www.aaai.org/ocs/index.php/KR/KR16/paper/view/12899.

[22] Tristan Charrier, Sophie Pinchinat, and François Schwarzentruber. Symbolic model checking of public announcement protocols. *JLC*, 29(8):1211–1249, 2019. doi: 10.1093/logcom/exz023. URL https://doi.org/10.1093/logcom/exz023.

[23] Krishnendu Chatterjee, Thomas A. Henzinger, and Nir Piterman. Strategy logic. *Information and Computation*, 208(6):677–693, 2010. doi: 10.1016/j.ic.2009.07.004. URL https://doi.org/10.1016/j.ic.2009.07.004.

[24] Harmen de Weerd, Rineke Verbrugge, and Bart Verheij. Negotiating with other minds: the role of recursive theory of mind in negotiation with incomplete information. *AAMAS*, 31(2):250–287, 2017.

[25] Volker Diekert and Paul Gastin. First-order definable languages. In Jörg Flum, Erich Grädel, and Thomas Wilke, editors, *Logic and Automata: History and Perspectives [in Honor of Wolfgang Thomas]*, volume 2 of *Texts in Logic and Games*, pages 261–306. Amsterdam University Press, 2008.

[26] Catalin Dima. Revisiting satisfiability and model-checking for CTLK with synchrony and perfect recall. In Michael Fisher, Fariba Sadri, and Michael Thielscher, editors, *Computational Logic in Multi-Agent Systems, 9th International Workshop, CLIMA IX, Dresden, Germany, September 29-30, 2008. Revised Selected and Invited Papers*, volume 5405 of *Lecture Notes in Computer Science*, pages 117–131. Springer, 2008. doi: 10.1007/978-3-642-02734-5_8. URL https://doi.org/10.1007/978-3-642-02734-5_8.

[27] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning About Knowledge*. MIT Press, 1995. ISBN 9780262562003. doi: 10.7551/MITPRESS/5803.001.0001. URL https://doi.org/10.7551/mitpress/5803.001.0001.

[28] Michael J. Fischer and Richard E. Ladner. Propositional dynamic logic of regular programs. *JCSS*, 18(2):194–211, 1979. doi: 10.1016/0022-0000(79)90046-1. URL https://doi.org/10.1016/0022-0000(79)90046-1.

[29] Tim French and Hans P. van Ditmarsch. Undecidability for arbitrary public announcement logic. *AiML*, pages 23–42, 2008.

[30] Malik Ghallab, Dana S. Nau, and Paolo Traverso. *Automated planning - theory and practice*. Elsevier, 2004. ISBN 978-1-55860-856-6.

[31] Sujata Ghosh and R. Ramanujam. Strategies in games: A logic-automata study. *LNCS*, 7388:110–159, 2012.

[32] Joseph Y. Halpern and Yoram Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artif. Intell.*, 54(2):319–379, 1992. doi: 10.1016/0004-3702(92)90049-4. URL https://doi.org/10.1016/0004-3702(92)90049-4.

[33] Joseph Y. Halpern and Moshe Y. Vardi. The complexity of reasoning about knowledge and time. i. lower bounds. *J. Comput. Syst. Sci.*, 38(1):195–237, 1989. doi: 10.1016/0022-0000(89)90039-1. URL https://doi.org/10.1016/0022-0000(89)90039-1.

[34] Joseph Y. Halpern, Ron van der Meyden, and Moshe Y. Vardi. Complete axiomatizations for reasoning about knowledge and time. *CoRR*, cs.LO/0208033, 2002. URL https://arxiv.org/abs/cs/0208033.

[35] David Harel, Jerzy Tiuryn, and Dexter Kozen. *Dynamic Logic*. MIT Press, 2000.

[36] Bernhard Heinemann. A pdl-like logic of knowledge acquisition. In Volker Diekert, Mikhail V. Volkov, and Andrei Voronkov, editors, *Computer Science – Theory and Applications*, pages 146–157, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. ISBN 978-3-540-74510-5.

[37] Jaakko Hintikka. Reasoning about knowledge in philosophy: The paradigm of epistemic logic. In Joseph Y. Halpern, editor, *Proceedings of the 1st Conference on Theoretical Aspects of Reasoning about Knowledge, Monterey, CA, USA, March 1986*, pages 63–80. Morgan Kaufmann, 1986.

[38] Martin Holm Jensen. Planning using dynamic epistemic logic: Correspondence and complexity. In Davide Grossi, Olivier Roy, and Huaxin Huang, editors, *Logic, Rationality, and Interaction - 4th International Workshop, LORI 2013, Hangzhou, China, October 9-12, 2013, Proceedings*, volume 8196 of *Lecture Notes in Computer Science*, pages 316–320. Springer, 2013. doi: 10.1007/978-3-642-40948-6_25. URL https://doi.org/10.1007/978-3-642-40948-6_25.

[39] Dexter Kozen. Lower bounds for natural proof systems. *FOCS*, pages 254–266, 1977. doi: 10.1109/SFCS.1977.16. URL https://doi.org/10.1109/SFCS.1977.16.

[40] Dexter Kozen. *Automata and computability*. Undergraduate texts in computer science. Springer, 1997. ISBN 978-0-387-94907-9.

[41] Yanjun Li. Tableaux for a combination of propositional dynamic logic and epistemic logic with interactions. *J. Log. Comput.*, 28(2):451–473, 2018. doi: 10.1093/logcom/exx040. URL https://doi.org/10.1093/logcom/exx040.

[42] Alessio Lomuscio and Jakub Michaliszyn. Model checking multi-agent systems against epistemic HS specifications with regular expressions. *KR*, pages 298–308, 2016.

[43] Carsten Lutz. Complexity and succinctness of public announcement logic. *AAMAS*, pages 137–143, 2006. doi: 10.1145/1160633.1160657. URL https://doi.org/10.1145/1160633.1160657.

[44] Minghui Ma, Katsuhiko Sano, François Schwarzentruber, and Fernando R. Velázquez-Quesada. Tableaux for non-normal public announcement logic. In Mohua Banerjee and Shankara Narayanan Krishna, editors, *Logic and Its Applications - 6th Indian Conference, ICLA 2015, Mumbai, India, January 8-10, 2015. Proceedings*, volume 8923 of *Lecture Notes in Computer Science*, pages 132–145. Springer, 2015. doi: 10.1007/978-3-662-45824-2_9. URL https://doi.org/10.1007/978-3-662-45824-2_9.

[45] Joseph S. Miller and Lawrence S. Moss. The undecidability of iterated modal relativization. *Stud Logica*, 79(3):373–407, 2005. doi: 10.1007/s11225-005-3612-9. URL https://doi.org/10.1007/s11225-005-3612-9.

[46] Christian Muise, Vaishak Belle, Paolo Felli, Sheila A. McIlraith, Tim Miller, Adrian R. Pearce, and Liz Sonenberg. Efficient multi-agent epistemic planning: Teaching planners about nested belief. *Artificial Intelligence*, 302, 2022. doi: 10.1016/j.artint.2021.103605. URL https://doi.org/10.1016/j.artint.2021.103605.

[47] Christos H. Papadimitriou. *Computational complexity*. Academic Internet Publ., 2007. ISBN 978-1-4288-1409-7.

[48] Jan Plaza. Logics of public communications. *Synthese*, 158(2):165–179, 2007. doi: 10.1007/s11229-007-9168-7. URL https://doi.org/10.1007/s11229-007-9168-7.

[49] Amir Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, pages 46–57, 1977. doi: 10.1109/SFCS.1977.32.

[50] Thomason Richmond. Logic-based artificial intelligence. *The Stanford Encyclopedia of Philosophy (Spring 2024 Edition)*. URL https://plato.stanford.edu/archives/spr2024/entries/logic-ai/.

[51] Abdallah Saffidine, François Schwarzentruber, and Bruno Zanuttini. Knowledge-based policies for qualitative decentralized pomdps. *AAAI*, pages 6270–6277, 2018. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17029.

[52] Walter J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *JCSS*, 4(2):177–192, 1970. doi: 10.1016/S0022-0000(70)80006-X. URL https://doi.org/10.1016/S0022-0000(70)80006-X.

[53] Philippe Schnoebelen. The complexity of temporal logic model checking. *AiML*, pages 393–436, 2002.

[54] Michael Sipser. *Introduction to the theory of computation*. PWS Publishing Company, 1997.

[55] R.M. Smullyan. *First-order Logic*. Dover books on advanced mathematics. Dover, 1995. ISBN 9780486683706. URL https://books.google.co.in/books?id=kgvhQ-oSZiUC.

[56] Johan van Benthem. *Logical dynamics of information and interaction*. Cambridge University Press, 2011.

[57] Johan van Benthem, Jan van Eijck, and Barteld P. Kooi. Logics of communication and change. *Information and Computation*, 204(11):1620–1662, 2006. doi: 10.1016/j.ic.2006.04.006. URL https://doi.org/10.1016/j.ic.2006.04.006.

[58] Johan van Benthem, Jan van Eijck, Malvin Gattinger, and Kaile Su. Symbolic model checking for dynamic epistemic logic - S5 and beyond. *JLC*, 28(2):367–402, 2018. doi: 10.1093/logcom/exx038. URL https://doi.org/10.1093/logcom/exx038.

[59] Hans van Ditmarsch and Tim French. Quantifying over boolean announcements. *Log. Methods Comput. Sci.*, 18(1), 2022. doi: 10.46298/lmcs-18(1:20)2022. URL https://doi.org/10.46298/lmcs-18(1:20)2022.

[60] Hans van Ditmarsch, Wiebe van der Hoek, and Barteld Kooi. *Dynamic Epistemic Logic*, volume 337 of *Synthese Library*. Springer, 2008.

[61] Hans van Ditmarsch, Wiebe van der Hoek, and Ji Ruan. Connecting dynamic epistemic and temporal epistemic logics. *Log. J. IGPL*, 21(3):380–403, 2013. doi: 10.1093/jigpal/jzr038. URL https://doi.org/10.1093/jigpal/jzr038.

[62] Hans van Ditmarsch, Sujata Ghosh, Rineke Verbrugge, and Yanjing Wang. Hidden protocols: Modifying our expectations in an evolving world. *Artificial Intelligence*, 208: 18–40, 2014.

[63] Peter van Emde Boas. The convenience of tilings. In *Complexity, Logic, and Recursion Theory*, pages 331–363. CRC Press, 2019.

[64] Pierre Wolper. Temporal logic can be more expressive. *Information and Control*, 56(1): 72–99, 1983. ISSN 0019-9958. doi: https://doi.org/10.1016/S0019-9958(83)80051-5. URL https://www.sciencedirect.com/science/article/pii/S0019995883800515.