An Inquisitive Dynamic Epistemic Logic with Factual Change

René Mellema
(s2348802)

8th July 2019

Master's Thesis

Artificial Intelligence
University of Groningen,
The Netherlands

Supervisors:
Prof. dr. Rineke Verbrugge
Artificial Intelligence
University of Groningen

Dr. Floris Roelofsen
Institute of Logic, Language and Computation
University of Amsterdam

Abstract

Dynamic Epistemic Logic allows us to model the knowledge that agents have and the effects of actions, such as announcements, on that knowledge. This knowledge also includes various notions of group knowledge, such as common and general knowledge. A good example of such a logic is the Logic of Communication and Change, since it allows us to express various notions of group knowledge in a very natural manner. However, some of the things that it cannot model are the questions that agents have, or the act of asking a question.

For this purpose, Inquisitive Semantics was created. Within this field lies the sub-field of Inquisitive Dynamic Epistemic Logic. Inquisitive Dynamic Epistemic Logic is a relatively new field that deals with knowledge, issues that agents have, and epistemic updates to that knowledge and those issues. While the field has shown to be very promising by creating conservative extensions for Epistemic Logic, Public Announcement Logic, and Action Model Logic, it cannot currently model actions with factual change, or (the effects of actions on) common knowledge and public issues.

In this thesis we combine these two forms of Dynamic Epistemic Logic into one unified framework. We will do this by first creating an inquisitive epistemic logic of relativized group knowledge, based on Propositional Dynamic Logic (PDL), which we will call Inquisitive Epistemic Propositional Dynamic Logic (IE-PDL). We will show that any Inquisitive Epistemic Logic formula can be translated into IE-PDL, and that it is sound and complete with respect to its semantics. This completeness proof differs from the standard construction in Inquisitive Epistemic Logic in that it only works for a finite number of worlds instead of the usual infinite construction.

After the creation of IE-PDL, we extend it with action models similar to Action Model Logic with Issues, which results in a logic which we call the Logic of Communication, Change, and Issues (LCCI). Unlike in Action Model Logic with Issues, these action models can also model factual change. We then show that LCCI can be reduced to IE-PDL using the idea of program transformers from the Logic of Communication and Change. We then used this reduction to show that LCCI is sound and complete with respect to finite models and that is is a conservative extension of Action Model Logic with Issues.

Acknowledgements

First of all, I would like to thank my supervisors, Rineke Verbrugge and Floris Roelofsen. Their comments helped shape not only the logics, but also the text of the thesis, and without their help I would have not been able to solve some of the problems that I encountered. I would like to thank them in particular for their patience, since this project has been going on for quite a bit longer than was planned.

Furthermore, I would like to thank Ivano Ciardelli and Thom van Gessel for sitting down and discussing my thesis with me. Their comments and questions have helped me greatly in improving my thesis. I would also like to thank Vít Punčochář for being willing to share his work with me. Seeing a different approach to the same problem was very insightful.

I would also like to thank Laura van de Braak, Jan van Houten, and Lotte Bouma for their comments on earlier versions of my thesis. Without their input, this thesis would not have been as clear or readable.

Lastly I would like to thank everyone that shared the gradation rooms with me, before and after the move. You guys provided me with nice distractions from time to time, and that made the whole process more pleasant. This goes in particular for Kim van Prooijen and Laura van de Braak, who have been at this for almost as long as I have.

Contents

1 Introduction **1**

2 Background **3**

2.1 Citadels . . . 3

2.2 Inquisitive Semantics . . . 4

2.2.1 Possible Worlds . . . 4

2.2.2 Information States . . . 5

2.2.3 Issues . . . 5

2.2.4 Support . . . 6

2.3 Inquisitive Epistemic Logic . . . 6

2.3.1 Syntax . . . 7

2.3.2 Semantics . . . 7

2.3.3 Entertaining and wondering . . . 8

2.3.4 Common knowledge and public issues . . . 8

2.3.5 Properties . . . 9

2.3.6 Resolutions . . . 10

2.4 Action Model Logic . . . 11

2.5 Logic of Communication and Change . . . 11

2.5.1 Propositional Dynamic Logic . . . 12

2.5.2 Substitutions . . . 13

2.5.3 Logic of Communication and Change . . . 14

2.6 AML and Inquisitive Semantics . . . 15

2.6.1 Syntax and Semantics . . . 16

2.6.2 Properties . . . 17

2.7 Structure of this Thesis . . . 18

3 Inquisitive Epistemic Propositional Dynamic Logic **19**

3.1 Introduction . . . 19

3.2 Syntax . . . 19

3.3 Semantics . . . 20

3.4 Examples . . . 22

3.5 Properties . . . 25

3.5.1 Resolutions . . . 28

3.5.2 Substitutions . . . 30

3.6 Reduction Axioms . . . 32

3.7 Translation from IEL . . . 36

3.8 Axiomatisation . . . 39

3.8.1 Entailment 39
3.8.2 Proof system 39
3.8.3 Completeness 41
3.9 Conclusion 57

4 Logic of Communication, Change, and Issues 58
4.1 Introduction 58
4.2 Definitions 58
4.2.1 Syntax 58
4.2.2 Update Models 59
4.2.3 Update Execution 59
4.2.4 Semantics 60
4.3 Examples 60
4.4 Properties 62
4.4.1 Declaratives 63
4.4.2 Resolution 64
4.5 Reduction 65
4.5.1 Atoms 65
4.5.2 Conjunction and Inquisitive Disjunction 66
4.5.3 Implication 67
4.5.4 Modalities 67
4.5.5 Full reduction 73
4.6 Axiomatisation 76
4.6.1 Completeness 77
4.7 Reduced Formulas 77
4.7.1 Differences with AMLI 78
4.7.2 Analysing Public Announcements 79
4.8 Conclusion 80

5 Comparison with other work 81
5.1 LCC and E-PDL 81
5.2 AMLI and IEL 82
5.3 InqPDL 82

6 Conclusion and further research 84
6.1 Conclusion 84
6.2 Further work 85
6.2.1 Implications for IEL 85
6.2.2 Extensions to IE-PDL 86
6.2.3 Uses for LCCI 86

A Implementation 88
A.1 The Hexa Game 88
A.1.1 The Game 88
A.1.2 Setting up 89
A.1.3 The Model 90
A.1.4 It's Time for Action 94
A.1.5 Other Uses of Formulas 97
A.2 Simplify 98
A.3 Conclusion 100

Bibliography **101**

Chapter 1

Introduction

In many situations in modern life we are dealing with situations where we have to reason about both the knowledge and issues of other people, and also about the changes that happen in the world. An example of such a situation would be the game of Citadels, also called Machiavelli in some parts of the world. In this game, the goal of the players is to build a city that gets them the most points by building several buildings. For this the players all get to pick a different character each round in order to use its abilities, which differ between the characters.

Because there are multiple characters and actions that players can take, the selection of a character is an important step in a round of Citadels. Just knowing what would be good for your city is not enough, since you also need to keep in mind what the other players need to achieve, and what their options are. The modelling of such a game could be done in a framework such as the Logic of Communication and Change (LCC) [18]. For this we need to model the knowledge relations for all the agents, and create update models for the different types of actions that can be taken in the game.

The only problem with this is that this leaves out a valuable part of the analysis, namely the issues that the different agents have at different points in time. For example, when an agent is presented with the choice whether or not to select the Condottiere as their character in order to destroy the building of one of the other players, they should instantly wonder whether or not that player has selected the Preacher as their character, which would block them from using their power. This issue can however not be represented within the framework of LCC.

A similar situation holds for the effect of certain actions. At the start of the game, it is not extremely relevant which cards get added to the closed hand of a fellow player. However, later on in the game the exact hand of each player can be quite relevant, since this can determine in how many rounds the game will end. In order to properly model such phenomena we will need to somehow enrich our models of knowledge.

For this, we can borrow from the field of Inquisitive Semantics [4], which has introduced questions in logic. Of particular interest in our case is Inquisitive Epistemic Logic (IEL) [2] and its action model variant Action Model Logic with Issues (AMLI) [22], since both these logics can already model knowledge, the issues that agents have, and announcements.

This thesis will set out to combine the ideas from LCC and AMLI into one framework that will incorporate the knowledge and issues of agents, announcements, and factual change. Since both these logics are generalizations of Action Model Logic (AML) [1] these ideas should be able to fit together quite nicely.

Along the way, we will also introduce an Inquisitive version of Epistemic Propositional Dynamic Logic (E-PDL). E-PDL is used in LCC as the basic epistemic language instead of the usual

S5 that IEL is modelled after. We will show that this new logic has a sound and complete axiomatisation, and that it can be used as a replacement for IEL.

The next chapter will start with an introduction into Citadels and the relevant background theories. The chapter will then end with the outline for the rest of this thesis.

Chapter 2

Background

First we will explain the rules of the game of Citadels, which we will use for the examples throughout this thesis. Then we will introduce the background ideas that go along with this thesis. These two ideas are the idea of Inquisitive Semantics [4], and the Logic of Communication and Change (LCC) [18]. We will end this chapter with an overview of how these ideas can be combined.

2.1 Citadels

Citadels is a card game in which the goal of the players is to build a set of eight buildings. They can do this over a number of rounds. During each round, the player plays as a certain character, which is chosen at the start of the round. There are eight characters in total. The game ends at the end of the first round in which at least one player has built their eight buildings.

Each of the buildings is worth a certain number of points. They also all have a colour, which is either yellow, blue, green, red, or purple. Building one of these buildings also costs a set number of coins, which is the same as the number of points they are worth. The player that has the most points is the player that wins. The players can also get bonus points for being the first to build their eight buildings, having eight buildings without being the first, or having one building of each colour.

The game is played in a number of rounds, of which the exact execution is dependent on the number of players. For simplicity's sake, we will limit ourselves to situations with three players, which we will call Alice (a), Bob (b), and Claire (c). In order to avoid a combinatorial explosion, we will also leave out three of the characters, one of the building colours (purple) and will constrain ourselves to a total of 12 buildings, three for each colour. Of these three, one is worth one point, one is worth three points, and the last is worth five points. An overview of the characters that we will use, and the order that they get to act in, can be found in Table 2.1.

A round starts with the starting player collecting and shuffling all the character cards. The starting player then takes the top card of the deck, looks at it, and places it on the table face down. The player then goes through the rest of the character cards and picks one. They then pass on the rest of the character cards to the second player, which then also picks a card, and hands the last two characters to the third player. This player picks a card, and puts the last remaining card face down on the table. This means that everybody has a character, but the players only know which character they themselves have.

After this, the players get to act in the order of their characters. So the first player to act is the player that has the character of the king. First they get to take either two coins, or one

Table 2.1: The roles in Citadels

Order	Role	Colour	Description
1	King	Yellow	The next starting player
2	Preacher	Blue	Buildings cannot be destroyed by the Condottiere
3	Merchant	Green	Gets one extra gold coin
4	Architect	—	Gets two extra building cards, can build two more buildings
5	Condottiere	Red	Can destroy one building each round for its cost minus one

building card. After this, they get to build a building from their hand. They can also use the special property that comes with their character at any point during their turn. If the character has an associated colour, they also get to get a number of coins equal to the number of buildings of that colour that they have build. After the turn of a character is done, the next character gets called and gets to act. In this way, the order of the characters is always the same, but the order of the players can differ, since they might have different characters each round. The last round is the round in which a player has built their eighth building. This round is finished as normal, so every player gets the same number of turns. Any player that has eight or more buildings cannot have any of their buildings destroyed.

2.2 Inquisitive Semantics

In general, logics and formal semantics are concerned with the truth and falsehood of statements. For single statements in two-valued logics this is not a problem, since most statements are either true or false, but for questions, this leads to problems. It is not possible to say whether a question is true or false, since a question does not necessarily carry information. This means that the standard definition that is used in logic for propositions will not work with questions.

For this purpose the framework of *Inquisitive Semantics* [4] was defined. This framework takes a slightly different look at sentences and propositions, and under this definition questions, or interrogative sentences, can be modelled in a natural way. The rest of this section will introduce these various notions.

2.2.1 Possible Worlds

In formal semantics, we normally discuss the truth or falsehood of a proposition with respect to a certain model. This model normally specifies a number of worlds, one of which is the actual world, while the other ones are alternatives for the actual world. These worlds are called the *possible worlds* of a model. The set of all possible worlds is traditionally denoted as $\mathcal{W}$. This is also sometimes called the *logical space*.

So if we, for example, wanted to model the propositions p: "It rains in Amsterdam", and q: "It rains in Groningen", we would need four worlds, one for each possible truth assignment. We could then draw this model as well, as seen in Figure 2.1.

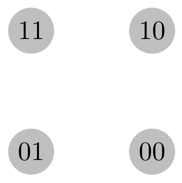


Figure 2.1: An example of a logical space, where the names of the world give the valuation for that world. In 11, both p and q are true, in 10 p is true, but q is false, etc.

2.2.2 Information States

The second notion that we need is the notion of an *information state*. If one of the possible worlds is the actual world, then an information state can be seen as a way to pinpoint the actual world by limiting the logical space.

Definition 2.1 (Information State)**.** An *information state* s is a subset of the possible worlds, $s \subseteq \mathcal{W}$.

We will also use the word *state* to refer to an information state.

Now, if we have two information states s and t, such that $t \subseteq s$, we then know that t holds at least as much information as s, since it locates the actual world with at least as much precision. This might seem counterintuitive, since t might contain fewer worlds, but this also means that it has a higher "signal-to-noise" ratio.

In our previous drawing of the possible worlds, we can also draw information states. If we for example wanted to show the information that it rains in Groningen, we could show that as in Figure 2.2.

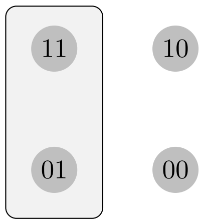


Figure 2.2: An example of an information state, namely the one that says that it rains in Groningen.

There is one information state that is special, namely $\emptyset$. In this information state all possible worlds have been discarded as candidates for the actual worlds. This means that the available information has become inconsistent. This is why we will also refer to it as the *inconsistent state*.

2.2.3 Issues

Issues are at the heart of Inquisitive Semantics, and they are used to actually model questions. The meaning of *issue* here is a point of discussion. For example this could be a question that was raised, or some other matter for which a solution has not yet been found. In Inquisitive semantics, an issue is represented by the situations in which it is *resolved*. That is to say, an issue is represented by all the information states that contain enough information to solve it. Since if s contains enough information to resolve the issue, any subset t will also contain enough

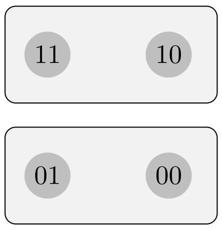


Figure 2.3: An example of an issue, namely the question "Does it rain in Amsterdam?"

information to resolve the issue, this means that if s is in the issue, t will also have to be contained in the issue.

Definition 2.2 (Issues)**.** An *Issue* is a non-empty set I of information states that is downward closed. This means that for all $s \subseteq \mathcal{W}$, if $s \in I$ and $t \subseteq s$, then $t \in I$. The set of all issues is denoted by $\mathcal{I}$.

Because issues are downward closed, if $\mathcal{W}$ is finite they can be defined in terms of their largest sets. For example, if we have the information state $\{w_1, w_2\}$, then we can construct an issue from this by taking the downward closure of the set $\{\{w_1, w_2\}\}$, which would be $\{\{w_1, w_2\}, \{w_1\}, \{w_2\}, \emptyset\}$. This operation is also written down as $^{\downarrow}$. So we could write the previous issue as $\{\{w_1, w_2\}\}^{\downarrow}$.

Since an issue can be defined in terms of its biggest elements, these elements also have a special name:

Definition 2.3 (Alternatives)**.** The maximal elements of an issue I are called its *alternatives*. The maximal elements are the elements that are not a subset of another element in I, $s \in I$ such that there is no t: $t \in I$ and $s \subset t$.

Like possible worlds and information states, Issues can also be shown graphically. An example of this can be seen in Figure 2.3, where the question "Does it rain in Amsterdam?" is represented. When drawing an issue, we only draw the alternatives to keep the figures free from clutter.

2.2.4 Support

As mentioned before, it is impossible to see if an issue is true or false in a given world. Because of this, we will use a different notion instead of truth, namely the notion of support. We will say that an information state s supports a formula φ in a model M in the case that it resolves the issue of φ. The notation for this stays the same, $M, s \models \varphi$.

For example, in Figure 2.3, if we take φ to be "Does it rain in Amsterdam?", then the upper alternative resolves the issue, since it tells us that it rains in Amsterdam. We would write this down as $M, \{\{11, 10\}\}^{\downarrow} \models \varphi$.

2.3 Inquisitive Epistemic Logic

An example of a logic based on Inquisitive Semantics is Inquisitive Epistemic Logic (IEL). IEL is similar to standard Epistemic Logic, but here it is also possible to model the issues that the agents entertain.

This section will give a short introduction into IEL, based on the work in [6].

2.3.1 Syntax

The syntax of IEL is similar to the syntax of Epistemic Logic, but with two small changes. The first of these is the introduction of the *inquisitive or* ($\mathbin{\rotatebox[origin=c]{-90}{$\geqslant$}}$) and the second is the addition of an *entertain* (E_a) modality.

The new *inquisitive or* is what we will use to introduce issues into our new language. If we only kept the classical operators, we would only be able to create classical structures.

The *entertain operator* works as a standard modal operator, but instead of going from worlds to information states, this one goes from worlds to issues. This in turn allows us to express what an agent entertains, instead of only what an agent knows. More explanation of how the entertain operator works can be found in Section 2.3.3.

Definition 2.4 (Formulas of IEL)**.** Given a finite set of propositional variables $\mathcal{P}$, and a finite set of agents $\mathcal{A}$, with p ranging over $\mathcal{P}$ and a ranging over $\mathcal{A}$, the formulas of IEL are given by:

$$\varphi := \bot \mid p \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \to \varphi_2 \mid \varphi_1 \mathbin{\rotatebox[origin=c]{-90}{$\geqslant$}} \varphi_2 \mid K_a\varphi \mid E_a\varphi$$

For the language, we will also use the following abbreviations:

$$\begin{aligned} \neg\varphi &:= \varphi \to \bot \\ \varphi_1 \vee \varphi_2 &:= \neg(\neg\varphi_1 \wedge \neg\varphi_2) \\ \varphi_1 \leftrightarrow \varphi_2 &:= (\varphi_1 \to \varphi_2) \wedge (\varphi_2 \to \varphi_2) \\ ?\varphi &:= \varphi \mathbin{\rotatebox[origin=c]{-90}{$\geqslant$}} \neg\varphi \end{aligned}$$

To continue on our earlier weather examples, we could now represent the issue in Figure 2.3 as $?p$, which means that we can represent that Alice (a) entertains that it rains in Amsterdam as $E_a?p$.

2.3.2 Semantics

The structure of models for IEL is similar to the structure of models for standard epistemic logic. The big difference is that the inquisitive state of an agent needs to be incorporated as well. This is done by assigning to each agent in a world an issue that that agent entertains in that world instead of the usual set of worlds that are indistinguishable. Because IEL is meant to treat knowledge, we would also like these inquisitive states to have properties similar to reflection, transitivity, and symmetry. The first one is now called factivity, and the second two are put together into introspection, to add the axiom for truth and introspection, respectively.

Definition 2.5 (Models for IEL)**.** An Inquisitive Epistemic Logic model for a finite set $\mathcal{P}$ of propositional variables and a set $\mathcal{A}$ of agents is a triple $M = \langle \mathcal{W}, \Sigma_{\mathcal{A}}, V\rangle$, where:

- $\mathcal{W}$ is a set of possible worlds,
- $\Sigma_{\mathcal{A}} = \{\Sigma_a \mid a \in \mathcal{A}\}$ is a set of *state maps*, each of which assigns to every world w an issue $\Sigma_a(w)$ such that it has the following properties:

  Factivity: for any $w \in \mathcal{W}$, $w \in \sigma_a(w)$

  Introspection: for any $w, v \in \mathcal{W}$, if $v \in \sigma_a(w)$, then $\Sigma_a(w) = \Sigma_a(v)$

  where $\sigma_a(w) = \bigcup \Sigma_a(w)$ represents the *information state* of the agent a in world w.
- $V : \mathcal{W} \to \wp(\mathcal{P})$[1] is a *valuation map*,

[1] $\wp(P)$ is the powerset of P, so a set of all subsets of P.

The conditions put on the state maps are necessary in order to ensure that the agent's information states are truthful, and that the agents are aware of their own knowledge and issues. This is similar to the conditions put on the knowledge relations in standard Epistemic Logic.

Now we can give the definition of the interpretation of IEL sentences.

Definition 2.6 (Interpretation of IEL formulas)**.** Let M be an IEL model and s an information state in M.

$$\begin{aligned}
&M, s \models p \iff p \in V(w) \text{ for all worlds } w \in s\\
&M, s \models \bot \iff s = \emptyset\\
&M, s \models \varphi_1 \wedge \varphi_2 \iff M, s \models \varphi_1 \text{ and } M, s \models \varphi_2\\
&M, s \models \varphi_1 \mathbin{\vee\!\!\!\vee} \varphi_2 \iff M, s \models \varphi_1 \text{ or } M, s \models \varphi_2\\
&M, s \models \varphi_1 \to \varphi_2 \iff \text{ for every } t \subseteq s,\ M, t \models \varphi_1 \text{ implies } M, t \models \varphi_2\\
&M, s \models K_a\varphi \iff \text{ for every } w \in s,\ M, \sigma_a(w) \models \varphi\\
&M, s \models E_a\varphi \iff \text{ for all } w \in s,\ \text{for all } t \in \Sigma_a(w),\ M, t \models \varphi
\end{aligned}$$

While truth is a derived notion in Inquisitive Semantics, we can still give a definition for the truth of a IEL formula in a world.

Definition 2.7 (Truth in IEL)**.** Let M be an IEL model and w be a world in M.

$$M, w \models \varphi \iff M, \{w\} \models \varphi$$

2.3.3 Entertaining and wondering

The entertain modality E_a is used in IEL to express that an agent a wants to know something. This is done by making $E_a\varphi$ supported in those cases where φ is supported in all the states in the inquisitive state of agent a. This works well for most cases. For example, if we look at $E_a?\varphi$ this is supported if in all the worlds in the inquisitive state of agent a, either φ or $\neg\varphi$ is supported.

However, the current entertain modality is not without its problems. A peculiar property that it has is that if an agent knows whether φ is true or false, they also entertain whether or not φ. This is because of how $K_a\varphi$ is interpreted. If we have that $M, s \models K_a\varphi$, we know that for all $w \in s$, $M, \sigma_a(w) \models \varphi$. This means that φ is supported in all worlds in $\sigma_a(w)$, and therefore also in all issue states in $\Sigma_a(w)$. Therefore we get that $K_a\varphi$ implies $E_a\varphi$. This is not a good definition of the natural language concept of entertaining.

Therefore, the entertain modality is mostly used in a technical sense, and for most cases we will actually use a new modality, called the wonder modality. The wonder modality is an abbreviation that will give us exactly what we need to specify that an agent wants to know something, but does not know it yet. It is defined in the following manner:

$$W_a\varphi := \neg K_a\varphi \wedge E_a\varphi$$

So if an agents wonders whether φ is supported, we can now express that in the following way: $W_a?\varphi$.

2.3.4 Common knowledge and public issues

The language of IEL can also be enriched by the notions of common knowledge, like standard epistemic logic, and its inquisitive counterpart, public issues. Since we are not working with relations, this cannot be done in the standard way, by taking the reflexive transitive closure of the union of relations for the agents. However, it is possible to do something similar. Consider public issues. Something is a public issue if both:

- all the agents entertain it;
- each agent knows that the others entertain it.

This can be seen as iteratively finding all the worlds that a number of agents consider, so including the ones the other agents in the group "believe" the others consider, and see which issues they have in this worlds. An issue would then be a public issue if it is an issue for all agents in those worlds. This is leads to the following definition of a new state map, called Σ_*, which is used for the interpretation for public issues:

Definition 2.8 (Definition of Σ_*)**.**

$$\begin{aligned}\Sigma_*(w) = \{s \mid &\text{there exist } v_0, \dots v_n \in \mathcal{W} \text{ and } a_0, \dots a_n \in \mathcal{A} \text{ such that} \\ &v_0 = w, v_{i+1} \in \sigma_{a_i}(v_i) \text{ for all } i < n, \text{ and } s \in \Sigma_{a_n}(v_n)\}\end{aligned}$$

We now also get a map called σ_* for the knowledge relations that correspond to what we would get if we took the common knowledge construction on the individual information maps σ_a. This map is defined in the usual way $\sigma_*(w) = \bigcup \Sigma_*(w)$.

Now two new operators can be defined, called K_* for common knowledge, and E_* for public issues. The support definitions for these are as follows.

$$\begin{aligned}M, s \models K_*\varphi &\iff \text{for every } w \in s,\ M, \sigma_*(w) \models \varphi \\ M, s \models E_*\varphi &\iff \text{for every } w \in s,\ \text{for every } t \in \Sigma_*(w),\ M, t \models \varphi\end{aligned}$$

With K_* and E_* we can also define an operator W_* that says that a group of agents publicly entertain φ and that φ is not publicly settled.

$$W_*\varphi := \neg K_*\varphi \wedge E_*\varphi$$

It should be noted, however, that while there is sound and complete axiomatisation for IEL, this axiomatisation is no longer complete when we introduce common knowledge and public issues into the language. We will have more to say about this in the conclusion.

2.3.5 Properties

Since IEL is the basic logic on which most other logics in this thesis are based, it is a good idea to look at the properties of IEL. Later on we will see if these properties also hold for the other logics. All of these come from [3].

Fact 2.1 (Persistency of support)**.** For all models M and all information states $s \subseteq \mathcal{W}$, for all $t \subseteq s$, if $M, s \models \varphi$ then $M, t \models \varphi$.

Fact 2.2 (Empty State Property)**.** For all models M and all formulas φ, $M, \emptyset \models \varphi$.

This second property is also why $\emptyset$ is called the inconsistent state, since everything is supported there.

There is a set of formulas in IEL that have the special property of being *truth-conditional.* This means that the support conditions in a state s boil down to the truth conditions in each $w \in s$.

Definition 2.9 (Truth-Conditional)**.** A formula φ is *truth-conditional* if for all models M and information states s, $M, s \models \varphi \iff M, w \models \varphi$ for all $w \in s$.

For some formulas it is possible to see that they are truth-conditional based on their syntax. We call these formulas the *declaratives*. Any formula that is not declarative will be called *interrogative*. The notation for the declarative fragment of a language $\mathcal{L}$ is $\mathcal{L}_!$.

In the rest of this thesis, we will us the following notational convention: α, β, γ will range over declaratives, μ, ν, λ will range over interrogatives, and φ, ψ, χ will range over the (then relevant) whole language.

Definition 2.10 (Declaratives of IEL)**.** The declarative fragment of IEL is given by:

$$\alpha := p \mid \bot \mid K_a\varphi \mid E_a\varphi \mid \alpha_1 \wedge \alpha_2 \mid \varphi \to \alpha$$

Fact 2.3. Any $\alpha \in \mathcal{L}_!^{\mathsf{IEL}}$ is truth-conditional.

Another nice property that declaratives have is that given a declarative α, $K_a\alpha$ and $E_a\alpha$ are equivalent.

Fact 2.4. For all models M and information states $s \subseteq \mathcal{W}$, given any declarative $\alpha \in \mathcal{L}_!^{\mathsf{IEL}}$, the following holds:

$$M, s \models K_a\alpha \iff M, s \models E_a\alpha$$

2.3.6 Resolutions

Resolutions form a notion that will play an important role in tying an interrogative sentence to a declarative counterpart. Intuitively, resolutions can be seen as declarative sentences that describe the different ways in which a formula can be settled.

Definition 2.11 (Resolutions in IEL)**.** The set $\mathcal{R}(\varphi)$ of resolutions of a formula φ is defined recursively as follows:

$$\begin{aligned}
&\mathcal{R}(\alpha) = \{\alpha\} \text{ if } \alpha \text{ is a declarative}\\
&\mathcal{R}(\varphi_1 \mathbin{\vee\!\!\!\vee} \varphi_2) = \mathcal{R}(\varphi_1) \cup \mathcal{R}(\varphi_2)\\
&\mathcal{R}(\mu \wedge \nu) = \{\alpha \wedge \beta \mid \alpha \in \mathcal{R}(\mu) \text{ and } \beta \in \mathcal{R}(\nu)\}\\
&\mathcal{R}(\varphi \to \mu) = \left\{ \bigwedge_{\alpha \in \mathcal{R}(\varphi)} \alpha \to f(\alpha) \mid f \text{ is a function from } \mathcal{R}(\varphi) \text{ to } \mathcal{R}(\mu) \right\}
\end{aligned}$$

The use of resolutions also gives us some new facts. The first says that to resolve an interrogative is to establish some resolution of it.

Fact 2.5. For any M, s, and φ, $M, s \models \varphi \iff M, s \models \alpha$ for some $\alpha \in \mathcal{R}(\varphi)$

This also leads to the second result, which defines a normal form: every formula φ is equivalent to an interrogative made up of its resolutions.

Fact 2.6 (Normal form)**.** For any φ, $\varphi \iff \mathbin{\vee\!\!\!\vee}_{\alpha \in \mathcal{R}(\varphi)} \alpha$

Using the resolutions, the *presupposition* of an interrogative can be defined.

Definition 2.12 (Presupposition of an interrogative)**.** The presupposition of an interrogative μ is the declarative $\bigvee_{\alpha \in \mathcal{R}(\mu)} \alpha$.

We can also generalize this notion of resolutions to sets of formulas, which will come in useful later.

Definition 2.13 (Resolutions of a set)**.** The set $\mathcal{R}(\Phi)$ of resolutions of a set Φ contains those sets Γ of declaratives such that:

- for all $\varphi \in \Phi$ there is an $\alpha \in \Gamma$ such that $\alpha \in \mathcal{R}(\varphi)$.
- for all $\alpha \in \Gamma$ there is a $\varphi \in \Phi$ such that $\alpha \in \mathcal{R}(\varphi)$.

This means to say that the resolutions of a set Φ is a set of declaratives which is obtained by replacing each formula in Φ by one or more resolutions. Since declaratives have themselves as their only resolutions, we get that the declaratives in Φ are also in every $\Gamma \in \mathcal{R}(\Phi)$. In particular, if Γ is a set of declaratives, then then $\mathcal{R}(\Gamma) = \Gamma$.

Fact 2.5 also generalizes to sets. Here $M, s \models \Phi$ where Φ is a set of formulas means $M, s \models \varphi$ for all $\varphi \in \Phi$.

Fact 2.7. For any M, s, and Φ, $M, s \models \Phi \iff M, s \models \Gamma$ for some $\Gamma \in \mathcal{R}(\Phi)$.

2.4 Action Model Logic

This section will give a quick, mostly conceptual, understanding of Action Model Logic (AML) [1]. The ideas presented in this section will be expanded upon in Sections 2.5 and 2.6 and can help in their understanding.

The main idea behind action model logic is to extend a standard epistemic logic, which is called the static language, with one new operator, $[\mathsf{U}, \mathsf{x}]\,\varphi$, that takes an action x and an action model U, whose intuitive reading is: after the execution of x, the formula φ holds. Formally, this operator is interpreted by augmenting the static language model M that $[\mathsf{x}]\,\varphi$ is interpreted in with the given action model U, which results in a new model called $M \circ \mathsf{U}$ in which the formula φ is then evaluated.

Of course, not every action can be executed in every world. For example, if Alice is not the architect, then she does not get to build more than one building in a turn. Because of this, every action x has a so called pre-condition, written as $\mathsf{pre}(\mathsf{x})$. All the preconditions for the actions are stored in the model U.

It is also not the case that every agent knows about the execution of every action. For example, if Claire gets to pick a character, then only she will know which character she picked. However, Bob will have more knowledge about which action occurred than that Alice has. Because of this, there is also a relation in U for each agent a, called R_a, that relates two events x and y if they are indistinguishable to agent a. This relation is similar to the relation R_a from the static language. So for example, if the static language is $\mathsf{S5}$, then it is normal to make sure that R_a is also reflexive, transitive, and symmetric.

The model $M \circ \mathsf{U}$ should be a valid model for the static language, and has as worlds ordered pairs of worlds and actions, where a propositional atom is in the valuation of world (w, x) if it is in the valuation of w. The new relation R'_a is created by combining the relations R_a and R_a, where two worlds (w, x) and (v, y) are related if agent a cannot distinguish between w and v, nor between x and y. This should give a complete model for the static language.

2.5 Logic of Communication and Change

The *Logic of Communication and Change* (LCC) is a logic that was introduced in [18]. It is a logic that combines announcements, both public and private, factual change, and epistemic operators. It achieves this by building a generalization of AML which allows for *substitutions* and which uses Propositional Dynamic Logic (PDL) as its static language. This section of the background will start out with an introduction into PDL, followed by an explanation of substitutions. The section ends with an explanation of the workings of LCC.

Since PDL also uses the word 'action' and we want to avoid ambiguity, we will refer to action models as *update models*, and the actions in such a model as *events* for the rest of this section.

2.5.1 Propositional Dynamic Logic

Propositional Dynamic Logic (PDL) is a modal logic which allows us to reason about the execution of sequences of actions, usually called a program [12]. This works by having a set of atomic programs that can influence the state of the world. This is modelled by having a modal operator that takes programs, where the relation for each program represents the possible worlds after the execution of that program. These atomic programs can also be combined into larger programs, using operators to combine the atomic programs.

The language of PDL is defined by mutual recursion over formulas and programs. The structure of formulas is mostly the same as that for propositional logic, with the addition of a modal program operator $[\pi]\varphi$. This formula says that φ is true after the execution of the program π.

For the creation of programs we also have several operators. The first of these is $\pi_1;\pi_2$, which is called *sequence*, and executes program π_2 after π_1. This is used to execute multiple complex programs in sequence, or to embed sequences of programs in other larger programs. The second is $\pi_1 \cup \pi_2$, which is called *choice*, and executes either π_1 or π_2 non-deterministically. This is useful to model branching paths in programs. The operator '$?\varphi$' is called *test*, and this operator tests if a formula φ holds in the current world, and aborts the program otherwise. The last operator is π^* which is called *unbounded iteration*, and executes the program π zero or more times. This is used to execute a program multiple times.

Definition 2.14 (Formulas of PDL)**.** Let a finite set of propositional variables $\mathcal{P}$ and a set of atomic programs $\mathcal{A}$ be given, with p ranging over $\mathcal{P}$ and a ranging over $\mathcal{A}$. The language of PDL is given by:

$$\varphi := \top \mid p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid [\pi]\varphi$$
$$\pi := a \mid \varphi? \mid \pi_1;\pi_2 \mid \pi_1 \cup \pi_2 \mid \pi^*$$

We employ the usual abbreviations:

$$\bot := \neg\top \qquad \varphi_1 \vee \varphi_2 := \neg(\neg\varphi_1 \wedge \neg\varphi_2)$$
$$\varphi_1 \rightarrow \varphi_2 := \neg(\varphi_1 \wedge \neg\varphi_2) \qquad \varphi_1 \leftrightarrow \varphi_2 := (\varphi_1 \rightarrow \varphi_2) \wedge (\varphi_2 \rightarrow \varphi_1)$$

These formulas can be interpreted in relation to standard multimodal Kripke models, with a relation for each atomic program.

Definition 2.15 (Models of PDL)**.** A Propositional Dynamic Logic model for a finite set $\mathcal{P}$ of propositional variables and a set $\mathcal{A}$ of atomic programs is a triple $M = \langle \mathcal{W}, R_{\mathcal{A}}, V \rangle$, where:

- $\mathcal{W}$ is a set of possible worlds
- $R_{\mathcal{A}} = \{R_a \mid a \in \mathcal{A}\}$ is a set of relations, $R_a : \mathcal{W} \times \mathcal{W}$, where every relation gives the worlds that are possible after the execution of action a
- $V : \mathcal{W} \rightarrow \wp(\mathcal{P})$

Currently, the relations are only given for the atomic programs. The relations for the composite programs are as follows:

$$\begin{aligned} R_{\pi_1;\pi_2} &:= R_{\pi_1} \circ R_{\pi_2} \\ R_{\pi_1 \cup \pi_2} &:= R_{\pi_1} \cup R_{\pi_2} \\ R_{\varphi?} &:= \{(w,w) \mid M, w \models \varphi\} \\ R_{\pi^*} &:= (R_\pi)^* \end{aligned}$$

where $R_{\pi_1} \circ R_{\pi_2}$ is the composition of R_{π_1} and R_{π_2}, and R^* is the reflexive transitive closure of R.

Now we can give the truth definitions for PDL formulas in models.

Definition 2.16 (Truth definitions for PDL)**.** Let M be an PDL models and w a world in M.

$$\begin{aligned} M, w \models \top &\iff \text{true} \\ M, w \models p &\iff p \in V(w) \\ M, w \models \neg\varphi &\iff M, w \not\models \varphi \\ M, w \models \varphi_1 \wedge \varphi_2 &\iff M, w \models \varphi_1 \text{ and } M, w \models \varphi_2 \\ M, w \models [\pi]\varphi &\iff \text{for every } wR_\pi v, M, v \models \varphi \end{aligned}$$

In LCC, instead of using PDL for its usual purpose, as a language for analysing the execution of programs, PDL is used as a rich epistemic language. This does not change any of the behaviours of the language, but only our interpretation of it. In order to make this distinction clearer, this epistemic reading is called E-PDL.

In E-PDL, the relations for atomic programs become the accessibility relations for agents, and are in that way similar to the state maps from IEL. Sequences like $a_1; a_2$ will be the 'levels of knowledge' of Parikh [14]. If a_1 and a_2 are agents, and φ is a formula of E-PDL, then $[a_1; a_2]\varphi$ means that a_1 knows that a_2 knows φ. Furthermore, if $A \subseteq \mathcal{A}$, then we will use $[\bigcup A]\varphi$ as a shorthand for $[a_1 \cup \cdots \cup a_n]\varphi$. In this way, we can express the common knowledge of a group of agents as $[(\bigcup A)^*]\varphi$, while general knowledge of φ becomes $[\bigcup A]\varphi$.

2.5.2 Substitutions

The substitutions are the part of the update models that drive the factual change. They can be seen as postconditions of an event that changes the world.

Definition 2.17 (Substitutions)**.** $\mathcal{L}$ substitutions are functions of type $\mathcal{L} \to \mathcal{L}$ that distribute over all language constructs, and that map all but a finite number of basic propositional atoms to themselves. $\mathcal{L}$ substitutions can be represented as a set of bindings

$$\{p_1 \mapsto \varphi_1, \ldots, p_n \mapsto \varphi_n\}$$

where all the p_i are different. If ρ is a $\mathcal{L}$ substitution, then the set $\{p \mid p \in \mathcal{P}, \rho(p) \neq p\}$ is called the *domain* of ρ, denoted $dom(\rho)$. Use ϵ for the identity substitution. Let $SUB_{\mathcal{L}}$ be the set of all $\mathcal{L}$ substitutions.

We can also update a model with a substitution. For this, we only have to change the valuation, so we will focus on that.

Definition 2.18 (Epistemic Models under a substitution)**.** If $M = \langle \mathcal{W}, V, R \rangle$ is an epistemic model and ρ is a $\mathcal{L}$ substitution (for an appropriate epistemic language $\mathcal{L}$), then V^ρ_M is the valuation given by

$$V^\rho_M(p) = \{w \mid M, w \models \rho(p)\}$$

In other words, V^ρ_M assigns to p the set of worlds w where $\rho(p)$ is true. For $M = \langle \mathcal{W}, V, R \rangle$, call M^ρ the model given by $\langle \mathcal{W}, V^\rho_M, R \rangle$.

The substitutions will be the driving force behind the factual changes, since they allow us to model the changes in the world. To see this, we can have a look at the event of a player in Citadels taking two coins at the start of their turn. For this substitution, we would want to map the proposition that a player has x coins to the proposition that a player has $x - 2$ coins. If we take $c(a, i)$ to mean that player a has at least i coins, this substitution would look like this:

$$\{c(a, 1) \mapsto \top, c(a, 2) \mapsto \top, c(a, i) \mapsto c(a, i - 2) \text{ where } 2 < i \leq 10\}$$

This means that after the application of this substitution, $c(a, 1)$ and $c(a, 2)$ are true, $c(a, 3)$ has the truth value that $c(a, 1)$ had before and so on.

2.5.3 Logic of Communication and Change

The Logic of Communication and Change (LCC) [18] is an action model logic that uses PDL as its static language. Its syntax is the same as that of PDL, with the addition of an update operator $[\mathsf{U}, \mathsf{e}]\,\varphi$, which means that after the execution of event e from update model U, φ is true.

In that respect LCC is not much different from AML, but it differs on two accounts. The first of these is that LCC allows for the addition of factual change to action models. While this was already mentioned in [1], the mechanisms for it were not yet incorporated into AML. The second difference is that LCC has a neater completeness proof, since it is in *dynamic-static* harmony thanks to the fact that it uses epistemic programs instead of the normal epistemic operators for knowledge in groups. This also makes it easier to study the effects of epistemic actions on aspects like general knowledge and common belief.

The update models from LCC are like action models from AML, but with an additional function sub to assign a substitution to each event.

Definition 2.19 (Update Models)**.** An update model for a finite set of agents $\mathcal{A}$ with a language $\mathcal{L}$ is a quadruple $\mathsf{U} = \langle \mathsf{E}, \mathsf{R}, \mathsf{pre}, \mathsf{sub} \rangle$ where:

- $\mathsf{E} = \{\mathsf{e}_1, \ldots, \mathsf{e}_n\}$ is a finite non-empty set of events
- $\mathsf{R} : \mathcal{A} \to \wp(E^2)$ assigns an accessibility relation $\mathsf{R}(a)$ to all agents $a \in \mathcal{A}$
- $\mathsf{pre} : E \to \mathcal{L}$ assigns a precondition to each event
- $\mathsf{sub} : E \to SUB_{\mathcal{L}_{\mathsf{PDL}}}$ assigns a substitution to each event

A pair U, e is an update model with a distinguished event $\mathsf{e} \in \mathsf{E}$.

These update models are then combined with the underlying static language, to create a new PDL model. This process is called update execution. The definition of the resulting model is given below.

Definition 2.20 (Update Execution)**.** Given a static epistemic model $M = \langle \mathcal{W}, V, R \rangle$, a world $w \in \mathcal{W}$, an update model $\mathsf{U} = \langle \mathsf{E}, \mathsf{R}, \mathsf{pre}, \mathsf{sub} \rangle$ and an event $\mathsf{e} \in \mathsf{E}$ with $M, w \models \mathsf{pre}(\mathsf{e})$, we say that the result of *executing* U, e in M, w is the model $M \circ \mathsf{U}, (w, \mathsf{e}) = \langle \mathcal{W}', V', R' \rangle, (w, \mathsf{e})$ where

- $\mathcal{W}' = \{(v, \mathsf{f}) \mid M, v \models \mathsf{pre}(\mathsf{f})\}$
- $V'(p) = \{(v, \mathsf{f}) \in \mathcal{W}' \mid M, v \models \mathsf{sub}(\mathsf{f})(p)\}$
- $R'_a = \{((v, \mathsf{f}), (u, \mathsf{g})) \mid (v, u) \in R_a \text{ and } (\mathsf{f}, \mathsf{g}) \in \mathsf{R}(a)\}$

The interpretation of LCC functions is also the same as the interpretation of PDL formulas with the following addition:

$$M, w \models [\mathsf{U}, \mathsf{e}]\,\varphi \iff M \circ \mathsf{U}, (w, \mathsf{e}) \models \varphi$$

As mentioned before, one of the nice properties of LCC is that it is in dynamic-static harmony, which means that the language is completely reducible to the underlying static language. This reduction is done by so called *program transformers*. These program transformers take an epistemic program that is supposed to be run in the updated model, and transform it into a program in the original model.

The base idea behind the program transformers is that if the formula $[\mathsf{U}, \mathsf{e}]\,[\pi]\varphi$ is true in some model M and world w, then φ is true in each world on all π-paths in $M \circ \mathsf{U}$. This means that there is some path $w \cdots v$ in M and some path $\mathsf{e} \cdots \mathsf{f}$ in U where each event $\mathsf{e_i}$ in $\mathsf{e} \cdots \mathsf{f}$ can be executed in the corresponding world w_i in $w \cdots v$. These paths can be specified using programs, as is normal in PDL, and the test whether an event can be executed can be done using the test operator. This makes it possible to find an equivalent formula to $[\mathsf{U}, \mathsf{e}]\,[\pi]\varphi$ that does not contain any action operators, which in turn allows for a full reduction of LCC into E-PDL.

This also explains why LCC needs E-PDL as its static language instead of classical epistemic logic, since it needs to have an operator for iteration, to enable common knowledge, as well as an operator to test if a formula is true, which needs to be able to be nested under iteration. As E-PDL adheres to both these constraints, it is a good fit for a static language for LCC.

2.6 Action Model Logics and Inquisitive Semantics

It is also possible to enrich Inquisitive Epistemic Languages, such as IEL, with action models. For this purpose two languages that combine IEL with action models were developed in [22]. Since we want to use one of these logics as our basis for our own update models, we will quickly go over the differences.

The two logics in question are *Action Model Logic with Questions* (AMLQ) and *Action Model Logic with Issues* (AMLI). AMLQ works a lot like AML, but also allows actions to have a question as a precondition, thereby allowing the logic to model the asking of a question. Since it changes nothing about the relations between actions, after an action every agent entertains which question was asked.

AMLI works differently, since it allows us to specify which agents are interested in which actions. The downside of this is that the logic no longer allows the asking of a question as an action. However, we can model the asking of a question as an action in AMLI by making multiple actions, one for each alternative in the question.

Since we want our action models to be able to deal with arbitrary actions, and not just announcements, being able to model the interest of agents in the different actions is important. Therefore, we will base our update models on AMLI. The rest of this section will give a quick overview of AMLI, based on the work in [22].

AMLI is an action model logic which uses IEL as its static language. This means that this section will also borrow from Section 2.3 for some definitions.

2.6.1 Syntax and Semantics

The syntax and semantics of AMLI are defined by mutual recursion. In order to also allow for AMLI formulas in preconditions, the logic is also defined in multiple levels, where each level allows for the nesting of more complex action models.

Definition 2.21 (Syntax of AMLI)**.** Let $\mathcal{L}^{\mathsf{AMLI}_0} = \mathcal{L}^{\mathsf{IEL}}$. For $i > 0$, $\mathcal{L}^{\mathsf{AMLI}_i}$ is defined as follows, where s is a set of actions within the AMLI action model M of at most level $i-1$.

$$\varphi := p \mid \bot \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \mathbin{\rotatebox[origin=c]{90}{$\geqslant$}} \varphi_2 \mid \varphi_1 \to \varphi_2 \mid K_a\varphi \mid E_a\varphi \mid [\mathsf{M}, \mathsf{s}]\varphi$$

The full language is the union of all $\mathcal{L}^{\mathsf{AMLI}_i}$ for all natural numbers i.

$$\bigcup_{i \geq 0} \mathcal{L}^{\mathsf{AMLI}_i}$$

We use the same abbreviations as for IEL with the following additions:

$$[\mathsf{s}]\varphi := [\mathsf{M}, \mathsf{s}]\varphi \qquad \text{Where } \mathsf{M} \text{ is clear from context}$$
$$[\mathsf{x}]\varphi := [\{\mathsf{x}\}]\varphi \qquad \text{Where } \mathsf{x} \text{ is a single action}$$

Because we are now working with a set of actions, the intuitive reading for $[\mathsf{s}]\,\varphi$ becomes a bit different. It now means: "After the execution of some action in s, φ is supported." The reading of $[\mathsf{x}]\,\varphi$ stays the same, since there is only one action x, so it is the only one that can be executed.

The Action Models with Issues are defined as follows.

Definition 2.22 (Action Models with Issues)**.** For $i \geq 0$, an AMLI action model of level i is a triple $\mathsf{M} = \langle \mathsf{S}, \Delta_{\mathcal{A}}, \mathsf{pre} \rangle$, where:

- S is a finite domain of actions.
- $\Delta_{\mathcal{A}} = \{\Delta_a \mid a \in \mathcal{A}\}$ where Δ_a maps every action to a non-empty downward closed set of sets of actions.
- $\mathsf{pre} : S \to \mathcal{L}_!^{\mathsf{AMLI}_i}$ is a function that assigns a precondition $\mathsf{pre} \in \mathcal{L}_!^{\mathsf{AMLI}_i}$ to each action $\mathsf{x} \in \mathsf{S}$.

Since we want the updated model to be an IEL model, there are a lot of similarities between $\Delta_{\mathcal{A}}$ and $\Sigma_{\mathcal{A}}$, since they serve similar goals. In particular, the factivity and introspection conditions apply to $\Delta_{\mathcal{A}}$, as they did to $\Sigma_{\mathcal{A}}$ in IEL. For each $a \in \mathcal{A}$, there is also a $\delta_a = \bigcup \Delta_a$, similar to σ_a in IEL.

These action models can then be combined with IEL models to get the model after execution. For this we will want two functions to map states in the updated model to their state in the IEL model or the Action Model.

Definition 2.23 (Projection operators)**.** If $s \subseteq (\mathcal{W} \times \mathsf{S})$, then:

$$\pi_1(s) = \{w \mid (w, \mathsf{x}) \in s \text{ for some } \mathsf{x}\}$$
$$\pi_2(s) = \{\mathsf{x} \mid (w, \mathsf{x}) \in s \text{ for some } w\}$$

The definition of the updated model is a follows.

Definition 2.24 (Product Update)**.** Let M be an IEL model and M an AMLI action model. Then $M \circ \mathsf{M}$ is the product update of M and M, defined as the triple $\langle \mathcal{W}', V', \Sigma'_{\mathcal{A}} \rangle$, where:

- $\mathcal{W}' = \{(w, \mathsf{x}) \mid M, w \models \mathsf{pre}(\mathsf{x})\}$
- $V'(p) = \{(w, \mathsf{x}) \mid w \in V(p), \mathsf{x} \in \mathsf{S}\}$
- $\Sigma'_{\mathcal{A}} = \{\Sigma'_a \mid a \in \mathcal{A}\}$ where $s \in \Sigma'_a((w, \mathsf{x}))$ iff
    1. $\pi_1(s) \in \Sigma_a(w)$
    2. $\pi_2(s) \in \Delta_a(\mathsf{x})$

Since we work with states and not worlds in Inquisitive Semantics, we will need to tie our states in the original model to states in our updated model. For this the definition of an *updated state* is used.

Definition 2.25 (Updated State). Given an IEL model M, an information state s in M, an AMLI action model M, and a set of actions s in M, the updated state $s[\mathsf{M}, \mathsf{s}]$ is the information state in $M \circ \mathsf{M}$ such that:

$$s[\mathsf{M}, \mathsf{s}] = \{(w, \mathsf{x}) \in \mathcal{W}' \mid w \in s, \mathsf{x} \in \mathsf{s}\}$$

The support conditions for the formulas are the same as for IEL with the following addition:

$$M, s \models [\mathsf{M}, \mathsf{s}]\varphi \iff (M \circ \mathsf{M}), s[\mathsf{M}, \mathsf{s}] \models \varphi$$

2.6.2 Properties

As one would suspect, AMLI has much of the same properties as IEL has. All of the following results come from [22].

Fact 2.8 (Persistency of Support). For all models M and all information states $s \subseteq \mathcal{W}$, for all $t \subseteq s$, if $M, s \models \varphi$ then $M, t \models \varphi$.

Fact 2.9 (Empty State Property). For all models M and all formulas φ, $M, \emptyset \models \varphi$.

Since AMLI's action operator also operates on sets, the action operator can also have these properties, which it in fact does.

Fact 2.10 (Modal Persistence). For all models M, all information states $s \subseteq \mathcal{W}$, all action models M, and all states s, if $M, s \models [\mathsf{M}, \mathsf{s}]\,\varphi$ and $\mathsf{t} \subseteq \mathsf{s}$, then $M, s \models [\mathsf{M}, \mathsf{t}]\,\varphi$.

Fact 2.11 (Modal Empty State). For all models M, all information states $s \subseteq \mathcal{W}$, all action models M, and all formulas φ, $M, s \models [\mathsf{M}, \emptyset]\,\varphi$.

AMLI also has a declarative fragment, which is given below. For AMLI the declaratives are even more important than they are for IEL, since all the preconditions of the actions are declaratives.

Definition 2.26 (Declaratives of AMLI). The declarative fragment of AMLI is given by:

$$\alpha := p \mid \bot \mid K_a\varphi \mid E_a\varphi \mid [\mathsf{s}]\alpha \mid \alpha_1 \wedge \alpha_2 \mid \varphi \to \alpha$$

As in IEL, the declaratives for AMLI are truth-conditional.

Fact 2.12. Any $\alpha \in \mathcal{L}_!^{\mathsf{AMLI}}$ is truth-conditional.

AMLI also has a set of resolutions. These are the same as for IEL, with an addition for the update modality.

Definition 2.27 (Resolutions for AMLI). The set $\mathcal{R}(\varphi)$ of resolutions of a formula φ is defined by extending Definition 2.11 with the following:

$$\mathcal{R}([\mathsf{M}, \mathsf{s}]\,\mu) = \{[\mathsf{M}, \mathsf{s}]\,\alpha \mid \alpha \in \mathcal{R}(\mu)\}$$

2.7 Structure of this Thesis

Since it is not immediately clear how all the logics from this section fit together and how they can be combined, we will now spend some time on laying out these plans. In order to better visualize this, we have included a diagram of the relations between these logics and the ones that have yet to be developed in Figure 2.4. This diagram will also aid us in determining what the rest of the thesis will look like.

$$\begin{tikzcd}
\mathsf{IEL} \arrow[r, blue] \arrow[dr, red] & \mathsf{AMLI} \arrow[dr, red] & \\
\mathsf{AML} \arrow[ur, green] \arrow[dr, red] & \mathsf{IE\text{-}PDL} \arrow[r, blue] & \mathsf{LCCI} \\
\mathsf{E\text{-}PDL} \arrow[r, blue] \arrow[ur, green] & \mathsf{LCC} \arrow[ur, green] &
\end{tikzcd}$$

Figure 2.4: The relations between the different Logics in this thesis. A blue arrow means that the logic on the left is used as the static language for the logic on the right. A green arrow means that the logic on the right is an inquisitive variant of the logic on the left. A red arrow means that the logic on the right borrows ideas from the logic on the left.

The first step is finding a logic that can function as our static language. As for LCC, this language needs to have PDL-like program operators, but it also needs to be an adequate replacement for IEL. Such a logic does not yet exist. Because of this, we will create an Inquisitive Epistemic Propositional Dynamic Logic (IE-PDL) which adheres precisely to these requirements. In Chapter 3, we will define this language, give examples of its use using Citadels, show that it generalizes IEL, and give a sound and complete axiomatisation. Because of the nature of IE-PDL, this will be the first language that we know of that will have both a sound and complete axiomatisation, and operators for common knowledge and public issues.

After that, we can start to define our main new language, the Logic of Communication, Change, and Issues (LCCI). Chapter 4 will start out with defining this new language and showing some examples of its use (including factual change). The rest of that chapter will show how to reduce LCCI to IE-PDL, how this leads to a sound and complete axiomatisation, and it will study some of these reductions.

Chapter 5 will then compare our work with logics with similar goals to IE-PDL and LCCI, in particular LCC and AMLI. Chapter 6 will conclude this thesis by giving some concluding remarks on the project and exploring some avenues for further research, such as further extensions that can be given to LCCI similar to the ones for LCC.

Chapter 3

Inquisitive Epistemic Propositional Dynamic Logic

3.1 Introduction

In this chapter we will start with the new additions that this thesis proposes. Here we will define the new logic called Inquisitive Epistemic Propositional Dynamic Logic (IE-PDL) which we will use as the basis for our later logic. IE-PDL is an inquisitive variant of E-PDL, and will therefore have the same basis on Propositional Dynamic Logic.

The logic of IE-PDL will give us an epistemic language that is richer than both E-PDL, since it will allow us to model the issues that agents have, and IEL, since it will allow us to model different knowledge relations in the same model.

We will start with the definition of the syntax, which is followed up by the definition of the models of IE-PDL, and how to interpret formulas in these models. Then we will discuss the properties of IE-PDL and how they differ from IEL and AMLI. We then show that IE-PDL allows us to model anything that we can model in IEL. This chapter will end with giving a sound and complete axiomatisation of IE-PDL.

3.2 Syntax

The definition of IE-PDL is similar to the definition of E-PDL, with the same set of operators to combine the epistemic relations of single agents. The differences are that there are two new operators. The first of these new operators is the *inquisitive or* ($\mathbin{\setminus\!\!/}$). We will need this operator to introduce issues into our new language. It operates in the same way as in IEL.

We will also need to introduce a new modal operator, which is similar to $E_a\varphi$ from IEL. For this we use $[\![\pi]\!]\varphi$, which means that the issues in φ are resolved in every state accessible via π, or in other words, that the issues in φ are entertained by π, where π is taken to be an agent, or a group of agents.

Like in E-PDL, the formulas of IE-PDL are defined by mutual recursion.

Definition 3.1 (Formulas of IE-PDL)**.** Given a finite set of propositional variables $\mathcal{P}$ and a finite set of agents $\mathcal{A}$, with p ranging over $\mathcal{P}$ and a ranging over $\mathcal{A}$. The language of IE-PDL is given by:

$$\begin{aligned}
\varphi &:= \bot \mid p \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \to \varphi_2 \mid \varphi_1 \mathbin{\rotatebox[origin=c]{-90}{$\geqslant$}} \varphi_2 \mid [\pi]\varphi \mid [\![\pi]\!]\varphi \\
\pi &:= a \mid ?\varphi \mid \pi_1;\pi_2 \mid \pi_1 \cup \pi_2 \mid \pi^*
\end{aligned}$$

We employ the following abbreviations for formulas:

$$\begin{aligned}
\neg\varphi &:= \varphi \to \bot \\
\top &:= \neg\bot \\
\varphi_1 \vee \varphi_2 &:= \neg(\neg\varphi_1 \wedge \neg\varphi_2) \\
\varphi_1 \leftrightarrow \varphi_2 &:= (\varphi_1 \to \varphi_2) \wedge (\varphi_2 \to \varphi_1)
\end{aligned}$$

Furthermore, if $B \subseteq \mathcal{A}$, we will take the program $\bigcup B$ to mean:

$$\bigcup_{b \in B} b$$

as in E-PDL. We will also use π^n as a shortcut for sequences, in the following manner:

$$\begin{aligned}
\pi^0 &= ?\top \\
\pi^1 &= \pi \\
\pi^{n+1} &= \pi;\pi^n
\end{aligned}$$

This means that, in IE-PDL, common knowledge of φ within a group of agents $B \subseteq \mathcal{A}$ is represented by $[(\bigcup B)^*]\varphi$, general knowledge of φ is $[\bigcup B]\varphi$, and φ is a public issue within a group of agents if $[\![(\bigcup B)^*]\!]\varphi$ is supported.

Also of note is that, unlike in IEL, we do not define $?\varphi := \varphi \mathbin{\rotatebox[origin=c]{-90}{$\geqslant$}} \neg\varphi$. The reason for this is that this operator would cause ambiguity with the program $?\varphi$, which could then be both a program and a formula, so it has been left out.

Now we have defined everything related to the syntax, we can move onto the definition of the semantic structures.

3.3 Semantics

Before we can define how we can interpret the formulas of IE-PDL, we will first have to define the relevant models in which we will interpret them.

Unlike what is normal in Inquisitive Modal logics, we will work with relations from information states to information states for IE-PDL. While the state map approach works well for encoding knowledge and issues for agents, it makes combining the actions harder, since operations like composition do not work on statemaps. However, for the knowledge and issues for the agents, it is easier to use a statemap, so we will use those as our basis, and later on transform a statemap into a relation from states to states for interpreting the composed epistemic relations.

Definition 3.2 (Models for IE-PDL)**.** An Inquisitive Epistemic Propositional Dynamic Logic model for a finite set $\mathcal{P}$ of propositional variables and a finite set $\mathcal{A}$ of atomic actions is a triple $M = \langle \mathcal{W}, \Sigma_{\mathcal{A}}, V \rangle$, where

- $\mathcal{W}$ is a finite set of possible worlds,

- $V : \mathcal{W} \to \wp(\mathcal{P})$ is a *valuation map*,
- $\Sigma_{\mathcal{A}} : \{\Sigma_a \mid a \in \mathcal{A}\}$ is a set of state maps, each of which assigns to every world w an issue $\Sigma_a(w)$.

  Note: Since we want our results to hold for all these state maps, we do not enforce factivity or introspection as requirements on Σ_a, since we want to show that these properties are not necessary for the technical results in this thesis.

Now we still have to define the relations for the epistemic relations. The one for single agents is easy, a state t is reachable from a state s iff there is a $w \in s$ such that $t \in \Sigma_a(w)$. This gives $E - a\varphi$ and $[\![a]\!]\varphi$ the same support conditions, as we will later see. For the composition programs, we will use the standard definitions from Propositional Dynamic Logic, with the exception of test and unbounded iteration.

Thanks to the fact that we no longer interpret formulas in worlds, but in sets of worlds, we cannot use the normal definitions which are dependent on reflexivity. With test, the problem with this is that it would break the persistence property (Proposition 3.1), since every formula that is not supported in a state might be supported in a subset of that state, where the formula after the modal operator would not be supported. Because of this, the definition for test is based upon the support condition for implication, where it returns all the subsets of a state that support the formula being tested. Because of this, $[\![?\varphi]\!]\psi$ can also be read as: "Assuming that φ holds, ψ holds".

For iteration, the problem is a bit different, and has to do with the declarativeness of $[\![\pi^*]\!]\varphi$. If π^* is reflexive, then the support condition for $[\![\pi^*]\!]\varphi$ in a state s would depend on the support condition of φ in that state s. This would mean that if φ was declarative, so would be $[\![\pi^*]\!]\varphi$, even when $[\![\pi]\!]\varphi$ was declarative. For our interpretation with IE-PDL as an epistemic language, this is a harmful property. To solve this problem, we have at least two options:

1. make reflexivity only matter on the level of worlds;
2. use only the transitive closure, instead of the reflexive transitive one.

Option 1 would be in line with the factivity requirement from IEL, which is why it seems like a good candidate to solve this problem. However, we think that option 2 is more in line with our current approach. Since the state maps for the agents are not factive either, it seems counter-intuitive to require that the "common knowledge" of one agent would have to be factive. Because of this, we will go with option 2. As we will see in Section 3.7, this will not lead to any problems for translating from IEL. We will also have more to say about this subject in Section 6.2.2.

Definition 3.3 (Compositional actions)**.** Given two relations R_{π_1} and R_{π_2}, the compositional action state maps are defined by:

$$\begin{aligned}
R_a &= \{(s,t) \mid w \in s,\ t \in \Sigma_a(w)\}\\
R_{\pi_1;\pi_2} &= R_{\pi_1} \circ R_{\pi_2}\\
R_{\pi_1 \cup \pi_2} &= R_{\pi_1} \cup R_{\pi_2}\\
R_{\varphi ?} &= \{(s,t) \mid M,t \models \varphi, t \subseteq s\}\\
R_{\pi^*} &= (R_\pi)^+
\end{aligned}$$

Here R^+ is the transitive closure of R. If $sR_{\pi^*}t$, then we will say that there is a π path from s to t.

We will also introduce the following notation:

$$\mathrm{R}_\pi(w) := \bigcup\{t \mid \{w\}R_\pi t\}$$

R_π is to R_π as σ_a is to Σ_a.

With Definitions 3.2 and 3.3 in place, we can now define the support conditions for the different formulas.

Definition 3.4 (Support of IE-PDL Formulas)**.** Let M be an IE-PDL model and s an information state in M.

$$\begin{aligned}
&M, s \models p \iff p \in V(w) \text{ for all worlds } w \in s\\
&M, s \models \bot \iff s = \emptyset\\
&M, s \models \varphi_1 \wedge \varphi_2 \iff M, s \models \varphi_1 \text{ and } M, s \models \varphi_2\\
&M, s \models \varphi_1 \vee\!\!\!\vee \varphi_2 \iff M, s \models \varphi_1 \text{ or } M, s \models \varphi_2\\
&M, s \models \varphi_1 \to \varphi_2 \iff \text{ for every } t \subseteq s,\ M, t \models \varphi_1 \text{ implies } M, t \models \varphi_2\\
&M, s \models [\pi]\varphi \iff \text{ for every } w \in s,\ M, \mathrm{R}_\pi(w) \models \varphi\\
&M, s \models [\![\pi]\!]\varphi \iff \text{ for every } sR_\pi t,\ M, t \models \varphi
\end{aligned}$$

This gives us the support conditions for information states, but truth in a world is still an important notion for inquisitive logics. Luckily, we can treat a world as a singleton information state, and define truth in that way.

Definition 3.5 (Truth in IE-PDL)**.** Let M be an IE-PDL model and w a world in M:

$$M, w \models \varphi \iff M, \{w\} \models \varphi$$

3.4 Examples

Let us now look at some examples of IE-PDL models. For these models, we will focus on one specific aspect of Citadels, this being the selection of roles. For a quick overview of the rules we use see Section 2.1. We will take a scenario where the players are sitting in alphabetical order, and where Alice is the first to select a character. In the example, *ak* means that Alice is the king, *ap* means that Alice is the Preacher, *aa* means that Alice is the Architect, *am* means that Alice is the Merchant, and *ac* means that Alice is the Condottiere. For Bob we have *bk* to mean that he is the King etcetera.

In this scenario, Alice only has one gold, but she does have four cards in her hand. She also has already built six buildings, meaning that she is close to the required eight. Bob on the other hand, has six coins, three buildings in his city, and no cards on hand. Claire has five buildings, has two coins and two cards to choose from.

Alice shuffles the stack and takes off the first card, which is the King. She puts it aside and looks at the rest of the cards. Since she has three blue buildings in her city, she can get three extra gold coins if she selects the Preacher, which would give her enough money to build one of the buildings in her hand. As a benefit, the Condottiere would not be able to destroy any of her buildings this round. She could also select the Architect and try to finish the game in this round, but it is not possible for the game to end this round otherwise, so this does not give Alice much benefit. Therefore, she selects the Preacher, and then hands over the cards to Bob.

Bob is behind on the number of buildings in his city, so he picks the Architect, in order to get more buildings build before the game ends. He then hands the remaining two cards to Claire.

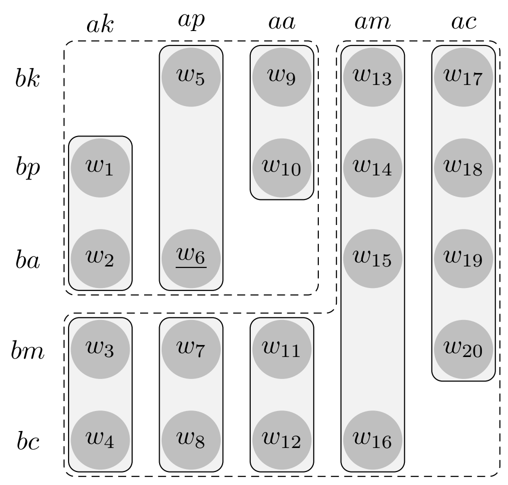


Figure 3.1: The epistemic and inquisitive state of Claire (c)

When we get to Claire, things get interesting. This is because Claire is trying to stop Alice from winning, so that she can win herself, and therefore has to try to guess which character Alice has picked to counteract that. Because Claire gets two cards, she knows that Alice has not picked the Condottiere or the Merchant, since she has those in her own hand. We will now use an IE-PDL model to study the inquisitive and epistemic state of Claire. Since giving the full relation R_c would not be extremely insightful, we will only give a visual representation. The epistemic and inquisitive state for Claire can be seen in Figure 3.1.

In this representation, we assume that a relation for a state with two or more members is built up from the relations for the singleton states that are its subsets. A singleton state is related to all the larger states within the dashed lines it is drawn in. The larger states are shown by drawing them within a grey box. So in this picture, $\{w_1\}R_c\{w_5, w_6\}$. All the rules from Definition 3.2 for relations are applied as well, so we also get $\{w_1\}R_c\{w_5\}$ and $\{w_1\}R_c\emptyset$. The actual world is shown by underlining the name of that world. In this example, that world will be w_6.

In the model, we can see that Claire knows which characters Alice and Bob do not have, since the actual world (w_6) has no relation to a state where any of am, ac, bm, or bc is supported. We can also see that Claire does not know which characters Alice and Bob have chosen, since we have $M, \{w_6\} \not\models [c]ak \vee [c]ap \vee [c]aa$ for Alice. For Bob a similar inference holds, namely $M, \{w_6\} \not\models [c]bk \vee [c]bp \vee [c]ba$. This is something which we could have modelled in standard Epistemic Logic as well. This model also gives us something which we could not model in standard Epistemic logic however, and that is the interest that Claire has in which character Alice has chosen.

In the scenario, Claire is only interested in the character that Alice has chosen, but not in the one that Bob has. We can also see that in the relation for Claire. We have $M, \{w_6\} \models [\![c]\!](ak \mathbin{\vee\!\!\!\vee} ap \mathbin{\vee\!\!\!\vee} aa)$, since for all the t such that $\{w_6\}R_ct$, $M, t \models ak \mathbin{\vee\!\!\!\vee} ap \mathbin{\vee\!\!\!\vee} aa$. This holds because in all those t, exactly one of ak, ap, and aa is supported. This does not hold for the character that Bob has chosen. This is because there are states t such that $\{w_6\}R_ct$ where neither bk nor bp or ba are supported. An example of such a state is $\{w_5, w_6\}$.

With IE-PDL we can also look at higher-order knowledge. For this we will also need to introduce the epistemic and inquisitive state of the other players. For Alice this can be found in Figure 3.2 and for Bob in Figure 3.3.

Now, we can express that Claire knows that Bob does not know which character Alice has

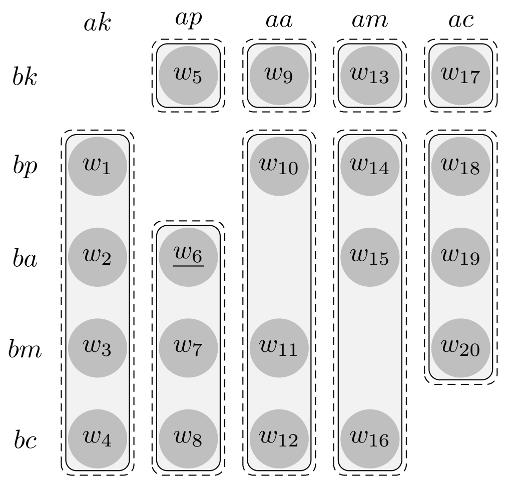


Figure 3.2: The epistemic and inquisitive state of Alice (a)

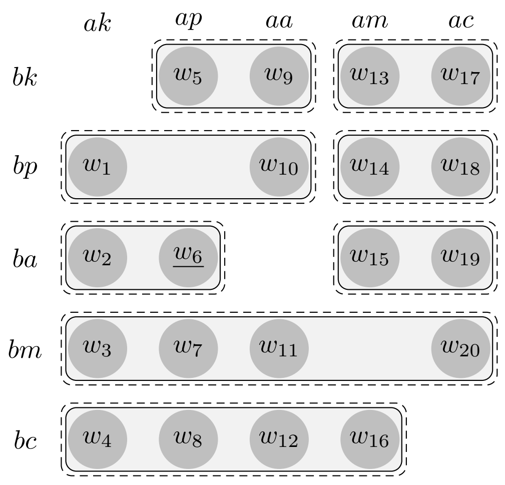


Figure 3.3: The epistemic and inquisitive state of Bob (b)

selected in the following way: $M, \{w_6\} \models [c]\neg[b](ak \vee\!\!\!\vee ap \vee\!\!\!\vee aa)$. This inference holds, since for all states that are reachable from $\{w_6\}$ via R_c there are states reachable via R_b where $ak \vee\!\!\!\vee ap \vee\!\!\!\vee aa$ is not supported, such as $\{w_2, w_6\}$.

Using IE-PDL, we can also study general and common knowledge and public issues in groups. While it is possible to build this into IEL, in its standard form this can only be done for the full group of agents. We will, however, start off with an example that shows common knowledge for all agents. Among the agents, there is the general knowledge that Alice has not selected the Merchant or the Condottiere, since both Bob and Claire have seen these cards. We can express the general knowledge about this by saying that all of the agents individually know. In IE-PDL we can express this in the following shorthand form: $[a \cup b \cup c]\neg(am \vee ac)$, which by our short-cut for groups of agents becomes $[\bigcup \mathcal{A}]\neg(am \vee ar)$. In order to express that this is common knowledge (which it would become after Bob reveals his role), we have to take the transitive closure for the knowledge of the set of agents. This we can do by simply applying the * operator to the program in the previous sentence to get the following result: $[(\bigcup \mathcal{A})^*]\neg(am \vee ar)$.

We can do the same thing for public issues. For example, we could say that it is a public issue for Alice and Claire which card Alice has selected in the following way: $[\![(a \cup c)^*]\!](ak \vee\!\!\!\vee ap \vee\!\!\!\vee aa)$[1]. These methods can also be combined to make more interesting claims. For example, for all the agents in the model it is common knowledge that which card Alice has selected is a public issue for Alice and Claire, which we can express in the following way: $[(\bigcup \mathcal{A})^*][\![(a \cup c)^*]\!](ak \vee\!\!\!\vee ap \vee\!\!\!\vee aa)$. This means that IE-PDL gives us a flexible way to create groups of agents and reason about their knowledge and issues.

3.5 Properties

We would also like to see which properties IE-PDL shares with IEL and AMLI. Most of the properties are the same, but some of them have some extra constraints put on them.

The persistence property and the empty state property still work for IE-PDL.

Proposition 3.1 (Persistency of Support)**.** *For all IE-PDL models M and information states $s \subseteq \mathcal{W}$, for all $t \subseteq s$, if $M, s \models \varphi$, then $M, t \models \varphi$.*

Proof. The proof is the same as for IEL except for $[\pi]\varphi$ and $[\![\pi]\!]\varphi$. Therefore, only those two cases are given.

$[\pi]\varphi$: $M, s \models [\pi]\varphi \iff$ for every $w \in s$, $M, \mathrm{R}_\pi(w) \models \varphi$. If this holds for every $w \in s$, then it also holds for every $w \in t$ where $t \subseteq s$. Therefore $M, t \models [\pi]\varphi$ for all $t \subseteq s$.

$[\![\pi]\!]\varphi$: This proof works by induction over the structure of programs.

a: $M, s \models [\![a]\!]\varphi \iff$ for every $sR_a u$, $M, u \models \varphi$. For all $t \subseteq s$, if $tR_a u$, then $sR_a u$, so if for every $sR_a u$, $M, u \models \varphi$, then it is also the case that for every $t \subseteq s$, for every $tR_a u$, $M, u \models \varphi$. Therefore $M, t \models [\![a]\!]\varphi$ for every $t \subseteq s$.

$\varphi?$: $M, s \models [\![\varphi?]\!]\psi \iff$ for every $sR_{\varphi?}u$, $M, u \models \psi$. Following Definition 3.3, these are all $t \subseteq s$ such that $M, t \models \varphi$. Therefore we get $M, t \models \varphi$ implies $M, t \models \psi$ for all $t \subseteq s$. Then by the inductive hypothesis we get that for all $t' \subseteq t$, $M, t' \models \varphi$ implies $M, t' \models \psi$. Therefore $M, t \models [\![\varphi?]\!]\psi$ for all $t \subseteq s$.

Take two arbitrary programs π_1 and π_2 such that if $M, s \models [\![\pi_1]\!]\varphi$, then $M, t \models [\![\pi_1]\!]\varphi$ for all $t \subseteq s$, and if $M, s \models [\![\pi_2]\!]\varphi$, then $M, t \models [\![\pi_2]\!]\varphi$ for all $t \subseteq s$.

[1]It might seem weird that this is an issue for Alice, but this is because of the definition of entertain. See Section 2.3.3

$\pi_1;\pi_2$: $M,s \models [\![\pi_1;\pi_2]\!]\varphi \iff$ for every $sR_{\pi_1;\pi_2}u$, $M,u \models \varphi \iff$ for every $sR_{\pi_1}t$, $tR_{\pi_1}u$, $M,u \models \varphi \iff M,s \models [\![\pi_1]\!][\![\pi_2]\!]\varphi$. Then by applying the inductive hypothesis we get that if $M,s \models [\![\pi_1;\pi_2]\!]\varphi$, then $M,t \models [\![\pi_1;\pi_2]\!]\varphi$ for all $t \subseteq s$.

$\pi_1 \cup \pi_2$: $M,s \models [\![\pi_1 \cup \pi_2]\!]\varphi \iff$ for every $sR_{\pi_1\cup\pi_2}t$, $M,t \models \varphi \iff$ for every $sR_{\pi_1}t$, $M,t \models \varphi$ and for every $sR_{\pi_2}u$, $M,u \models \varphi \iff M,s \models [\![\pi_1]\!]\varphi$ and $M,s \models [\![\pi_2]\!]\varphi$. By the inductive hypothesis we get if $M,s \models [\![\pi_1 \cup \pi_2]\!]\varphi$, then $M,t \models [\![\pi_1 \cup \pi_2]\!]\varphi$ for all $t \subseteq s$.

π_1^*: Take some model M and some state s in M such that $M,s \models [\![\pi_1^*]\!]\varphi$. Now assume that there is a $t \subseteq s$ such that $M,t \not\models [\![\pi_1^*]\!]\varphi$. That means that there must be some $tR_{\pi_1^*}u$ such that $M,u \not\models \varphi$. However, then we either have that $sR_{\pi_1^*}u$ or $sR_{\pi_1^*}u'$ such that $u \subseteq u'$, so we would have $M,s \not\models [\![\pi_1^*]\!]\varphi$. This is however not the case, so our assumption must be false.

This concludes the proof. □

Proposition 3.2 (Empty state property)**.** *For all IE-PDL models M and all formulas φ, $M,\emptyset \models \varphi$.*

Proof. The proof is the same as for IEL except for $[\pi]\varphi$ and $[\![\pi]\!]\varphi$. Therefore it is only given for those formulas.

$[\pi]\varphi$: By the support definition of the modal operator, we get that $M,\emptyset \models [\pi]\varphi \iff$ for every $w \in \emptyset$, $M, \mathrm{R}_\pi(w) \models \varphi$. But since there is no $w \in \emptyset$ this means that $M,\emptyset \models [\pi]\varphi \iff M,\emptyset \models \varphi$, which holds by the inductive hypothesis.

$[\![\pi]\!]\varphi$: By the support condition of $[\![\pi]\!]\varphi$ we get that $M,\emptyset \models [\![\pi]\!]\varphi \iff$ for every $\emptyset R_\pi t$, $M,t \models \varphi$. However, by Definition 3.2 there is no $t \subseteq \mathcal{W}$ such that $\emptyset R_\pi t$, so this statement is vacuously true. Therefore, $M,\emptyset \models [\![\pi]\!]\varphi$.

This concludes the proof. □

Before we can define the declaratives of IE-PDL, we will first have to denote the declarative programs. This is subset of all the programs that allows $[\![\pi]\!]\varphi$ to be declarative. This is necessary because $[\![\varphi?]\!]\psi$ functions like a conditional, and therefore is only declarative if ψ is. This means that there are two conditions when $[\![\pi]\!]\varphi$ is declarative, when π does not contain test, or when φ is declarative. We will now give the definition of the declarative programs, which are the programs which contain no test.

Definition 3.6 (Declarative Programs)**.** The declarative programs, written as $\Pi_!$ and ranged over by ϖ (pronounced varpi) are given by:

$$\varpi := a \mid \varpi;\pi \mid \varpi_1 \cup \varpi_2 \mid \varpi^*$$

Besides being useful in the definition of the declaratives for IE-PDL, the declarative programs also have an interesting property that is worth mentioning.

Proposition 3.3. *For any declarative program ϖ, $sR_\varpi t \iff$ there is some $w \in s$, such that $\{w\}R_\varpi t$.*

Proof. This proof works by induction over the structure of the declarative programs.

a: $sR_a t \iff$ for some $w \in s$, $t \in \Sigma_a(w) \iff$ for some $w \in s$, $\{w\}R_a t$

For the inductive hypothesis, take two arbitrary declarative programs ϖ_1 and ϖ_2 and assume that:

- $sR_{\varpi_1}t \iff$ there is some $w \in s$, such that $\{w\}R_{\varpi_1}t$
- $sR_{\varpi_2}t \iff$ there is some $w \in s$, such that $\{w\}R_{\varpi_2}t$

$\varpi_1;\pi$: Take some arbitrary program π, such that $sR_{\varpi_1;\pi}u$. This means there is some state t such that $sR_{\varpi_1}t$ and $tR_\pi u$. By the inductive hypothesis, there is some $w \in s$ such that $\{w\}R_{\varpi_1}t$. This also means that there is a $w \in s$ such that $\{w\}R_{\varpi_1;\pi}u$. Since we took an arbitrary π, we can assume this holds for all program π.

$\varpi_1 \cup \varpi_2$: $sR_{\varpi_1\cup\varpi_2}t \iff sR_{\varpi_1}t$ or $sR_{\varpi_2}t \iff$ there is some $w \in s$, such that $\{w\}R_{\varpi_1}t$ or there is some $v \in s$, such that $\{v\}R_{\varpi_2}t \iff$ there is some $w \in s$, such that $\{w\}R_{\varpi_1\cup\varpi_2}t$.

ϖ_1^*: This step follows from the step for sequence.

Since we took ϖ_1 and ϖ_2 arbitrary, we can assume that for all declarative programs ϖ: $sR_\varpi t \iff$ there is some $w \in s$, such that $\{w\}R_\varpi t$ □

Now we can go on to the definition of the declaratives for IE-PDL.

Definition 3.7 (Declaratives of IE-PDL)**.** The declarative fragment of IE-PDL, written as $\mathcal{L}_!^{\text{IE-PDL}}$ is given by:

$$\alpha := p \mid \bot \mid [\pi]\varphi \mid [\![\varpi]\!]\varphi \mid [\![\pi]\!]\alpha \mid \alpha_1 \wedge \alpha_2 \mid \varphi \rightarrow \alpha$$

Proposition 3.4. *The declarative fragment of IE-PDL is truth-conditional.*

Proof. The proof for all the formulas except for $[\pi]\varphi$, $[\![\varpi]\!]\varphi$ and $[\![\pi]\!]\alpha$ are as for IEL. Therefore, we will only discuss the proof for $[\pi]\varphi$, $[\![\varpi]\!]\varphi$ and $[\![\pi]\!]\alpha$.

$[\pi]\varphi$: $M, s \models [\pi]\varphi \iff$ for every $w \in s$, $M, \mathrm{R}_\pi(w) \models \varphi \iff$ for every $w \in s$, $M, w \models [\pi]\varphi$.

$[\![\varpi]\!]\varphi$: The proof for $[\![\varpi]\!]\varphi$ works by induction over $\Pi_!$.

a: $M, s \models [\![a]\!]\varphi \iff$ for every $sR_a t$, $M, t \models \varphi$. By Definition 3.3, this means that for all $w \in s$, for all $t \in \Sigma_a(w)$, $M, t \models \varphi$. This means that for all $w \in s$, for all $\{w\}R_a t$, $M, t \models \varphi$. Therefore, for all $w \in s$, $M, w \models [\![a]\!]\varphi$.

Take two arbitrary declarative programs ϖ_1 and ϖ_2 such that $[\![\varpi_1]\!]\varphi$ and $[\![\varpi_2]\!]\varphi$ are truth-conditional for all formulas φ.

$\varpi_1;\pi$: $M, s \models [\![\varpi_1;\pi]\!]\varphi \iff$ for every $sR_{\varpi_1;\pi}t$, $M, t \models \varphi \iff$ for every $sR_{\varpi_1}t$, for all $tR_\pi u$, $M, u \models \varphi \iff$ for every $sR_{\varpi_1}t$, $M, t \models [\![\pi]\!]\varphi \iff M, s \models [\![\varpi_1]\!][\![\pi]\!]\varphi$. Now by the inductive hypothesis we get that $[\![\varpi_1;\pi]\!]\varphi$ is truth-conditional.

$\varpi_1 \cup \varpi_2$: $M, s \models [\![\varpi_1 \cup \varpi_2]\!]\varphi \iff$ for every $sR_{\varpi_1\cup\varpi_2}t$, $M, t \models \varphi \iff$ for every $sR_{\varpi_1}t$, $M, t \models \varphi$ and for every $sR_{\varpi_2}u$, $M, u \models \varphi$. Now by the inductive hypothesis we get that $[\![\varpi_1 \cup \varpi_2]\!]\varphi$ is truth-conditional.

ϖ_1^*: Take some model M and some state s in M such that $M, s \models [\![\varpi_1^*]\!]\varphi$. $M, s \models [\![\varpi_1^*]\!]\varphi \iff$ for all $sR_{\varpi_1^*}t$, $M, t \models \varphi \iff$ for all $sR_{\varpi_1}t$, $M, t \models \varphi$ and for all $tR_{\varpi_1^*}u$, $M, u \models \varphi \iff$ for all $sR_{\varpi_1}t$, $M, t \models \varphi \wedge [\![\varpi_1^*]\!]\varphi \iff M, s \models [\![\varpi_1]\!](\varphi \wedge [\![\varpi_1^*]\!]\varphi)$. Now by the inductive hypothesis we have that $[\![\varpi_1^*]\!]\varphi$ is truth-conditional.

$[\![\pi]\!]\alpha$: The proof for $[\![\pi]\!]\alpha$ works by induction over the structure of programs. The only part that is different from the case for $[\![\varpi]\!]\varphi$ is the case for $\varphi?$, so that is the only case that is given.

$\varphi?$: $M, s \models [\![\varphi?]\!]\alpha \iff$ for every $sR_{\varphi?}t$, $M, t \models \alpha \iff$ for every $t \subseteq s$, $M, t \models \varphi$ implies $M, t \models \alpha$. This formula is truth-conditional, so therefore $[\![\varphi?]\!]\alpha$ is also truth-conditional.

Since it holds for all cases, all declarative formulas are truth-conditional. ☐

Furthermore, we also have $[\pi]$-$[\![\pi]\!]$ equivalence for declaratives.

Proposition 3.5 ($[\pi]$-$[\![\pi]\!]$ Equivalence)**.** *For all IE-PDL models M, all information states s in M, all programs π, and all declaratives α, $M, s \models [\pi]\alpha \iff M, s \models [\![\pi]\!]\alpha$.*

Proof.

$$\begin{aligned}
& M, s \models [\pi]\alpha \\
\iff & \text{for all } w \in s,\ M, \mathrm{R}_\pi(w) \models \alpha \\
\iff & \text{for all } w \in s,\ M, \bigcup\{t \mid \{w\}R_\pi t\} \models \alpha \\
\iff & \text{for all } w \in s, \text{ for all } v \in \bigcup\{t \mid \{w\}R_\pi t\},\ M, v \models \alpha \\
\iff & \text{for all } v \in \bigcup\{t \mid sR_\pi t\},\ M, v \models \alpha \\
\iff & \text{for all } sR_\pi t,\ M, t \models \alpha \\
\iff & M, s \models [\![\pi]\!]\alpha
\end{aligned}$$

☐

3.5.1 Resolutions

Since the resolutions will play an important role in proving the completeness of the logic later on, we will also have to define these for IE-PDL. Here the existence of test poses a problem, since we can now embed a question, so to speak, inside of the epistemic operators. This means that we cannot find a simple resolution for for example $[\![(?\varphi \cup b)^*]\!]\psi$.

Since giving the resolutions for complex programs is quite tricky, we will derive the resolutions from the reductions for IE-PDL (see Section 3.6). We can use this methodology since every formula in IE-PDL can be rewritten as a formula without complex programs, thanks to the fact that we only have a finite number of worlds in $\mathcal{W}$.

However, the exact form of this reduced formula depends upon the number of worlds in a model. Since there is no general upper bound upon this size, we will instead parametrize the resolutions, based on the number of worlds.

Definition 3.8 (Resolutions for IE-PDL)**.** The set $\mathcal{R}_n(\varphi)$ of resolutions of a formula φ in models with up to n worlds, is defined by changing every $\mathcal{R}$ in Definition 2.11 to a $\mathcal{R}_n$ and extending it with the following:

$$\begin{aligned}
\mathcal{R}_n([\![\varphi?]\!]\psi) &= \mathcal{R}_n(\varphi \to \psi) \\
\mathcal{R}_n([\![\pi_1;\pi_2]\!]\psi) &= \mathcal{R}_n([\![\pi_1]\!][\![\pi_2]\!]\psi) \\
\mathcal{R}_n([\![\pi_1 \cup \pi_2]\!]\psi) &= \mathcal{R}_n([\![\pi_1]\!]\psi \wedge [\![\pi_2]\!]\psi) \\
\mathcal{R}_n([\![\pi^*]\!]\psi) &= \mathcal{R}_n([\![\pi]\!]\psi \wedge [\![\pi^2]\!]\psi \cdots [\![\pi^{(2^n)}]\!]\psi)
\end{aligned}$$

To note here is that, since $[\![a]\!]\varphi$, $[\![\varpi]\!]\varphi$, and $[\![\pi]\!]\alpha$ are always declarative, and thus have themselves as their only resolution, every formula will eventually boil down to a formula for which we can calculate a resolution.

The properties that resolutions have in IEL can also be found in IE-PDL.

Proposition 3.6. *For any M, s, and φ, $M, s \models \varphi \iff$ for some $\alpha \in \mathcal{R}_{|\mathcal{W}|}(\varphi)$, $M, s \models \alpha$*

Proof. The proof works by induction over the structure of formulas of IE-PDL. The proof for all formulas except for $[\![\pi]\!]\varphi$ is as in IEL (Fact 2.5). The proof for $[\![\pi]\!]\varphi$ is given below. This part works by induction over the structure of programs.

a: Since $[\![a]\!]\varphi$ is declarative, this case is trivially true.

$\varphi?$: The resolutions for $[\![\varphi?]\!]\mu$ are the same as those for $\varphi \to \mu$, and since those sentences are also equivalent, this holds.

For the inductive hypothesis take two arbitrary programs π_1 and π_2 such that for any M, s, and φ, $M, s \models [\![\pi_1]\!]\varphi \iff$ for some $\alpha \in \mathcal{R}_{|\mathcal{W}|}([\![\pi_1]\!]\varphi)$, $M, s \models \alpha$ and $M, s \models [\![\pi_2]\!]\varphi \iff$ for some $\alpha \in \mathcal{R}_{|\mathcal{W}|}([\![\pi_2]\!]\varphi)$, $M, s \models \alpha$.

$\pi_1;\pi_2$: $M, s \models [\![\pi_1;\pi_2]\!]\varphi \iff M, s \models [\![\pi_1]\!][\![\pi_2]\!]\varphi$. Now, by the inductive hypothesis we know that $M, s \models [\![\pi_1]\!]\psi \iff$ for some $\alpha \in \mathcal{R}_{|\mathcal{W}|}([\![\pi_1]\!]\psi)$, $M, s \models \alpha$. So if we take $\psi := [\![\pi_2]\!]\varphi$, we get the wanted outcome.

$\pi_1 \cup \pi_2$: $M, s \models [\![\pi_1 \cup \pi_2]\!]\varphi \iff M, s \models [\![\pi_1]\!]\varphi \wedge [\![\pi_2]\!]\varphi$. The resolutions for the sentence $[\![\pi_1]\!]\varphi \wedge [\![\pi_2]\!]\varphi$ are $\{\alpha \wedge \beta \mid \alpha \in \mathcal{R}_{|\mathcal{W}|}([\![\pi_1]\!]\varphi), \beta \in \mathcal{R}_{|\mathcal{W}|}([\![\pi_2]\!]\varphi)\}$. By the inductive hypothesis we know that there is an $\alpha \in \mathcal{R}_{|\mathcal{W}|}([\![\pi_1]\!]\varphi)$ and a $\beta \in \mathcal{R}_{|\mathcal{W}|}([\![\pi_2]\!]\varphi)$ such that $M, s \models \alpha$ and $M, s \models \beta$. Therefore, there is also an $\alpha \wedge \beta \in \mathcal{R}_{|\mathcal{W}|}([\![\pi_1]\!]\varphi \wedge [\![\pi_2]\!]\varphi)$ such that $M, s \models \alpha \wedge \beta$.

${\pi_1}^*$: $M, s \models [\![\pi_1^*]\!]\varphi \iff M, s \models [\![\pi_1^1]\!]\varphi \wedge [\![\pi_1^2]\!]\varphi \cdots \left[\!\!\left[\pi_1^{|\mathcal{P}(\mathcal{W})|}\right]\!\!\right]\varphi$. The resolutions for this sentence are $\left\{\bigwedge_{1 \le n \le |\mathcal{W}|} \beta_n \,\middle|\, \beta_n \in \mathcal{R}_{|\mathcal{P}(\mathcal{W})|}([\![\pi_1^n]\!]\varphi)\right\}$. We know that there is a $\beta_n \in \mathcal{R}_{|\mathcal{W}|}([\![\pi_1^n]\!]\varphi)$ for each $1 \le n \le |\mathcal{P}(\mathcal{W})|$ by the inductive hypothesis and repeated applications of the case for sequence, such that we have that $M, s \models \beta_n$ for all $1 \le n \le |\mathcal{P}(\mathcal{W})|$. Therefore, there is also an $\beta_1 \wedge \beta_2 \cdots \beta_{|\mathcal{P}(\mathcal{W})|} \in \mathcal{R}_{|\mathcal{W}|}([\![\pi_1^1]\!]\varphi \wedge [\![\pi_1^2]\!]\varphi \cdots \left[\!\!\left[\pi_1^{|\mathcal{P}(\mathcal{W})|}\right]\!\!\right]\varphi)$ such that $M, s \models \beta_1 \wedge \beta_2 \cdots \beta_{|\mathcal{P}(\mathcal{W})|}$.

This concludes our proof. ☐

Now we also get the normal form result as a corollary.

Corollary 3.7 (Normal form)**.** *For any M, s, and φ, $M, s \models \varphi \iff M, s \models \bigvee\!\!\!\!\bigvee \mathcal{R}_{|\mathcal{W}|}(\varphi)$.*

Proof. From Proposition 3.6 we know that $M, s \models \varphi \iff M, s \models \alpha$ for some $\alpha \in \mathcal{R}_{|\mathcal{W}|}(\varphi)$. By Definition 3.4 we also have that $M, s \models \alpha \mathbin{\bigvee\!\!\!\!\bigvee} \beta \iff M, s \models \alpha$ or $M, s \models \beta$. Therefore, if we know that $M, s \models \alpha$, we can use the definition to get $M, s \models \beta_1 \mathbin{\bigvee\!\!\!\!\bigvee} \cdots \alpha \cdots \mathbin{\bigvee\!\!\!\!\bigvee} \beta_n$ for any β_i. Therefore we can also get that for some $\alpha \in \mathcal{R}_{|\mathcal{W}|}(\varphi)$, $M, s \models \alpha \iff M, s \models \bigvee\!\!\!\!\bigvee \mathcal{R}_{|\mathcal{W}|}(\varphi)$. From this we can conclude $M, s \models \varphi \iff M, s \models \bigvee\!\!\!\!\bigvee \mathcal{R}_{|\mathcal{W}|}(\varphi)$. ☐

For our purposes, it will also be useful to define the resolutions of a set, for which we will use Definition 2.13, as in IEL. Proposition 3.6 also generalizes to sets of resolutions.

Corollary 3.8. *For any M, s, and Φ, $M, s \models \Phi \iff$ for some $\Gamma \in \mathcal{R}_{|\mathcal{W}|}(\Phi)$, $M, s \models \Gamma$.*

Proof. We will start with the direction from left to right. Suppose that $M, s \models \Phi$. We will now build a resolution Γ of Φ and show that $M, s \models \Gamma$. For each formula $\varphi_i \in \Phi$ take a resolution $\alpha_i \in \mathcal{R}_{|\mathcal{W}|}(\varphi)$ such that $M, s \models \alpha_i$. By Proposition 3.6 we know that this resolutions exists. Now we take $\Gamma = \{\alpha_i \mid \varphi_i \in \Phi\}$. Now $M, s \models \Gamma$ by construction.

For the right to left direction, suppose that $M, s \models \Gamma$. Now, since $\Gamma \in \mathcal{R}_{|\mathcal{W}|}(\Phi)$, for each $\alpha \in \Gamma$, there is a $\varphi \in \Phi$. By Proposition 3.6 we also have that $M, s \models \varphi$ for all those φ. Now, by Definition 2.13 there are also no $\varphi' \in \Phi$ such that there is no $\alpha' \in \Gamma$ with $\alpha' \in \mathcal{R}_{|\mathcal{W}|}(\varphi')$. Therefore $M, s \models \varphi$ for all $\varphi \in \Phi$, and thus $M, s \models \Phi$, as required. □

3.5.2 Substitutions

As was the case with LCC, the factual change that we will build in Chapter 4 will depend in a large part on the existence of substitutions over the language of IE-PDL. Because of this, we will have to define substitutions for IE-PDL, and we will show some properties that the substitutions have.

However, before we can start with defining what a substitution is, we should first consider what properties we would want a substitution to have. Since the substitutions represent the post-condition for an action and should not raise any new issues, we would want that a substitution will map the propositional atoms to truth-conditional sentences. This is something that we can enforce by only allowing the binding to go from propositional atoms to declaratives.

Definition 3.9 (IE-PDL Substitution)**.** IE-PDL substitutions are functions of type $\mathcal{L}^{\text{IE-PDL}} \to \mathcal{L}^{\text{IE-PDL}}$ that distribute over all language constructs, and that map all but a finite number of basic propositional atoms to themselves. IE-PDL substitutions can be represented as a finite set of bindings

$$\{p_1 \mapsto \alpha_1, \ldots, p_n \mapsto \alpha_n\}$$

where all the p_i are different, and each $\alpha_i \in \mathcal{L}_!^{\text{IE-PDL}}$. If ρ is an IE-PDL substitution, then the set $\{p \mid p \in \mathcal{P}, \rho(p) \neq p\}$ is called the *domain* of ρ, denoted $dom(\rho)$. We use ϵ for the identity substitution. Let $SUB_{\text{IE-PDL}}$ be the set of all IE-PDL substitutions.

Now we know what IE-PDL substitutions look like, we can also define how they should be distributed over formulas. If $\rho = \{p_1 \mapsto \alpha_1, \ldots, p_n \mapsto \alpha_n\}$ is an IE-PDL substitution, we use φ^ρ for $\rho(\varphi)$ and π^ρ for $\rho(\pi)$. The definition of ρ for formulas is as follows:

$$\begin{aligned}
\bot^\rho &= \bot \\
p^\rho &= \rho(p) \\
(\neg\varphi)^\rho &= \neg\varphi^\rho \\
(\varphi_1 \wedge \varphi_2)^\rho &= \varphi_1^\rho \wedge \varphi_2^\rho \\
(\varphi_1 \to \varphi_2)^\rho &= \varphi_1^\rho \to \varphi_2^\rho \\
(\varphi_1 \vee\!\!\!\vee \varphi_2)^\rho &= \varphi_1^\rho \vee\!\!\!\vee \varphi_2^\rho \\
([\pi]\varphi)^\rho &= [\pi^\rho]\varphi^\rho \\
([\![\pi]\!]\varphi)^\rho &= [\![\pi^\rho]\!]\varphi^\rho
\end{aligned}
\qquad
\begin{aligned}
a^\rho &= a \\
(\varphi?)^\rho &= \varphi^\rho? \\
(\pi_1;\pi_2)^\rho &= \pi_1^\rho;\pi_2^\rho \\
(\pi_1 \cup \pi_2)^\rho &= \pi_1^\rho \cup \pi_2^\rho \\
(\pi^*)^\rho &= (\pi^\rho)^*
\end{aligned}$$

Since our definition for the valuation is a bit different from the one in [18], we will also have to update the definition of a model under a substitution.

Definition 3.10 (IE-PDL model under a substitution)**.** If $M = \langle \mathcal{W}, V, R \rangle$ is an IE-PDL model and ρ is an IE-PDL substitution, then V_M^ρ is the valuation given by

$$V_M^\rho(w) = \{p \mid M, w \models \rho(p)\}$$

In other words, V_M^ρ assigns to w the set of propositional atoms for which $\rho(p)$ is true in w. For $M = \langle \mathcal{W}, V, R \rangle$, call M^ρ the model given by $\langle \mathcal{W}, V_M^\rho, R \rangle$. Furthermore, we use R^ρ for a relation where $?\varphi$ is interpreted using M^ρ instead of M, but which is otherwise the same as R.

This definition is defined in terms of truth, but since the binding only maps truth-conditional formulas to other truth-conditional formulas, this is not a problem.

Besides these two definitions, there is also one last property that we would like the substitutions to have.

Lemma 3.9 (Substitution)**.** *For all IE-PDL models M, all IE-PDL formulas φ, all IE-PDL programs π, and all IE-PDL substitutions ρ:*

$$M, s \models \varphi^\rho \iff M^\rho, s \models \varphi$$

$$(s,t) \in R_{\pi^\rho} \iff (s,t) \in R_\pi^\rho$$

Where R_π^ρ is the relation of π in M^ρ.

Proof. The proof goes by simultaneous induction on the structure of formulas and programs.

p: $M, s \models p^\rho \iff M, s \models \rho(p) \iff M, w \models \rho(p)$ for all $w \in s \iff p \in V_M^\rho(w)$ for all $w \in s \iff M^\rho, s \models p$

$\bot$: $M, s \models \bot^\rho \iff M, s \models \bot \iff s = \emptyset \iff M^\rho, s \models \bot$

a: $(s,t) \in R_{a^\rho} \iff (s,t) \in R_a \iff (s,t) \in R_a^\rho$

As inductive hypothesis, assume we have two arbitrary formulas φ and ψ and two arbitrary programs π_1 and π_2 such that

$$\begin{aligned} M, s \models \varphi^\rho &\iff M^\rho, s \models \varphi & \qquad (s,t) \in R_{\pi_1^\rho} &\iff (s,t) \in R_{\pi_1}^\rho \\ M, s \models \psi^\rho &\iff M^\rho, s \models \psi & \qquad (s,t) \in R_{\pi_2^\rho} &\iff (s,t) \in R_{\pi_2}^\rho \end{aligned}$$

$\varphi \wedge \psi$: $M, s \models (\varphi \wedge \psi)^\rho \iff M, s \models \varphi^\rho \wedge \psi^\rho \iff M, s \models \varphi^\rho$ and $M, s \models \psi^\rho \iff M^\rho, s \models \varphi$ and $M^\rho, s \models \psi \iff M^\rho, s \models \varphi \wedge \psi$

$\varphi \Vvee \psi$: $M, s \models (\varphi \Vvee \psi)^\rho \iff M, s \models \varphi^\rho \Vvee \psi^\rho \iff M, s \models \varphi^\rho$ or $M, s \models \psi^\rho \iff M^\rho, s \models \varphi$ or $M^\rho, s \models \psi \iff M^\rho, s \models \varphi \Vvee \psi$

$\varphi \to \psi$: $M, s \models (\varphi \to \psi)^\rho \iff M, s \models \varphi^\rho \to \psi^\rho \iff$ for every $t \subseteq s$, $M, t \models \varphi^\rho$ implies $M, t \models \psi^\rho \iff$ for every $t \subseteq s$, $M^\rho, t \models \varphi$ implies $M^\rho, t \models \psi \iff M^\rho, s \models \varphi \to \psi$

$[\pi_1]\varphi$: $M, s \models ([\pi_1]\varphi)^\rho \iff M, s \models [\pi_1^\rho]\varphi^\rho \iff$ for every $w \in s$, $M, \mathrm{R}_{\pi_1}(w) \models \varphi^\rho \iff$ for every $w \in s$, $M, \mathrm{R}_\pi^\rho(w) \models \varphi^\rho \iff$ for every $w \in s$, $M^\rho, \mathrm{R}_\pi^\rho(w) \models \varphi \iff M^\rho, s \models [\pi_1]\varphi$

$[\![\pi_1]\!]\varphi$: $M, s \models ([\![\pi]\!]\varphi)^\rho \iff M, s \models [\![\pi^\rho]\!]\varphi^\rho \iff$ for every $sR_{\pi_1^\rho}t$, $M, t \models \varphi^\rho \iff$ for every $sR_{\pi_1}^\rho t$, $M, t \models \varphi^\rho \iff$ for every $sR_{\pi_1}^\rho t$, $M^\rho, t \models \varphi \iff M^\rho, s \models [\![\pi]\!]\varphi$

$\varphi?$: $(s,t) \in R_{(\varphi?)^\rho} \iff (s,t) \in R_{\varphi^\rho?} \iff (s,t) \in \{(s',t') \mid M, t' \models \varphi^\rho, t' \subseteq s'\} \iff (s,t) \in \{(s',t') \mid M^\rho, t' \models \varphi, t' \subseteq s'\} \iff (s,t) \in R_{\varphi?}^\rho$

$\pi_1;\pi_2$: $(s,t) \in R_{(\pi_1;\pi_2)^\rho} \iff (s,t) \in R_{\pi_1^\rho;\pi_2^\rho} \iff (s,t) \in R_{\pi_1^\rho} \circ R_{\pi_2^\rho} \iff (s,u) \in R_{\pi_1^\rho}$ and $(u,t) \in R_{\pi_2^\rho} \iff (s,u) \in R^\rho_{\pi_1}$ and $(u,t) \in R^\rho_{\pi_2} \iff (s,t) \in (R^\rho_{\pi_1} \circ R^\rho_{\pi_2}) \iff (s,t) \in R^\rho_{\pi_1;\pi_2}$

$\pi_1 \cup \pi_2$: $(s,t) \in R_{(\pi_1\cup\pi_2)^\rho} \iff (s,t) \in R_{\pi_1^\rho\cup\pi_2^\rho} \iff (s,t) \in (R_{\pi_1^\rho} \cup R_{\pi_2^\rho}) \iff (s,t) \in R_{\pi_1^\rho}$ or $(s,t) \in R_{\pi_2^\rho} \iff (s,t) \in R^\rho_{\pi_1}$ or $(s,t) \in R^\rho_{\pi_2} \iff (s,t) \in (R^\rho_{\pi_1} \cup R^\rho_{\pi_2}) \iff (s,t) \in R^\rho_{\pi_1\cup\pi_2}$

π_1^*: $(s,t) \in R_{(\pi_1^*)^\rho} \iff (s,t) \in R_{(\pi_1^\rho)^*} \iff (s,t) \in (R_{\pi_1^\rho})^+ \iff (s,t) \in (R^\rho_{\pi_1})^+ \iff (s,t) \in R^\rho_{\pi_1^*}$

This concludes the proof. □

3.6 Reduction Axioms

While the compositional operators allow us a great degree of freedom in defining epistemic relations, it would also be nice if we could simplify statements. For this we found the following reduction axioms. The first are a subset of the classical axioms for Propositional Dynamic Logic, all of which still work, except for the ones for unbound iteration. This is thanks to the fact that * is now no longer the reflexive transitive closure, but just the transitive one. The reduction lemmas for π^* have now been replaced by new ones.

Lemma 3.10 (K)**.** *For all models M, all information states s, all programs π and all formulas φ and ψ, the following holds:*

$$M, s \models [\pi](\varphi \to \psi) \leftrightarrow ([\pi]\varphi \to [\pi]\psi)$$

Proof.

$$\begin{aligned}
& M, s \models [\pi](\varphi \to \psi) \\
\iff & \text{for every } w \in s,\ M, \mathrm{R}_\pi(w) \models \varphi \to \psi \\
\iff & \text{for every } w \in s, \\
& M, \mathrm{R}_\pi(w) \models \varphi \text{ implies } M, \mathrm{R}_\pi(w) \models \psi \\
\iff & M, s \models [\pi]\varphi \text{ implies } M, s \models [\pi]\psi \\
\iff & M, s \models [\pi]\varphi \to [\pi]\psi
\end{aligned}$$

□

This lemma allows us to push the modal operator into an implication.

While test can still be applied, it does work a bit differently than it does in PDL. This is because it only works on declarative consequents. As we will see later, the inquisitive version of test works for all formulas.

Lemma 3.11 (Test)**.** *For all models M, all information states s, all formulas φ, all declaratives α, the following holds:*

$$M, s \models [\varphi?]\alpha \leftrightarrow (\varphi \to \alpha)$$

Proof. Since both $[\varphi?]\alpha$ and $\varphi \to \alpha$ are truth-conditional, we only have to look at their truth conditions.

$$\begin{aligned} & M, w \models [\varphi?]\alpha \\ \iff & M, \mathrm{R}_{?\varphi}(w) \models \alpha \\ \iff & M, w \models \varphi \text{ implies } M, w \models \alpha \\ \iff & M, w \models \varphi \to \alpha \end{aligned}$$

□

For the following two proofs, we will not use the abbreviation $\mathrm{R}_\pi(w)$ because this makes the proofs easier to follow.

Lemma 3.12 (Sequence). *For all models M, all information states s, all programs π_1 and π_2 and all formulas φ, the following holds:*

$$M, s \models [\pi_1; \pi_2]\varphi \leftrightarrow [\pi_1][\pi_2]\varphi$$

Proof.

$$\begin{aligned} & M, s \models [\pi_1; \pi_2]\varphi \\ \iff & \text{for every } w \in s,\ M, \bigcup \{t \mid \{w\}R_{\pi_1;\pi_2}t\} \models \varphi \\ \iff & \text{for every } w \in s,\ M, \bigcup \{t \mid \{w\}R_{\pi_1} \circ R_{\pi_2}t\} \models \varphi \\ \iff & \text{for every } w \in s,\ M, \bigcup \{t \mid u \subseteq \mathcal{W}, \{w\}R_{\pi_1}u, uR_{\pi_2}t\} \models \varphi \\ \iff & \text{for every } w \in s, \text{ and every } \{w\}R_{\pi_1}u,\ M, \bigcup \{t \mid uR_{\pi_2}t\} \models \varphi \\ \iff & \text{for every } w \in s,\ M, \bigcup \{u \mid \{w\}R_{\pi_1}u\} \models [\pi_2]\varphi \\ \iff & M, s \models [\pi_1][\pi_2]\varphi \end{aligned}$$

□

This lemma simply states that having knowledge of some else's knowledge, is the same as knowing that the other agents knows something.

Lemma 3.13 (Choice). *For all models M, all information states s, all programs π_1 and π_2 and all formulas φ, the following holds:*

$$M, s \models [\pi_1 \cup \pi_2]\varphi \leftrightarrow ([\pi_1]\varphi \wedge [\pi_2]\varphi)$$

Proof.

$$\begin{aligned} & M, s \models [\pi_1 \cup \pi_2]\varphi \\ \iff & \text{for every } w \in s,\ M, \bigcup \{t \mid \{w\}R_{\pi_1 \cup \pi_2}t\} \models \varphi \\ \iff & \text{for every } w \in s,\ M, \bigcup \{t \mid \{w\}(R_{\pi_1} \cup R_{\pi_2})t\} \models \varphi \\ \iff & \text{for every } w \in s,\ M, \bigcup \{t \mid \{w\}R_{\pi_1}t\} \models \varphi \text{ and} \\ & \qquad\qquad\qquad M, \bigcup \{u \mid \{w\}R_{\pi_2}u\} \models \varphi \\ \iff & M, s \models [\pi_1]\varphi \text{ and } M, s \models [\pi_2]\varphi \\ \iff & M, s \models [\pi_1]\varphi \wedge [\pi_2]\varphi \end{aligned}$$

□

This lemma states that if something is general knowledge is a group of agents, then all the agents must know it.

Lemma 3.14 (Unrolling)**.** *For all models M, all informations states s, all programs π and all formulas φ, the following holds:*

$$M, s \models [\pi^*]\varphi \iff M, s \models [\pi^1]\varphi \wedge [\pi^2]\varphi \cdots [\pi^{|\mathcal{P}(\mathcal{W})|}]\varphi$$

Proof. This proof needs to be given in both directions, we start by going from the left to the right.

$\Rightarrow$: Take arbitrary model M and information state s such that $M, s \models [\pi^*]\varphi$. By the definition of the support conditions, this means that we have that for all $w \in s$, $M, \mathrm{R}_{\pi^*}(w) \models \varphi$. Now suppose that $M, s \not\models [\pi]\varphi \wedge [\pi^2]\varphi \cdots [\pi^{|\mathcal{P}(\mathcal{W})|}]\varphi$. Then there must be some $1 \leq n \leq |\mathcal{P}(\mathcal{W})|$ such that $M, s \not\models [\pi^n]\varphi$ and thus for some $w \in s$, $M, \mathrm{R}_{\pi^n}(w) \not\models \varphi$. However, by definition $\mathrm{R}_{\pi^n}(w) \subseteq \mathrm{R}_{\pi^*}(w)$, which leads to a contradiction. Therefore, our earlier assumption must be wrong, and $M, s \models [\pi]\varphi \wedge [\pi^2]\varphi \cdots [\pi^{|\mathcal{P}(\mathcal{W})|}]\varphi$, as required.

$\Leftarrow$: This proof also works by contradiction. Assume that there is some M, s, and φ such that $M, s \models \varphi \wedge [\pi^1] \cdots [\pi^{|\mathcal{P}(\mathcal{W})|}]\varphi$ but not $M, s \models [\pi^*]\varphi$. Then there must be some t such that $sR_{\pi^*}t$, while $s \neq t$, not $sR_{\pi}t$ ...not $sR_{\pi^{|\mathcal{P}(\mathcal{W})|}}t$. This t must be a subset of $\mathcal{W}$ however, since otherwise we could not have $sR_{\pi^*}t$. Since we only have a finite number of worlds, only $|\mathcal{P}(\mathcal{W})|$ information states are possible in the model M. Since there is a π path from s to t, this path can therefore also not be longer than $|\mathcal{P}(\mathcal{W})|$ steps. But that means that there must be a π^n with $0 \leq n \leq \mathcal{P}(\mathcal{W})$ such that $sR_{\pi^n}t$. This means that $M, s \not\models [\pi^n]\varphi$. This conflicts with our earlier assumption, leading to a contradiction. Therefore, $M, s \models \varphi \wedge [\pi^1]\varphi \cdots [\pi^{|\mathcal{P}(\mathcal{W})|}]\varphi \Rightarrow M, s \models [\pi^*]\varphi$.

□

This lemma states that every formula with a star can be rewritten into one without the star. While this is not true in PDL, this is a side effect of the fact that we have a finite number of worlds, and will be used later in the completeness proof for IE-PDL.

Besides these standard ones, we also found that similar axioms hold for the Inquisitive Action Operator.

Lemma 3.15 (Inquisitive K)**.** *For all models M, all information states s, all programs π and all formulas φ and ψ, the following holds:*

$$M, s \models [\![\pi]\!](\varphi \to \psi) \leftrightarrow ([\![\pi]\!]\varphi \to [\![\pi]\!]\psi)$$

Proof.

$$\begin{aligned}
& M, s \models [\![\pi]\!](\varphi \to \psi) \\
\iff & \text{for every } sR_\pi t,\ M, t \models \varphi \to \psi \\
\iff & \text{for every } sR_\pi t,\ M, t \models \varphi \text{ implies } M, t \models \psi \\
\iff & M, s \models [\![\pi]\!]\varphi \text{ implies } M, s \models [\![\pi]\!]\psi \\
\iff & M, s \models [\![\pi]\!]\varphi \to [\![\pi]\!]\psi
\end{aligned}$$

□

Lemma 3.16 (Inquisitive Sequence)**.** *For all models M, all information states s, all programs π_1 and π_2 and all formulas φ, the following holds:*

$$M, s \models [\![\pi_1; \pi_2]\!]\varphi \leftrightarrow [\![\pi_1]\!][\![\pi_2]\!]\varphi$$

Proof.

$$\begin{aligned}
& M, s \models [\![\pi_1; \pi_2]\!]\varphi \\
\iff & \text{for all } sR_{\pi_1;\pi_2}t,\ M, t \models \varphi \\
\iff & \text{for all } s(R_{\pi_1} \circ R_{\pi_2})t,\ M, t \models \varphi \\
\iff & \text{for all } sR_{\pi_1}u, \text{ and all } uR_{\pi_2}t,\ M, t \models \varphi \\
\iff & \text{for all } sR_{\pi_1}u,\ M, u \models [\![\pi_2]\!]\varphi \\
\iff & M, s \models [\![\pi_1]\!][\![\pi_2]\!]\varphi
\end{aligned}$$

□

Lemma 3.17 (Inquisitive Choice)**.** *For all models M, all information states s, all programs π_1 and π_2 and all formulas φ, the following holds:*

$$M, s \models [\![\pi_1 \cup \pi_2]\!]\varphi \leftrightarrow ([\![\pi_1]\!]\varphi \wedge [\![\pi_2]\!]\varphi)$$

Proof.

$$\begin{aligned}
& M, s \models [\![\pi_1 \cup \pi_2]\!]\varphi \\
\iff & \text{for all } sR_{\pi_1 \cup \pi_2}t,\ M, s \models \varphi \\
\iff & \text{for all } s(R_{\pi_1} \cup R_{\pi_2})t,\ M, s \models \varphi \\
\iff & \text{for all } sR_{\pi_1}t,\ M, s \models \varphi \text{ and all } sR_{\pi_2}u,\ M, u \models \varphi \\
\iff & M, s \models [\![\pi_1]\!]\varphi \text{ and } M, s \models [\![\pi_2]\!]\varphi \\
\iff & M, s \models [\![\pi_1]\!]\varphi \wedge [\![\pi_2]\!]\varphi
\end{aligned}$$

□

Lemma 3.18 (Inquisitive Test)**.** *For all models M, all information states s, and all formulas φ and ψ, the following holds:*

$$M, s \models [\![\varphi?]\!]\psi \leftrightarrow (\varphi \rightarrow \psi)$$

Proof.

$$\begin{aligned}
& M, s \models [\![\varphi?]\!]\psi \\
\iff & \text{for all } sR_{\varphi?}t,\ M, t \models \psi \\
\iff & \text{for all } t \subseteq s,\ M, t \models \varphi \text{ implies } M, t \models \psi \\
\iff & M, s \models \varphi \rightarrow \psi
\end{aligned}$$

□

This lemma states that if ψ holds in a world where φ is true, then φ implies ψ.

Lemma 3.19 (Inquisitive Unrolling)**.** *For all models M, all informations states s, all programs π and all formulas φ, the following holds:*

$$M, s \models [\![\pi^*]\!]\varphi \iff M, s \models [\![\pi^1]\!]\varphi \wedge [\![\pi^2]\!]\varphi \cdots [\![\pi^{|\mathcal{P}(\mathcal{W})|}]\!]\varphi$$

Proof. This proof needs to be given in both directions, we start by going from the left to the right.

- $\Rightarrow$: Takes some arbitrary M and s such that $M, s \models [\![\pi^*]\!]\varphi$. Now assume that $M, s \not\models [\![\pi]\!]\varphi \wedge [\![\pi^2]\!]\varphi \cdots [\![\pi^{|\mathcal{P}(\mathcal{W})|}]\!]\varphi$. That means that there is some $1 \leq n \leq |\mathcal{P}(\mathcal{W})|$ such that $M, s \not\models [\![\pi^n]\!]\varphi$. In turn, this means that there is an $sR_{\pi^n}t$ such that $M, t \not\models \varphi$. By Definition 3.3 this means that $sR_{\pi^*}t$, but this leads to a contradiction. Therefore, our earlier assumption must be wrong, and we get $M, s \models [\![\pi]\!]\varphi \wedge [\![\pi^2]\!]\varphi \cdots [\![\pi^{|\mathcal{P}(\mathcal{W})|}]\!]\varphi$, as required.
- $\Leftarrow$: This proof also works by contradiction. Assume that there is some M, s, and φ such that $M, s \models \varphi \wedge [\![\pi^1]\!] \cdots [\![\pi^{|\mathcal{P}(\mathcal{W})|}]\!]\varphi$ but not $M, s \models [\![\pi^*]\!]\varphi$. Then there must be some t such that $sR_{\pi^*}t$, while $s \neq t$, not $sR_\pi t$ ... not $sR_{\pi^{|\mathcal{P}(\mathcal{W})|}}t$. This t must be a subset of $\mathcal{W}$ however, since otherwise we could not have $sR_{\pi^*}t$. Since we only have a finite number of worlds, only $|\mathcal{P}(\mathcal{W})|$ information states are possible in the model M. Since there is a π path from s to t, this path can therefore also not be longer than $|\mathcal{P}(\mathcal{W})|$ steps. But that means that there must be a π^n with $0 \leq n \leq \mathcal{P}(\mathcal{W})$ such that $sR_{\pi^n}t$. This means that $M, s \not\models [\![\pi^n]\!]\varphi$. This conflicts with our earlier assumption, leading to a contradiction. Therefore, $M, s \models \varphi \wedge [\![\pi^1]\!]\varphi \cdots [\![\pi^{|\mathcal{P}(\mathcal{W})|}]\!]\varphi \Rightarrow M, s \models [\![\pi^*]\!]\varphi$.

□

Beside the standard axioms from PDL, we also get an axiom from IEL, namely the distributivity of the $[\pi]$ operator over interrogatives, as follows:

Lemma 3.20 ($[\pi]$ distribution over interrogatives)**.** *For all models M, all information states s, all programs π, and all formulas φ and ψ, the following holds:*

$$M, s \models [\pi](\varphi \mathbin{\vee\!\!\!\vee} \psi) \leftrightarrow ([\pi]\varphi \vee [\pi]\psi)$$

Proof. Since both $[\pi](\varphi \mathbin{\vee\!\!\!\vee} \psi)$ and $[\pi]\varphi \vee [\pi]\psi$ are truth-conditional, we only have to show that they have the same truth conditions.

$$\begin{aligned} & M, w \models [\pi](\varphi \mathbin{\vee\!\!\!\vee} \psi) \\ \iff & M, \mathrm{R}_\pi(w) \models \varphi \mathbin{\vee\!\!\!\vee} \psi \\ \iff & M, \mathrm{R}_\pi(w) \models \varphi \text{ or } M, \mathrm{R}_\pi(w) \models \psi \\ \iff & M, w \models [\pi]\varphi \text{ or } M, w \models [\pi]\psi \\ \iff & M, w \models [\pi]\varphi \vee [\pi]\psi \end{aligned}$$

□

3.7 Translation from IEL

Since both IE-PDL and IEL are languages that contain epistemic operators, one might wonder what the relation between the two languages is. The relation of these two languages is that every formula in IEL can be translated into a equivalent IE-PDL formula. In this section, we will show that this claim holds.

Since the models for IEL and IE-PDL are almost the same, we won't have to translate between the two. The only problem is that the results in this section only hold for finite IEL models, since IE-PDL only allows for finite models. With this note out of the way, we can start with the definition of the translation of formulas.

Definition 3.11 (Translation of IEL formula to IE-PDL)**.** Given an IEL formula φ we will define the IE-PDL formula φ' to be:

$$\begin{aligned}
p' &= p \text{ for propositional atoms}\\
\bot' &= \bot\\
(\varphi_1 \wedge \varphi_2)' &= \varphi_1' \wedge \varphi_2'\\
(\varphi_1 \vee\!\!\vee \varphi_2)' &= \varphi_1' \vee\!\!\vee \varphi_2'\\
(\varphi_1 \to \varphi_2)' &= \varphi_1' \to \varphi_2'\\
(K_a\varphi)' &= [a]\varphi'\\
(E_a\varphi)' &= [\![a]\!]\varphi'\\
(K_*\varphi)' &= \left[\left(\bigcup \mathcal{A}\right)^*\right]\varphi'\\
(E_*\varphi)' &= \left[\!\!\left[\left(\bigcup \mathcal{A}\right)^*\right]\!\!\right]\varphi'
\end{aligned}$$

Before we can move on to prove that φ and φ' are equivalent, we will first have to prove that $\sigma_a(w)$ is equal to $\mathrm{R}_a(w)$ and that $sR_{A^*}t$ iff for some $w \in s$, $t \in \Sigma_*(w)$, since we will need these in the proof for the epistemic operators.

Lemma 3.21. *For any model M, any world $w \in \mathcal{W}$, and any agent $a \in \mathcal{A}$:*

$$\sigma_a(w) = \mathrm{R}_a(w)$$

Proof. For a singleton state, $\{w\}$, Definition 3.3 becomes $\{(\{w\}, t) \mid t \in \Sigma_a(w)\}$. This means that $\{t \mid \{w\}R_a t\}$ is equivalent to $\{t \mid t \in \Sigma_a(w)\} = \Sigma_a(w)$. So therefore, $\mathrm{R}_a(w) = \bigcup\{t \mid \{w\}R_a t\} = \bigcup \Sigma_a(w) = \sigma_a(w)$. □

Lemma 3.22. *For any model M, information states $s, t \subseteq \mathcal{W}$, and set of agents $\mathcal{A}$:*

$$sR_{(\bigcup \mathcal{A})^*}t \textit{ iff for some } w \in s, t \in \Sigma_*(w)$$

Proof. This proof works in two parts. We will start with the direction from left to right.

$\Rightarrow$: Take an arbitrary information state t such that $sR_{(\bigcup \mathcal{A})^*}t$. This means that there is an $\bigcup \mathcal{A}$ path from s to t, which in turn means that there exist a $u_0, \ldots u_n \subseteq \mathcal{W}$ and $a_0, \ldots a_n \in \mathcal{A}$, such that $u_0 = \{w\}, u_i R_{a_i} u_{i+1}$ for all $i < n$, and $u_n R_{a_n} t$. Since $(\bigcup \mathcal{A})^*$ and all its subprograms are declarative programs, we know that all u_i can be singleton states $\{v_i\}$. This means that it is equivalent to Definition 2.8, and therefore $t \in \Sigma_*(w)$.

$\Leftarrow$: Suppose there is some information state t such that $t \in \Sigma_*(w)$, but not $sR_{(\bigcup \mathcal{A})^*}t$. By Definition 2.8, this means that there must be an agent a_n and a world v_n such that $t \in \Sigma_{a_n}(v_n)$ and $v_0, \ldots v_n \in \mathcal{W}$ such that $v_{i+1} \in \sigma_{a_i}(v_i)$. But if $v_1 \in \sigma_{a_1}(w)$, then, by Lemma 3.21, it is also in some u such that $\{w\}R_{a_1}u$. By Definition 3.1, this means that $\{w\}R_{(\bigcup \mathcal{A})^*}u$. This also means that $v_i \in \mathrm{R}_{(\bigcup \mathcal{A})^*}(w)$ for any $i \leq n$, by the same reasoning. Then we must also have $\{w\}R_{(\bigcup \mathcal{A})^*}t$. By Proposition 3.3 we get $sR_{(\bigcup \mathcal{A})^*}t$. This contradicts our earlier assumption, so the assumption must be false. Therefore, $sR_{(\bigcup \mathcal{A})^*}t$ if for some $w \in s$, $t \in \Sigma_*(w)$.

□

To show that the formula φ' corresponds to the formula φ, we will have to show that if $M, s \models \varphi$, then $M, s \models \varphi'$.

Theorem 3.23. *For every* IEL *formula* φ *the corresponding* IE-PDL *formula* φ' *is such that:*

$$M, s \models \varphi \text{ iff } M, s \models \varphi'$$

Proof. The proof goes by induction. The base cases are trivial, since $p = p'$ and $\bot = \bot'$.

Now suppose we have two arbitrary formulas φ and ψ such that $M, s \models \varphi$ iff $M, s \models \varphi'$ and $M, s \models \psi$ iff $M, s \models \psi'$.

$\varphi \wedge \psi$: $M, s \models \varphi \wedge \psi$ iff $M, s \models \varphi$ and $M, s \models \psi$. By the inductive hypothesis we get that $M, s \models \varphi \wedge \psi$ iff $M, s \models \varphi'$ and $M, s \models \psi'$. By Definition 3.4 and Definition 3.11 we get that $M, s \models \varphi \wedge \psi$ iff $M, s \models (\varphi \wedge \psi)'$.

$\varphi \vee\!\!\!\vee \psi$: $M, s \models \varphi \vee\!\!\!\vee \psi$ iff $M, s \models \varphi$ or $M, s \models \psi$. By the inductive hypothesis we get that $M, s \models \varphi \vee\!\!\!\vee \psi$ iff $M, s \models \varphi'$ or $M, s \models \psi'$. By Definition 3.4 and Definition 3.11 we get that $M, s \models \varphi \vee\!\!\!\vee \psi$ iff $M, s \models (\varphi \vee\!\!\!\vee \psi)'$.

$\varphi \to \psi$: $M, s \models \varphi \to \psi$ iff for every $t \subseteq s$, $M, t \models \varphi$ implies $M, t \models \psi$. By the inductive hypothesis we get that $M, s \models \varphi \to \psi$ iff for every $t \subseteq s$ $M, t \models \varphi'$ implies $M, t \models \psi'$. By Definition 3.4 and Definition 3.11 we get that $M, s \models \varphi \to \psi$ iff $M, s \models (\varphi \to \psi)'$.

$K_a\varphi$: $M, s \models K_a\varphi \iff$ for every $w \in s$, $M, \sigma_a(w) \models \varphi$. By the inductive hypothesis we get that for every $w \in s$, $M, \sigma_a(w) \models \varphi'$. By Lemma 3.21 we get that $M, \sigma_a(w) \models \varphi'$ iff $M, \mathrm{R}_a(w) \models \varphi'$. By Definitions 3.4 and 3.11, this is the same as $M, s \models (K_a\varphi)'$. So $M, s \models K_a\varphi \iff M, s \models (K_a\varphi)'$.

$E_a\varphi$: $M, s \models E_a\varphi \iff$ for every $w \in s$, for every $t \in \Sigma_a(w)$, $M, t \models \varphi$. Now, by the inductive hypothesis, for all $w \in s$, and every $t \in \Sigma_a(w)$, $M, t \models \varphi$ iff $M, t \models \varphi'$. By Definition 3.3 this gives that for all $w \in s$, and every $\{w\}R_a t$, $M, t \models \varphi'$. By Proposition 3.3 this means that for every $sR_a t$, $M, t \models \varphi'$, so $M, s \models [\![a]\!]\varphi'$ which by Definition 3.11 is equivalent to $M, s \models (E_a\varphi')$. So $M, s \models E_a\varphi \iff M, s \models (E_a\varphi)'$.

$K_*\varphi$: $M, s \models K_*\varphi \iff$ for every $w \in s$, $M, \sigma_*(w) \models \varphi$. By the inductive hypothesis we get, for all $w \in s$, $M, \sigma_*(w) \models \varphi$, iff $M, \sigma_*(w) \models \varphi'$. This is the same as for every $w \in s$, $M, \bigcup \Sigma_*(w) \models \varphi'$. Now, by Lemma 3.22 we get that this is the same as for every $w \in s$, $M, \bigcup \left\{t \mid \{w\}R_{(\bigcup \mathcal{A})^*}t\right\} \models \varphi'$. In turn, this is the same as $M, s \models \left[(\bigcup \mathcal{A})^*\right]\varphi'$, which by Definition 3.11 is the same as $M, s \models (K_*\varphi)'$. So, $M, s \models K_*\varphi \iff M, s \models (K_*\varphi)'$.

$E_*\varphi$: $M, s \models E_*\varphi \iff$ for every $w \in s$, for every $t \in \Sigma_*(w)$, $M, t \models \varphi$. By the inductive hypothesis we get, for all $w \in s$, for every $t \in \Sigma_*(w)$, $M, t \models \varphi$ iff $M, t \models \varphi'$. Now, by Lemma 3.22 we get that this is the same as for every $sR_{(\bigcup \mathcal{A})^*}t$, $M, t \models \varphi'$. In turn, this is the same as $M, s \models \left[\!\!\left[(\bigcup \mathcal{A})^*\right]\!\!\right]\varphi'$, which by Definition 3.11 is the same as $M, s \models (E_*\varphi)'$. So, $M, s \models E_*\varphi \iff M, s \models (E_*\varphi)'$.

□

Something to note is that this translation only goes one way. While every formula in IEL has a corresponding formula in IE-PDL, the reverse is not true. Take for example the following formula in an arbitrary IE-PDL model: $[a](\varphi \wedge [a]\varphi)$. This formula means that agent a beliefs φ, and believes that he believes φ. This has no corresponding formula in IEL, since IEL only deals with knowledge and not belief.

3.8 Axiomatisation

In this section we will give a complete axiomatisation for IE-PDL, based on the proof system for IEL [2] and the standard axioms from PDL [12]. Because the logic of IE-PDL is a bit nonstandard for Inquisitive Semantics, since it is only defined for finite models, we will also incorporate the ideas from [9] with regard to sets of relevant formulas. We will start with defining a notion of entailment. Then we will introduce the new proof system and we will end with proving that it is complete.

3.8.1 Entailment

The notion of entailment that we will use is the same as the one used in IEL and classical logic.

Definition 3.12 (Entailment)**.** $\Phi \models_n \psi \iff$ for any model M with at most n worlds and for all states s, if $M, s \models \Phi$ then $M, s \models \psi$

Since we now have a notion of entailment, we can also prove that $[\pi]$ and $[\![\pi]\!]$ are monotonic, which is one of the steps for the soundness proof for our axiomatisation.

Proposition 3.24 (Monotonicity of $[\pi]$)**.** *For all $\Phi \subseteq \mathcal{L}^{\textsf{IE-PDL}}$ and $\psi \in \mathcal{L}^{\textsf{IE-PDL}}$, if $\Phi \models \psi$, then $[\pi]\Phi \models [\pi]\psi$, where $[\pi]\Phi = \{[\pi]\varphi \mid \varphi \in \Phi\}$.*

Proof. Suppose $\Phi \models \psi$. Now take an arbitrary IE-PDL model M and information state s such that $M, s \models [\pi]\Phi$. This means that for all $w \in s$, $M, \mathrm{R}_\pi(w) \models \Phi$. Since we have $\Phi \models \psi$, we now get for all $w \in s$, $M, \mathrm{R}_\pi(w) \models \psi$. This in turn leads to $M, s \models [\pi]\psi$. □

Proposition 3.25 (Monotonicity of $[\![\pi]\!]$)**.** *For all $\Phi \subseteq \mathcal{L}^{\textsf{IE-PDL}}$ and $\psi \in \mathcal{L}^{\textsf{IE-PDL}}$, if $\Phi \models \psi$, then $[\![\pi]\!]\Phi \models [\![\pi]\!]\psi$, where $[\![\pi]\!]\Phi = \{[\![\pi]\!]\varphi \mid \varphi \in \Phi\}$.*

Proof. The proof follows the same structure as the proof for Proposition 3.25 and is omitted. □

3.8.2 Proof system

We will give a system for natural deduction, based on the proof system for IEL [2]. The propositional fragment of the proof system is the same as the proof system for IEL, and can be found in Figure 3.4. Here we follow the usual conventions that φ and ψ are arbitrary IE-PDL formulas, and that α and β are declaratives of the same language.

While most of the rules are available for both interrogative and declarative sentences, there are some exceptions. These are the introduction of an interrogative, which can only be done from declarative sentences, and the double negation axiom, which can also only be applied if the consequent is a declarative.

The proof system will also need rules for the modalities, which can be found in Figure 3.5. Here we run into the same problem as we had with the definition of resolutions with that some axioms depend on the number of worlds in the underlying model. Therefore we will also parametrize the proof system on the number of worlds, n, in the model. The axioms here are mostly related to the reduction axioms from Section 3.6. Furthermore, we also have $[\pi]$-$[\![\pi]\!]$ Equivalence (Proposition 3.5) and Necessitation (Propositions 3.24 and 3.25).

Now we have the rules of the proof system defined, we can define some notation to specify that something is provable for models with at most n worlds.

Definition 3.13 (Proof system)**.** We write $P : \Phi \vdash_n \psi$ if there is a proof P for models with at most n worlds whose conclusion is ψ, and whose premises are included in Φ. Furthermore, we write $\varphi \dashv\vdash_n \psi$ if φ and ψ are provably equivalent, which means that $\varphi \vdash_n \psi$ and $\psi \vdash_n \varphi$.

Conjunction

Implication

$$\frac{\varphi \quad \psi}{\varphi \wedge \psi} \qquad \frac{\varphi \wedge \psi}{\varphi} \quad \frac{\varphi \wedge \psi}{\psi} \qquad \frac{\begin{array}{c}[\varphi]\\ \vdots\\ \psi\end{array}}{\varphi \to \psi} \qquad \frac{\varphi \quad \varphi \to \psi}{\psi}$$

Interrogative

Falsum

$$\frac{\alpha_i}{\alpha_1 \mathbin{\mathbb{V}} \dots \mathbin{\mathbb{V}} \alpha_i \mathbin{\mathbb{V}} \dots \mathbin{\mathbb{V}} \alpha_m} \qquad \frac{\begin{array}{c}[\alpha_1]\\ \vdots\\ \varphi\end{array} \quad \cdots \quad \begin{array}{c}[\alpha_m]\\ \vdots\\ \varphi\end{array} \quad \alpha_1 \mathbin{\mathbb{V}} \cdots \mathbin{\mathbb{V}} \alpha_m}{\varphi} \qquad \frac{\bot}{\varphi}$$

Kreisel-Putnam axiom

$(\alpha \to (\beta_1 \mathbin{\mathbb{V}} \cdots \mathbin{\mathbb{V}} \beta_m)) \to (\alpha \to \beta_1 \mathbin{\mathbb{V}} \cdots \mathbin{\mathbb{V}} \alpha \to \beta_m)$

Double negation

$\neg\neg\alpha \to \alpha$

Figure 3.4: The propositional fragment of the proof systems for every $n \in \mathbb{N}$.

Distributivity

$[\pi](\varphi \to \psi) \to ([\pi]\varphi \to [\pi]\psi)$ $\qquad$ $[\![\pi]\!](\varphi \to \psi) \to ([\![\pi]\!]\varphi \to [\![\pi]\!]\psi)$

$[\pi](\varphi \mathbin{\mathbb{V}} \psi) \to ([\pi]\varphi \vee [\pi]\psi)$

Sequence

$[\pi_1;\pi_2]\varphi \leftrightarrow [\pi_1][\pi_2]\varphi$ $\qquad$ $[\![\pi_1;\pi_2]\!]\varphi \leftrightarrow [\![\pi_1]\!][\![\pi_2]\!]\varphi$

Choice

$[\pi_1 \cup \pi_2]\varphi \leftrightarrow ([\pi_1]\varphi \wedge [\pi_2]\varphi)$ $\qquad$ $[\![\pi_1 \cup \pi_2]\!]\varphi \leftrightarrow ([\![\pi_1]\!]\varphi \wedge [\![\pi_2]\!]\varphi)$

Test

$[\varphi?]\alpha \leftrightarrow (\varphi \to \alpha)$ $\qquad$ $[\![\varphi?]\!]\psi \leftrightarrow (\varphi \to \psi)$

Unrolling

$[\pi^*]\varphi \leftrightarrow \big([\pi^1]\varphi \wedge [\pi^2]\varphi \cdots [\pi^{2^n}]\varphi\big)$ $\qquad$ $[\![\pi^*]\!]\varphi \leftrightarrow \big([\![\pi^1]\!]\varphi \wedge [\![\pi^2]\!]\varphi \cdots [\![\pi^{2^n}]\!]\varphi\big)$

Equivalence

$[\pi]\alpha \leftrightarrow [\![\pi]\!]\alpha$

Necessitation

$$\frac{\varphi}{[\pi]\varphi} \qquad \frac{\varphi}{[\![\pi]\!]\varphi}$$

Figure 3.5: The modal fragment of the proof system, for models with at most n possible worlds.

It is straightforward to check that the proof system is *sound* with respect to the semantics of IE-PDL.

Theorem 3.26 (Soundness of IE-PDL)**.** *If $\Phi \vdash_n \psi$ then $\Phi \models_n \psi$.*

Proof. The propositional part of IE-PDL is proven to be sound in [5], so we will focus on the modal part of the axiomatisation, for which we will give a proof sketch.

The soundness of the axioms has already been proven in Proposition 3.5 and Section 3.6. This means that only the proof for necessitation still has to follow. Assume we have $\Phi \vdash_n \psi$. By necessitation it follows that $[\pi]\Phi \vdash_n [\pi]\psi$. We also get $\Phi \models_n \psi$ by the inductive hypothesis. Then by Proposition 3.24 we get $[\pi]\Phi \models_n [\pi]\psi$. Therefore necessitation preserves validity for $[\pi]\psi$. The proof for the necessitation of $[\![\pi]\!]$ is similar. □

Another useful result about the proof system, is that if you can prove something for models with at most n worlds, then you can prove it for models with $m \leq n$ worlds.

Proposition 3.27. *If $\Phi \vdash_n \psi$, then for all $m \leq n$, $\Phi \vdash_m \psi$ for all $n \in \mathbb{N}$.*

Proof. Assume some arbitrary $n \in \mathbb{N}$, Φ, and ψ such that $\Phi \vdash_n \psi$. This proof goes by the induction over the structure of proofs, but all steps are trivial except for the ones for unrolling.

Take some arbitrary $m \in \mathbb{N}$ such that $m \leq n$. Then ψ is $[\pi^*]\chi$ and $[\pi^1]\chi \wedge [\pi^2]\chi \cdots [\pi^{2^n}]\chi \in \Phi$. Since $m \leq n$, $\Phi \vdash_n [\pi^1]\chi \wedge [\pi^2]\chi \cdots [\pi^{2^m}]\chi$, and thus by the inductive hypothesis and conjunction elimination and introduction $\Phi \vdash_m [\pi^1]\chi \wedge [\pi^2]\chi \cdots [\pi^{2^m}]\chi$. Then via Unrolling, $\Phi \vdash_m [\pi^*]\chi$, as required. The proof for the other direction is similar, and the proof for $[\![\pi]\!]$ is the same. □

Since the number of worlds actually only plays a role when unrolling is used, we can also prove something else.

Proposition 3.28. *If Φ does not contain any formula that contains π^*, and ψ is a π^* free formula, then $\Phi \vdash_n \psi$ implies $\Phi \vdash_m \psi$ for any $n, m \in \mathbb{N}$.*

Proof. This proof works by induction over the structure of proofs. In each step, it needs to be shown that the value of n does not matter, which is trivial, except for the unrolling axioms. But there either Φ or ψ is not π^* free, so those cases are irrelevant. □

This proposition is especially useful for formulas that can be proven in the propositional fragment of the language, since this means that everything that can be proven in the propositional fragment of IEL also holds for IE-PDL.

3.8.3 Completeness

To prove that the proof system for IE-PDL is complete, we will follow the structure from [2], but only for finite models. This means that the structure is as follows: First we will establish a tight connection between the formulas of IE-PDL and their resolutions. Then we will construct a counter-model for each declarative α, whose worlds are relevant theories of declaratives. We will prove that this counter-model is a valid IE-PDL model, and that the compositional relations are also still available. Then we will prove the finite support lemma, which is like the truth lemma from classical logics, but augmented for the support relation. Then we will use the support lemma to show that whenever $\nvdash_n \varphi$, it is also the case that $\not\models \varphi$, using the states in the counter-model.

We will start with establishing the connection between formulas and their resolutions by showing that a formula and its resolution are provably equivalent.

Lemma 3.29. *For any formula φ and all $n \in \mathbb{N}$, $\varphi \dashv\vdash_n \mathbb{V}\mathcal{R}_n(\varphi)$.*

Proof. The proof for the propositional fragment of IE-PDL is as in [5], and the proofs for $[\pi]\varphi$, $[\![\varpi]\!]\varphi$, and $[\![\pi]\!]\alpha$ are trivial, since they are their own only resolutions. Therefore we only give the proof for $[\![\pi]\!]\mu$.

Consider an interrogative formula $[\![\pi]\!]\mu$ and suppose that the inductive hypothesis holds for μ, that is, suppose that $\mu \dashv\vdash_n \bigvee\!\!\!\!\bigvee \mathcal{R}_n(\mu)$. We now need to show that $[\![\pi]\!]\mu \dashv\vdash_n \bigvee\!\!\!\!\bigvee \mathcal{R}_n([\![\pi]\!]\mu)$. We will do this by induction over the structure of programs.

a: For atomic programs, $[\![a]\!]\mu \dashv\vdash_n \bigvee\!\!\!\!\bigvee \mathcal{R}_n([\![a]\!]\mu)$ holds by definition, since the only resolution of $[\![a]\!]\mu$ is the formula itself.

$?\varphi$: The resolutions for $[\![?\varphi]\!]\mu$ are the resolutions for $\varphi \to \mu$, and these two formulas are provably equivalent (via test). The rest of this proof is then the same as the proof for implication.

For the inductive hypothesis assume that we have two arbitrary programs π_1 and π_2 such that for any formula φ, $[\![\pi_1]\!]\varphi \dashv\vdash_n \bigvee\!\!\!\!\bigvee \mathcal{R}_n([\![\pi_1]\!]\varphi)$ and $[\![\pi_2]\!]\varphi \dashv\vdash_n \bigvee\!\!\!\!\bigvee \mathcal{R}_n([\![\pi_2]\!]\varphi)$.

$\pi_1;\pi_2$: $[\![\pi_1;\pi_2]\!]\mu \dashv\vdash_n [\![\pi_1]\!][\![\pi_2]\!]\mu$. Now by the inductive hypothesis we get that $[\![\pi_1]\!][\![\pi_2]\!]\mu \dashv\vdash_n \bigvee\!\!\!\!\bigvee \mathcal{R}_n([\![\pi_1]\!][\![\pi_2]\!]\mu)$. Since the resolutions for $[\![\pi_1;\pi_2]\!]\mu$ are the resolutions for $[\![\pi_1]\!][\![\pi_2]\!]\mu$, we have that $[\![\pi_1;\pi_2]\!]\mu \dashv\vdash_n \bigvee\!\!\!\!\bigvee \mathcal{R}_n([\![\pi_1;\pi_2]\!]\mu)$.

$\pi_1 \cup \pi_2$: $[\![\pi_1 \cup \pi_2]\!]\mu \dashv\vdash_n [\![\pi_1]\!]\mu \wedge [\![\pi_2]\!]\mu$. We know from the inductive hypothesis that $[\![\pi_1]\!]\mu \dashv\vdash_n \bigvee\!\!\!\!\bigvee \mathcal{R}_n([\![\pi_1]\!]\mu)$ and $[\![\pi_2]\!]\mu \dashv\vdash_n \bigvee\!\!\!\!\bigvee \mathcal{R}_n([\![\pi_2]\!]\mu)$. Putting these together we get that $[\![\pi_1]\!]\mu \wedge [\![\pi_2]\!]\mu \dashv\vdash_n \bigvee\!\!\!\!\bigvee \mathcal{R}_n([\![\pi_1]\!]\mu) \wedge \bigvee\!\!\!\!\bigvee \mathcal{R}_n([\![\pi_2]\!]\mu)$. Then the proof works the same for conjunction to get that $\bigvee\!\!\!\!\bigvee \mathcal{R}_n([\![\pi_1]\!]\mu) \wedge \bigvee\!\!\!\!\bigvee \mathcal{R}_n([\![\pi_2]\!]\mu) \dashv\vdash_n \bigvee\!\!\!\!\bigvee \mathcal{R}_n([\![\pi_1]\!]\mu \wedge [\![\pi_2]\!]\mu)$. Now, since the resolutions for $[\![\pi_1 \cup \pi_2]\!]\mu$ are the same as those for $[\![\pi_1]\!]\mu \wedge [\![\pi_2]\!]\mu$, we get that $[\![\pi_1 \cup \pi_2]\!]\mu \dashv\vdash_n \bigvee\!\!\!\!\bigvee \mathcal{R}_n([\![\pi_1 \cup \pi_2]\!]\mu)$.

π_1^*: $[\![\pi_1^*]\!]\mu \dashv\vdash_n [\![\pi_1^1]\!]\mu \wedge [\![\pi_1^2]\!]\mu \cdots [\![\pi_1^{2^n}]\!]\mu$. If we now use the proofs for sequence we get $[\![\pi_1^1]\!]\mu \wedge [\![\pi_1^2]\!]\mu \cdots [\![\pi_1^{2^n}]\!]\mu \dashv\vdash_n \bigvee\!\!\!\!\bigvee \mathcal{R}_n([\![\pi_1^1]\!]\mu) \wedge \bigvee\!\!\!\!\bigvee \mathcal{R}_n([\![\pi_1^2]\!]\mu) \cdots \bigvee\!\!\!\!\bigvee \mathcal{R}_n([\![\pi_1^{2^n}]\!]\mu)$. If we then use the proof for conjunction then we get that $\bigvee\!\!\!\!\bigvee \mathcal{R}_n([\![\pi_1^1]\!]\mu) \wedge \bigvee\!\!\!\!\bigvee \mathcal{R}_n([\![\pi_1^2]\!]\mu) \cdots \bigvee\!\!\!\!\bigvee \mathcal{R}_n([\![\pi_1^{2^n}]\!]\mu) \dashv\vdash_n \bigvee\!\!\!\!\bigvee \mathcal{R}_n([\![\pi_1^1]\!]\mu \wedge [\![\pi_1^2]\!]\mu \cdots [\![\pi_1^{2^n}]\!]\mu)$. Since the resolutions for $[\![\pi_1^*]\!]\mu$ are the same as those for $[\![\pi_1^1]\!]\mu \wedge [\![\pi_1^2]\!]\mu \cdots [\![\pi_1^{2^n}]\!]\mu$, we get that $[\![\pi_1^*]\!]\mu \dashv\vdash_n \bigvee\!\!\!\!\bigvee \mathcal{R}_n([\![\pi_1^*]\!]\mu)$.

□

As a corollary, we now get that every formula is derivable from any of its resolutions.

Corollary 3.30. *If $\alpha \in \mathcal{R}_n(\varphi)$, then $\alpha \vdash_n \varphi$.*

Proof. If $\alpha \in \mathcal{R}_n(\varphi)$, then by a single application of $\vee\!\!\!\vee$ introduction we have $\alpha \vdash_n \bigvee\!\!\!\!\bigvee \mathcal{R}_n(\varphi)$. Then by Lemma 3.29 we get that $\alpha \vdash_n \varphi$. □

Furthermore, thanks to the distributivity axioms and the necessitation rules we can ensure that the modalities are monotonic.

Lemma 3.31. *If $\varphi_1, \ldots, \varphi_m \vdash_n \psi$, then $\Box\varphi_1, \ldots, \Box\varphi_m \vdash_n \Box\psi$ for all $\Box \in \{[\![\pi]\!], [\pi] \mid \pi \in \Pi\}$.*

Proof. This proof works via the distribution and the necessitation rules. From $\varphi_1, \ldots, \varphi_m \vdash_n \psi$ we can prove $\vdash_n (\varphi_1 \wedge \cdots \wedge \varphi_m) \to \psi$, which using necessitation gives us $\vdash_n \Box\left((\varphi_1 \wedge \cdots \wedge \varphi_m) \to \psi\right)$ for all $\Box \in \{[\![\pi]\!], [\pi] \mid \pi \in \Pi\}$. Using the distribution rules we then get $\vdash_n (\Box\varphi_1 \wedge \cdots \wedge \Box\varphi_m) \to \Box\psi$ for all $\Box \in \{[\![\pi]\!], [\pi] \mid \pi \in \Pi\}$, from which we can prove $\Box\varphi_1, \ldots \Box\varphi_m \vdash_n \Box\psi$ for all $\Box \in \{[\![\pi]\!], [\pi] \mid \pi \in \Pi\}$. □

Like in IEL and AMLI we also get the result that we can express a formula of the form $[\pi]\varphi$ in terms of $[\![\pi]\!]$. As a result of this, we can completely paraphrase $[\pi]$ out of the language.

Lemma 3.32. *For any φ, $[\pi]\varphi \dashv\vdash_n \bigvee_{\alpha \in \mathcal{R}_n(\varphi)} [\pi]\alpha \dashv\vdash_n \bigvee_{\alpha \in \mathcal{R}_n(\varphi)} [\![\pi]\!]\alpha$.*

Proof. From Lemma 3.29 we have that $\varphi \dashv\vdash_n \bigvee\!\!\!\!\bigvee \mathcal{R}_n(\varphi)$. From Lemma 3.31, we get that $[\pi]\varphi \dashv\vdash_n [\pi] \bigvee\!\!\!\!\bigvee \mathcal{R}(\varphi) \dashv\vdash_n \bigvee_{\alpha \in \mathcal{R}_n(\varphi)} [\pi]\alpha$. Now with a simple application of the equivalence of $[\pi]$ and $[\![\pi]\!]$ on declaratives, we get that $[\pi]\varphi \dashv\vdash_n \bigvee_{\alpha \in \mathcal{R}_n(\varphi)} [\pi]\alpha \dashv\vdash_n \bigvee_{\alpha \in \mathcal{R}_n(\varphi)} [\![\pi]\!]\alpha$. ☐

Then lastly we have the resolution theorem. This theorem states that we can derive ψ from Φ iff we can derive some resolution α of ψ from some resolution Γ of Φ. Before we can cover this theorem, we will first have to establish two other lemmas. The first of these is a standard lemma from IEL, but the second one is new. Normally, a modal formula is its own resolution, which simplifies the Resolution Theorem. However, since $[\![?\varphi]\!]\psi$ is not its own resolution, this does not hold, which means that we will need an additional lemma. We will first cover the first lemma.

Lemma 3.33. *If $\Phi \nvdash_n \psi$ then there exists some $\Gamma \in \mathcal{R}_n(\Phi)$ such that $\Gamma \nvdash_n \psi$ for all $n \in \mathbb{N}$.*

Proof. The proof of this lemma is as in [2], but with the addition of taking some arbitrary n for the proof system and resolutions. ☐

Lemma 3.34. *If $\beta \in \mathcal{R}_n(\varphi)$ and $\alpha \in \mathcal{R}_n([\![\pi]\!]\varphi)$, then $[\![\pi]\!]\beta \vdash_n \alpha$ for all $n \in \mathbb{N}$.*

Proof. This proof works by induction over programs π. We fix some arbitrary $n \in \mathbb{N}$.

a: $[\![a]\!]\varphi$ is its own resolution, so this follows from $\beta \vdash_n \varphi$ and Lemma 3.31.

$?\psi$: $[\![?\psi]\!]\varphi$ has the same resolutions as $\psi \to \varphi$, so we have to find an $\alpha \in \mathcal{R}_n(\psi \to \varphi)$ such that $[\![?\psi]\!]\beta \vdash_n \alpha$. Now, for all $\gamma_i \in \mathcal{R}_n(\psi)$, we can get a proof $Q_i : \gamma_i \vdash_n \beta$ thanks to Corollary 3.30 and $\psi \to \beta$, which means we get $\bigwedge_{\gamma_i \in \mathcal{R}_n(\psi)} \gamma_i \to \beta$, which is a resolution of $\psi \to \varphi$. And since we have $[\![?\psi]\!]\beta \vdash_n \psi \to \beta$, we get that $[\![?\psi]\!]\beta \vdash_n \alpha$ where $\alpha \in \mathcal{R}_n([\![?\psi]\!]\varphi)$.

For the inductive hypothesis we take some arbitrary π_1 and π_2 and suppose that:

- if $\beta \in \mathcal{R}_n(\varphi)$ and $\alpha \in \mathcal{R}_n([\![\pi_1]\!]\varphi)$, then $[\![\pi_1]\!]\beta \vdash_n \alpha$;
- if $\beta \in \mathcal{R}_n(\varphi)$ and $\alpha \in \mathcal{R}_n([\![\pi_2]\!]\varphi)$, then $[\![\pi_2]\!]\beta \vdash_n \alpha$.

$\pi_1; \pi_2$: Take some β such that $\beta \in \mathcal{R}_n(\varphi)$. By the inductive hypothesis we get that $[\![\pi_2]\!]\beta \vdash_n \gamma$ where $\gamma \in \mathcal{R}_n([\![\pi_2]\!]\varphi)$. By applying the inductive hypothesis again we get that $[\![\pi_1]\!]\gamma \vdash_n \alpha$ where $\alpha \in \mathcal{R}_n([\![\pi_1]\!][\![\pi_2]\!]\varphi)$ and thus also $\mathcal{R}_n([\![\pi_1; \pi_2]\!]\varphi)$. By applying Lemma 3.31 on the first one, we get that $[\![\pi_1]\!][\![\pi_2]\!]\beta \vdash_n [\![\pi_1]\!]\gamma$, and from sequence we get that $[\![\pi_1; \pi_2]\!]\beta \vdash_n [\![\pi_1]\!][\![\pi_2]\!]\beta$. Putting this all together we get $[\![\pi_1; \pi_2]\!]\beta \vdash_n \alpha$.

$\pi_1 \cup \pi_2$: So we have some $\alpha \in \mathcal{R}_n([\![\pi_1 \cup \pi_2]\!]\varphi)$ which is of the form $\gamma_1 \wedge \gamma_2$ with $\gamma_1 \in \mathcal{R}_n([\![\pi_1]\!]\varphi)$ and $\gamma_2 \in \mathcal{R}_n([\![\pi_2]\!]\varphi)$. Then we get $[\![\pi_1]\!]\beta \vdash_n \gamma_1$ and $[\![\pi_2]\!]\beta \vdash_n \gamma_2$ for all $\beta \in \mathcal{R}_n(\varphi)$ by the inductive hypothesis. Since we get $[\![\pi_1 \cup \pi_2]\!]\beta \vdash_n [\![\pi_1]\!]\beta \wedge [\![\pi_2]\!]\beta$, by choice, if we put it all together we get $[\![\pi_1 \cup \pi_2]\!]\beta \vdash_n \alpha$.

π^*: A resolution $\alpha \in \mathcal{R}_n([\![\pi^*]\!]\varphi)$ has the form $\bigwedge_{1 \le m \le n} \gamma_m$ where $\gamma_m \in \mathcal{R}_n([\![\pi^m]\!]\varphi)$. Now take some $\beta \in \mathcal{R}_n(\varphi)$. Since $[\![\pi^*]\!]\beta \vdash_n \bigwedge_{1 \le m \le n} [\![\pi^m]\!]\beta$ and $[\![\pi^n]\!]\beta \vdash_n \gamma_m$ (by the inductive hypothesis), we get that $[\![\pi^*]\!]\beta \vdash_n \bigwedge_{1 \le m \le n} \gamma_m$.

Since we fixed some arbitrary $n \in \mathbb{N}$, we can conclude that this holds for all $n \in \mathbb{N}$. ☐

Basically, this lemma tells us how we can give the proof $[\![\pi]\!]\beta \vdash_n [\![\pi]\!]\varphi$ for all $\beta \in \mathcal{R}_n(\varphi)$. Corollary 3.30 and Lemma 3.31 already guaranteed us that this proof exists, but the proofs for Lemma 3.34 and Corollary 3.30 also give us the structure for this proof itself.

Theorem 3.35 (Resolution theorem)**.** *For all $n \in \mathbb{N}$: $\Phi \vdash_n \psi \iff$ for all $\Gamma \in \mathcal{R}_n(\Phi)$ there exists some $\alpha \in \mathcal{R}_n(\psi)$ such that $\Gamma \vdash_n \alpha$.*

Proof. We will start with the left to right direction of the proof: if Φ derives ψ, then any resolution Γ of Φ derives some resolution α of ψ. The proof is by induction on the complexity of a proof $P : \Phi \vdash_n \psi$. In the interest of space, we will leave out the rules from the propositional fragment of the proof system, since these have already been shown in [2]. For all of these steps we will fix some arbitrary $n \in \mathbb{N}$.

- ψ is an undischarged assumption, $\psi \in \Phi$. In this case, any resolution Γ of Φ contains a resolution α of ψ by definition, so $\Gamma \vdash_n \alpha$.

- ψ is an axiom. If ψ is declarative, then the claim is trivially true. If ψ is interrogative, it is either an instance of the Kreisel-Putnam axiom, for which a proof can be found in [2], or one of the reduction axioms for $[\![\pi]\!]$. This last group we can put into two categories: the distributivity axiom and the reduction axioms for programs.

  We will start with the distributivity axiom, $\psi := [\![\pi]\!](\varphi \to \mu) \to ([\![\pi]\!]\varphi \to [\![\pi]\!]\mu)$. Take $\alpha = \bigwedge_{\beta \in \mathcal{R}_n([\![\pi]\!](\varphi \to \mu))} \beta \to f(\beta)$: α is a resolution of ψ, and because of the function f, itself an instance of the distributivity axiom. Therefore we have $\Gamma \vdash_n \alpha$ for any set Γ.

  For the reduction axioms for programs, we will pick one to use as an example, so we take $\psi = [\![\pi_1 \cup \pi_2]\!]\varphi \leftrightarrow ([\![\pi_1]\!]\varphi \wedge [\![\pi_2]\!]\varphi) \iff ([\![\pi_1 \cup \pi_2]\!]\varphi \to ([\![\pi_1]\!]\varphi \wedge [\![\pi_2]\!]\varphi)) \wedge (([\![\pi_1]\!]\varphi \wedge [\![\pi_2]\!]\varphi) \to [\![\pi_1 \cup \pi_2]\!]\varphi)$. Now take $\alpha = (\beta_1 \to \beta_2) \wedge (\beta_3 \to \beta_4)$ where $\beta_1, \beta_4 \in \mathcal{R}_n([\![\pi_1 \cup \pi_2]\!]\varphi)$ and $\beta_2, \beta_3 \in \mathcal{R}_n([\![\pi_1]\!]\varphi \wedge [\![\pi_2]\!]\varphi)$. Since $\mathcal{R}_n([\![\pi_1 \cup \pi_2]\!]\varphi) = \mathcal{R}_n([\![\pi_1]\!]\varphi \wedge [\![\pi_2]\!]\varphi)$, we can take $\beta_1 = \beta_2$ and $\beta_3 = \beta_4$. Then we get $\alpha = (\beta_1 \to \beta_1) \wedge (\beta_3 \to \beta_3)$ which is a resolution for ψ and, since it is a classical tautology, we get $\Gamma \vdash_n \alpha$ for any set Γ.

  The proof for the rest of the reduction axioms, except for unrolling is similar. The unrolling axioms are given in separate steps.

- $\psi = [\![\pi^*]\!]\varphi$ was obtained from an application of the unrolling axiom. Then the immediate subproof of P is a proof $P_0 : \Phi \vdash_n \bigwedge_{1 \leq m \leq n} [\![\pi^m]\!]\varphi$ and a proof $P_1 : \Phi \cup \left\{\bigwedge_{1 \leq m \leq n} [\![\pi^m]\!]\varphi\right\} \vdash [\![\pi^*]\!]\varphi$ using the unrolling axiom. Take any resolution Γ of Φ, and take α_m to be a resolution of $[\![\pi^m]\!]\varphi$. Then by the inductive hypothesis we get a proof $Q_0 : \Gamma \vdash \bigwedge_{1 \leq m \leq n} \alpha_m$, of which the conclusion is a resolution of $[\![\pi^*]\!]\varphi$.

- $\psi = \bigwedge_{1 \leq m \leq n} [\![\pi^m]\!]\varphi$ was obtained from an application of the unrolling axiom. Then the immediate subproof of P is a proof $P_0 : \Phi \vdash_n [\![\pi^*]\!]\varphi$ and a proof $P_1 : \Phi \cup \{[\![\pi^*]\!]\varphi\} \vdash_n \bigwedge_{1 \leq m \leq n} [\![\pi^m]\!]\varphi$ using the unrolling axiom. Take any resolution Γ of Φ, and take α_m to be a resolution of $[\![\pi^m]\!]\varphi$. Then by the inductive hypothesis we get a proof $Q_0 : \Gamma \vdash \bigwedge_{1 \leq m \leq n} \alpha_m$, of which the conclusion is a resolution of $\bigwedge_{1 \leq m \leq n} [\![\pi^m]\!]\varphi$.

- $\psi = [\![\pi]\!]\varphi$ was obtained from a $[\![\pi]\!]$-necessitation rule from φ. Then the immediate subproof of P is a proof $P_0 : \Phi \vdash_n \varphi$. Since there can be no undischarged assumptions for the necessitation rule, we also know that $\Phi = \emptyset$. Now consider a resolution Γ of Φ, where $\Gamma = \emptyset$, since that is the only resolution of Φ. Then by the inductive hypothesis we have some proof $Q : \Gamma \vdash_n \beta$ for some $\beta \in \mathcal{R}_n(\varphi)$. Then by Lemma 3.31 we get that $\Gamma \vdash_n [\![\pi]\!]\beta$. Now by Lemma 3.34 we get that $[\![\pi]\!]\beta \vdash_n \alpha$ for some $\alpha \in \mathcal{R}_n([\![\pi]\!]\varphi)$, which in turn allows us to conclude $\Gamma \vdash_n \alpha$, which is what we needed to prove.

This covers the left to right direction of the resolution theorem. The right to left direction depends on Lemma 3.33 and is the same as in [2]. $\square$

As a corollary of this theorem, we also get the following, since the resolutions of a set of declaratives are that set itself.

Corollary 3.36 (Split)**.** *Let Γ be a set of declaratives. If $\Gamma \vdash_n \psi$ then $\Gamma \vdash_n \alpha$ for some $\alpha \in \mathcal{R}_n(\psi)$ for all $n \in \mathbb{N}$.*

Counter-Model

Unlike what is usual in modal logic, we will not be using the usual method of defining a canonical model. This is because our models need to be finite, so the model we use in the completeness proof also needs to be finite. Since our logic is an inquisitive logic, we will also need a richer structure than a standard Kripke model, which also presents a problem.

The second of these problems we will solve by using a similar model construction as in [2], the most relevant part of which right now is the use of *complete theories of declaratives* (CTD) as the worlds instead of the maximally consistent sets from modal logic.

To then solve the second problem we will constrain our CTDs to sets with only relevant formulas for the proof, as is also done in [9]. As [9] only does this for Epistemic Logic, we will use the definition of a subformula from [11] for defining the set of relevant formulas, which we will call the closure of a formula φ. We can then use this set to build *relevant theories of declaratives* (RTD) for a given declarative formula α.

Before we will jump into the definitions, we will first have to consider what the relevant formulas are. Like in [9, 11] we are interested in the subformula of a given formula φ. However, in the RTD, we will only want declarative formulas. Therefore, one might be tempted to build a closure of a formula based on the resolutions of that formula. While this would work well in the general case, we can only calculate the resolutions with respect to a certain number of worlds, a number that we currently do not know. This means that we will have to go about this in a bit of a roundabout way and postpone the taking of the resolutions when this number of worlds is known. However, when we talk about *relevant formulas*, we mean both the subformulas of φ, and their resolutions with respect to a certain maximum number of worlds.

Definition 3.14 (Closure of a formula)**.** The closure of a formula φ, $\mathcal{C}(\varphi)$ is the smallest set such that:

- $\varphi \in \mathcal{C}(\varphi)$
- if $\neg\psi \in \mathcal{C}(\varphi)$, then $\psi \in \mathcal{C}(\varphi)$.
- if $\psi \in \mathcal{C}(\varphi)$ and ψ is not a negation, then $\neg\psi \in \mathcal{C}(\varphi)$.
- if $\psi \wedge \chi \in \mathcal{C}(\varphi)$, then $\psi \in \mathcal{C}(\varphi)$ and $\chi \in \mathcal{C}(\varphi)$.
- if $\psi \vee\!\!\!\vee \chi \in \mathcal{C}(\varphi)$, then $\psi \in \mathcal{C}(\varphi)$ and $\chi \in \mathcal{C}(\varphi)$.
- if $\psi \to \chi \in \mathcal{C}(\varphi)$, then $\psi \in \mathcal{C}(\varphi)$ and, if $\chi \neq \bot$, $\chi \in \mathcal{C}(\varphi)$.
- if $\Box\psi \in \mathcal{C}(\varphi)$, then $\psi \in \mathcal{C}(\varphi)$ for $\Box \in \{[\pi], [\![\pi]\!]\}$.
- if $[?\psi]\chi \in \mathcal{C}(\varphi)$ or $[\![?\psi]\!]\chi \in \mathcal{C}(\varphi)$, then $\psi \in \mathcal{C}(\varphi)$.
- if $[\pi_1;\pi_2]\psi \in \mathcal{C}(\varphi)$, then $[\pi_1][\pi_2]\psi \in \mathcal{C}(\varphi)$.
- if $[\![\pi_1;\pi_2]\!]\psi \in \mathcal{C}(\varphi)$, then $[\![\pi_1]\!][\![\pi_2]\!]\psi \in \mathcal{C}(\varphi)$.
- if $[\pi_1 \cup \pi_2]\psi \in \mathcal{C}(\varphi)$, then $[\pi_1] \in \mathcal{C}(\varphi)$ and $[\pi_2] \in \mathcal{C}(\varphi)$.

- if $[\![\pi_1 \cup \pi_2]\!]\psi \in \mathcal{C}(\varphi)$, then $[\![\pi_1]\!] \in \mathcal{C}(\varphi)$ and $[\![\pi_2]\!] \in \mathcal{C}(\varphi)$.
- if $[\pi^*]\psi \in \mathcal{C}(\varphi)$, then $[\pi][\pi^*]\psi \in \mathcal{C}(\varphi)$.
- if $[\![\pi^*]\!]\psi \in \mathcal{C}(\varphi)$, then $[\![\pi]\!][\![\pi^*]\!]\psi \in \mathcal{C}(\varphi)$.

Since the negation of a formula ψ is an abbreviation for $\psi \to \bot$, but $\bot$ is not relevant for the closure of this formula, we specifically do not add it to the closure when it appears in the consequent position of an implication.

Before we will continue with the definition of the model, we will first prove some things about the closures. The first of these have to do with the size of the closure of a given formula, which we would like to express in terms of the length of that formula. For this, we will first have to introduce the length of a formula.

Definition 3.15 (Length of a formula)**.** The length of a formula is defined by simultaneous induction on the structure of formulas and programs:

$$\begin{aligned} |p| &= 1 & |a| &= 1 \\ |\varphi \wedge \psi| &= |\varphi| + 1 + |\psi| & |?\varphi| &= 1 + |\varphi| \\ |\varphi \mathbin{\vee\!\!\!\vee} \psi| &= |\varphi| + 1 + |\psi| & |\pi_1;\pi_2| &= |\pi_1| + 1 + |\pi_2| \\ |\varphi \to \psi| &= |\varphi| + 1 + |\psi| & |\pi_1 \cup \pi_2| &= |\pi_1| + 1 + |\pi_2| \\ |[\pi]\varphi| &= |\pi| + |\varphi| + 1 & |\pi^*| &= |\pi| + 1 \\ |[\![\pi]\!]\varphi| &= |\pi| + |\varphi| + 1 \end{aligned}$$

Now we can go on and determine the maximum size of a closure and its maximal consistent subsets.

Proposition 3.37. $|\mathcal{C}(\varphi)| \leq 2|\varphi|$.

Proof. This proof works by induction over the structure of φ.

p: $|\mathcal{C}(p)| = |\{p, \neg p\}| = 2 = 2|p|$.

Now take two arbitrary formula φ_1 and φ_2, such that:

- $|\mathcal{C}(\varphi_1)| \leq 2|\varphi_1|$
- $|\mathcal{C}(\varphi_2)| \leq 2|\varphi_2|$

$\varphi_1 \wedge \varphi_2$: This proof also works by rewriting.

$$\begin{aligned} |\mathcal{C}(\varphi_1 \wedge \varphi_2)| &= |\{\varphi_1 \wedge \varphi_2, \neg(\varphi_1 \wedge \varphi_2)\} \cup \mathcal{C}(\varphi_1) \cup \mathcal{C}(\varphi_2)| \\ &\leq |\{\varphi_1 \wedge \varphi_2, \neg(\varphi_1 \wedge \varphi_2)\}| + |\mathcal{C}(\varphi_1)| + |\mathcal{C}(\varphi_2)| \\ &\leq 2 + 2|\varphi_1| + 2|\varphi_2| \\ &= 2 \times (|\varphi_1| + |\varphi_2| + 1) \\ &= 2|\varphi_1 \wedge \varphi_2| \end{aligned}$$

$\varphi_1 \mathbin{\vee\!\!\!\vee} \varphi_2$: This step is analogous to the step for conjunction.

$\varphi_1 \to \varphi_2$: This step is analogous to the step for conjunction.

$[\pi]\varphi_1$: This step goes by induction on the structure of programs.

a: $|\mathcal{C}([a]\varphi_1)| = |\{[a]\varphi_1, \neg[a]\varphi_1\} \cup \mathcal{C}(\varphi_1)| \leq |\{[a]\varphi_1, \neg[a]\varphi_1\}| + |\mathcal{C}(\varphi_1)| \leq 2 + 2|\varphi_1| = 2 \times (1 + |\varphi_1|) \leq 2 \times (2 + |\varphi_1|) = 2|[a]\varphi_1|$

$?\varphi_2$: $|\mathcal{C}([?\varphi_2]\varphi_1)| = |\{[?\varphi_2]\varphi_1, \neg[?\varphi_2]\varphi_1\} \cup \mathcal{C}(\varphi_2) \cup \mathcal{C}(\varphi_1)|$. From this point onward it is the same as the case for conjunction.

Now take some arbitrary programs π_1 and π_2 such that for φ_1:

- $\mathcal{C}([\pi_1]\varphi_1) \leq 2|[\pi_1]\varphi_1|$
- $\mathcal{C}([\pi_2]\varphi_1) \leq 2|[\pi_2]\varphi_1|$

$\pi_1;\pi_2$: In this step we need to built the closure of $\mathcal{C}([\pi_1][\pi_2])$ out of the closure of $\mathcal{C}([\pi_1]\varphi_1)$ and $\mathcal{C}([\pi_2]\varphi_1)$, which is a bit tricky. This is done by taking the closure of $\mathcal{C}([\pi_1]\varphi_1)$, and subtracting the extra elements, which are the elements of $\mathcal{C}(\varphi_1)$ and $[\pi_1]\varphi_1$ and $\neg[\pi_1]\varphi_1$.

$$\begin{aligned}|\mathcal{C}([\pi_1;\pi_2]\varphi_1)| &= |\{[\pi_1;\pi_2]\varphi_1, \neg[\pi_1;\pi_2]\varphi_1, [\pi_1][\pi_2]\varphi_1, \neg[\pi_1][\pi_2]\varphi_1\} \cup \mathcal{C}([\varphi_2]\varphi_1) \cup \\ &\quad (\mathcal{C}([\pi_1]\varphi_1) \setminus (\mathcal{C}(\varphi_1) \cup \{[\pi_1]\varphi_1, \neg[\pi_1]\varphi_1\}))| \\ &\leq |\{[\pi_1;\pi_2]\varphi_1, \neg[\pi_1;\pi_2]\varphi_1, [\pi_1][\pi_2]\varphi_1, \neg[\pi_1][\pi_2]\varphi_1\}| + |\mathcal{C}([\varphi_2]\varphi_1) \cup \\ &\quad (\mathcal{C}([\pi_1]\varphi_1) \setminus (\mathcal{C}(\varphi_1) \cup \{[\pi_1]\varphi_1, \neg[\pi_1]\varphi_1\}))| \\ &\leq 4 + 2|\mathcal{C}([\varphi_2]\varphi_1)| + 2|\mathcal{C}([\pi_1]\varphi_1)| - 2 \\ &\leq 2 + 2|\mathcal{C}([\varphi_2]\varphi_1)| + 2|\mathcal{C}([\pi_1]\varphi_1)| \\ &= 2 \times (|\mathcal{C}([\pi_1]\varphi_1)| + |\mathcal{C}([\pi_2]\varphi_1)| + 1) \\ &\leq 2|[\pi_1;\pi_2]\varphi_1|\end{aligned}$$

$\pi_1 \cup \pi_2$: This step is analogous to the case for conjunction.

$[\pi_1^*]\varphi_1$: Here we run into the same problem as with sequence, so we will yet again have to remove $[\pi_1]\varphi_1$ and $\neg[\pi_1]\varphi_1$ from the set.

$$\begin{aligned}|\mathcal{C}([\pi_1^*]\varphi_1)| &= |\{[\pi^*]\varphi_1, \neg[\pi^*]\varphi_1, [\pi_1][\pi^*]\varphi_1, \neg[\pi_1][\pi_1^*]\varphi_1\} \cup \\ &\quad \mathcal{C}([\pi_1]\varphi_1) \setminus \{[\pi_1]\varphi_1, \neg[\pi_1]\varphi_1\}| \\ &= |\{[\pi^*]\varphi_1, \neg[\pi^*]\varphi_1, [\pi_1][\pi^*]\varphi_1, \neg[\pi_1][\pi_1^*]\varphi_1\}| + \\ &\quad |\mathcal{C}([\pi_1]\varphi_1)| - |\{[\pi_1]\varphi_1, \neg[\pi_1]\varphi_1\}| \\ &\leq 4 + 2|[\pi_1]\varphi_1| - 2 \\ &= 2 + 2|[\pi_1]\varphi_1| \\ &= 2|[\pi^*]\varphi_1|\end{aligned}$$

$[\![\pi]\!]\varphi_1$: This step is the same as the step for $[\pi]\varphi_1$.

□

Definition 3.16. A maximal consistent subset of a closure $\mathcal{C}(\varphi)$ are all $\Phi \subseteq \mathcal{C}(\varphi)$ such that:

- Φ is consistent, which means that $\Phi \nvdash_n \bot$ for all $n \in \mathbb{N}$;
- there is no set $\Phi' \subseteq \mathcal{C}(\varphi)$ such that $\Phi \subset \Phi'$ and Φ' is consistent.

Proposition 3.38. *A closure $\mathcal{C}(\varphi)$ has at most $2^{\frac{|\mathcal{C}(\varphi)|}{2}}$ maximal consistent subsets.*

Proof. In order to prove this, we will have to establish the following two things:

1. a maximal consistent subset Φ of $\mathcal{C}(\varphi)$ has either $\psi \in \Phi$ or $\neg\psi \in \Phi$ for all non-negated $\psi \in \mathcal{C}(\varphi)$;
2. $|\mathcal{C}(\varphi)|$ is even;

The first follows from the fact that Φ is both maximal and consistent. If it would have both, it would not be consistent, if it had neither, there would be some set, namely either $\Phi \cup \{\psi\}$ or $\Phi \cup \{\neg\psi\}$, such that Φ would be a proper subset. So it must contain either ψ or $\neg\psi$ for all non-negated $\psi \in \mathcal{C}(\varphi)$. The second is true by Proposition 3.37.

This means that if we were to build a maximally consistent subset Φ of $\mathcal{C}(\varphi)$, for each formula $\psi \in \mathcal{C}(\varphi)$ we can have maximally 2 choices, and we only get this choice for half of the formulas, for if we have $\neg\psi \in \Phi$, we cannot include ψ is Φ, because it needs to be consistent, and we were to include neither, Φ could not be maximal. So we have $2^{\frac{|\mathcal{C}(\varphi)|}{2}}$ different choices for maximally consistent subsets of $\mathcal{C}(\varphi)$. ☐

Corollary 3.39. *A closure $\mathcal{C}(\varphi)$ has at most $2^{|\varphi|}$ maximal consistent subsets.*

Now we can use the closure of a formula to build up a RTD. While closures are defined for all formulas φ, RTDs will only be defined for declaratives. This is because they are only required for declaratives in order to make the completeness proof work. However, since the closure of a declarative can contain interrogatives — $[\![\varpi]\!]\mu$ has μ as a subformula — we will have to replace those interrogatives in such a way that they still prove φ. Luckily, we can use the resolutions of φ for that purpose. And since the RTDs will function as worlds in the counter-model, we will now also know for how many worlds we maximally need to calculate the resolutions.

We will define these RTDs in terms of certain functions, called resolution functions, that will map every formula in a closure $\mathcal{C}(\alpha)$ onto exactly one of its resolutions.

Definition 3.17 (Resolution function)**.** A *resolution function* of a set Φ is a function $f_n : \Phi \to \mathcal{R}_n(\Phi)$ such that for each $\varphi \in \Phi$, $f_n(\varphi) \in \mathcal{R}_n(\varphi)$. A resolution function is called *n-consistent*, meaning consistent for models with at most n worlds, if $\{f_n(\varphi) \mid \varphi \in \Phi\} \nvdash_n \bot$ for that $n \in \mathbb{N}$.

By Lemma 3.33 we know that for every consistent set of formulas and every $n \in \mathbb{N}$, there is at least one n-consistent resolution function.

Definition 3.18 (Relevant theory of declaratives)**.** A relevant theory of declaratives (RTD) Γ for a given declarative α is the image of $\mathcal{C}(\alpha)$ under an $2^{|\alpha|}$-consistent resolution function for $\mathcal{C}(\alpha)$.

We will use $\text{RTD}_f(\alpha)$ for the set of all relevant theories of declaratives for α under resolution function $f_{2^{|\alpha|}}$.

Since we now replace every maximally consistent subset of $\mathcal{C}(\alpha)$ with at most one new set, there are also maximally $2^{|\alpha|}$ RTDs for a given formula α.

For ease of reading, we will also introduce notation for the set of relevant formula.

Definition 3.19 (Relevant formula)**.** For a given formula α, its set of relevant formula $\mathcal{RC}(\alpha)$ is defined as:

$$\mathcal{RC}(\alpha) = \mathcal{C}(\alpha) \cup \bigcup \mathcal{R}_{2^{|\alpha|}}(\mathcal{C}(\alpha))$$

Like CTDs, the RTDs also have the disjunction property.

Fact 3.1 (Disjunction Property)**.** If Γ is a RTD and $\alpha_1 \vee \cdots \vee \alpha_n \in \Gamma$, then $\alpha_i \in \Gamma$ for some i.

However, it is now no longer possible to extend an arbitrary set of formulas into an RTD. Luckily, we also only have to deal with a subset of the formulas in IE-PDL, which allows us to prove a similar lemma. Before we can do this however, we will first have to prove the Lindenbaümchen Lemma, which shows that every subset of $\mathcal{C}(\varphi)$ is extendible into a maximally consistent one.

Lemma 3.40 (Lindenbaümchen Lemma)**.** *If Ψ is a consistent subset of $\mathcal{C}(\varphi)$, then Ψ is a subset of some maximally consistent subset of $\mathcal{C}(\varphi)$.*

Proof. We can enumerate all of the formulas $\varphi_0, \varphi_1, \dots \varphi_m \in \mathcal{C}(\alpha)$ and define a sequence of sets of formulas

$$\Psi_0 \subseteq \Psi_1 \subseteq \cdots \subseteq \Psi_m$$

as follows, by induction on m:

$$\begin{aligned} \Psi_0 &= \Psi \\ \Psi_{m+1} &= \begin{cases} \Psi_m \cup \{\psi_m\} & \text{If } \Psi_m \cup \{\psi_m\} \text{ is consistent} \\ \Psi_m & \text{otherwise} \end{cases} \\ \Psi' &= \bigcup_{m' \leq m} \Psi_{m'} \end{aligned}$$

We now need to show that this set Ψ' is a maximally consistent subset of $\mathcal{C}(\varphi)$. Since it is consistent by construction, we will focus on the second claim.

Suppose that Ψ' is not a complete subset of $\mathcal{C}(\varphi)$. Then there must be some non-negated $\chi \in \mathcal{C}(\varphi)$ such that neither $\chi \in \Psi'$ nor $\neg\chi \in \Psi'$, while either $\Psi' \cup \{\chi\}$ or $\Psi' \cup \{\neg\chi\}$ is consistent. Take χ to be ψ_i in the enumeration. Then $\Psi_i \cup \{\psi_i\}$ must be inconsistent, since otherwise ψ_i would have been included in Ψ'. But then $\Psi' \cup \{\psi_i\}$ is also inconsistent, which contradicts the earlier assumption. Therefore, Ψ' must be a complete subset of $\mathcal{C}(\varphi)$. □

Since we will have to deal with both interrogatives in $\mathcal{C}(\alpha)$ and the resolutions of $\mathcal{C}(\alpha)$ in the following completeness proof, we will have to prove the following lemma as well.

Lemma 3.41. *If $\Phi \subseteq \mathcal{RC}(\chi)$ is consistent, then there is some $\Phi' \subseteq \mathcal{C}(\chi)$ such that for all $\psi \in \mathcal{C}(\chi)$, $\Phi \vdash_{2^{|\chi|}} \psi$ iff $\Phi' \vdash_{2^{|\chi|}} \psi$.*

Proof. First, we will start by building a set $\Phi' \subseteq \mathcal{C}(\chi)$ where all of the resolutions of formulas from $\mathcal{C}(\chi)$ have been replaced by the formulas for which they were added as a resolution.

$$\Phi' = (\Phi \cap \mathcal{C}(\chi)) \cup \{\mu \mid \mu \in \mathcal{C}(\chi), \beta \in (\Phi \setminus \mathcal{C}(\chi)), \beta \in \mathcal{R}_{2^{|\chi|}}(\mu)\}$$

Since a formula might be a resolution of multiple other formulas in $\mathcal{C}(\chi)$, the set Φ' might be larger than the set Φ. However, it is still consistent. For if it was not consistent, then there must have been added some μ such that $\Phi \cup \{\mu\}$ is not consistent. But, since there was some $\beta \in \mathcal{R}_{2^{|\chi|}}(\mu)$ in Φ, $\Phi \vdash_{2^{|\chi|}} \mu$, which would mean Φ is not consistent. This is in contradiction with our earlier assumption, so Φ' must be consistent.

These two sets also share one other property, since for all $\psi \in \mathcal{C}(\chi)$, $\Phi \vdash_{2^{|\chi|}} \psi$ iff $\Phi' \vdash_{2^{|\chi|}} \psi$. The right to left direction comes from the fact that for every formula $\varphi' \in \Phi'$, either $\varphi' \in \Phi$ or there is some $\beta \in \mathcal{R}_{2^{|\chi|}}(\varphi')$ such that $\beta \in \Phi'$, so via Corollary 3.30 $\Phi \vdash_{2^{|\chi|}} \varphi'$.

The left to right direction is a bit more involved, and proceeds by induction on the structure of χ. The case for propositional atoms and $\bot$ are trivial, since then $\Phi = \Phi'$ by construction. For the inductive hypothesis, take two arbitrary formulas χ_1 and χ_2 such that:

- For all $\Phi_{\chi_1} \subseteq \mathcal{RC}(\chi_1)$ that are consistent, there is some $\Phi'_{\chi_1} \subseteq \mathcal{C}(\chi_1)$ such that for all $\psi \in \mathcal{C}(\chi_1)$, if $\Phi_{\chi_1} \vdash_{2^{|\chi_1|}} \psi$, then $\Phi'_{\chi_1} \vdash_{2^{|\chi_1|}} \psi$.
- For all $\Phi_{\chi_2} \subseteq \mathcal{RC}(\chi_2)$ that are consistent, there is some $\Phi'_{\chi_2} \subseteq \mathcal{C}(\chi_2)$ such that for all $\psi \in \mathcal{C}(\chi_2)$, if $\Phi_{\chi_2} \vdash_{2^{|\chi_2|}} \psi$, then $\Phi'_{\chi_2} \vdash_{2^{|\chi_2|}} \psi$.

$\chi_1 \wedge \chi_2$: Now $\mathcal{C}(\chi_1 \wedge \chi_2) = \{\chi_1 \wedge \chi_2, \neg(\chi_1 \wedge \chi_2)\} \cup \mathcal{C}(\chi_1) \cup \mathcal{C}(\chi_2)$. Now take some arbitrary set $\Phi_{\chi_1 \wedge \chi_2} \subseteq \mathcal{RC}(\chi_1 \wedge \chi_2)$. If $\Phi_{\chi_1 \wedge \chi_2} \vdash_{2^{|\chi|}} \psi$, for some $\psi \in \mathcal{C}(\chi_1 \wedge \chi_2)$, then either it can be proven using a subset of $\mathcal{RC}(\chi_1)$ or $\mathcal{RC}(\chi_2)$, in which case it can be proven from $\Phi'_{\chi_1 \wedge \chi_2}$ by the inductive hypothesis, or ψ is either $\chi_1 \wedge \chi_2$ or $\neg(\chi_1 \wedge \chi_2)$. In the first case, we need two subsets $\Phi_{\chi_i} \subseteq \Phi_{\chi_1 \wedge \chi_2}$ to prove $\Phi_{\chi_i} \vdash_{2^{|\chi_i|}} \chi_i$ for $i \in \{1, 2\}$, which by the inductive hypothesis means that $\Phi'_{\chi_i} \vdash_{2^{|\chi|}} \chi_i$ for $i \in \{1, 2\}$. Since $\Phi'_{\chi_i} \subseteq \Phi'_{\chi_1 \wedge \chi_2}$, $\Phi'_{\chi_1 \wedge \chi_2} \vdash_{2^{|\chi|}} \chi_1 \wedge \chi_2$. The argument for the second case is similar. Since $\Phi_{\chi_1 \wedge \chi_2}$ was chosen arbitrarily, it holds for all subsets of $\mathcal{RC}(\chi_1 \wedge \chi_2)$.

$\chi_1 \mathbin{\vee\!\!\!\vee} \chi_2$: Now $\mathcal{C}(\chi_1 \mathbin{\vee\!\!\!\vee} \chi_2) = \{\chi_1 \mathbin{\vee\!\!\!\vee} \chi_2, \neg(\chi_1 \mathbin{\vee\!\!\!\vee} \chi_2)\} \cup \mathcal{C}(\chi_1) \cup \mathcal{C}(\chi_2)$. Now take some arbitrary set $\Phi_{\chi_1 \mathbin{\vee\!\!\!\vee} \chi_2} \subseteq \mathcal{RC}(\chi_1 \mathbin{\vee\!\!\!\vee} \chi_2)$. If $\Phi_{\chi_1 \mathbin{\vee\!\!\!\vee} \chi_2} \vdash_{2^{|\chi|}} \psi$, for some $\psi \in \mathcal{C}(\chi_1 \mathbin{\vee\!\!\!\vee} \chi_2)$, then either it can be proven using a subset of $\mathcal{RC}(\chi_1)$ or $\mathcal{RC}(\chi_2)$, in which case it can be proven from $\Phi'_{\chi_1 \mathbin{\vee\!\!\!\vee} \chi_2}$ by the inductive hypothesis, or ψ is either $\chi_1 \mathbin{\vee\!\!\!\vee} \chi_2$ or $\neg(\chi_1 \mathbin{\vee\!\!\!\vee} \chi_2)$. In the first case, the proof is trivial, so we will focus on the second case. This formula is actually $(\chi_1 \mathbin{\vee\!\!\!\vee} \chi_2) \to \bot$, so we will need a proof for $\neg\chi_1$ and $\neg\chi_2$ and then use Interrogative Elimination to arrive at $\bot$, which means we can introduce the implication. Since $\neg\chi_1 \in \mathcal{C}(\chi_1)$ and $\neg\chi_2 \in \mathcal{C}(\chi_2)$, this holds by the inductive hypothesis.

$\chi_1 \to \chi_2$: Now $\mathcal{C}(\chi_1 \to \chi_2) = \{\chi_1 \to \chi_2, \neg(\chi_1 \to \chi_2)\} \cup \mathcal{C}(\chi_1) \cup \mathcal{C}(\chi_2)$. Now take some arbitrary set $\Phi_{\chi_1 \to \chi_2} \subseteq \mathcal{RC}(\chi_1 \to \chi_2)$. If $\Phi_{\chi \to \chi_2} \vdash_{2^{|\chi|}} \psi$, for some $\psi \in \mathcal{C}(\chi_1 \to \chi_2)$, then either it can be proven using a subset of $\mathcal{RC}(\chi_1)$ or $\mathcal{RC}(\chi_2)$, in which case it can be proven from $\Phi'_{\chi_1 \to \chi_2}$ by the inductive hypothesis, or ψ is either $\chi_1 \to \chi_2$ or $\neg(\chi_1 \to \chi_2)$. In the first case, $\Phi \vdash_{2^{|\chi_1 \to \chi_2|}} \chi_1 \to \chi_2$. Either $\chi_1 \to \chi_2 \in \Phi_{\chi_1 \to \chi_2}$, in which case $\Phi'_{\chi_1 \to \chi_2} \vdash_{2^{|\chi_1 \to \chi_2|}} \psi$ by construction, or there are two subsets $\Phi_{\chi_1}, \Phi_{\chi_2} \subseteq \Phi_{\chi_1 \to \chi_2}$ such that $\Phi_{\chi_1} \cup \Phi_{\chi_2} \vdash_{2^{|\chi_1 \to \chi_2|}} \chi_1 \to \chi_2$. Since for $i \in \{1, 2\}$, $\chi_i \in \mathcal{C}(\chi_i)$ and $\Phi_{\chi_i} \subseteq \mathcal{RC}(\chi_i)$, by the inductive hypothesis we get that $\Phi'_{\chi_1} \cup \Phi'_{\chi_2} \vdash_{2^{|\chi_1 \to \chi_2|}} \chi_1 \to \chi_2$, which means that $\Phi'_{\chi_1 \to \chi_2} \vdash_{2^{|\chi_1 \to \chi_2|}} \psi$. Since we chose an arbitrary $\Phi_{\chi_1 \to \chi_2}$, this holds for all subsets of $\mathcal{RC}(\chi_1 \to \chi_2)$.

$[\![\pi]\!]\chi_1$: In order to do this, we will need to use induction of the structure of π. The case for atomic programs follows from the inductive hypothesis, and the case for $[\![?\chi_2]\!]\chi_1$ follows from the step for implication.

For the inductive hypothesis, take two arbitrary programs π_1 and π_2 such that for some formulas χ':

- For all $\Phi_{[\![\pi_1]\!]\chi'} \subseteq \mathcal{RC}([\![\pi_1]\!]\chi')$ that are consistent, there is some $\Phi'_{[\![\pi_1]\!]\chi'} \subseteq \mathcal{C}([\![\pi_1]\!]\chi')$ such that for all $\psi \in \mathcal{C}([\![\pi_1]\!]\chi')$, if $\Phi_{[\![\pi_1]\!]\chi'} \vdash_{2^{|[\![\pi_1]\!]\chi'|}} \psi$, then $\Phi'_{[\![\pi_1]\!]\chi'} \vdash_{2^{|[\![\pi_1]\!]\chi'|}} \psi$.
- For all $\Phi_{[\![\pi_2]\!]\chi'} \subseteq \mathcal{RC}([\![\pi_2]\!]\chi')$ that are consistent, there is some $\Phi'_{[\![\pi_2]\!]\chi'} \subseteq \mathcal{C}([\![\pi_2]\!]\chi')$ such that for all $\psi \in \mathcal{C}([\![\pi_2]\!]\chi')$, if $\Phi_{[\![\pi_2]\!]\chi'} \vdash_{2^{|[\![\pi_2]\!]\chi'|}} \psi$, then $\Phi'_{[\![\pi_2]\!]\chi'} \vdash_{2^{|[\![\pi_2]\!]\chi'|}} \psi$.

$\pi_1; \pi_2$: Now $\mathcal{C}([\![\pi_1; \pi_2]\!]\chi') = \{[\![\pi_1; \pi_2]\!]\chi', \neg[\![\pi_1; \pi_2]\!]\chi'\} \cup \mathcal{C}([\![\pi_1]\!][\![\pi_2]\!]\chi')$. Now take some arbitrary set $\Phi_{[\![\pi_1;\pi_2]\!]\chi'} \subseteq \mathcal{RC}([\![\pi_1; \pi_2]\!]\chi')$. If $\Phi_{[\![\pi_1;\pi_2]\!]\chi'} \vdash_{2^{|\chi|}} \psi$, for some $\psi \in \mathcal{C}([\![\pi_1; \pi_2]\!]\chi')$, then either it can be proven using a subset of $\mathcal{RC}(\chi_2)$, in which case it can be proven from $\Phi'_{[\![\pi_1;\pi_2]\!]\chi'}$ by the inductive hypothesis, or ψ is either $[\![\pi_1; \pi_2]\!]\chi'$, $\neg[\![\pi_1; \pi_2]\!]\chi'$, $[\![\pi_1]\!]\psi'$ or $\neg[\![\pi_1]\!]\psi'$, for some $\psi' \in \mathcal{C}([\![\pi_2]\!]\chi')$. Since the first two are special cases of the latter two, we will focus on those. Since $\psi' \in \mathcal{C}([\![\pi_2]\!]\chi')$, $\Phi'_{[\![\pi_2]\!]\chi'} \vdash_{2^{|[\![\pi_2]\!]\chi'|}} \psi'$ by the inductive hypothesis. And since $\mathcal{C}([\![\pi_1]\!]\chi')$ contains all subformulas of π_1, $\Phi_{[\![\pi_1]\!]\chi'} \cup \Phi_{[\![\pi_2]\!]\chi'} \vdash_{2^{|[\![\pi_1;\pi_2]\!]\chi'|}} [\![\pi_1]\!]\psi'$. This, in turn, means that $\Phi_{[\![\pi_1;\pi_2]\!]\chi'} \vdash_{2^{|[\![\pi_1;\pi_2]\!]\chi'|}} [\![\pi_1]\!]\psi'$. The second case is similar. Since $\Phi_{[\![\pi_1;\pi_2]\!]\chi'}$ was chosen arbitrarily, this holds for all subsets of $\mathcal{RC}([\![\pi_1;\pi_2]\!]\chi')$.

$\pi_1 \cup \pi_2$: This step is similar to the step for conjunction. The only difference is that now there are extra steps to go from $[\![\pi_1 \cup \pi_2]\!]\chi'$ to $[\![\pi_1]\!]\chi' \wedge [\![\pi_2]\!]\chi'$ and back.

π_1^*: Now $\mathcal{C}([\![\pi_1^*]\!]\chi') = \{[\![\pi_1^*]\!]\chi', \neg[\![\pi_1^*]\!]\chi', [\![\pi]\!][\![\pi^*]\!]\chi', \neg[\![\pi]\!][\![\pi_1^*]\!]\chi'\} \cup \mathcal{C}([\![\pi]\!]\chi')$. Now take some arbitrary set $\Phi_{[\![\pi_1^*]\!]\chi'} \subseteq \mathcal{RC}([\![\pi_1^*]\!]\chi')$. Now $\Phi_{[\![\pi_1^*]\!]\chi'}$ can only prove a formula $\psi \in \mathcal{C}([\![\pi_1^*]\!]\chi')$ if that formula is already in $\Phi_{[\![\pi_1^*]\!]\chi'}$, or if it is provable via the (contraposition of) the unrolling axiom, in which case the substeps must be provable from some subset contained in $\mathcal{RC}([\![\pi]\!]\chi')$, which means that, by the inductive hypothesis, we get $\Phi'_{[\![\pi_1^*]\!]\chi'} \vdash_{2^{|[\![\pi_1^*]\!]\chi'|}} \psi$. Since $\Phi_{[\![\pi_1^*]\!]\chi'}$ was chosen arbitrarily, this holds for all subsets of $\mathcal{RC}([\![\pi_1^*]\!]\chi')$.

$[\pi]\chi'$: The proofs here are the same as the one for $[\![\pi]\!]\chi'$, but using the axioms for $[\pi]$.

Since χ_1 and χ_2 were chosen arbitrarily, this holds for all formulas χ. This concludes the right to left direction, and the proof as a whole. □

Lemma 3.42. *If $\Phi \subseteq \mathcal{RC}(\alpha)$ is consistent, then there is some $\Gamma \in RTD_f(\alpha)$ such that for all $\psi \in \mathcal{C}(\alpha)$, if $\Phi \vdash_{2^{|\alpha|}} \psi$, then $\Gamma \vdash_{2^{|\alpha|}} \psi$ for all $2^{|\alpha|}$-consistent resolution functions f.*

Proof. Take some arbitrary $2^{|\alpha|}$-consistent formula $f_{2^{|\alpha|}}$. Using Lemma 3.41, we can replace Φ by a set $\Phi' \subseteq \mathcal{C}(\alpha)$, which we can extend to a maximally consistent subset Φ'' of $\mathcal{C}(\alpha)$. Then the $\Gamma \in \mathrm{RTD}_f(\alpha)$ is simply the RTD that is the image of Φ'' under $f_{2^{|\alpha|}}$. To show that this proves all the $\psi \in \mathcal{C}(\alpha)$, assume that $\Gamma \nvdash_{2^{|\alpha|}} \psi$ for some $\psi \in \mathcal{C}(\alpha)$. This means that there is no $\beta \in \mathcal{R}_{2^{|\alpha|}}(\psi)$ such that $\Gamma \vdash_{2^{|\alpha|}} \beta$, and therefore, $\Phi'' \nvdash_{2^{|\alpha|}} \psi$. Since $\Phi' \subseteq \Phi''$, $\Phi' \nvdash_{2^{|\alpha|}} \psi$. Then by Lemma 3.41, $\Phi \nvdash_{2^{|\alpha|}} \psi$. But that contradicts our earlier assumption, so $\Gamma \vdash_{2^{|\alpha|}} \psi$ for all $\psi \in \mathcal{C}(\alpha)$ such that $\Phi \vdash_{2^{|\alpha|}} \psi$. Since we took some arbitrary $f_{2^{|\alpha|}}$, this holds for all $2^{|\alpha|}$-consistent resolution functions. □

Now that we know that for every set of relevant formulas there is an RTD that proves the same formulas of $\mathcal{C}(\alpha)$, we can define our counter-models. The definition of the model should be familiar from PDL, but it differs in three big aspects. The first of these is that the models are defined with respect to a certain formula α, and the second difference is that this is done using RTDs. The third has to do with the definition of Σ^α, since this is now a state map, and not a relation between states. For this we will borrow the definition from [2].

Definition 3.20 (Countermodel for IE-PDL)**.** The counter-model for a given declarative IE-PDL formula α is the model $M^\alpha = \langle \mathcal{W}^\alpha, V^\alpha, \mathcal{R}^\alpha_{\mathcal{A}} \rangle$, where:

- $\mathcal{W}^\alpha$ is $\mathrm{RTD}_f(\alpha)$ for some consistent resolution function $f_{2^{|\alpha|}}$.
- $V^\alpha(\Gamma) = \{p \mid p \in \mathcal{C}(\alpha), p \in \Gamma\}$
- $T \in \Sigma^\alpha_a(\Gamma)$ iff $\bigcap T \vdash_{2^{|\alpha|}} \varphi$ whenever $[\![a]\!]\varphi \in \Gamma$, where $\bigcap \emptyset = \mathcal{L}_!^{\text{IE-PDL}}$

Now we will still need to show that M^α is a proper IE-PDL model by showing that $\Sigma^\alpha_a(\Gamma)$ is downward closed and non-empty for every a and Γ.

Lemma 3.43. *The model M^α is a proper IE-PDL model.*

Proof. In order to show this, we have to show that:

1. $\Sigma_a^\alpha(\Gamma)$ is downward closed for all agents a and worlds Γ;
2. $\emptyset \in \Sigma_a^\alpha(\Gamma)$ for all agents a and worlds Γ.

Take some arbitrary agent a and world $\Gamma \in \mathcal{W}^\alpha$.

1. Assume that $T \in \Sigma_a^\alpha(\Gamma)$. We now have to show that for all $T' \subseteq T$, $T' \in \Sigma_a^\alpha(\Gamma)$. Since $\bigcap T \subseteq \bigcap T'$, and we know that $\bigcap T \vdash_{2^{|\alpha|}} \varphi$ for any $[\![a]\!]\varphi \in \Gamma$, we also have $\bigcap T' \vdash_{2^{|\alpha|}} \varphi$ for any $[\![a]\!]\varphi \in \Gamma$ and therefore also $T' \in \Sigma_a^\alpha(\Gamma)$.
2. Since $\bigcap \emptyset$ is defined as $\mathcal{L}^{\mathsf{IE\text{-}PDL}}$, $\bigcap \emptyset \vdash_{2^{|\alpha|}} \varphi$ for any formula φ, so also every formula φ such that $[\![a]\!]\varphi \in \Gamma$. Therefore, $\emptyset \in \Sigma_a^\alpha(\Gamma)$.

Since a and Γ were chosen arbitrarily, we can conclude that this holds for any agent a and world $\Gamma \in \mathcal{W}^\alpha$. From this, we can conclude that M^α is a proper IE-PDL model. □

Since the counter-model only gives the state map, we will also want to know that the complex programs for the counter-model are well behaved.

Lemma 3.44. *If $SR_\pi^\alpha T$ and $\bigcap S \vdash_{2^{|\alpha|}} [\![\pi]\!]\varphi$, then $\bigcap T \vdash_{2^{|\alpha|}} \varphi$.*

Proof. This proof goes by induction over the structure of π.

a: Suppose $\bigcap S \vdash_{2^{|\alpha|}} [\![a]\!]\varphi$. Then consider any $\Gamma \in S$. Since $\bigcap S \subseteq \Gamma$, we have $\Gamma \vdash_{2^{|\alpha|}} [\![a]\!]\varphi$. Then by Definition 3.20 we have that $SR_a^\alpha T$ if and only if $\bigcap T \vdash_{2^{|\alpha|}} \varphi$, as required.

$?\psi$: Suppose $\bigcap S \vdash_{2^{|\alpha|}} [\![?\psi]\!]\varphi$, which means we also have $\bigcap S \vdash_{2^{|\alpha|}} \psi \to \varphi$. By Definition 3.3 $SR_{?\psi}^\alpha T$ if $T \subseteq S$ and $M^\alpha, T \models \psi$. Since $T \subseteq S$, $\bigcap S \subseteq \bigcap T$, and therefore $\bigcap T \vdash_{2^{|\alpha|}} \psi \to \varphi$. Combining this, we get that $\bigcap T \vdash_{2^{|\alpha|}} \varphi$, as required.

For the inductive hypothesis, take two arbitrary programs π_1 and π_2 and suppose that for all states S and formulas φ:

- if $SR_{\pi_1}^\alpha T$ and $\bigcap S \vdash_{2^{|\alpha|}} [\![\pi_1]\!]\varphi$, then $\bigcap T \vdash_{2^{|\alpha|}} \varphi$;
- if $SR_{\pi_2}^\alpha T$ and $\bigcap S \vdash_{2^{|\alpha|}} [\![\pi_2]\!]\varphi$, then $\bigcap T \vdash_{2^{|\alpha|}} \varphi$.

$\pi_1;\pi_2$: Suppose $\bigcap S \vdash_{2^{|\alpha|}} [\![\pi_1;\pi_2]\!]\varphi$. Then we also have that $\bigcap S \vdash_{2^{|\alpha|}} [\![\pi_1]\!][\![\pi_2]\!]\varphi$. Now by applying the inductive hypothesis once we get that if $SR_{\pi_1}^\alpha U$, then $\bigcap U \vdash_{2^{|\alpha|}} [\![\pi_2]\!]\varphi$. By applying the inductive hypothesis on this formula we get that if $UR_{\pi_2}^\alpha T$, then $\bigcap T \vdash_{2^{|\alpha|}} \varphi$. So all we have left is to show that $SR_{\pi_1;\pi_2}^\alpha T$. However, this holds by Definition 3.3, so we have the required result.

$\pi_1 \cup \pi_2$: Suppose $\bigcap S \vdash_{2^{|\alpha|}} [\![\pi_1 \cup \pi_2]\!]\varphi$. Then we also have $\bigcap S \vdash_{2^{|\alpha|}} [\![\pi_1]\!]\varphi \wedge [\![\pi_2]\!]\varphi$ and thus $\bigcap S \vdash_{2^{|\alpha|}} [\![\pi_1]\!]\varphi$ and $\bigcap S \vdash_{2^{|\alpha|}} [\![\pi_2]\!]\varphi$. By the inductive hypothesis we therefore get that if $SR_{\pi_1}^\alpha U$, then $\bigcap U \vdash_{2^{|\alpha|}} \varphi$ and if $SR_{\pi_2}^\alpha U'$, then $\bigcap U' \vdash_{2^{|\alpha|}} \varphi$. So all we have left to show is that $SR_{\pi_1 \cup \pi_2}^\alpha T$ if $SR_{\pi_1}^\alpha T$ or $SR_{\pi_2}^\alpha T$, but this is true by Definition 3.3, so we have the required result.

π_1^*: Suppose $\bigcap S \vdash_{2^{|\alpha|}} [\![\pi_1^*]\!]\varphi$. Then we also get that $\bigcap S \vdash_{2^{|\alpha|}} \bigwedge_{1 \leq m \leq 2^{|\alpha|}} [\![\pi_1^n]\!]\varphi$. This means that by repeated applications of the inductive hypothesis and the steps for sequence we get that if $SR_{\pi_1^*}^\alpha T$, then $\bigcap T \vdash_{2^{|\alpha|}} \varphi$, as required.

This concludes the proof. □

Lemma 3.45. *If $\bigcap S \vdash_{2^{|\alpha|}} [\pi]\varphi$, then for all $\Gamma \in S$, $\bigcap \mathrm{R}^{\alpha}_{\pi}(\Gamma) \vdash_{2^{|\alpha|}} \varphi$.*

Proof. This lemma is analogous to Lemma 3.44. $\square$

Now we have the counter-model we can continue onward to the finite support lemma. Before we can establish this lemma we will need some extra results.

Lemma 3.46. *For any state $S \subseteq \mathcal{W}^{\alpha}$ and any $\beta \in \bigcup RTD_f(\alpha)$, $\bigcap S \vdash_{2^{|\alpha|}} \beta \iff \beta \in \bigcap S$.*

Proof. If $\beta \in \bigcap S$ then also $\bigcap S \vdash_{2^{|\alpha|}} \beta$. For the other direction, suppose that $\bigcap S \vdash_{2^{|\alpha|}} \beta$. Then, for each $\Gamma \in S$, since $\bigcap S \subseteq \Gamma$, $\Gamma \vdash_{2^{|\alpha|}} \beta$. Then, since Γ is complete, $\beta \in \Gamma$. Therefore, $\beta \in \bigcap S$. $\square$

We will also have to show that whenever $[\![\pi]\!]\varphi \notin \Gamma$, there is some state $\{\Gamma\}R^{\alpha}_{\pi}T$ such that $\bigcap T \nvdash_{2^{|\alpha|}} \varphi$.

Lemma 3.47. *Let $\Gamma \in \mathcal{W}^{\alpha}$ and let $[\![\pi]\!]\varphi \in \mathcal{C}(\alpha)$ be a declarative formula. If $[\![\pi]\!]\varphi \notin \Gamma$, then there exists a state $\{\Gamma\}R^{\alpha}_{\pi}T$ such that $\bigcap T \nvdash_{2^{|\alpha|}} \varphi$.*

Proof. Suppose $[\![\pi]\!]\varphi \notin \Gamma$. Now create $\Gamma^{[\![\pi]\!]} = \{\psi \mid [\![\pi]\!]\psi \in \Gamma\}$. Note that not all formulas in $\Gamma^{[\![\pi]\!]}$ are declaratives, since $[\![\pi]\!]\psi$ can be declarative even if ψ is not.

We claim that $\Gamma^{[\![\pi]\!]} \nvdash_{2^{|\alpha|}} \varphi$. This proof works by contradiction. Suppose that $\Gamma^{[\![\pi]\!]} \vdash_{2^{|\alpha|}} \varphi$. Let $\psi_1, \dots, \psi_n \in \Gamma^{[\![\pi]\!]}$ be assumptions such that $\psi_1, \dots, \psi_n \vdash_{2^{|\alpha|}} \varphi$. By Lemma 3.31, we then have $[\![\pi]\!]\psi_1, \dots, [\![\pi]\!]\psi_n \vdash_{2^{|\alpha|}} [\![\pi]\!]\varphi$. Since $\psi_1, \dots \psi_n \in \Gamma^{[\![\pi]\!]}$, we have that $[\![\pi]\!]\psi_1, \dots, [\![\pi]\!]\psi_n \in \Gamma$, and therefore, $\Gamma \vdash_{2^{|\alpha|}} [\![\pi]\!]\varphi$. Since Γ is complete and $[\![\pi]\!]\varphi \notin \Gamma$, $\neg[\![\pi]\!]\varphi \in \Gamma$. But this leads to a contradiction. Therefore $\Gamma^{[\![\pi]\!]} \nvdash_{2^{|\alpha|}} \varphi$.

Since $\Gamma^{[\![\pi]\!]} \nvdash_{2^{|\alpha|}} \varphi$, by Theorem 3.35 we have that there is some $\Theta \in \mathcal{R}(\Gamma^{[\![\pi]\!]})$ which derives no resolution of φ. Now take any $\beta \in \mathcal{R}(\varphi)$. Since $\Theta \nvdash_{2^{|\alpha|}} \beta$, the set $\Theta \cup \{\neg\beta\}$ must be consistent. Otherwise, we would have had $\Theta \vdash_{2^{|\alpha|}} \beta$ since we would have had that $\Theta, \neg\beta \vdash_{2^{|\alpha|}} \bot$, which by the rules of negation leads to $\Theta \vdash_{2^{|\alpha|}} \neg\neg\beta$ and then we can apply double negation elimination since β is declarative. This means that $\Theta \cup \{\neg\beta\}$ is consistent, and since it is build up of relevant formulas for α, by Lemma 3.42 we can extend it to some $\Delta_\beta \in \mathrm{RTD}_f(\alpha)$.

Now consider the state $T = \{\Delta_\beta \mid \beta \in \mathcal{R}_{2^{|\alpha|}}(\varphi)\}$. We claim that $\{\Gamma\}R^{\alpha}_{a}T$ and that $\bigcap T \nvdash_{2^{|\alpha|}} \varphi$. We will first prove the first claim.

For this, we will take some $[\![\pi]\!]\psi \in \Gamma$, which means that $\psi \in \Gamma^{[\![\pi]\!]}$. Since Θ is a resolution of $\Gamma^{[\![\pi]\!]}$, it contains some resolution γ of ψ. This means that every $\Delta_\beta \vdash_{2^{|\alpha|}} \psi$ as well, and thus, in turn, there is some resolution $\gamma' \in \mathcal{R}_{2^{|\alpha|}}(\psi)$ in each Δ_β. Since Δ_β is an RTD of α, we can conclude that $f_{2^{|\alpha|}}(\psi) \in \Delta_\beta$ for each $\beta \in \mathcal{R}_{2^{|\alpha|}}(\psi)$. Therefore, $f_{2^{|\alpha|}}(\psi) \in \bigcap T$, and thus, $\bigcap T \vdash_{2^{|\alpha|}} \psi$. Therefore, $\bigcap T \vdash_{2^{|\alpha|}} \psi$ whenever $[\![\pi]\!]\psi \in \Gamma$, which means that $\{\Gamma\}R^{\alpha}_{\pi}T$.

Then we still have to prove that $\bigcap T \nvdash_{2^{|\alpha|}} \varphi$. Towards a contradiction, suppose that $\bigcap T \vdash_{2^{|\alpha|}} \varphi$. Since $\bigcap T$ is a set of declaratives, by Corollary 3.36 we have that $\bigcap T \vdash_{2^{|\alpha|}} \beta$ for some $\beta \in \mathcal{R}_{2^{|\alpha|}}(\varphi)$. Since $\Delta_\beta \in T$, we have that $\bigcap T \subseteq \Delta_\beta$, so we would also have $\Delta_\beta \vdash_{2^{|\alpha|}} \beta$. However, this leads to a contradiction, since by construction $\neg\beta \in \Delta_\beta$ and Δ_β is consistent. Therefore, $\bigcap T \nvdash_{2^{|\alpha|}} \varphi$, which proofs the lemma. $\square$

However, unlike in IEL, this will not be enough to prove the finite support lemma for $[\![\pi]\!]\varphi$. Normally, one would suppose that $\bigcap S \nvdash_{2^{|\alpha|}} [\![\pi]\!]\varphi$, and therefore conclude that there must be a $\Gamma \in S$ such that $[\![\pi]\!]\varphi \notin \Gamma$. However, this does not work for IE-PDL, since $[\![\pi]\!]\varphi$ is not necessarily declarative. Therefore, we will need to prove an additional result.

Lemma 3.48. *For all formulas $\varphi \in \mathcal{C}(\alpha)$, if we have that $\bigcap T \nvdash_{2^{|\alpha|}} \varphi$ implies $M^{\alpha}, T \not\models \varphi$ for all states T, then for each S, $\bigcap S \nvdash_{2^{|\alpha|}} [\![\pi]\!]\varphi$ implies $M^{\alpha}, S \not\models [\![\pi]\!]\varphi$.*

Proof. This proof works by induction over π. We will start by assuming that for all states T, $\bigcap T \nvdash_{2^{|\alpha|}} \varphi$ implies $M^\alpha, T \not\models \varphi$.

a: Suppose $\bigcap S \nvdash_{2^{|\alpha|}} [\![a]\!]\varphi$. Since $[\![a]\!]\varphi$ is declarative, this means that there is some $\Gamma \in S$ such that $[\![a]\!]\pi \notin \Gamma$. Then by Lemma 3.47 we get that there exists a state $\{\Gamma\}R^\alpha_a T$ such that $\bigcap T \nvdash_{2^{|\alpha|}} \varphi$. By the assumption we then get that $M^\alpha, T \not\models \varphi$. Then by Definition 3.3, we get that $SR^\alpha_a T$. Therefore we get that $M^\alpha, S \not\models_{2^{|\alpha|}} [\![a]\!]\varphi$.

$?\psi$: Suppose $\bigcap S \nvdash_{2^{|\alpha|}} [\![?\psi]\!]\varphi$. Then we also get that $\bigcap S \nvdash_{2^{|\alpha|}} \psi \to \varphi$. Now by the step for implication we get that $M^\alpha, S \not\models \psi \to \varphi$, which means that we get $M^\alpha, S \not\models [\![?\psi]\!]\varphi$ by Lemma 3.18, as required.

For the inductive hypothesis, take some arbitrary programs π_1 and π_2 and assume that for all states S and formulas φ:

- if $\bigcap S \nvdash_{2^{|\alpha|}} [\![\pi_1]\!]\varphi$ then $M^\alpha, S \not\models [\![\pi_1]\!]\varphi$.
- if $\bigcap S \nvdash_{2^{|\alpha|}} [\![\pi_2]\!]\varphi$ then $M^\alpha, S \not\models [\![\pi_2]\!]\varphi$.

$\pi_1;\pi_2$: Suppose $\bigcap S \nvdash_{2^{|\alpha|}} [\![\pi_1;\pi_2]\!]\varphi$. Then we also get that $\bigcap S \nvdash_{2^{|\alpha|}} [\![\pi_1]\!][\![\pi_2]\!]\varphi$, and by the assumption $M^\alpha, S \not\models [\![\pi_1]\!][\![\pi_2]\!]\varphi$. By Lemma 3.16 we then get $M^\alpha, S \not\models [\![\pi_1;\pi_2]\!]\varphi$.

$\pi_1 \cup \pi_2$: Suppose $\bigcap S \nvdash_{2^{|\alpha|}} [\![\pi_1 \cup \pi_2]\!]\varphi$. Then we also get that $\bigcap S \nvdash_{2^{|\alpha|}} [\![\pi_1]\!]\varphi \wedge [\![\pi_2]\!]\varphi$, which means that either $\bigcap S \nvdash_{2^{|\alpha|}} [\![\pi_1]\!]\varphi$ or $\bigcap S \nvdash_{2^{|\alpha|}} [\![\pi_2]\!]\varphi$. By the inductive hypothesis we get that $M^\alpha, S \not\models [\![\pi_1]\!]\varphi$ or $M^\alpha, S \not\models [\![\pi_2]\!]\varphi$, which means that there must be some state T such that $SR^\alpha_{\pi_1} T$ or $SR^\alpha_{\pi_2} T$ where $M^\alpha, T \not\models \varphi$. This means that $SR^\alpha_{\pi_1 \cup \pi_2} T$ as well, and therefore $M^\alpha, S \not\models [\![\pi_1 \cup \pi_2]\!]\varphi$.

π_1^*: Suppose $\bigcap S \nvdash_{2^{|\alpha|}} [\![\pi_1^*]\!]\varphi$. Then we also get that $\bigcap S \nvdash_{2^{|\alpha|}} \bigwedge_{1 \le m \le 2^{|\alpha|}} [\![\pi_1^m]\!]\varphi$, and thus $\bigcap S \nvdash_{2^{|\alpha|}} [\![\pi_1^m]\!]\varphi$ for some $1 \le m \le 2^{|\alpha|}$. By the steps for sequence and the inductive hypothesis, we then get that $M^\alpha, S \not\models [\![\pi^m]\!]\varphi$ for that $1 \le m \le 2^{|\alpha|}$. By Definition 3.4, this means that $M^\alpha, T \not\models \varphi$ for some $SR^\alpha_{\pi_1^m} T$. By Definition 3.3 we get that $SR^\alpha_{\pi_1^*} T$. Therefore, $M^\alpha, S \not\models [\![\pi^*]\!]\varphi$.

This proves the lemma. □

Since $[\pi]\varphi$ is always a declarative formula, we can use the same method as in IEL here, which means we will only need to establish an analogue to Lemma 3.47.

Lemma 3.49. *Let $\Gamma \in \mathcal{W}^\alpha$. For all formulas $[\pi]\varphi \in \mathcal{C}(\alpha)$, if $[\pi]\varphi \notin \Gamma$, then $\bigcap \mathrm{R}^\alpha_\pi(\Gamma) \nvdash_{2^{|\alpha|}} \varphi$.*

Proof. Suppose that $[\pi]\varphi \notin \Gamma$, and let $\mathcal{R}(\varphi) = \{\beta_1, \dots, \beta_n\}$. By Lemma 3.32 we know that $[\pi]\varphi \dashv\vdash [\![\pi]\!]\beta_1 \vee \cdots \vee [\![\pi]\!]\beta_n$, which means that we also have $[\![\pi]\!]\beta_1 \vee \cdots \vee [\![\pi]\!]\beta_n \notin \Gamma$. Since Γ is consistent, it can therefore not contain any of $[\![\pi]\!]\beta_1, \dots, [\![\pi]\!]\beta_n$.

Because of this, by Lemma 3.47, there exists a state T_i such that $\{\Gamma\}R^\alpha_\pi T_i$ and $\bigcap T_i \nvdash_{2^{|\alpha|}} \beta_i$. Now we have that $T_i \subseteq \mathrm{R}^\alpha_\pi(\Gamma)$, and therefore $\bigcap \mathrm{R}^\alpha_\pi(\Gamma) \subseteq \bigcap T_i$. Since $\bigcap T_i \nvdash_{2^{|\alpha|}} \beta_i$, we also have that $\bigcap \mathrm{R}^\alpha_\pi(\Gamma) \nvdash_{2^{|\alpha|}} \beta_i$. Since $\bigcap \mathrm{R}^\alpha_\pi(\Gamma)$ is a set of declaratives and does not derive any resolution of φ, by Corollary 3.36 we get that $\bigcap \mathrm{R}^\alpha_\pi(\Gamma) \nvdash_{2^{|\alpha|}} \varphi$. □

Now we have all the ingredients to go and prove the support lemma.

Lemma 3.50 (Finite Support Lemma)**.** *For any $S \subseteq \mathcal{W}^\alpha$ and any $\varphi \in \mathcal{C}(\alpha)$, $M^\alpha, S \models \varphi \iff \bigcap S \vdash_{2^{|\alpha|}} \varphi$.*

Proof. This proof goes by induction on the structure of formulas φ. The cases for atoms and falsum are trivial, so we will only show the proofs for the other connectives.

For the inductive hypothesis we will take two arbitrary formulas $\varphi_1, \varphi_2 \in \mathcal{C}(\alpha)$ such that for all S:

- $\bigcap S \vdash_{2^{|\alpha|}} \varphi_1$ iff $M^\alpha, S \models \varphi_1$
- $\bigcap S \vdash_{2^{|\alpha|}} \varphi_2$ iff $M^\alpha, S \models \varphi_2$

conjunction: $\bigcap S \vdash_{2^{|\alpha|}} \varphi_1 \wedge \varphi_2$ is the case iff $\bigcap S \vdash_{2^{|\alpha|}} \varphi_1$ and $\bigcap S \vdash_{2^{|\alpha|}} \varphi_2$. Then by the inductive hypothesis we get that this holds iff $M^\alpha, S \models \varphi_1$ and $M^\alpha, S \models \varphi_2$. Then the support condition for conjunction tell us that this is the case iff $M^\alpha, S \models \varphi_1 \wedge \varphi_2$, as required.

implication: Suppose $\bigcap S \vdash_{2^{|\alpha|}} \varphi_1 \to \varphi_2$. Now consider any state $T \subseteq S$ such that $M^\alpha, T \models \varphi_1$. By the inductive hypothesis, we get that $\bigcap T \vdash_{2^{|\alpha|}} \varphi_1$, and since $T \subseteq S$, $\bigcap S \subseteq \bigcap T$, so $\bigcap T \vdash_{2^{|\alpha|}} \varphi_1 \to \varphi_2$. Therefore, we get that $\bigcap T \vdash_{2^{|\alpha|}} \varphi_2$, and thus by the inductive hypothesis $M^\alpha, T \models \varphi_2$. And since this works for any substate T with $M^\alpha, T \models \varphi_1$, we get that $M^\alpha, S \models \varphi_1 \to \varphi_2$.

For the other direction, suppose that $\bigcap S \nvdash_{2^{|\alpha|}} \varphi_1 \to \varphi_2$. This implies that $\bigcap S, \varphi_1 \nvdash_{2^{|\alpha|}} \varphi_2$, and thus, by Lemma 3.33, $\bigcap S, \beta \nvdash_{2^{|\alpha|}} \varphi_2$ for some $\beta \in \mathcal{R}_{2^{|\alpha|}}(\varphi_1)$. Then by Lemma 3.42 there is also some resolution $\beta' \in \bigcup \mathrm{RTD}_f(\alpha)$ such that $\beta' \in \mathcal{R}_{2^{|\alpha|}}(\varphi_1)$ and $\bigcap S, \beta' \nvdash_{2^{|\alpha|}} \varphi_2$. Now let $T_{\beta'} = \{\Gamma \in S \mid \beta' \in \Gamma\}$. By definition we have $\beta' \in \bigcap T_{\beta'}$, and thus $\bigcap T_{\beta'} \vdash_{2^{|\alpha|}} \varphi_1$ by Corollary 3.30. All is now left to show that $\bigcap T_{\beta'} \nvdash_{2^{|\alpha|}} \varphi_2$.

Assume that $\bigcap T_{\beta'} \vdash_{2^{|\alpha|}} \varphi_2$. Then by Corollary 3.36, $\bigcap T_{\beta'} \vdash_{2^{|\alpha|}} \gamma$ for some $\gamma \in \mathcal{R}_{2^{|\alpha|}}(\varphi_2)$. This means that by Lemma 3.42 there is also a $\gamma' \in \mathcal{R}_{2^{|\alpha|}}(\varphi_2)$ such that $\gamma' \in \bigcup \mathrm{RTD}_f(\alpha)$ and $\bigcap T_{\beta'}, \gamma' \vdash_{2^{|\alpha|}} \varphi_2$. And since $T_{\beta'} \subseteq \mathrm{RTD}_f(\alpha)$, $\gamma' \in \bigcap T_{\beta'}$. Thus for any $\Gamma \in T_{\beta'}$, $\gamma' \in \Gamma$. As a consequence of this, $\Gamma \vdash_{2^{|\alpha|}} \beta' \to \gamma'$ and by Lemma 3.46 thus also $\beta' \to \gamma' \in \Gamma$.

Now consider any $\Gamma \in (S \setminus T_{\beta'})$. Then $\beta' \notin \Gamma$, so $\neg\beta' \in \Gamma$. The we get $\Gamma \vdash_{2^{|\alpha|}} \beta' \to \gamma'$ and thus $\beta' \to \gamma' \in \Gamma$. Therefore $\beta' \to \gamma' \in \Gamma$ for all $\Gamma \in S$. We can now conclude $\beta' \to \gamma' \in \bigcap S$, and thus $\bigcap S, \beta' \vdash_{2^{|\alpha|}} \gamma'$ and $\bigcap S, \beta' \vdash_{2^{|\alpha|}} \varphi_2$. But this contradicts our earlier assumption of $\bigcap S, \beta' \nvdash_{2^{|\alpha|}} \varphi_2$, thus $\bigcap T_{\beta'} \nvdash_{2^{|\alpha|}} \varphi_2$. By the inductive hypothesis, this gives us that $M^\alpha, T_{\beta'} \not\models \varphi_2$. And since $\bigcap T_{\beta'} \vdash_{2^{|\alpha|}} \varphi_1$, by the inductive hypothesis $M^\alpha, T_{\beta'} \models \varphi_1$. Then, by the support conditions for implication we get $M^\alpha, S \not\models \varphi_1 \to \varphi_2$, as required.

inquisitive disjunction: Suppose $\bigcap S \vdash_{2^{|\alpha|}} \varphi_1 \vee\!\!\!\vee \varphi_2$. Because $\bigcap S$ is a set of declaratives, by Corollary 3.36 we get that $\bigcap S \vdash \beta$ for some $\beta \in \mathcal{R}_{2^{|\alpha|}}(\varphi_1 \vee\!\!\!\vee \varphi_2)$. By the definition of the resolutions, this β is either a resolution of φ_1 or φ_2, and therefore, $\bigcap S \vdash_{2^{|\alpha|}} \varphi_1$ or $\bigcap S \vdash_{2^{|\alpha|}} \varphi_2$, and thus $M^\alpha, S \models \varphi_1$ or $M^\alpha, S \models \varphi_2$ by the inductive hypothesis. In both cases, $M^\alpha, S \models \varphi_1 \vee\!\!\!\vee \varphi_2$.

For the other direction, suppose $M^\alpha, S \models \varphi_1 \vee\!\!\!\vee \varphi_2$. Then by the support conditions for inquisitive disjunction we get that $M^\alpha, S \models \varphi_1$ or $M^\alpha, S \models \varphi_2$. By the inductive hypothesis, this means that either $\bigcap S \vdash_{2^{|\alpha|}} \varphi_1$ or $\bigcap S \vdash_{2^{|\alpha|}} \varphi_2$. In both cases $\bigcap S \vdash_{2^{|\alpha|}} \varphi_1 \vee\!\!\!\vee \varphi_2$, as required.

$[\![\pi]\!]$ modality: Suppose $\bigcap S \vdash_{2^{|\alpha|}} [\![\pi]\!]\varphi$. Then by Lemma 3.44 we have that for all $SR^\alpha_\pi T$, $\bigcap T \vdash_{2^{|\alpha|}} \varphi$, which by the inductive hypothesis entails that $M^\alpha, T \models \varphi$. By the support condition for $[\![\pi]\!]\varphi$, this means that $M^\alpha, S \models [\![\pi]\!]\varphi$.

For the other direction, we make use of Lemma 3.48. By the inductive hypothesis we have that $\bigcap T \nvdash_{2^{|\alpha|}} \varphi$ implies $M^\alpha, T \not\models \varphi$ for all states T, so we can simply invoke the lemma, which then gives us that $\bigcap S \nvdash_{2^{|\alpha|}} [\![\pi]\!]\varphi$ implies $M^\alpha, S \not\models [\![\pi]\!]\varphi$. After an application of contrapostion we then get $M^\alpha, S \models [\![\pi]\!]\varphi$ implies $\bigcap S \vdash_{2^{|\alpha|}} [\![\pi]\!]\varphi$, as required.

$[\pi]$ **modality:** Suppose $\bigcap S \vdash_{2^{|\alpha|}} [\pi]\varphi$, which means that by Lemma 3.45 we get that for all $\Gamma \in S$, $\bigcap \mathrm{R}^\alpha_\pi(\Gamma) \vdash_{2^{|\alpha|}} \varphi$. Then by the inductive hypothesis we get that for all $\Gamma \in S$, $M^\alpha, \mathrm{R}^\alpha_\pi(\Gamma) \models \varphi$, which, by the support condition of $[\pi]\varphi$ gives us $M^\alpha, S \models [\varphi]$, as required.

For the other direction, suppose $\bigcap S \nvdash_{2^{|\alpha|}} [\pi]\varphi$. Then there is a $\Gamma \in S$ such that $[\pi]\varphi \notin \Gamma$, since $[\pi]\varphi$ is declarative and Γ is complete with respect to $\mathcal{C}(\alpha)$. Then, by Lemma 3.49 then we get that $\bigcap \mathrm{R}^\alpha_\pi(\Gamma) \nvdash_{2^{|\alpha|}} \varphi$. By the inductive hypothesis this leads to $M^\alpha, \mathrm{R}^\alpha_\pi(\Gamma) \not\models \varphi$, which in turn means that $M^\alpha, S \not\models [\pi]\varphi$. By contraposition, we now get that $M^\alpha, S \models [\pi]\varphi$ if and only if $\bigcap S \vdash_{2^{|\alpha|}} [\pi]\varphi$, as required.

This completes the lemma. □

With the finite support lemma in place, we can now prove the Completeness theorem. The following proof is adapted from [2] and [9]. Since this proof will heavily use the resolutions of φ, here we run into the problem that we will also have to deal with formulas that might not be in the closure of some resolution α of φ, which adds some extra complexity to this proof.

Theorem 3.51 (Completeness theorem). *If for some $n \in \mathbb{N}$, $\nvdash_n \varphi$, then there is some $m \in \mathbb{N}$ such that $\nvdash_m \varphi$ and there is a model M and a state s such that $M, s \not\models \varphi$, where $|\mathcal{W}| \leq m$.*

Proof. Suppose $\nvdash_n \psi$ for some arbitrary n. Now we have to find an $m \in \mathbb{N}$ such that $\nvdash_m \psi$ and construct a model M with a state s such that $M, s \not\models \psi$. But before we do that, we will first look at the resolutions of ψ.

By Theorem 3.35, we know that $\nvdash_n \alpha$ for any $\alpha \in \mathcal{R}_n(\psi)$. Now pick the longest $\alpha \in \mathcal{R}_n(\psi)$. We now claim that we can take the m to be $2^{|\alpha|}$ or n and the model M to be M^α as defined in Definition 3.20. For the first we will have to consider two sub-cases, either ψ is π^* free, or it is not.

In the first case, $\not\models_{2^{|\alpha|}} \psi$ and $\nvdash_{2^{|\alpha|}} \psi$ by the contraposition of Proposition 3.28. In the second case, we yet again have to consider two sub-cases, either the π^* in ψ is declarative, or it is not. In the latter, we can use the contraposition of Proposition 3.27 and the fact that for formulas with a non-declarative π^*, the length of the resolutions will at least be 2^n to get $\not\models_{2^{|\alpha|}} \psi$ and $\nvdash_{2^{|\alpha|}} \psi$. In the former, either $n \leq 2^{|\alpha|}$, in which case we can apply the same reasoning as before, or $2^{|\alpha|} \leq n$, in which case we can take $n = m$.

Now we have found an m, we only have to build a model for which $|\mathcal{W}| \leq m$. The model M^α as defined in Definition 3.20 has this property. Left to show is that there is a state s such that $M^\alpha, s \not\models \psi$. Since $\neg\alpha \in \mathcal{C}(\alpha)$ and $\nvdash_{2^{|\alpha|}} \alpha$, the set $\{\neg\alpha\}$ is consistent, and a subset of $\mathcal{C}(\alpha)$, so it can be extended to a $\Gamma_\alpha \in \mathrm{RTD}_f(\alpha)$. Since $\mathcal{C}(\alpha)$ could contain other resolutions of φ, we will have to repeat this procedure for every $\beta \in (\mathcal{C}(\alpha) \cap \mathcal{R}_{2^{|\alpha|}}(\psi))$.

Now we can take the state $S = \{\Gamma_\beta \mid \beta \in (\mathcal{C}(\alpha) \cap \mathcal{R}_{2^{|\alpha|}}(\varphi))\}$. We claim that $M^\alpha, S \not\models \psi$. In order to show this, assume that $M^\alpha, S \models \psi$. Then by Proposition 3.6 there must be some $\beta'_i \in \mathcal{R}_{2^{|\alpha|}}(\psi)$ such that $M^\alpha, S \models \beta'_i$. Now there are two cases:

- $\beta'_i \notin \mathcal{C}(\alpha)$
- $\beta'_i \in \mathcal{C}(\alpha)$

We will start with the second one. This would mean that $\bigcap S \vdash_{2^{|\alpha|}} \beta'_i$, and thus that $\beta'_i \in \Gamma_{\beta'_i}$, since $\Gamma_{\beta'_i} \subseteq \bigcap S$. But this is impossible, since $\Gamma_{\beta'_i}$ is consistent and contains $\neg\beta'_i$ by construction. Thus $M^\alpha, S \not\models \beta'_i$.

The proof for the first case goes by induction on the structure of ψ. The cases where ψ is declarative are trivial, for if ψ is declarative, $\psi = \alpha = \beta'_i$, so $\beta'_i \in \mathcal{C}(\alpha)$ by the definition of $\mathcal{C}(\alpha)$. In particular, this proves the base cases for ψ is a propositional atom and for $\psi = \bot$.

For the inductive hypothesis, take two arbitrary formulas χ_1 and χ_2 such that:

- for all $\gamma_1 \in \mathcal{R}_{2^{|\alpha|}}(\chi_1)$, $M^\alpha, S \not\models \gamma_1$;
- for all $\gamma_2 \in \mathcal{R}_{2^{|\alpha|}}(\chi_2)$, $M^\alpha, S \not\models \gamma_2$.

$\chi_1 \wedge \chi_2$: For any $\gamma_1 \wedge \gamma_2 \in \mathcal{R}_{2^{|\alpha|}}(\chi_1 \wedge \chi_2)$ not in $\mathcal{C}(\alpha)$, either $\gamma_1 \notin \mathcal{C}(\alpha)$ or $\gamma_2 \notin \mathcal{C}(\alpha)$. So by the inductive hypothesis, $M^\alpha, T \not\models \gamma_1 \wedge \gamma_2$.

$\chi_1 \mathbin{\vee\!\!\!\vee} \chi_2$: This one follows directly from the inductive hypothesis.

$\chi_1 \to \chi_2$: If $\psi = \chi_1 \to \chi_2$, then every $\Gamma \in S$ is based upon the negation of a conjunction of implications. That means that there must be some $\gamma_1 \in (\mathcal{R}_{2^{|\alpha|}}(\chi_1) \cap \mathcal{C}(\alpha))$ such that $M^\alpha, \Gamma \models \gamma_1$ but not $M^\alpha, \Gamma \models \chi_2$ for some $\chi_2 \in (\mathcal{R}_{2^{|\alpha|}}(\chi_2) \cap \mathcal{C}(\alpha))$ for each $\Gamma \in S$. This means that any resolution of $\chi_1 \to \chi_2$ is going to run into some world $\Gamma \in S$ where on of the implications in its conjunction will be false, making the resolution false.

$[\pi]\chi_1$: This formula is declarative, thus this is trivial.

$[\![\pi]\!]\chi_1$: This case goes by induction on the structure of programs. The case for atomic programs is trivial, since $[\![a]\!]\varphi$ is declarative for all φ, and the other cases will all use the step for implication, since π cannot be a declarative program, since then $[\![\pi]\!]\chi_1$ is declarative as well. Therefore, all the resolutions for $[\![\pi]\!]\chi_1$ that need to be considered are conjunctions of implications, which go by the step for implication.

This ends the proof for the second case and shows that $M^\alpha, S \not\models \beta'_i$ for all $\beta'_i \in \mathcal{R}_{2^{|\alpha|}}(\psi)$. But this contradicts our earlier assumption, so therefore $M^\alpha, S \not\models \psi$, as required. ☐

This concludes our proof that the given axiomatisation of IE-PDL is complete with respect to its semantics.

3.9 Conclusion

In this chapter we introduced the new logic IE-PDL, about which we have proven some properties which are familiar from inquisitive semantics. We have also found several reduction axioms for the new language, and we have shown that every IEL formula can be translated into an equivalent sentence in IE-PDL. Lastly, we have given an axiomatisation for IE-PDL and proven that this axiomatisation is complete.

Chapter 4

Logic of Communication, Change, and Issues

4.1 Introduction

This chapter will introduce the main new result of this thesis, namely the Logic of Communication, Change, and Issues (LCCI). Here we will combine the ideas from [18] and [22] to build an inquisitive version of LCC.

This chapter will start off with extending the action models from AMLI into update models for LCCI by defining a substitution for each action. We will then give an update mechanism that will execute the update models in IE-PDL models. We will then discuss some examples of things that we can now model in LCCI. After that, we will show that LCCI has the same properties that one would expect from an inquisitive logic. Then we will give a reduction procedure to reduce LCCI to IE-PDL and use that to give a sound and complete axiomatisation of the system. We will end this chapter with examples of those reduced sentences.

4.2 Definitions

4.2.1 Syntax

Unlike in AMLI, the syntax for LCCI will be given in one definition. This is because in AMLI is it possible to have an AMLI formula as a precondition for an action. For our purposes, this extra expressivity is not necessary, but does lead to complications down the line. Therefore, we will say that the preconditions for all actions have to be IE-PDL formulas.

The language of LCCI is the same as the language of IE-PDL extended with an update operator $[\mathsf{U},\mathsf{s}]\,\varphi$, which says that after the execution of some event in s, which is a set of events, φ is supported.

Definition 4.1 (Language of LCCI)**.** Given a finite set of propositions $\mathcal{P}$, a set of relational atoms $\mathcal{A}$, and an LCCI update model U, with p ranging over $\mathcal{P}$ and a ranging over $\mathcal{A}$. The language of LCCI is given by:

$$\begin{aligned}
&\varphi := p \mid \bot \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \mathbin{\vee\!\!\!\vee} \varphi_2 \mid \varphi_1 \to \varphi_2 \mid [\pi]\varphi \mid [\![\pi]\!]\varphi \mid [\mathsf{U},\mathsf{s}]\,\varphi \\
&\pi := a \mid ?\varphi \mid \pi_1;\pi_2 \mid \pi_1 \cup \pi_2 \mid \pi^*
\end{aligned}$$

We will use the same abbreviations as with $\mathsf{IE\text{-}PDL}$ with the following additions:

$[\mathsf{s}]\,\varphi = [\mathsf{U}, \mathsf{s}]\,\varphi$	where U is clear from context
$[\mathsf{e}]\,\varphi = [\{\mathsf{e}\}]\,\varphi$	where e is a single event

4.2.2 Update Models

The update models of LCCI combine the insights of AMLI and LCC into one model structure. This means that we take the structure from LCC, but replace the relationships between events by an epistemic state map as in AMLI. This gives us the substitutions from LCC, which drive factual change, and allows us to model agents interest into the different actions, as in AMLI. Unlike in AMLI we will not enforce the properties of **factivity** and **introspection** on the state maps. This is to show that these properties are not necessary for the technical results of this section.

Definition 4.2 (Update Models for LCCI)**.** Given a set of agents $\mathcal{A}$ an update model for LCCI is a quadruple $\mathsf{U} = \langle \mathsf{E}, \Delta_{\mathcal{A}}, \mathsf{pre}, \mathsf{sub}\rangle$ where:

- $\mathsf{E} = \{\mathsf{e}_1, \ldots, \mathsf{e}_\mathsf{n}\}$ is a set of events
- $\Delta_{\mathcal{A}} = \{\Delta_a \mid a \in \mathcal{A}\}$ is a set of inquisitive state maps that assigns an inquisitive state to each event e_i.
- $\mathsf{pre} : \mathsf{E} \to \mathcal{L}_!^{\mathsf{IE\text{-}PDL}}$ a function that maps every event to its precondition
- $\mathsf{sub} : \mathsf{E} \to SUB_{\mathsf{IE\text{-}PDL}}$, a function that maps every event to its substitution.

These models are just like the Inquisitive Action Models from Chapter 4 of [22], with the sub function from LCC, which is what drives factual change. This means that for events that do not require factual change we can simply use the empty substitution ϵ, which maps every formula onto itself. Examples of this can be seen in Examples 4.1 and 4.2 below.

One thing to note about the substitutions is that they only map propositional atoms to declarative formulas. This is because we deal with factual change, and not with the creation of questions. However, this does not mean that events with a substitution cannot raise issues. This happens in the same way as announcements raise issues, and can be seen in Example 4.4.

4.2.3 Update Execution

Update execution is also a combination of update execution in LCC and AMLI. Here we take the update of the valuation from LCC, and the update of the state maps from AMLI. This gives the following update procedure:

Definition 4.3 (Update Procedure for LCCI)**.** Given an IE-PDL model M and a LCCI update model U, $M \circ \mathsf{U} = \langle \mathcal{W}', V', R'_{\mathcal{A}}\rangle$ is the result of executing U in M, defined as follows:

- $\mathcal{W}' = \{(w, \mathsf{e}) \mid M, w \models \mathsf{pre}(\mathsf{e})\}$ is the new set of possible worlds
- $V'(p) = \{(w, \mathsf{e}) \mid M, w \models \mathsf{sub}(\mathsf{e})(p)\}$ is the new valuation
- $t \in \Sigma'_a((w, \mathsf{e}))$ iff $\pi_1(t) \in \Sigma_a(w)$ and $\pi_2(t) \in \Delta_a(\mathsf{e})$. Here π_1 and π_2 are the projection operators from Definition 2.23.

Since our possible worlds are now ordered pairs of worlds and events, our states are now sets of ordered pairs of worlds and events, as in AMLI.

We will now need to show that the model $M \circ \mathsf{U}$ is an IE-PDL model.

Proposition 4.1. *For any IE-PDL model M and any LCCI update model U, $M \circ \mathsf{U}$ is an IE-PDL model. That is to say, for any agent a and any wordl $(w, \mathsf{e}) \in \mathcal{W}'$, $\Sigma'_a((w, \mathsf{e}))$ is non-empty and downward closed.*

Proof. Since both $\Sigma_a(w)$ and $\Delta_a(\mathsf{e})$ were downward closed and non-empty, $\Sigma'_a((w, \mathsf{e}))$ is also downward closed and non-empty. □

4.2.4 Semantics

Lastly we have to extend the support definition for IE-PDL a bit to account for the new $[\mathsf{U}, \mathsf{e}]$ operator. For this, we will have to give a definition of updated states in LCCI as well.

Definition 4.4 (Updated state)**.** Given an IE-PDL model M, a LCCI update model U, an information state s in M, and a set of events s in U, the updated state in $M \circ \mathsf{U}$ is $s\,[\mathsf{U}, \mathsf{s}]$ such that:

$$s\,[\mathsf{U}, \mathsf{s}] = \{(w, \mathsf{e}) \mid w \in s, \mathsf{e} \in \mathsf{s}, M, w \models \mathsf{pre}(\mathsf{e})\}$$

Definition 4.5 (Support for LCCI)**.** The support conditions for LCCI are the same as those for IE-PDL, with the following addition:

$$M, s \models [\mathsf{U}, \mathsf{s}]\,\varphi \iff M \circ \mathsf{U}, s\,[\mathsf{U}, \mathsf{s}] \models \varphi$$

4.3 Examples

Before we look at the properties of this system, we will first give some examples of how the system works. For this we will yet again refer to the game of Citadels. We will start by using the same situation as in Section 3.4. We will assume that Claire has selected the Condottiere as her character. Now the game starts off with Alice calling for the king to reveal themselves. Since no one has this card the turn of the king is skipped, and we move on to the next character.

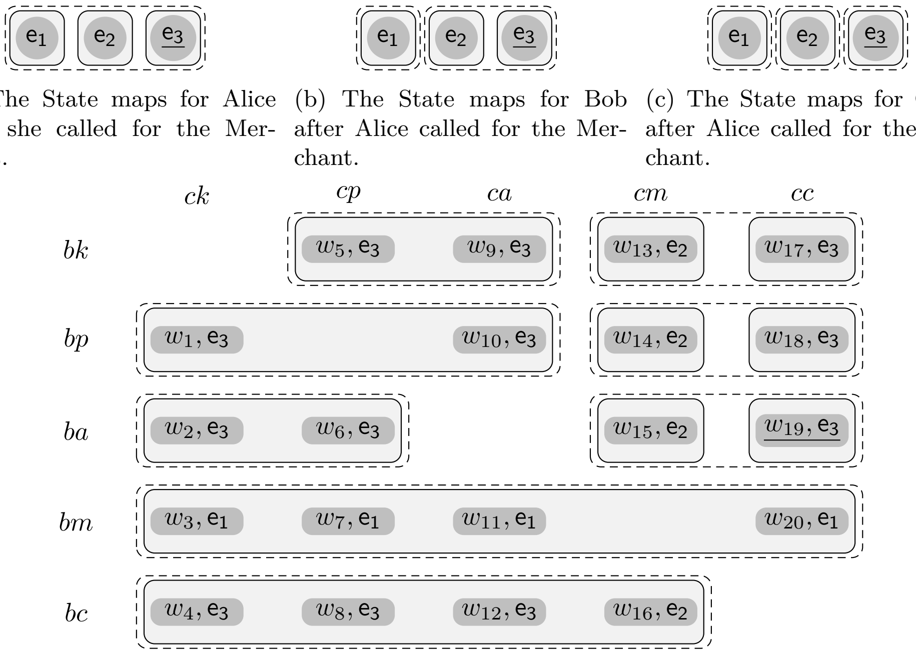
Example 4.1 (Public Announcements)**.** Since no one has the king, Alice calls for the Preacher, and reveals that she has that character. This is similar to a public announcement that Alice is the Preacher. In LCCI we could model this by making an update model with one action, namely the action with ap as a precondition, the empty substitution ϵ as the substitution, and an epistemic state map which for all agents is $\Delta(\mathsf{e}) = \{\{\mathsf{e}\}, \emptyset\}$. This closely mimics the standard behaviour for public announcements.

Example 4.2 (Raising Issues)**.** The situation becomes more interesting when Alice is done with her turn and calls for the Merchant. Every player has passed on the merchant, but this is only known to Claire. However, calling for the Merchant does raise the question whether anyone picked that character. We can also model this in LCCI. For this we need three events: one in which Bob picked the Merchant as a character, one in which Claire did that, and one in which neither of them picked that card. The precondition for the first event, e_1, is bm, since in this event Bob picked the merchant. The precondition for the second event, e_2, the one where Claire picked the merchant, is cm. The precondition for e_3 is $\neg bm \wedge \neg cm$, since in this event, neither player picked the card. Since this example does not involve factual change, all the events will have an empty substitution. The state maps for the events for the different agents are as in Figure 4.1.

Since Alice does not know which cards the other players have selected, all the events are possible for her. She is interested in which of the events is the actual one, since this is relevant for the continuation of the game. Bob knows that e_1 cannot be executed, since bm is not true in the actual world, so he can distinguish between e_1 and the other events. Since Claire put away the Merchant, she can distinguish between all the events, since she actually knows where the card is. This question does not raise any questions for her, since she already knows the answer.

For each possible world, only one event can be executed, thanks to the preconditions. This makes showing the outcome of an event easier to see. In Figure 4.1d the accessibility relation for Bob is given, in the same manner as in Section 3.4. Note, however, that the worlds on the horizontal axis differ in which card Claire has, since we now know that Alice is the Preacher. We can see that after Alice has called for the Merchant, Bob now wants to know if Claire has the Merchant $(M, \{(w_{19}, \mathsf{e}_3)\} \models [\![b]\!](cm \vee\!\!\!\vee \neg cm))$. This is because there are three states reachable from R_b, which are $\emptyset$, where everything is supported, $\{w_{19}, \mathsf{e}_3\}$, where $\neg cm$ is supported, and $\{w_{15}, \mathsf{e}_2\}$, where cm is supported. This is also the wanted outcome of the update.

Now as soon as Claire says that she is not the merchant, this counts as a public announcement which works similarly to Example 4.1.



(a) The State maps for Alice after she called for the Merchant.

(b) The State maps for Bob after Alice called for the Merchant.

(c) The State maps for Claire after Alice called for the Merchant.

(d) The relations for Bob after Alice called for the merchant.

Figure 4.1: The state maps and relation for Bob for Example 4.2

Example 4.3 (Public Factual Change)**.** Now we have arrived at Bob's turn. He starts his turn with taking two gold coins, in order to have more money to spend on building buildings. We can also model this action in LCCI, by the use of our factual change mechanism. For this, we need to make an update model with a set of events, one for each number of coins that a player can get at once. Since this event happens out in the open, all of these events are distinguishable, so for all agents and each event it is the case that the state maps only contain a set with the event itself and the empty set. None of the actions have preconditions, so for each event we have that

$\mathsf{pre}(\mathsf{e}) = \top$. The substitutions do get more interesting now. Let's have a look at the substitution for the event in which Bob takes two coins. Let $cb1$ stand for the proposition that Bob has one coin, let $cb2$ stand for the proposition that Bob has two coins, etc.

For this substitution we want to represent the idea that Bob has picked up two coins. We can do this by mapping each proposition to the proposition that represents that he has two coins less. If that second proposition was supported before the update, then the first will be supported after the update. We can write this in the following way:

$$\mathsf{sub}(\mathsf{e}_2) = \{cb10 \mapsto cb8, \ldots, cb3 \mapsto cb1, cb2 \mapsto \top, cb1 \mapsto \top\}$$

We have to map $cb2$ and $cb1$ to $\top$, since these are most certainly true after picking two new coins. If we were to do this for every number of coins that a player could pick, we would get a full model for the economics in Citadels.

Example 4.4 (Private Factual Change)**.** Besides factual change that is known to everybody, we can also model factual change that is only known by some. An example of such a situation in Citadels is the acquisition of building cards. If we continue where we left off in Example 4.3, we now get to the point where Bob uses his character's special ability, which allows him to pick up two cards, which he can then use for building. We will model the picking up of two cards as the picking up of one card twice. Let $hy1b$ stand for the fact that the yellow building of cost one is in the hand of Bob, $hy3b$ stand for the fact that the yellow building of cost three is in the hand of Bob, etc. and let $by1b$ stand for the proposition that Bob has built the yellow building of cost one. For the building of the events, we will only focus on one example for the precondition and substitution, namely Bob picking up the green building of cost 5.

You cannot pick up a card if this card is already in your hand, or if you have already built the building corresponding to it. We can encode this in the precondition by stating this in a formula:

$$\mathsf{pre}(\mathsf{e}) = \neg hg5b \wedge \neg bg5b$$

The factual change that happens is that you now have this card in your hand. We can do this by mapping the proposition onto $\top$:

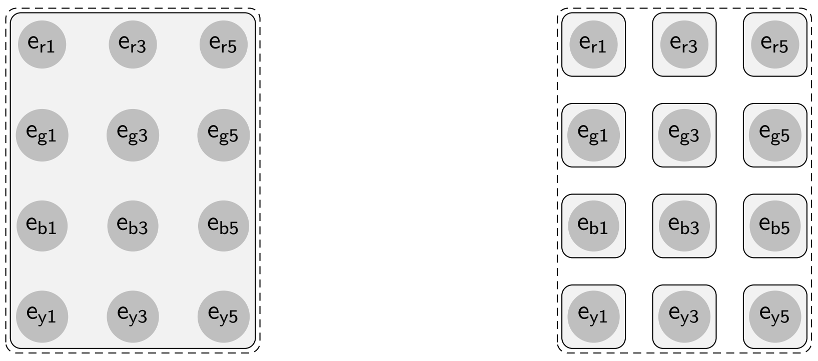
$$\mathsf{sub}(\mathsf{e}) = \{hg5b \mapsto \top\}$$

The state maps can differ per person. For the sake of explanation, let us assume that Alice does not care which card Bob picked, but Claire does. We can use this to shape the state maps. As before, this means that for Alice we need to have that from each state, the state with all events is reachable, whereas for Claire all the relations for the singleton states only need to be reflexive. These state maps can be found in Figure 4.2.

As can be seen in the examples, using LCCI we can model some aspects of the game which could not be modelled beforehand. On the one hand we can now model the agents interest in the questions and events that transpire around them, as in Example 4.2, which is something that was not possible to do in LCC and similar languages. On the other hand, we can now have events that effect the world in other ways than just epistemic events, as in Example 4.3, which is an improvement upon AMLI. The combination of both is also a possiblity, as can be seen in Example 4.4. This makes LCCI more expressive than either of its inspirations.

4.4 Properties

For LCCI we also want the properties of persistence and the empty state property to hold.



(a) The state map for Alice for Bob picking a card.

(b) The state map for Claire for Bob picking a card.

Figure 4.2: The state maps for Alice and Claire for the action of Bob picking a card.

Proposition 4.2 (Persistence for LCCI)**.** *For any LCCI model M and any state s in M, if $M, s \models \varphi$ and $t \subseteq s$, then $M, t \models \varphi$*

Proposition 4.3 (Empty state property)**.** *For any LCCI model M and any LCCI formula φ, $M, \emptyset \models \varphi$.*

Proof. The proof for both propositions is the same as that for IE-PDL, (Propositions 3.1 and 3.2) with the exception of the proof for $[\mathsf{U}, \mathsf{s}]\,\varphi$, so only the proof for that formula will be given. $M, s \models [\mathsf{U}, \mathsf{s}]$ iff $M \circ \mathsf{U}, s\,[\mathsf{U}, \mathsf{s}] \models \varphi$. Since $t\,[\mathsf{U}, \mathsf{s}] \subseteq s\,[\mathsf{U}, \mathsf{s}]$, $M \circ \mathsf{U}, t\,[\mathsf{U}, \mathsf{s}] \models \varphi$. This is the same as $M, t \models [\mathsf{U}, \mathsf{s}]\,\varphi$.

$M, \emptyset \models [\mathsf{U}, \mathsf{s}]\,\varphi$ iff $M \circ \mathsf{U}, \emptyset\,[\mathsf{U}, \mathsf{s}] \models \varphi$. Since $\emptyset\,[\mathsf{U}, \mathsf{s}] = \emptyset$, this follows from the inductive hypothesis. ☐

Like in AMLI, our $[\mathsf{U}, \mathsf{s}]\,\varphi$ operator has these properties as well.

Proposition 4.4 (Modal persistence property)**.** *For any IE-PDL model M, any state s in M, any LCCI update model U, and any set of events s, if $M, s \models [\mathsf{U}, \mathsf{s}]\,\varphi$ and $\mathsf{t} \subseteq \mathsf{s}$, then $M, s \models [\mathsf{U}, \mathsf{t}]\,\varphi$.*

Proof. $M, s \models [\mathsf{U}, \mathsf{s}]\,\varphi \iff M \circ \mathsf{U}, s\,[\mathsf{U}, \mathsf{s}] \models \varphi$. Since $s\,[\mathsf{U}, \mathsf{t}] \subseteq s\,[\mathsf{U}, \mathsf{s}]$, from Proposition 4.2 we get $M \circ \mathsf{U}, s\,[\mathsf{U}, \mathsf{t}] \models \varphi \iff M, s \models [\mathsf{U}, \mathsf{t}]\,\varphi$. ☐

Proposition 4.5 (Modal empty state property)**.** *For any IE-PDL model M, and state s in M, any LCCI update model U, and any LCCI formula φ, $M, s \models [\mathsf{U}, \emptyset]\,\varphi$*

Proof. $M, s \models [\mathsf{U}, \emptyset]\,\varphi \iff M \circ \mathsf{U}, s\,[\mathsf{U}, \emptyset] \models \varphi$. Since $s\,[\mathsf{U}, \emptyset] = \emptyset$, we get $M \circ \mathsf{U}, \emptyset \models \varphi$, which is true by Proposition 4.3. Therefore, $M, s \models [\mathsf{U}, \emptyset]\,\varphi$. ☐

4.4.1 Declaratives

Just like IEL, AMLI, and IE-PDL, LCCI has a declarative fragment. This fragment is the same as that for IE-PDL, with the addition of $[\mathsf{U}, \mathsf{s}]\,\alpha$.

Definition 4.6 (Declaratives of LCCI)**.** The declarative fragment of LCCI is given by:

$$\alpha := p \mid \bot \mid [\pi]\varphi \mid [\![\varpi]\!]\varphi \mid [\![\pi]\!]\alpha \mid [\mathsf{U}, \mathsf{s}]\,\alpha \mid \alpha_1 \wedge \alpha_2 \mid \varphi \to \alpha$$

Proposition 4.6. *The declarative fragment of* LCCI *is truth-conditional.*

Proof. The proof for most of the formulas is the same as that for IE-PDL, (Proposition 3.4) so we will only show the proof for $[\mathsf{U},\mathsf{s}]\,\alpha$.

$$
\begin{aligned}
M, s \models [\mathsf{U},\mathsf{s}]\,\alpha \iff\ & M \circ \mathsf{U}, s\,[\mathsf{U},\mathsf{s}] \models \alpha \\
\iff\ & \text{for all } (w,\mathsf{e}) \in s\,[\mathsf{U},\mathsf{s}]\,, M \circ \mathsf{U}, (w,\mathsf{e}) \models \alpha \\
\iff\ & \text{for all } (w,\mathsf{e}) \in \mathcal{W}' \text{ such that } w \in s \text{ and } \mathsf{e} \in \mathsf{s}, \\
& \quad M \circ \mathsf{U}, (w,\mathsf{e}) \models \alpha \\
\iff\ & \text{for all } w \in s \text{ such that } (w,\mathsf{e}) \in \mathcal{W}' \text{ and } \mathsf{e} \in \mathsf{s}, \\
& \quad M \circ \mathsf{U}, (w,\mathsf{e}) \models \alpha \\
\iff\ & \text{for all } w \in s, M, w\,[\mathsf{U},\mathsf{s}] \models \alpha \\
\iff\ & \text{for all } w \in s, M, w \models [\mathsf{U},\mathsf{s}]\,\alpha
\end{aligned}
$$

□

As one might expect, we also have a similar result for sets of actions, as in AMLI.

Proposition 4.7. *If α is truth-conditional, then* $[\mathsf{s}]\,\alpha \iff \bigwedge_{\mathsf{e}\in\mathsf{s}} [\mathsf{e}]\,\alpha$.

Proof. Since both formulas are truth conditional, we only have to show that they have the same truth conditions.

$$
\begin{aligned}
M, w \models [\mathsf{U},\mathsf{s}]\,\alpha \iff\ & M \circ \mathsf{U}, w\,[\mathsf{s}] \models \alpha \\
\iff\ & \text{for all } \mathsf{e} \in \mathsf{s}, M \circ \mathsf{U}, (w,\mathsf{e}) \models \alpha \\
\iff\ & \text{for all } \mathsf{e} \in \mathsf{s},\ \text{if } (w,\mathsf{e}) \in \mathcal{W}',\ \text{then } M \circ \mathsf{U}, (w,\mathsf{e}) \models \alpha \\
\iff\ & \text{for all } \mathsf{e} \in \mathsf{s}, M \circ \mathsf{U}, w\,[\mathsf{e}] \models \alpha \\
\iff\ & \text{for all } \mathsf{e} \in \mathsf{s}, M, w \models [\mathsf{e}]\,\alpha \\
\iff\ & M, w \models \bigwedge_{\mathsf{e}\in\mathsf{s}} [\mathsf{e}]\,\alpha
\end{aligned}
$$

□

4.4.2 Resolution

Like in IE-PDL and its predecessors, LCCI also has resolutions. Since LCCI is based on IE-PDL, the resolutions for LCCI also need to be based on the maximum number of worlds in a model.

Definition 4.7 (Resolutions for LCCI)**.** The resolutions for LCCI are defined as the resolutions for IE-PDL (Definition 3.8) extended with the following:

$$\mathcal{R}_n([\mathsf{U},\mathsf{s}]\,\varphi) = \{[\mathsf{U},\mathsf{s}]\,\alpha \mid \alpha \in \mathcal{R}_n(\varphi)\}$$

We now also get all the usual properties for resolutions.

Proposition 4.8. *For any M, s, and φ, $M, s \models \varphi \iff$ for some $\alpha \in \mathcal{R}_{|\mathcal{W}|}(\varphi)$, $M, s \models \alpha$.*

Proof. This proof works the same as Proposition 3.6, with the addition of $[\mathsf{U},\mathsf{s}]\,\varphi$, so only that proof is given.

$$\begin{aligned} M,s\models[\mathsf{U},\mathsf{s}]\,\varphi &\iff M\circ\mathsf{U},s\,[\mathsf{s}]\models\varphi\\ &\iff M\circ\mathsf{U},s\,[\mathsf{s}]\models\alpha \text{ for some } \alpha\in\mathcal{R}_{|\mathcal{W}|}(\varphi)\\ &\iff M,s\models[\mathsf{U},\mathsf{s}]\,\alpha \text{ for some } \alpha\in\mathcal{R}_{|\mathcal{W}|}(\varphi)\\ &\iff M,s\models\alpha \text{ for some } \alpha\in\mathcal{R}_{|\mathcal{W}|}([\mathsf{U},\mathsf{s}]\,\varphi) \end{aligned}$$

□

Corollary 4.9 (Normal Form)**.** *For any M, s, and φ, $M,s\models\varphi \iff M,s\models\bigvee\!\!\!\!\bigvee\mathcal{R}_{|\mathcal{W}|}(\varphi)$.*

Proof. The proof is the same as Corollary 3.7. □

Corollary 4.10. *For any M, s, and Φ, $M,s\models\Phi \iff$ for some $\Gamma\in\mathcal{R}_{|\mathcal{W}|}(\Phi)$, $M,s\models\Gamma$.*

4.5 Reduction

Now that we have seen how the system works and what its properties are, we can start to think about what the axioms for this system are. For this, we can look at the axioms from AMLI and LCC and combine their features so it fits with our system. For both of these systems, all axioms are reduction axioms, since these languages are not more expressive than their underlying static languages. Since this is also the case for LCCI, we should therefore be able to use a similar technique.

4.5.1 Atoms

As is usual, we start with giving the reductions for propositional atoms and $\bot$. Since any propositional atom is truth conditional, and the substitution for a propositional atom will also always be truth conditional, we can prove the reduction for propositional atoms in terms of single actions and worlds.

Proposition 4.11. *For any IE-PDL model M and any state s in M, $M,s\models[\mathsf{U},\mathsf{e}]\,p \iff M,s\models\mathsf{pre}(\mathsf{e})\to p^{\mathsf{sub}(\mathsf{e})}$.*

Proof. Since both statements are truth conditional, we only have to show that the truth conditions are the same.

$$\begin{aligned} M,w\models[\mathsf{U},\mathsf{e}]\,p &\iff M\circ\mathsf{U},w\,[\mathsf{e}]\models p\\ &\iff w\,[\mathsf{e}]=\emptyset \text{ or } M\circ\mathsf{U},(w,\mathsf{e})\models p\\ &\iff w\,[\mathsf{e}]=\emptyset \text{ or } M,w\models p^{\mathsf{sub}(\mathsf{e})}\\ &\iff \text{ if } M,w\models\mathsf{pre}(\mathsf{e}), \text{ then } M,w\models p^{\mathsf{sub}(\mathsf{e})}\\ &\iff M,w\models\mathsf{pre}(\mathsf{e})\to p^{\mathsf{sub}(\mathsf{e})} \end{aligned}$$

□

Now the following follows from the previous proposition and Proposition 4.7:

Corollary 4.12. *For any IE-PDL model M and any state s in M, $M,s\models[\mathsf{U},\mathsf{s}]\,p \iff M,s\models \bigwedge_{\mathsf{e}\in\mathsf{s}}(\mathsf{pre}(\mathsf{e})\to p^{\mathsf{sub}(\mathsf{e})})$.*

Using a similar method we can also derive a reduction equivalence for $[\mathsf{U}, \mathsf{s}]\,\bot$.

Proposition 4.13. *For any IE-PDL model M and any state s in M, $M, s \models [\mathsf{U}, \mathsf{e}]\,\bot \iff M, s \models \neg\mathsf{pre}(\mathsf{e})$.*

Proof. Since both formulas are truth conditional, we only have to show that the truth conditions are the same.

$$\begin{aligned} M, w \models [\mathsf{U}, \mathsf{e}]\,\bot &\iff M \circ \mathsf{U}, w\,[\mathsf{e}] \models \bot \\ &\iff w\,[\mathsf{e}] = \emptyset \\ &\iff M, w \not\models \mathsf{pre}(\mathsf{e}) \\ &\iff M, w \models \neg\mathsf{pre}(\mathsf{e}) \end{aligned}$$

□

Corollary 4.14. *For any IE-PDL model M and any state s in M, $M, s \models [\mathsf{U}, \mathsf{s}]\,\bot \iff M, s \models \bigwedge_{\mathsf{e} \in \mathsf{s}} \neg\mathsf{pre}(\mathsf{e})$.*

4.5.2 Conjunction and Inquisitive Disjunction

Like in both AMLI and LCC, we can distribute a dynamic modality across a conjunction.

Proposition 4.15. *For any IE-PDL model M and any state s in M, $M, s \models [\mathsf{U}, \mathsf{s}]\,(\varphi \wedge \psi) \iff M, s \models [\mathsf{U}, \mathsf{s}]\,\varphi \wedge [\mathsf{U}, \mathsf{s}]\,\psi$.*

Proof.

$$\begin{aligned} M, s \models [\mathsf{U}, \mathsf{s}]\,(\varphi \wedge \psi) &\iff M \circ \mathsf{U}, s\,[\mathsf{s}] \models \varphi \wedge \psi \\ &\iff M \circ \mathsf{U}, s\,[\mathsf{s}] \models \varphi \text{ and } M \circ \mathsf{U}, s\,[\mathsf{s}] \models \psi \\ &\iff M, s \models [\mathsf{U}, \mathsf{s}]\,\varphi \text{ and } M, s \models [\mathsf{U}, \mathsf{s}]\,\psi \\ &\iff M, s \models [\mathsf{U}, \mathsf{s}]\,\varphi \wedge [\mathsf{U}, \mathsf{s}]\,\psi \end{aligned}$$

□

We can use a similar reduction for Inquisitive Disjunction.

Proposition 4.16. *For any IE-PDL model M and any state s in M, $M, s \models [\mathsf{U}, \mathsf{s}]\,(\varphi \vee\!\!\!\vee \psi) \iff M, s \models [\mathsf{U}, \mathsf{s}]\,\varphi \vee\!\!\!\vee [\mathsf{U}, \mathsf{s}]\,\psi$.*

Proof.

$$\begin{aligned} M, s \models [\mathsf{U}, \mathsf{s}]\,(\varphi \vee\!\!\!\vee \psi) &\iff M \circ \mathsf{U}, s\,[\mathsf{s}] \models \varphi \vee\!\!\!\vee \psi \\ &\iff M \circ \mathsf{U}, s\,[\mathsf{s}] \models \varphi \text{ or } M \circ \mathsf{U}, s\,[\mathsf{s}] \models \psi \\ &\iff M, s \models [\mathsf{U}, \mathsf{s}]\,\varphi \text{ or } M, s \models [\mathsf{U}, \mathsf{s}]\,\psi \\ &\iff M, s \models [\mathsf{U}, \mathsf{s}]\,\varphi \vee\!\!\!\vee [\mathsf{U}, \mathsf{s}]\,\psi \end{aligned}$$

□

4.5.3 Implication

As in [22] we do not have that $[\mathsf{s}]\,(\varphi \to \psi) \iff [\mathsf{s}]\,\varphi \to [\mathsf{s}]\,\psi$. This still makes it possible to reduce the formula $[\mathsf{s}]\,(\varphi \to \psi)$, but we will have to do it using the normal form. However, like in AMLI, we can reduce $[\mathsf{s}]\,(\varphi \to \psi)$ if s is a singleton.

Proposition 4.17. *For any IE-PDL model M and any state s in M, $M, s \models [\mathsf{U},\mathsf{e}]\,(\varphi \to \psi) \iff M, s \models [\mathsf{U},\mathsf{e}]\,\varphi \to [\mathsf{U},\mathsf{e}]\,\psi$.*

Proof.

- $\Rightarrow$: Suppose $M, s \models [\mathsf{U},\mathsf{e}]\,(\varphi \to \psi)$. Then we get $M \circ \mathsf{U}, s\,[\mathsf{e}] \models \varphi \to \psi$ from the support condition for $[\mathsf{s}]$. Now take any $t \subseteq s$ such that $M, t \models [\mathsf{U},\mathsf{e}]\,\varphi$. This means we get $M \circ \mathsf{U}, t\,[\mathsf{e}] \models \varphi$ and by the support condition for implication $M \circ \mathsf{U}, t\,[\mathsf{e}] \models \psi$. In turn, this means that $M, t \models [\mathsf{U},\mathsf{e}]\,\psi$, which means we have $M, s \models [\mathsf{U},\mathsf{e}]\,\varphi \to [\mathsf{U},\mathsf{e}]\,\psi$.
- $\Leftarrow$: Suppose $M, s \models [\mathsf{U},\mathsf{e}]\,\varphi \to [\mathsf{U},\mathsf{e}]\,\psi$. Take any $t' \subseteq s\,[\mathsf{e}]$ such that $M \circ \mathsf{U}, t' \models \varphi$. Let $t = \pi_1(t')$, so by the definition of updated states $t\,[\mathsf{e}] = t'$. Since we have $M \circ \mathsf{U}, t\,[\mathsf{e}] \models \varphi$, $M, t \models [\mathsf{U},\mathsf{e}]\,\varphi$. From the support condition for implication we then get $M, t \models [\mathsf{U},\mathsf{e}]\,\psi$, and therefore $M \circ \mathsf{U}, t\,[\mathsf{e}] \models \psi$. Since t' was chosen arbitrarily, we can then conclude that $M \circ \mathsf{U}, s\,[\mathsf{e}] \models \varphi \to \psi$ and therefore $M, s \models [\mathsf{U},\mathsf{e}]\,(\varphi \to \psi)$.

$\square$

While we cannot lift this to a full set s like we did with $[\mathsf{U},\mathsf{e}]\,p$, we can use this together with the normal form to get a full reduction of the formula.

4.5.4 Modalities

Because of the way in which programs work in LCCI, reducing formulas with modalities happens differently from the way it happens in AMLI. However, since the logic also borrows heavily from LCC, we can borrow some of their tricks for reduction as well.

The reduction for LCC works via so-called *program transformations*, which take a program in the updated model and transform it into a program in the original model. The basic idea is that $[\mathsf{U},\mathsf{s}]\,[\![\pi]\!]\varphi$ is true in some model M iff there is a π path in $M \circ \mathsf{U}$ from $s\,[\mathsf{s}]$ to $t\,[\mathsf{t}]$ where φ is supported. This in turn means there is some path $s \cdots t$ in M and some path $\mathsf{s} \cdots \mathsf{t}$ in U such that in all the worlds in t, an event from t can be executed. Using the programs from IE-PDL we can luckily specify exactly this condition in a formula, which we will call $T^{\mathsf{U}}_{\mathsf{st}}(\pi)$. Eventually, we will want to prove that $s\,[\mathsf{s}]\,R'_\pi t\,[\mathsf{t}] \iff sR_{T^{\mathsf{U}}_{\mathsf{st}}(\pi)}t$.

However, we will need to take into account that we are now not dealing with single events, as was the case in LCC, but with sets of events instead. In particular, we cannot just check whether agent a cannot distinguish between s and t, as we will illustrate with the following example. Assume that $s\,[\mathsf{s}]\,R'_a t\,[\mathsf{t}]$. Now take some extra event e, such that there is no world w in t such that $M, w \models \mathsf{pre}(\mathsf{e})$. In this case, $t\,[\mathsf{t}] = t\,[\mathsf{t} \cup \{\mathsf{e}\}]$, so $s\,[\mathsf{s}]\,R'_a t\,[\mathsf{t} \cup \{\mathsf{e}\}]$. However, agent a might be able to distinguish between an event in s and $\mathsf{t} \cup \{\mathsf{e}\}$ happening, even if they cannot distinguish between s and t. So in this case, when looking at $T^{\mathsf{U}}_{\mathsf{s}(\mathsf{t}\cup\{\mathsf{e}\})}(a)$, we want to disregard the event e, since it cannot be executed in t anyway. We can do this by only looking at the largest subsets u of $\mathsf{t} \cup \{\mathsf{e}\}$ that a cannot distinguish from s, and making sure that the events in the rest of $\mathsf{t} \cup \{\mathsf{e}\}$ cannot be executed in t, while all the events in u can be executed. We will have a similar problem for the set $s\,[\mathsf{s}]$, so we will also have to account for this there.

Luckily, we can check whether a set of events cannot be executed in a state by checking whether none of the events in that set can be executed in any world in that state. Since all preconditions are declarative, we can test this for a set of events s' in a state s' with $M, s' \models \bigwedge_{\mathsf{e}\in\mathsf{s}'} \neg\mathsf{pre}(\mathsf{e})$. This makes it possible to also trace all the 'paths' through $\bigcup_{\mathsf{e}\in\mathsf{s}} \Delta_a(\mathsf{e})$ that end up in a subset of t that can be executed in full in t. If we then start all these paths from all the subsets of s that can be executed in full in s, we alleviate all these problems. This gives us the following definition of the translations.

Definition 4.8 (Program Transformations)**.**

$$\begin{aligned}
T^{\mathsf{U}}_{\mathsf{st}}(a) &= ?\bigvee_{\mathsf{e}\in\mathsf{s}} \mathsf{pre}(\mathsf{e}); \bigcup_{\mathsf{s}'\subseteq\mathsf{s}} \left(?\bigwedge_{\mathsf{e}\in(\mathsf{s}-\mathsf{s}')} \neg\mathsf{pre}(\mathsf{e}); a; ?\bigvee_{\mathsf{t}'\in\bigcup_{\mathsf{e}\in\mathsf{s}'}\Delta_a(\mathsf{e}),\mathsf{t}'\subseteq\mathsf{t},\mathsf{t}'\neq\emptyset} \left(\wedge_{\mathsf{e}\in(\mathsf{t}-\mathsf{t}')}\neg\mathsf{pre}(\mathsf{e})\right)\right)\\
T^{\mathsf{U}}_{\mathsf{st}}(?\varphi) &= \begin{cases} ?\,[\mathsf{U},\mathsf{s}]\,\varphi & \text{if } \mathsf{t}\subseteq\mathsf{s}\\ ?\bot & \text{otherwise}\end{cases}\\
T^{\mathsf{U}}_{\mathsf{st}}(\pi_1;\pi_2) &= \bigcup_{\mathsf{u}\subseteq\mathsf{E}} \left(T^{\mathsf{U}}_{\mathsf{su}}(\pi_1); T^{\mathsf{U}}_{\mathsf{ut}}(\pi_2)\right)\\
T^{\mathsf{U}}_{\mathsf{st}}(\pi_1\cup\pi_2) &= T^{\mathsf{U}}_{\mathsf{st}}(\pi_1)\cup T^{\mathsf{U}}_{\mathsf{st}}(\pi_2)\\
T^{\mathsf{U}}_{\mathsf{st}}(\pi^*) &= K^{\mathsf{U}}_{\mathsf{stE}}(\pi)
\end{aligned}$$

Here we take the empty disjunction to be $\bot$, so if there is no t' for any s' in the atomic case, then $T^{\mathsf{U}}_{\mathsf{st}}(a) = ?\bigvee_{\mathsf{e}\in\mathsf{s}}\mathsf{pre}(\mathsf{e}); \bigcup_{\mathsf{s}'\subseteq\mathsf{s}}(?\bot) = ?\bigvee_{\mathsf{e}\in\mathsf{s}}\mathsf{pre}(\mathsf{e}); ?\bot = ?\bot$, as is also the case in LCC.

We need the additional program $K^{\mathsf{U}}_{\mathsf{stu}}$ for building paths that correspond to the transitive closure of π in the updated model. For this, we need to take all the possible information states between two states s and t into account. Intuitively, $K^{\mathsf{U}}_{\mathsf{stu}}(\pi)$ is a program for all the π paths from $s\,[\mathsf{s}]$ to $t\,[\mathsf{t}]$ that can be traced through $M\circ\mathsf{U}$ while avoiding a pass through intermediate states which are not proper subsets of u.

An observation that helps in understanding these transformations is that $?\top$ works well for the empty path, which we now have to specifically account for, since, unlike in LCC, π^* is no longer reflexive. This is done by adding $?\top$ to each position that beforehand could have been empty due to π^*.

Definition 4.9 ($K^{\mathsf{U}}_{\mathsf{stu}}$ transformer.)**.** $K^{\mathsf{U}}_{\mathsf{stu}}$ is defined by recursion on E, the set with all events.

$$K^{\mathsf{U}}_{\mathsf{st}\emptyset}(\pi) = T^{\mathsf{U}}_{\mathsf{st}}(\pi)$$

$$K^{\mathsf{U}}_{\mathsf{stu}}(\pi) = \begin{cases}
\bigcup_{\mathsf{e}\in\mathsf{u}}\left(K^{\mathsf{U}}_{\mathsf{st}(\mathsf{u}-\{\mathsf{e}\})}(\pi)\right)^* & \text{if } \mathsf{s}=\mathsf{u}=\mathsf{t}\\
\bigcup_{\mathsf{e}\in\mathsf{u}}\left(K^{\mathsf{U}}_{\mathsf{su}(\mathsf{u}-\{\mathsf{e}\})}(\pi)^*\cup ?\top\right); \bigcup_{\mathsf{e}\in\mathsf{u}} K^{\mathsf{U}}_{\mathsf{ut}(\mathsf{u}-\{\mathsf{e}\})}(\pi) & \text{if } \mathsf{s}=\mathsf{u}\neq\mathsf{t}\\
\bigcup_{\mathsf{e}\in\mathsf{u}} K^{\mathsf{U}}_{\mathsf{su}(\mathsf{u}-\{\mathsf{e}\})}(\pi); \bigcup_{\mathsf{e}\in\mathsf{u}}\left(K^{\mathsf{U}}_{\mathsf{ut}(\mathsf{u}-\{\mathsf{e}\})}(\pi)^*\cup ?\top\right) & \text{if } \mathsf{s}\neq\mathsf{u}=\mathsf{t}\\
\bigcup_{\mathsf{e}\in\mathsf{u}} K^{\mathsf{U}}_{\mathsf{st}(\mathsf{u}-\{\mathsf{e}\})}(\pi)\cup\left(\bigcup_{\mathsf{e}\in\mathsf{u}} K^{\mathsf{U}}_{\mathsf{su}(\mathsf{u}-\{\mathsf{e}\})};\right. & \\
\qquad\left.\bigcup_{\mathsf{e}\in\mathsf{u}}\left(K^{\mathsf{U}}_{\mathsf{uu}(\mathsf{u}-\{\mathsf{e}\})}(\pi)^*\cup ?\top\right); \bigcup_{\mathsf{e}\in\mathsf{u}} K^{\mathsf{U}}_{\mathsf{ut}(\mathsf{u}-\{\mathsf{e}\})}(\pi)\right) & \text{otherwise}
\end{cases}$$

Now we have these transformers, we need one more result before we can get to proving that formulas of the form $[\mathsf{U},\mathsf{e}]\,[\![\pi]\!]\varphi$ and $[\mathsf{U},\mathsf{e}]\,[\pi]\varphi$ can be reduced. This result is that the translations specify that the paths in the updated model only exist if and only if there is a path for the translated program.

Theorem 4.18 (Program transformation into $\mathsf{IE\text{-}PDL}$.)**.** *For all update models* U *and all* $\mathsf{\mathit{IE\text{-}PDL}}$ *programs* π*,*

$$sR_{T^{\mathsf{U}}_{\mathsf{st}}(\pi);?\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre}(\mathsf{e})}t \iff s\,[\mathsf{s}]\,R'_{\pi}t\,[\mathsf{t}]$$

Before we can prove this, we will need two auxiliary results.

Lemma 4.19. *Suppose*

$$sR_{T^{\mathsf{U}}_{\mathsf{st}}(\pi);?\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre}(\mathsf{e})}t \iff s\,[\mathsf{s}]\,R'_{\pi}t\,[\mathsf{t}]$$

Then $sR_{K^{\mathsf{U}}_{\mathsf{stu}}(\pi);?\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre}(\mathsf{e})}t$ *iff there is a* π *path from* $s\,[\mathsf{s}]$ *to* $t\,[\mathsf{t}]$ *in* $M\circ\mathsf{U}$ *that does not have intermediate states* $u'\,[\mathsf{u}']$ *with* $\mathsf{u}'\not\subseteq\mathsf{u}$*.*

Proof. The proof works by induction over the set of all events E.

Base case: A π path from $s\,[\mathsf{s}]$ to $t\,[\mathsf{t}]$ that does not visit any intermediate states can now only be a single π step from $s\,[\mathsf{s}]$ to $t\,[\mathsf{t}]$. Such a path exists iff

$$\begin{aligned} s\,[\mathsf{s}]\,R'_{\pi}t\,[\mathsf{t}] &\iff sR_{T^{\mathsf{U}}_{\mathsf{st}}(\pi);?\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre}(\mathsf{e})}t \\ &\iff sR_{K^{\mathsf{U}}_{\mathsf{st}\emptyset}(\pi);?\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre}(\mathsf{e})}t \end{aligned}$$

Induction step: Assume that $sR_{K^{\mathsf{U}}_{\mathsf{st}(\mathsf{u}-\{\mathsf{e}\})}(\pi);?\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre}(\mathsf{e})}t$ for some $\mathsf{e}\in\mathsf{u}$ iff there is a π path from $s\,[\mathsf{s}]$ to $t\,[\mathsf{t}]$ in $M\circ\mathsf{U}$ that does not have intermediate states $u'\,[\mathsf{u}']$ with $\mathsf{u}'\not\subseteq(\mathsf{u}-\{\mathsf{e}\})$. Now we need to show that $sR_{K^{\mathsf{U}}_{\mathsf{stu}}(\pi);?\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre}(\mathsf{e})}t$ iff there is a π path from $s\,[\mathsf{s}]$ to $t\,[\mathsf{t}]$ in $M\circ\mathsf{U}$ that does not pass through any intermediate states $u'\,[\mathsf{u}']$ with $\mathsf{u}'\not\subseteq\mathsf{u}$.

Case $\mathsf{s}=\mathsf{u}=\mathsf{t}$: A π path from $s\,[\mathsf{s}]$ to $t\,[\mathsf{t}]$ in $M\circ\mathsf{U}$ that does not have intermediate states $u'\,[\mathsf{u}']$ with $\mathsf{u}'\not\subseteq\mathsf{u}$ consists of an arbitrary composition of π paths from s to t. By the inductive hypothesis, this is the case iff $sR_{K^{\mathsf{U}}_{\mathsf{st}(\mathsf{u}-\{\mathsf{e}\})}(\pi)^*;?\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre}(\mathsf{e})}t$ for some $\mathsf{e}\in\mathsf{u}$ iff $sR_{\bigcup_{\mathsf{e}\in\mathsf{u}}\left(K^{\mathsf{U}}_{\mathsf{st}(\mathsf{u}-\{\mathsf{e}\})}(\pi)\right)^*;?\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre}(\mathsf{e})}t$ iff $sR_{K^{\mathsf{U}}_{\mathsf{stu}}(\pi);?\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre}(\mathsf{e})}t$.

Case $\mathsf{s}=\mathsf{u}\neq\mathsf{t}$: A π path from $s\,[\mathsf{s}]$ to $t\,[\mathsf{t}]$ in $M\circ\mathsf{U}$ that does not have intermediate states $u'\,[\mathsf{u}']$ with $\mathsf{u}'\not\subseteq\mathsf{u}$ consists of an arbitrary composition of π paths from s to u without intermediate states $u'\,[\mathsf{u}']$ with $\mathsf{u}'\not\subseteq\mathsf{u}$ or an empty path, followed by a π path from u to t without intermediate states $u'\,[\mathsf{u}']$ with $\mathsf{u}'\not\subseteq\mathsf{u}$.

By the inductive hypothesis, this first requirement can be met iff

$$\begin{aligned} & sR_{(K^{\mathsf{U}}_{\mathsf{su}(\mathsf{u}-\{\mathsf{e}\})}(\pi)^*\cup?\top);?\bigvee_{\mathsf{e}\in\mathsf{u}}\mathsf{pre}(\mathsf{e})}u \text{ for some } \mathsf{e}\in\mathsf{u} \\ \iff & sR_{\bigcup_{\mathsf{e}\in\mathsf{u}}\left(K^{\mathsf{U}}_{\mathsf{su}(\mathsf{u}-\{\mathsf{e}\})}(\pi)^*\cup?\top\right);\bigvee_{\mathsf{e}\in\mathsf{u}}\mathsf{pre}(\mathsf{e})}u \end{aligned}$$

The second parts holds iff

$$\begin{aligned} & uR_{K^{\mathsf{U}}_{\mathsf{ut}(\mathsf{u}-\{\mathsf{e}\})}(\pi);?\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre}(\mathsf{e})}t \text{ for some } \mathsf{e}\in\mathsf{u} \\ \iff & uR_{\bigcup_{\mathsf{e}\in\mathsf{u}}K^{\mathsf{U}}_{\mathsf{ut}(\mathsf{u}-\{\mathsf{e}\})}(\pi);\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre}(\mathsf{e})}t \end{aligned}$$

Putting these together we get

$$\begin{aligned} & s\left(R_{\bigcup_{\mathsf{e}\in\mathsf{u}}\left(K^{\mathsf{U}}_{\mathsf{su}(\mathsf{u}-\{\mathsf{e}\})}(\pi)^*\cup?\top\right);?\bigvee_{\mathsf{e}\in\mathsf{u}}\mathsf{pre}(\mathsf{e})}\circ R_{\bigcup_{\mathsf{e}\in\mathsf{u}}K^{\mathsf{U}}_{\mathsf{ut}(\mathsf{u}-\{\mathsf{e}\})}(\pi);?\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre}(\mathsf{e})}\right)t \\ \iff & sR_{\bigcup\mathsf{e}\in\mathsf{u}\left(K^{\mathsf{U}}_{\mathsf{su}(\mathsf{u}-\{\mathsf{e}\})}(\pi)^*\cup?\top\right);\bigcup_{\mathsf{e}\in\mathsf{u}}K^{\mathsf{U}}_{\mathsf{ut}(\mathsf{u}-\{\mathsf{e}\})}(\pi);?\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre}(\mathsf{e})}t \\ \iff & sR_{K^{\mathsf{U}}_{\mathsf{stu}}(\pi);?\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre}(\mathsf{e})}t \end{aligned}$$

Case $\mathsf{s} \neq \mathsf{u} = \mathsf{t}$: A π path from $s\,[\mathsf{s}]$ to $t\,[\mathsf{t}]$ in $M \circ \mathsf{U}$ that does not have intermediate states $u'\,[\mathsf{u}']$ with $\mathsf{u}' \not\subseteq \mathsf{u}$ now consists of a π path from $s\,[\mathsf{s}]$ to $u\,[\mathsf{u}]$ that does not have intermediate states $u'\,[\mathsf{u}']$ with $\mathsf{u}' \not\subseteq \mathsf{u}$ and either an arbitrary composition of π paths from $u\,[\mathsf{u}]$ to $t\,[\mathsf{t}]$ without intermediate states $u'\,[\mathsf{u}']$ with $\mathsf{u}' \not\subseteq \mathsf{u}$ or an empty path.

By the inductive hypothesis, this first requirement can be met iff

$$\begin{aligned} & sR_{K^{\mathsf{U}}_{\mathsf{su(u-\{e\})}}(\pi);?\bigvee_{\mathsf{e}\in\mathsf{u}}\mathsf{pre(e)}}u \text{ for some } \mathsf{e}\in\mathsf{u} \\ \iff & sR_{\bigcup_{\mathsf{e}\in\mathsf{u}} K^{\mathsf{U}}_{\mathsf{su(u-\{e\})}}(\pi);?\bigvee_{\mathsf{e}\in\mathsf{u}}\mathsf{pre(e)}}u \end{aligned}$$

The second step can be done iff

$$\begin{aligned} & uR_{K^{\mathsf{U}}_{\mathsf{ut(u-\{e\})}}(\pi)^*\cup?\top;\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre(e)}}t \text{ for some } \mathsf{e}\in\mathsf{u} \\ \iff & uR_{\bigcup_{\mathsf{e}\in\mathsf{u}}\left(K^{\mathsf{U}}_{\mathsf{ut(u-\{e\})}}(\pi)^*\cup?\top\right);?\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre(e)}}t \end{aligned}$$

Putting these together we get

$$\begin{aligned} & s\left(R_{\bigcup_{\mathsf{e}\in\mathsf{u}} K^{\mathsf{U}}_{\mathsf{su(u-\{e\})}}(\pi);?\bigvee_{\mathsf{e}\in\mathsf{u}}\mathsf{pre(e)}} \circ R_{\bigcup_{\mathsf{e}\in\mathsf{u}}\left(K^{\mathsf{U}}_{\mathsf{ut(u-\{e\})}}(\pi)^*\cup?\top\right);?\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre(e)}}\right) t \\ \iff & sR_{\left(\bigcup_{\mathsf{e}\in\mathsf{u}} K^{\mathsf{U}}_{\mathsf{su(u-\{e\})}}(\pi);\bigcup_{\mathsf{e}\in\mathsf{u}}\left(K^{\mathsf{U}}_{\mathsf{ut(u-\{e\})}}(\pi)^*\cup?\top\right);?\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre(e)}\right)}t \\ \iff & sR_{K^{\mathsf{U}}_{\mathsf{stu}}(\pi);?\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre(e)}}t. \end{aligned}$$

Case $\mathsf{s} \neq \mathsf{u} \neq \mathsf{t}$: A π path from $s\,[\mathsf{s}]$ to $t\,[\mathsf{t}]$ in $M \circ \mathsf{U}$ that does not have intermediate states $u'\,[\mathsf{u}']$ with $\mathsf{u}' \not\subseteq \mathsf{u}$ now consists of a π path from s to t, or a π path with intermediate steps. This inner π path would consist of a π path from s to u, an arbitrary composition of π paths from $u\,[\mathsf{u}]$ to $u\,[\mathsf{u}]$ or an empty path, and a π path from $u\,[\mathsf{u}]$ to $t\,[\mathsf{t}]$, all without intermediate states $u'\,[\mathsf{u}']$ with $\mathsf{u}' \not\subseteq \mathsf{u}$.

This first requirement can be met, by the inductive hypothesis, iff

$$\begin{aligned} & sR_{K^{\mathsf{U}}_{\mathsf{st(u-\{e\})}}(\pi);?\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre(e)}}t \text{ for some } \mathsf{e}\in\mathsf{u} \\ \iff & sR_{\bigcup_{\mathsf{e}\in\mathsf{u}} K^{\mathsf{U}}_{\mathsf{st(u-\{e\})}}(\pi);?\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre(e)}}t \end{aligned}$$

The first step of the inner path exists, by the inductive hypothesis, iff

$$\begin{aligned} & sR_{K^{\mathsf{U}}_{\mathsf{su(u-\{e\{)}}}(\pi);?\bigvee_{\mathsf{e}\in\mathsf{u}}\mathsf{pre(e)}}u \text{ for some } \mathsf{e}\in\mathsf{u} \\ \iff & sR_{\bigcup_{\mathsf{e}\in\mathsf{u}} K^{\mathsf{U}}_{\mathsf{su(u-\{e\})}}(\pi);?\bigvee_{\mathsf{e}\in\mathsf{u}}\mathsf{pre(e)}}u \end{aligned}$$

By the inductive hypothesis, the second exists iff

$$\begin{aligned} & uR_{K^{\mathsf{U}}_{\mathsf{uu(u-\{e\})}}(\pi)^*\cup?\top;?\bigvee_{\mathsf{e}\in\mathsf{u}}\mathsf{pre(e)}}u \text{ for some } \mathsf{e}\in\mathsf{u} \\ \iff & uR_{\bigcup_{\mathsf{e}\in\mathsf{u}}\left(K^{\mathsf{U}}_{\mathsf{uu(u-\{e\})}}(\pi)^*\cup?\top\right);?\bigvee_{\mathsf{e}\in\mathsf{u}}\mathsf{pre(e)}}u \end{aligned}$$

The last step of the inner path exists, by the inductive hypothesis iff

$$\begin{aligned} & uR_{K^{\mathsf{U}}_{\mathsf{ut(u-\{e\})}}(\pi);?\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre(e)}}t \text{ for some } \mathsf{e}\in\mathsf{u} \\ \iff & uR_{\bigcup_{\mathsf{e}\in\mathsf{u}} K^{\mathsf{U}}_{\mathsf{ut(u-\{e\})}}(\pi);?\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre(e)}}t \end{aligned}$$

Putting it all together we get

$$\begin{aligned}
&s\left(R_{\bigcup_{\mathsf{e}\in\mathsf{u}} K^{\mathsf{U}}_{\mathsf{st}(\mathsf{u}-\{\mathsf{e}\})}(\pi);?\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre}(\mathsf{e})} \cup \left(R_{\bigcup_{\mathsf{e}\in\mathsf{u}} K^{\mathsf{U}}_{\mathsf{su}(\mathsf{u}-\{\mathsf{e}\})}(\pi);?\bigvee_{\mathsf{e}\in\mathsf{u}}\mathsf{pre}(\mathsf{e})} \circ \right.\right.\\
&\qquad \left.\left. R_{\bigcup_{\mathsf{e}\in\mathsf{u}}\left(K^{\mathsf{U}}_{\mathsf{uu}(\mathsf{u}-\{\mathsf{e}\})}(\pi)^*\cup ?\top\right);?\bigvee_{\mathsf{e}\in\mathsf{u}}\mathsf{pre}(\mathsf{e})} \circ R_{\bigcup_{\mathsf{e}\in\mathsf{u}} K^{\mathsf{U}}_{\mathsf{ut}(\mathsf{u}-\{\mathsf{e}\})}(\pi);?\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre}(\mathsf{e})}\right)\right) t\\
\iff & s\left(R_{\bigcup_{\mathsf{e}\in\mathsf{u}} K^{\mathsf{U}}_{\mathsf{st}(\mathsf{u}-\{\mathsf{e}\})}(\pi);?\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre}(\mathsf{e})} \cup \right.\\
&\qquad \left. R_{\bigcup_{\mathsf{e}\in\mathsf{u}} K^{\mathsf{U}}_{\mathsf{su}(\mathsf{u}-\{\mathsf{e}\})}(\pi);\bigcup_{\mathsf{e}\in\mathsf{u}}\left(K^{\mathsf{U}}_{\mathsf{uu}(\mathsf{u}-\{\mathsf{e}\})}(\pi)^*\cup ?\top\right);\bigcup_{\mathsf{e}\in\mathsf{u}} K^{\mathsf{U}}_{\mathsf{ut}(\mathsf{u}-\{\mathsf{e}\})}(\pi);?\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre}(\mathsf{e})}\right) t\\
\iff & sR_{\left(\bigcup_{\mathsf{e}\in\mathsf{u}} K^{\mathsf{U}}_{\mathsf{st}(\mathsf{u}-\{\mathsf{e}\})}(\pi)\cup\left(\bigcup_{\mathsf{e}\in\mathsf{u}} K^{\mathsf{U}}_{\mathsf{su}(\mathsf{u}-\{\mathsf{e}\})}(\pi);\bigcup_{\mathsf{e}\in\mathsf{u}}\left(K^{\mathsf{U}}_{\mathsf{uu}(\mathsf{u}-\{\mathsf{e}\})}(\pi)^*\cup ?\top\right);\bigcup_{\mathsf{e}\in\mathsf{u}} K^{\mathsf{U}}_{\mathsf{ut}(\mathsf{u}-\{\mathsf{e}\})}(\pi)\right);?\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre}(\mathsf{e})\right)} t\\
\iff & sR_{K^{\mathsf{U}}_{\mathsf{stu}}(\pi);?\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre}(\mathsf{e})} t
\end{aligned}$$

This concludes the proof. □

Lemma 4.20. *Suppose* $sR_{T^{\mathsf{U}}_{\mathsf{st}}(\pi);?\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre}(\mathsf{e})}t$ *iff there is a* π *step from* $s\,[\mathsf{s}]$ *to* $t\,[\mathsf{t}]$ *in* $M\circ\mathsf{U}$*. Then* $sR_{K^{\mathsf{U}}_{\mathsf{stu}}(\pi);?\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre}(\mathsf{e})}t$ *iff there is a* π *path from* $s\,[\mathsf{s}]$ *to* $t\,[\mathsf{t}]$ *in* $M\circ\mathsf{U}$*.*

Proof. Suppose that $sR_{T^{\mathsf{U}}_{\mathsf{st}}(\pi);?\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre}(\mathsf{e})}t$ iff there is a π path from $s\,[\mathsf{s}]$ to $t\,[\mathsf{t}]$ in $M\circ\mathsf{U}$. Then, assuming that $\mathsf{u}=\mathsf{E}$, an application of Lemma 4.19 yields that $K^{\mathsf{U}}_{\mathsf{stu}}(\pi)$ is a program for all the π paths from $s\,[\mathsf{s}]$ to $t\,[\mathsf{t}]$ that can be traced through $M\circ\mathsf{U}$, without stopovers at any $u'\,[\mathsf{u}']$ with $\mathsf{u}'\not\subseteq\mathsf{u}$. Therefore, $sR_{K^{\mathsf{U}}_{\mathsf{stu}}(\pi);?\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre}(\mathsf{e})}t$ iff there is a π path from $s\,[\mathsf{s}]$ to $t\,[\mathsf{t}]$ in $M\circ\mathsf{U}$. □

Now that we have these auxiliary results, we can go back to the matter at hand.

Theorem 4.18 (Program transformation into IE-PDL.)**.** *For all update models* U *and all IE-PDL programs* π*,*

$$sR_{T^{\mathsf{U}}_{\mathsf{st}}(\pi);?\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre}(\mathsf{e})}t \iff s\,[\mathsf{s}]\,R'_{\pi}t\,[\mathsf{t}]$$

Proof. The proof works by induction over the structure of programs π.

Base case a: Assume that $sR_{T^{\mathsf{U}}_{\mathsf{st}}(a);?\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre}(\mathsf{e})}t$. This then means that there is some $s'\subseteq s$ such that $M,s'\models\bigvee_{\mathsf{e}\in\mathsf{s}}\mathsf{pre}(\mathsf{e})$, some $\mathsf{s}'\subseteq\mathsf{s}$ such that $M,s'\models\bigwedge_{\mathsf{e}\in(\mathsf{s}-\mathsf{s}')}\neg\mathsf{pre}(\mathsf{e})$, that $M,t\models\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre}(\mathsf{e})$, that there is a $\mathsf{t}'\subseteq\mathsf{t}$ such that $M,t\models\bigwedge_{\mathsf{e}\in(\mathsf{t}-\mathsf{t}')}\neg\mathsf{pre}(\mathsf{e})$, and that $s'R_a t$ and $\mathsf{t}'\in\bigcup_{\mathsf{e}\in\mathsf{s}'}\Delta_a(\mathsf{e})$. By Definition 4.3 this is the case iff $s\,[\mathsf{s}]\,R'_a t\,[\mathsf{t}]$, as required.

Base Case $?\varphi$*, subcase* $\mathsf{t}\subseteq\mathsf{s}$*:*

$$\begin{aligned}
s\,[\mathsf{s}]\,R'_{?\varphi}t\,[\mathsf{t}] &\iff t\,[\mathsf{t}]\subseteq s\,[\mathsf{s}] \text{ and } M\circ\mathsf{U},t\,[\mathsf{t}]\models\varphi\\
&\iff t\subseteq s \text{ and } M,t\models[\mathsf{U},\mathsf{s}]\,\varphi\\
&\iff sR_{?[\mathsf{U},\mathsf{s}]\varphi;?\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre}(\mathsf{e})}t\\
&\iff sR_{T^{\mathsf{U}}_{\mathsf{st}}(?\varphi);?\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre}(\mathsf{e})}t
\end{aligned}$$

Base Case $?\varphi$*, subcase* $\mathsf{t}\not\subseteq\mathsf{s}$*:*

$$\begin{aligned}
s\,[\mathsf{s}]\,R'_{?\varphi}t\,[\mathsf{t}] &\iff sR_{?\bot}t\\
&\iff sR_{T^{\mathsf{U}}_{\mathsf{st}}(?\varphi);?\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre}(\mathsf{e})}t
\end{aligned}$$

Inductive Step: Now take two arbitrary programs π_1 and π_2 and assume that

$$sR_{T^{\mathsf{U}}_{\mathsf{st}}(\pi_1);?\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre(e)}}t \iff s\,[\mathsf{s}]\,R'_{\pi_1}t\,[\mathsf{t}] \text{ and } sR_{T^{\mathsf{U}}_{\mathsf{st}}(\pi_2);?\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre(e)}}t \iff s\,[\mathsf{s}]\,R'_{\pi_2}t\,[\mathsf{t}]$$

Case $\pi_1;\pi_2$:

$$\begin{aligned}
s\,[\mathsf{s}]\,R'_{(\pi_1;\pi_2)}t\,[\mathsf{t}] &\iff \text{for some } u\,[\mathsf{u}]\subseteq\mathcal{W}', s\,[\mathsf{s}]\,R'_{\pi_1}u\,[\mathsf{u}] \text{ and } u\,[\mathsf{u}]\,R'_{\pi_2}t\,[\mathsf{t}]\\
&\iff \text{for some } u\,[\mathsf{u}]\subseteq\mathcal{W}', sR_{T^{\mathsf{U}}_{\mathsf{su}}(\pi_1);?\bigvee_{\mathsf{e}\in\mathsf{u}}\mathsf{pre(e)}}u \text{ and } uR_{T^{\mathsf{U}}_{\mathsf{ut}}(\pi_2);?\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre(e)}}t\\
&\iff sR_{\bigcup_{\mathsf{u}\subseteq\mathsf{E}}\left(T^{\mathsf{U}}_{\mathsf{su}}(\pi_1);T^{\mathsf{U}}_{\mathsf{ut}}(\pi_2)\right);?\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre(e)}}t\\
&\iff sR_{T^{\mathsf{U}}_{\mathsf{st}}(\pi_1;\pi_2);?\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre(e)}}t
\end{aligned}$$

Case $\pi_1\cup\pi_2$:

$$\begin{aligned}
s\,[\mathsf{s}]\,R'_{(\pi_1\cup\pi_2)}t\,[\mathsf{t}] &\iff s\,[\mathsf{s}]\,R'_{\pi_1}t\,[\mathsf{t}] \text{ or } s\,[\mathsf{s}]\,R'_{\pi_2}t\,[\mathsf{t}]\\
&\iff sR_{T^{\mathsf{U}}_{\mathsf{st}}(\pi_1);?\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre(e)}}t \text{ or } sR_{T^{\mathsf{U}}_{\mathsf{st}}(\pi_2);?\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre(e)}}t\\
&\iff sR_{\left(T^{\mathsf{U}}_{\mathsf{st}}(\pi_1)\cup T^{\mathsf{U}}_{\mathsf{st}}(\pi_2)\right);?\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre(e)}}t\\
&\iff sR_{T^{\mathsf{U}}_{\mathsf{st}}(\pi_1\cup\pi_2);?\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre(e)}}t
\end{aligned}$$

Case ${\pi_1}^*$:

$$\begin{aligned}
s\,[\mathsf{s}]\,R'_{{\pi_1}^*}t\,[\mathsf{t}] &\iff \text{there is a } \pi_1 \text{ path from } s\,[\mathsf{s}] \text{ to } t\,[\mathsf{t}] \text{ in } M\circ\mathsf{U}\\
&\iff sR_{K^{\mathsf{U}}_{\mathsf{stE}}(\pi_1);?\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre(e)}}t\\
&\iff sR_{T^{\mathsf{U}}_{\mathsf{st}}({\pi_1}^*);?\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre(e)}}t
\end{aligned}$$

□

Using this theorem, we can give the reductions for the modal operators, starting with $[a]\varphi$. Here, we run into the same problem as in [22] Proposition 14. However, since we are working with more complex epistemic programs, we cannot use the solution employed there, which means that we will go with the other proposed solution, which is quantifying over the resolutions.

An additional "problem" here is that the reduction can only be given for single events. However, since both formulas are declarative, we can use Proposition 4.7 to still give a reduction for formulas where there is more than one event.

Proposition 4.21. *For any IE-PDL model M and any state s in M, $M,s\models[\mathsf{U},\mathsf{e}]\,[\pi]\varphi \iff M,s\models\bigvee_{\alpha\in\mathcal{R}_{|\mathcal{W}|}(\varphi)}\left(\bigwedge_{\mathsf{t}\subseteq\mathsf{E}}[T^{\mathsf{U}}_{\{\mathsf{e}\}\mathsf{t}}(\pi)]\,[\mathsf{t}]\,\alpha\right)$.*

Proof. Since both formulas are declarative, we will show that it works for worlds.

$$\begin{aligned}
M,w\models[\mathsf{U},\mathsf{e}]\,[\pi]\varphi &\iff M\circ\mathsf{U},w\,[\mathsf{e}]\models[\pi]\varphi\\
&\iff M,w\models\mathsf{pre(e)} \text{ implies } M\circ\mathsf{U},\bigcup\{t\,[\mathsf{t}]\mid\{(w,\mathsf{e})\}R'_{\pi}t\,[\mathsf{t}]\}\models\varphi\\
&\iff M\circ\mathsf{U},\bigcup\left\{t\,[\mathsf{t}]\;\middle|\;\mathsf{t}\subseteq\mathsf{E},\{w\}R_{T^{\mathsf{U}}_{\{\mathsf{e}\}\mathsf{t}}(\pi);?\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre(e)}}t\right\}\models\varphi\\
&\iff M\circ\mathsf{U},\bigcup\left\{t\,[\mathsf{t}]\;\middle|\;\mathsf{t}\subseteq\mathsf{E},\{w\}R_{T^{\mathsf{U}}_{\{\mathsf{e}\}\mathsf{t}}(\pi);?\bigvee_{\mathsf{e}\in\mathsf{t}}\mathsf{pre(e)}}t\right\}\models\bigvee\!\!\!\bigvee\mathcal{R}_{|\mathcal{W}|}(\varphi)
\end{aligned}$$

$$\begin{aligned}
&\iff \text{for some } \alpha \in \mathcal{R}_{|\mathcal{W}|}(\varphi),\\
&\quad M \circ \mathsf{U}, \bigcup \left\{ t\,[\mathsf{t}] \;\middle|\; \mathsf{t} \subseteq \mathsf{E}, \{w\} R_{T^{\mathsf{U}}_{\{\mathsf{e}\}\mathsf{t}}(\pi);?\bigvee_{\mathsf{e}\in\mathsf{t}} \mathsf{pre}(\mathsf{e})} t \right\} \models \alpha\\
&\iff \text{for some } \alpha \in \mathcal{R}_{|\mathcal{W}|}(\varphi),\ \text{for all } (v, \mathsf{f}) \in\\
&\quad \bigcup \left\{ t\,[\mathsf{t}] \;\middle|\; \mathsf{t} \subseteq \mathsf{E}, \{w\} R_{T^{\mathsf{U}}_{\{\mathsf{e}\}\mathsf{t}}(\pi);?\bigvee_{\mathsf{e}\in\mathsf{t}} \mathsf{pre}(\mathsf{e})} t \right\}, M \circ \mathsf{U}, (v, \mathsf{f}) \models \alpha\\
&\iff \text{for some } \alpha \in \mathcal{R}_{|\mathcal{W}|}(\varphi),\ \text{for all } \mathsf{t} \subseteq \mathsf{E},\\
&\quad \text{for all } \{w\} R_{T^{\mathsf{U}}_{\{\mathsf{e}\}\mathsf{t}}(\pi);?\bigvee_{\mathsf{e}\in\mathsf{t}} \mathsf{pre}(\mathsf{e})} t,\ \text{for all } (v, \mathsf{f}) \in t\,[\mathsf{t}]\,, M \circ \mathsf{U}, (v, \mathsf{f}) \models \alpha\\
&\iff \text{for some } \alpha \in \mathcal{R}_{|\mathcal{W}|}(\varphi),\ \text{for all } \mathsf{t} \subseteq \mathsf{E},\\
&\quad \text{for all } \{w\} R_{T^{\mathsf{U}}_{\{\mathsf{e}\}\mathsf{t}}(\pi);?\bigvee_{\mathsf{e}\in\mathsf{t}} \mathsf{pre}(\mathsf{e})} t, M \circ \mathsf{U}, t\,[\mathsf{t}] \models \alpha\\
&\iff \text{for some } \alpha \in \mathcal{R}_{|\mathcal{W}|}(\varphi),\ \text{for all } \mathsf{t} \subseteq \mathsf{E},\\
&\quad \text{for all } \{w\} R_{T^{\mathsf{U}}_{\{\mathsf{e}\}\mathsf{t}}(\pi);?\bigvee_{\mathsf{e}\in\mathsf{t}} \mathsf{pre}(\mathsf{e})} t, M, t \models [\mathsf{t}]\,\alpha\\
&\iff \text{for some } \alpha \in \mathcal{R}_{|\mathcal{W}|}(\varphi),\ \text{for all } \mathsf{t} \subseteq \mathsf{E}, M, s \models [T^{\mathsf{U}}_{\{\mathsf{e}\}\mathsf{t}}(\pi)]\,[\mathsf{t}]\,\alpha^{1}\\
&\iff \text{for some } \alpha \in \mathcal{R}_{|\mathcal{W}|}(\varphi), M, w \models \bigwedge_{\mathsf{t}\subseteq\mathsf{E}} [T^{\mathsf{U}}_{\{\mathsf{e}\}\mathsf{t}}(\pi)]\,[\mathsf{t}]\,\alpha\\
&\iff M, w \models \bigvee_{\alpha\in\mathcal{R}_{|\mathcal{W}|}(\varphi)} \left(\bigwedge_{\mathsf{t}\subseteq\mathsf{E}} [T^{\mathsf{U}}_{\{\mathsf{e}\}\mathsf{t}}(\pi)]\,[\mathsf{t}]\,\alpha \right)
\end{aligned}$$

□

Proposition 4.22. *For any* IE-PDL *model M and any state s in M, $M, s \models [\mathsf{U}, \mathsf{s}]\,[\![\pi]\!]\varphi \iff M, s \models \bigwedge_{\mathsf{t}\subseteq\mathsf{E}} [\![T^{\mathsf{U}}_{\mathsf{st}}(\pi)]\!]\,[\mathsf{t}]\,\varphi$.*

Proof.

$$\begin{aligned}
M, s \models [\mathsf{U}, \mathsf{s}]\,[\![\pi]\!]\varphi &\iff M \circ \mathsf{U}, s\,[\mathsf{s}] \models [\![\pi]\!]\varphi\\
&\iff \text{for all } s\,[\mathsf{s}]\,R'_{\pi} t\,[\mathsf{t}]\,, M \circ \mathsf{U}, t\,[\mathsf{t}] \models \varphi\\
&\iff \text{for all } \mathsf{t} \subseteq \mathsf{E},\ \text{if } sR_{T^{\mathsf{U}}_{\mathsf{st}}(\pi)} t \text{ then } M, t \models [\mathsf{U}, \mathsf{t}]\,\varphi\\
&\iff M, s \models \bigwedge_{\mathsf{t}\subseteq\mathsf{E}} [\![T^{\mathsf{U}}_{\mathsf{st}}(\pi)]\!]\,[\mathsf{U}, \mathsf{t}]\,\varphi
\end{aligned}$$

□

Using these two axioms, we can push the Action operator through any modal operator, which will allow us to fully reduce any formula to an IE-PDL formula.

4.5.5 Full reduction

As in AMLI, we can now use these equivalences to reduce a formula of LCCI into an equivalent IE-PDL formula. Here we also inherit the problem from AMLI that $[\mathsf{s}]\,(\varphi \to \psi)$ can only be reduced if $\varphi \to \psi$ is declarative. Luckily, the solution used there translates pretty well to our setting.

The solution as used in [22] is as follows. They define a complexity measure over formulas, since they cannot do straightforward induction on the structure of formulas. This is the case since the resolutions of $\varphi \to \psi$ and $[\![\pi^*]\!]\varphi$ are more complex then the formulas themselves. This complexity measure c has to have the following properties:

- If ψ is a proper subformula of φ, then $\mathsf{c}(\varphi) > \mathsf{c}(\psi)$;
- If φ is not a declarative, then $\alpha \in \mathcal{R}_n(\varphi)$ implies $\mathsf{c}(\varphi) > \mathsf{c}(\alpha)$.

After this measure is defined, it is shown by induction over the structure of formulas, using c, that every AMLI formula can be translated into an IEL formula.

It is fairly easy to adapt this procedure for LCCI, but we have to keep in mind that for LCCI the resolutions are defined for a maximum number of worlds, so we will also have to change the complexity measure to account for that. Because of this, we will have to also take the maximum number of worlds into account for the complexity measure.

This means that for LCCI we get that c should have the following properties:

1. If ψ is a proper subformula of φ, then $\mathsf{c}_n(\varphi) > \mathsf{c}_n(\psi)$, for all $n \in \mathbb{N}$.
2. If φ is not a declarative, then $\alpha \in \mathcal{R}_n(\varphi)$ implies $\mathsf{c}_n(\varphi) > \mathsf{c}_n(\alpha)$, for all $n \in \mathbb{N}$.

A complexity measure that has both of these properties can be made by adapting and extending the complexity measure from [22]:

Definition 4.10 (Complexity for LCCI formulas)**.** The complexity for an LCCI formula φ with respect to models with at most n worlds, $\mathsf{c}_n(\varphi)$ is defined as:

$$\begin{aligned}
\mathsf{c}_n(p) &= 1 \\
\mathsf{c}_n(\bot) &= 1 \\
\mathsf{c}_n(\psi_1 \wedge \psi_2) &= 1 + \max(\mathsf{c}_n(\psi_1), \mathsf{c}_n(\psi_2)) \\
\mathsf{c}_n(\psi_1 \vee\!\!\!\vee \psi_2) &= 1 + \max(\mathsf{c}_n(\psi_1), \mathsf{c}_n(\psi_2)) \\
\mathsf{c}_n(\psi_1 \to \psi_2) &= \begin{cases} 1 + \max(\mathsf{c}_n(\psi_1), \mathsf{c}_n(\psi_2)) & \text{if } \varphi \text{ is declarative} \\ 1 + |\mathcal{R}_n(\psi_1)| + \max(\mathsf{c}_n(\psi_1), \mathsf{c}_n(\psi_2)) & \text{otherwise.} \end{cases} \\
\mathsf{c}_n([\pi]\psi_1) &= 1 + \mathsf{c}_n(\psi_1) \\
\mathsf{c}_n([\![\pi]\!]\psi_1) &= \begin{cases} 1 + \mathsf{c}_n(\psi_1) & \text{if } [\![\pi]\!]\psi_1 \text{ is declarative} \\ \mathsf{c}_n(\psi_2 \to \psi_1) & \text{if } \pi = ?\psi_2 \\ \mathsf{c}_n([\![\pi_1]\!][\![\pi_2]\!]\psi_1) & \text{if } \pi = \pi_1;\pi_2 \\ 1 + \max(\mathsf{c}_n([\![\pi_1]\!]\psi_1), \mathsf{c}_n([\![\pi_2]\!]\psi_1)) & \text{if } \pi = \pi_1 \cup \pi_2 \\ 1 + 2^n + \mathsf{c}_n([\![\pi_1^{2^n}]\!]\psi_1) & \text{if } \pi = \pi_1^* \end{cases} \\
\mathsf{c}_n([\mathsf{s}]\,\psi_1) &= 1 + \mathsf{c}_n(\psi_1)
\end{aligned}$$

This definition has property 1 by construction, but we will have to show that the second property holds as well.

Proposition 4.23. *For all $n \in \mathbb{N}$, if φ is not a declarative, then $\alpha \in \mathcal{R}_n(\varphi)$ implies $\mathsf{c}_n(\varphi) > \mathsf{c}_n(\alpha)$.*

Proof. For this proof we fix some arbitrary $n \in \mathbb{N}$. The steps for all but the modal operators are as in [22], and the step for $[\pi]\psi$ is trivial. Therefore, we only give the steps for $[\![\pi]\!]\psi$. The step for $[\![?\chi]\!]\psi$ goes via the step for implication, the step for $[\![\pi_1 \cup \pi_2]\!]\psi$ is the same as the one for conjunction, and the one for $[\![\pi_1;\pi_2]\!]\psi$ follows directly from the inductive hypothesis. That leaves the step for $[\![\pi^*]\!]\psi$.

Take an arbitrary $\alpha \in \mathcal{R}_n([\![\pi^*]\!]\psi)$. By the definition of resolutions, this will be a formula of the form $\bigwedge_{1\leq m\leq 2^n}\alpha_m$ where $\alpha_m \in \mathcal{R}_n([\![\pi^m]\!]\psi)$. By the definition of c, its complexity is at most the number of conjuncts minus 1, plus the complexity of the most complex conjunct, which by definition is α_{2^n}. This makes $\mathsf{c}_n(\alpha) = 2^n + \mathsf{c}_n(\alpha_{2^n})$.

By the inductive hypothesis we get that if ψ is declarative, $\mathsf{c}_n(\psi) > \mathsf{c}_n(\beta)$ for all $\beta \in \mathcal{R}_n(\psi)$. Then by the step for sequence we get that $\mathsf{c}_n([\![\pi^{2^n}]\!]\psi) > \mathsf{c}_n(\beta)$ for all $\beta \in \mathcal{R}_n([\![\pi^{2^n}]\!]\psi)$. Therefore, the complexity of our arbitrary resolution α cannot be bigger than $2^n + \mathsf{c}_n([\![\pi^{2^n}]\!]\psi)$, which is in turn smaller than $\mathsf{c}_n([\![\pi^*]\!]\psi)$ by definition. Thus if $[\![\pi^*]\!]\psi$ is not a declarative, then $\alpha \in \mathcal{R}_n([\![\pi^*]\!]\psi)$ implies $\mathsf{c}_n([\![\pi^*]\!]\psi) > \mathsf{c}_n(\alpha)$.

Since we took some arbitrary $n \in \mathbb{N}$, this holds for all $n \in \mathbb{N}$. □

Now that we have our complexity measure, we can show that every LCCI formula can be reduced into an equivalent IE-PDL formula.

Theorem 4.24. *For every LCCI formula φ there is an equivalent IE-PDL formula φ' such that for all IE-PDL models M and state s:*

$$M, s \models \varphi \iff M, s \models \varphi'$$

Proof. This proof works by induction on the structure of formulas. All steps are trivial, except for the step for $[\mathsf{s}]\,\psi$. By the inductive hypothesis there is an IE-PDL formula ψ' such that for some arbitrary IE-PDL model M and state s: $M, s \models \psi \iff M, s \models \psi'$.

The case for $[\mathsf{s}]\,\psi'$ now proceeds by induction on $\mathsf{c}_{|\mathcal{W}|}(\psi')$.

- p: In this case we can apply Corollary 4.12. Since $\mathsf{pre}(\mathsf{e})$ and $\mathsf{sub}(\mathsf{e})(p)$ are IE-PDL formulas by definition, $\bigwedge_{\mathsf{e}\in\mathsf{s}}(\mathsf{pre}(\mathsf{e}) \rightarrow p^{\mathsf{sub}(\mathsf{f})})$ is an IE-PDL formula as well.

- $\bot$: In this case we can apply Corollary 4.14 with a similar reasoning as for p.

For the inductive hypothesis: for any formula χ less complex than ψ' and all sets of events t, there is some IE-PDL formula χ' such that $M, s \models [\mathsf{t}]\,\chi \iff M, s \models \chi'$.

- $\psi' = \chi_1 \wedge \chi_2$: Now by Proposition 4.15, $M, s \models [\mathsf{s}]\,\psi' \iff M, s \models [\mathsf{s}]\,\chi_1 \wedge [\mathsf{s}]\,\chi_2$. Then by the inductive hypothesis, we get χ_1' and χ_2' such that $M, s \models \psi' \iff M, s \models \chi_1' \wedge \chi_2'$, where $\chi_1' \wedge \chi_2'$ is an IE-PDL formula.

- $\psi' = \chi_1 \vee\!\!\!\vee\, \chi_2$: The same as the step for conjunction, but using Proposition 4.16.

- $\psi' = \chi_1 \rightarrow \chi_2$: Here there are two cases, either $\chi_1 \rightarrow \chi_2$ is declarative, or it is not. Lets first consider the former. Then by Propositions 4.7 and 4.17 $M, s \models [\mathsf{s}]\,(\chi_1 \rightarrow \chi_2) \iff M, s \models \bigwedge_{\mathsf{e}\in\mathsf{s}}([\mathsf{e}]\,\chi_1 \rightarrow [\mathsf{e}]\,\chi_2)$. Now take any $\mathsf{e} \in \mathsf{s}$. By the inductive hypothesis, there are two IE-PDL formulas χ_1^{e} and χ_2^{e} such that $M, s \models [\mathsf{e}]\,\chi_1 \rightarrow [\mathsf{e}]\,\chi_2 \iff M, s \models \chi_1^{\mathsf{e}} \rightarrow \chi_2^{\mathsf{e}}$. Since these formula exists for each $\mathsf{e} \in \mathsf{s}$, we get that $M, s \models [\mathsf{s}]\,(\chi_1 \rightarrow \chi_2) \iff M, s \models \bigwedge_{\mathsf{e}\in\mathsf{s}}(\chi_1^{\mathsf{e}} \rightarrow \chi_2^{\mathsf{e}})$, where $\bigwedge_{\mathsf{e}\in\mathsf{s}}(\chi_1^{\mathsf{e}} \rightarrow \chi_2^{\mathsf{e}})$ is an IE-PDL formula.

  Now suppose that $\chi_1 \rightarrow \chi_2$ is not declarative. In this case, we cannot apply Proposition 4.7, so we will have to use a different method. For this different method, we can use the normal form of a formula. Take any $\alpha \in \mathcal{R}_{|\mathcal{W}|}([\mathsf{s}]\,\psi')$. This α will always be of the form $[\mathsf{s}]\,\beta$ where $\beta \in \mathcal{R}_{|\mathcal{W}|}(\psi')$. Then by the inductive hypothesis, we will have some IE-PDL formula α' such that $M, s \models [\mathsf{s}]\,\beta \iff M, s \models \alpha'$. Therefore, $M, s \models \chi_1 \rightarrow \chi_2 \iff M, s \models \mathbb{W}_{\alpha\in\mathcal{R}_{|\mathcal{W}|}([\mathsf{s}](\chi_1\rightarrow\chi_2))}\alpha'$, where $\mathbb{W}_{\alpha\in\mathcal{R}_{|\mathcal{W}|}([\mathsf{s}](\chi_1\rightarrow\chi_2))}\alpha'$ is an IE-PDL formula.

!Proposition
$$\frac{[\mathsf{e}]\,p}{\mathsf{pre}(\mathsf{e}) \to p^{\mathsf{sub}(\mathsf{e})}}$$

$!\wedge$
$$\frac{[\mathsf{s}]\,(\varphi \wedge \psi)}{[\mathsf{s}]\,\varphi \wedge [\mathsf{s}]\,\psi}$$

$![\pi]$
$$\frac{[\mathsf{U},\mathsf{e}]\,[\pi]\varphi}{\bigvee_{\alpha\in\mathcal{R}_n(\varphi)}\left(\bigwedge_{\mathsf{t}\subseteq\mathsf{E}}[T^{\mathsf{U}}_{\mathsf{et}}(\pi)]\,[\mathsf{t}]\,\alpha\right)}$$

$\bot$
$$\frac{[\mathsf{e}]\,\bot}{\neg\mathsf{pre}(\mathsf{e})}$$

$!\vee\!\!\!\vee$
$$\frac{[\mathsf{s}]\,(\varphi \vee\!\!\!\vee \psi)}{[\mathsf{s}]\,\varphi \vee\!\!\!\vee [\mathsf{s}]\,\psi}$$

$![\![\pi]\!]$
$$\frac{[\mathsf{U},\mathsf{s}]\,[\![\pi]\!]\varphi}{\bigwedge_{\mathsf{t}\subseteq\mathsf{E}}[\![T^{\mathsf{U}}_{\mathsf{st}}(\pi)]\!]\,[\mathsf{s}]\,\varphi}$$

!AUD
$$\frac{[\mathsf{s}]\,\alpha}{\bigwedge_{\mathsf{e}\in\mathsf{s}}[\mathsf{e}]\,\alpha}$$

$!\to$
$$\frac{[\mathsf{e}]\,(\varphi \to \psi)}{[\mathsf{e}]\,\varphi \to [\mathsf{e}]\,\psi}$$

RE
$$\frac{\varphi \leftrightarrow \psi}{\chi[\varphi/p] \leftrightarrow \chi[\psi/p]}$$

Figure 4.3: The new fragment from the proof system for LCCI, for models with at most n possible worlds.

$\psi' = [\pi]\chi_1$: Now by Propositions 4.7 and 4.21, $M, s \models [\mathsf{U},\mathsf{s}]\,[\pi]\chi_1 \iff M, s \models \bigwedge_{\mathsf{e}\in\mathsf{s}}\left(\bigvee_{\alpha\in\mathcal{R}_{|\mathcal{W}|}(\chi_1)}\left(\bigwedge_{\mathsf{t}\subseteq\mathsf{E}}[T^{\mathsf{U}}_{\mathsf{st}}(\pi)]\,[\mathsf{t}]\,\alpha\right)\right)$. Since $\alpha \in \mathcal{R}_{|\mathcal{W}|}(\varphi)$, it is less complex than ψ'. Then by the inductive hypothesis we get some IE-PDL formula α' such that $M, s \models [\mathsf{t}]\,\alpha \iff M, s \models \alpha'$. Then we get that $M, s \models [\mathsf{U},\mathsf{s}]\,[\pi]\chi_1 \iff M, s \models \bigwedge_{\mathsf{e}\in\mathsf{s}}\left(\bigvee_{\alpha\in\mathcal{R}_{|\mathcal{W}|}(\chi_1)}\left(\bigwedge_{\mathsf{t}\subseteq\mathsf{E}}[T^{\mathsf{U}}_{\mathsf{st}}(\pi)]\,\alpha'\right)\right)$, where the latter is an IE-PDL formula.

$\psi' = [\![\pi]\!]\chi_1$: Now by Proposition 4.22, $M, s \models [\mathsf{U},\mathsf{s}]\,[\![\pi]\!]\chi_1 \iff M, s \models \bigwedge_{\mathsf{t}\subseteq\mathsf{E}}[\![T^{\mathsf{U}}_{\mathsf{st}}(\pi)]\!]\,[\mathsf{s}]\,\chi_1$. Then by the inductive hypothesis we get $M, s \models [\mathsf{U},\mathsf{s}]\,[\![\pi]\!]\chi_1 \iff M, s \models \bigwedge_{\mathsf{t}\subseteq\mathsf{E}}[\![T^{\mathsf{U}}_{\mathsf{st}}(\pi)]\!]\,\chi_1'$, where $\bigwedge_{\mathsf{t}\subseteq\mathsf{E}}[\![T^{\mathsf{U}}_{\mathsf{st}}(\pi)]\!]\,\chi_1'$ is an IE-PDL formula.

This concludes the step for $[\mathsf{s}]\,\psi$, and the proof. □

Now that we have shown that every LCCI formula is equivalent to an IE-PDL formula, we can use this fact to create a complete axiomatisation for LCCI, as well as analyse some of the reduced formulas for simple situations. The following two sections will be about these reductions.

4.6 Axiomatisation

This gives us everything we need to give a sound and complete axiomatisation for LCCI using the reductions from Section 4.5 and the inference rules for IE-PDL. Since it is based on the proof system for IE-PDL, this proof system will also be parametrized on the number of worlds. The new rules for models with at most n worlds can be found in Figure 4.3.

As was the case for IE-PDL, there is one case where a rule is only applicable to declaratives which is the rule !AUD. It is easy to see that this must be the case, as this rule corresponds to Proposition 4.7, which also only works for declaratives. Also of note is that none of the new rules actually depend on the number n of new worlds, meaning that Propositions 3.27 and 3.28 also hold for LCCI.

It is easy to show that this proof system is sound with respect to the semantics of LCCI.

Theorem 4.25 (Soundness of LCCI)**.** *If $\Phi \vdash_n \psi$ then $\Phi \models_n \psi$.*

Proof. We will only focus on the new rules, since we have already shown that the other parts are sound in Theorem 3.26. Since these rules come directly from Section 4.5, they are sound by Propositions 4.7, 4.11, 4.13, 4.15 to 4.17, 4.21 and 4.22. □

4.6.1 Completeness

To prove that the proof system for LCCI is complete, it is enough to show that every LCCI formula is provably equivalent to an LCCI formula. In order to show this, we will adapt the completeness proof for AMLI from [22]. First we have to show that every formula is provably equivalent to its normal form.

Lemma 4.26. *For any formula φ, $\varphi \dashv\vdash_n \bigvee\!\!\!\!\bigvee \mathcal{R}_n(\varphi)$.*

Proof. This proof works from Lemma 3.29, so only the case for $[\mathsf{s}]\,\varphi$ is given. By the inductive hypothesis, we have that $\varphi \dashv\vdash_n \bigvee\!\!\!\!\bigvee \mathcal{R}_n(\varphi)$. Then we can use RE to get $[\mathsf{s}]\,\varphi \dashv\vdash_n [\mathsf{s}]\bigvee\!\!\!\!\bigvee \mathcal{R}_n(\varphi)$. Then by using $!\bigvee\!\!\!\!\bigvee$ we get $[\mathsf{s}] \dashv\vdash_n \bigvee\!\!\!\!\bigvee_{\alpha\in\mathcal{R}_n(\varphi)} [\mathsf{s}]\,\alpha$, of which the right hand side is the resolution of $[\varphi]$, as required. □

Now we can show that every LCCI formula is provably equivalent with an IE-PDL formula.

Lemma 4.27. *For any LCCI formula φ, there is an IE-PDL formula φ' such that $\varphi \dashv\vdash_n \varphi'$ for all $n \in \mathbb{N}$.*

Proof. This goes by the structure of Theorem 4.24, but using provable equivalence instead of equivalent support conditions in a given model. Instead of the propositions, we use the deduction rules from Figure 4.3 and we use RE for substitution of equivalents. We use Lemma 4.26 instead of Corollary 4.9. □

This only leaves the completeness theorem for LCCI.

Theorem 4.28 (Completeness for LCCI)**.** *If for some $n \in \mathbb{N}$, $\nvdash_n \varphi$, then there is some $m \in \mathbb{N}$ such that $\nvdash_m \varphi$ and there is a model M and a state s such that $M, s \not\models \varphi$, where $|\mathcal{W}| \leq m$.*

Proof. Suppose that $\nvdash_n \varphi$ for some $n \in \mathbb{N}$. Then by Lemma 4.27, $\nvdash_n \varphi'$. Then by Theorem 3.51, there is some $m \in \mathbb{N}$ such that $\nvdash_m \varphi'$, and some model M and state s such that $M, s \not\models \varphi'$ and $|\mathcal{W}| \leq m$. Then by Theorem 4.24 we have that $M, s \not\models \varphi$, as required. □

This shows that the given axiomatisation of LCCI is sound and complete with respect to its semantics.

4.7 Reduced Formulas

Lastly, we also want to show some of the reductions that this system gives, and compare them to the equivalent reductions in LCC. But first, we want to talk about the differences between the reductions for the modalities between LCCI and AMLI.

4.7.1 Differences with AMLI

The reductions for the modalities for AMLI are as follows:

Proposition 4.29 (AMLI reductions)**.** *Let M be an IEL model, s be a state in M, and let* e *be an event in the AMLI update model U. We then have the following equivalences:*

$$M, s \models [\mathsf{e}]\, K_a\varphi \iff M, s \models \mathsf{pre}(\mathsf{e}) \to K_a\, [\delta_{\mathsf{a}}(\mathsf{e})]\, \varphi$$
$$M, s \models [\mathsf{e}]\, E_a\varphi \iff M, s \models \mathsf{pre}(\mathsf{e}) \to \bigwedge_{\mathsf{s}\in\Delta_a(\mathsf{e})} E_a\, [\mathsf{s}]\, \varphi$$

Now it would be interesting to see what the reductions for LCCI give us in these situations, and to see what the differences are, if any.

Using the definition of T, we get the following reduction for the knowledge operator in LCCI:

$$\begin{aligned} M, s \models [\mathsf{e}]\, [a]\varphi &\iff M, s \models \bigvee_{\alpha\in\mathcal{R}_{|\mathcal{W}|}(\varphi)} \left(\bigwedge_{\mathsf{t}\subseteq\mathsf{E}} [T^{\mathsf{U}}_{\{\mathsf{e}\}\mathsf{t}}(a)]\, [\mathsf{t}]\, \alpha \right) \\ &\iff M, s \models \bigvee_{\alpha\in\mathcal{R}_{|\mathcal{W}|}(\varphi)} \left(\bigwedge_{\mathsf{t}\subseteq\mathsf{E}} [?\mathsf{pre}(\mathsf{e}); ?\top; a; ? \bigvee_{\mathsf{t}'\in\Delta_a(\mathsf{e}),\mathsf{t}'\subseteq\mathsf{t},\mathsf{t}'\neq\emptyset} \left(\wedge_{\mathsf{f}\in(\mathsf{t}-\mathsf{t}')}\neg\mathsf{pre}(\mathsf{f})\right)]\, [\mathsf{t}]\, \alpha \right) \\ &\iff M, s \models \bigvee_{\alpha\in\mathcal{R}_{|\mathcal{W}|}(\varphi)} \left([?\mathsf{pre}(\mathsf{e})] \bigwedge_{\mathsf{t}\subseteq\mathsf{E}} [a; ? \bigvee_{\mathsf{t}'\in\Delta_a(\mathsf{e}),\mathsf{t}'\subseteq\mathsf{t},\mathsf{t}'\neq\emptyset} \left(\wedge_{\mathsf{f}\in(\mathsf{t}-\mathsf{t}')}\neg\mathsf{pre}(\mathsf{f})\right)]\, [\mathsf{t}]\, \alpha \right) \end{aligned}$$

Since we only allow for $\mathsf{t}' \in \Delta_a(\mathsf{e})$, we can leave the other subsets of E out of the conjunction, since if there is no $\mathsf{t}' \in \Delta_a(\mathsf{e})$, the second test will test if t cannot be executed, leaving us in an inconsistent state if it can be executed, and otherwise $t\, [\mathsf{t}]$ will be empty, meaning that in those cases α is supported anyway, so we can ignore them.

$$\begin{aligned} &\iff M, s \models \bigvee_{\alpha\in\mathcal{R}_{|\mathcal{W}|}(\varphi)} \left([?\mathsf{pre}(\mathsf{e})] \bigwedge_{\mathsf{t}\in\Delta_a(\mathsf{e})} [a]\, [\mathsf{t}]\, \alpha \right) \\ &\iff M, s \models \bigvee_{\alpha\in\mathcal{R}_{|\mathcal{W}|}(\varphi)} \left(\mathsf{pre}(\mathsf{e}) \to \bigwedge_{\mathsf{t}\in\Delta_a(\mathsf{e})} [a]\, [\mathsf{t}]\, \alpha \right) \end{aligned}$$

As can be easily seen, there are differences between this reduction and the reduction for the knowledge operator in AMLI, which are the disjunction over the resolutions of φ, and the conjunction over $\Delta_a(\mathsf{e})$. This is because we took a different approach to find a reduction for $[\mathsf{s}]\, [a]\varphi$ then that was done in [22]. However, it can in fact be shown that this formula is logically equivalent to the corresponding AMLI reduction, meaning that the reduction given for AMLI is only a special case for the LCCI reduction.

If we do the same for the entertains operator we get:

$$\begin{aligned}
M, s \models [\mathsf{e}]\, [\![a]\!]\varphi &\iff M, s \models \bigwedge_{\mathsf{t} \subseteq \mathsf{E}} [\![T^{\mathsf{U}}_{\mathsf{et}}(a)]\!]\, [\mathsf{t}]\, \varphi \\
&\iff M, s \models \bigwedge_{\mathsf{t} \subseteq \mathsf{E}} [\![?\mathsf{pre}(\mathsf{e}); ?\top; a; ? \bigvee_{\mathsf{t}' \in \Delta_a(\mathsf{e}), \mathsf{t}' \subseteq \mathsf{t}, \mathsf{t}' \neq \emptyset} (\wedge_{\mathsf{f} \in (\mathsf{t} - \mathsf{t}')} \neg \mathsf{pre}(\mathsf{f}))]\!]\, [\mathsf{t}]\, \varphi \\
&\iff M, s \models [\![?\mathsf{pre}(\mathsf{e})]\!] \bigwedge_{\mathsf{t} \subseteq \mathsf{E}} [\![a; ? \bigvee_{\mathsf{t}' \in \Delta_a(\mathsf{e}), \mathsf{t}' \subseteq \mathsf{t}, \mathsf{t}' \neq \emptyset} (\wedge_{\mathsf{f} \in (\mathsf{t} - \mathsf{t}')} \neg \mathsf{pre}(\mathsf{f}))]\!]\, [\mathsf{t}]\, \varphi
\end{aligned}$$

Using the same reasoning as before, we can simplify this to

$$\begin{aligned}
&\iff M, s \models [\![?\mathsf{pre}(\mathsf{e})]\!] \bigwedge_{\mathsf{t} \in \Delta_a(\mathsf{e})} [\![a]\!]\, [\mathsf{t}]\, \varphi \\
&\iff M, s \models \mathsf{pre}(\mathsf{e}) \to \bigwedge_{\mathsf{t} \in \Delta_a(\mathsf{e})} [\![a]\!]\, [\mathsf{t}]\, \varphi
\end{aligned}$$

Which is the same as the reduction for the entertains operator for a single agent that AMLI gives us, since here we did not have to deal with the added complexity of quantifying over the resolutions of φ. This means that here as well, the reduction for AMLI is only a special case of the reduction for LCCI.

Since both reductions for AMLI are special cases for the LCCI reductions, namely the ones where we only allow single agents as epistemic programs, it is probably possible to generalize the state maps for AMLI in a similar way to the program transformations for LCCI to get a version of AMLI with common knowledge.

4.7.2 Analysing Public Announcements

Like in the LCC paper [18], we also want to analyse the program transformations that are needed to reduce formulas with update models for major communication types, like public announcements and their effect on common knowledge. Studying these transformations can give rise to reduction axioms for other systems, like Inquisitive Dynamic Epistemic Logic with common knowledge and public issues, or the aforementioned extensions for AMLI. In order to make this task easier, these reductions were done by a computer. The software used is introduced in Appendix A.

However, since the reductions for LCCI are quite complex, we will only look at the simplest one, which is the public announcement of a declarative. Here we want to look at its effect on common knowledge and public issues.

As was shown in Example 4.1, a public announcement of a declarative in LCCI is an action model with one action, which has the declarative formula that is being announced as its precondition. The state map for each agent a is then $\Delta_a(\mathsf{e}) = \{\{e\}, \emptyset\}$, and the substitution for the event is empty. We will call this action model P_α, since it is the public announcement of some declarative α.

To test its effect on common knowledge, we will have to look at the reduction of the formula $[\mathsf{P}_\alpha, \mathsf{e}]\, [(\bigcup A)^*]\psi$, where A is some group of agents. The reduction of this formula depends on $T^{\mathsf{P}_\alpha}_{\{\mathsf{e}\}\{\mathsf{e}\}}((\bigcup A)^*)$ and $T^{\mathsf{P}_\alpha}_{\{\mathsf{e}\}\emptyset}((\bigcup A)^*)$. The latter instantly devolves into $?\bot$, since it has no events, so we only look at the former. In order to get the value for $T^{\mathsf{P}_\alpha}_{\{\mathsf{e}\}\{\mathsf{e}\}}((\bigcup A)^*)$, we need to know

$K^{\mathsf{P}_\alpha}_{\{\mathsf{e}\}\{\mathsf{e}\}\mathsf{E}}((\bigcup A)) = K^{\mathsf{P}_\alpha}_{\{\mathsf{e}\}\{\mathsf{e}\}\{\mathsf{e}\}}((\bigcup A))$. Following the definition for K we get:

$$\begin{aligned}
K^{\mathsf{P}_\alpha}_{\{\mathsf{e}\}\{\mathsf{e}\}\{\mathsf{e}\}}(\left(\bigcup A\right)) &= \bigcup_{\mathsf{f}\in\{e\}} \left(K^{\mathsf{P}_\alpha}_{\{\mathsf{e}\}\{\mathsf{e}\}\{\mathsf{e}\}-\{\mathsf{e}\}}(\left(\bigcup A\right))\right)^* \\
&= K^{\mathsf{P}_\alpha}_{\{\mathsf{e}\}\{\mathsf{e}\}\emptyset}(\left(\bigcup A\right))^* \\
&= T^{\mathsf{P}_\alpha}_{\{\mathsf{e}\}\{\mathsf{e}\}}(\left(\bigcup A\right))^* \\
&= \left(\bigcup_{a\in A} T^{\mathsf{P}_\alpha}_{\{\mathsf{e}\}\{\mathsf{e}\}}(a)\right)^* \\
&= \left(\bigcup_{a\in A} ?\mathsf{pre}(\mathsf{e}); a\right)^* \\
&= \left(?\mathsf{pre}(\mathsf{e}); \bigcup A\right)^* \\
&= \left(?\alpha; \bigcup A\right)^*
\end{aligned}$$

Therefore, the reduction for $[\mathsf{P}_\alpha, \mathsf{e}]\,[(\bigcup A)^*]\psi$ with respect to models with at most n worlds becomes $\bigvee_{\beta\in\mathcal{R}_n(\psi)}[(?\alpha; \bigcup A)^*]\,[\mathsf{P}_\alpha, \mathsf{e}_1]\,\beta$. The reduction tells us that commonly knowing whether ψ after the public announcement of α is equivalent to a group of agents commonly knowing one of the resolutions of ψ if we assume that α is true.

What this formula expresses, is that it is commonly known whether ψ after the public announcement of α if one of the resolutions of ψ is common knowledge when we assume that α is true.

Interestingly enough, this is the same reduction that was found for public announcements for LCC, meaning that LCCI, despite using different program transformations that use sets, does arrive at the same reductions in some cases.

Of course, looking at the raising of an issue would be more interesting. However, since the program transformers are dependent upon the number of events in the model, the complexity of the reduced formulas quickly increases. This happens so quickly, in fact, that the reduction for an action model with two events is already several lines long. The problem is that it has to go through all the possible combinations of events, of which there are 3 (we ignore the empty set) multiple times. This makes the resulting formulas quite long, and it would be a study in and of itself to find the exact structure of these formulas. Therefore, we will leave the study of these reductions for further research.

4.8 Conclusion

In this chapter we introduced the new logic LCCI, and showed that it has the normal properties for inquisitive semantics, including the modal persistency and modal empty state properties from AMLI. Then we found and discussed reductions that show that every LCCI formula is equivalent to a corresponding IE-PDL formula. We then used this reduction to give a sound and complete axiomatisation of LCCI, and studied some reductions.

Chapter 5

Comparison with other work

In this chapter, we will compare IE-PDL and LCCI with some other logics with similar goals. We will sometimes refer back to earlier chapters, if part of the comparison has already been made before.

5.1 LCC and E-PDL

We will start with comparing IE-PDL to E-PDL [12, 18]. If we look at the proof system for IE-PDL as presented in Section 3.8.2, and then in particular at the modal fragment as presented in Figure 3.5, we can see that all the standard proof rules for E-PDL are available. For $[\pi]$-distribution, $[\pi]$-necessitation, sequence, and choice, this is easy to see, as they are directly present. For test, it might look like this is not the case because the rule is only applicable if the consequent is declarative, but all functions in E-PDL are declarative, so the rule functions the same in both systems. The only differences lie in the rules for unbounded iteration.

Part of this difference comes from the difference in rules, but a part of this also comes from the fact that in IE-PDL π^* is interpreted on the transitive, not reflexive-transitive closure. This means that the rules also need to be slightly different. However, these different versions of mix and induction can also be proven to hold in IE-PDL, either semantically or using the proof system. Therefore, IE-PDL is a conservative extension of E-PDL, if we take π^* to be interpreted on the transitive closure.

When we start to compare LCCI and LCC [18] however, we start to see more serious differences. While the two logics employ a similar mechanism for the reductions, LCC does this by recursion over the set of events, while LCCI has to do this by recursion of the power set of the set of events. This means that LCCI is not a conservative extension of LCC. However, this is not a real problem, since the rules for LCC do not make sense if the agents wonder which event occurred. One would probably find more LCC-like program transformers if one would redo this research but using AMLQ as a basis instead of AMLI, since in AMLQ agents do not wonder about which event occurred.

However, everything that we could express in LCC can be expressed in LCCI, since LCC update models are a special case of LCCI update models. This special case is the case where every agent is only interested in which event exactly occurred. This means that LCCI adds exactly the expressivity to LCC that we wanted to bring, namely that we can express that the agents wonder which event happened. While this cannot be directly expressed in the logic (since there is no "event was executed" operator), we can see this in the state maps of the updated model. As a side-effect, this also means that the agents are interested in the post-conditions of the events.

Because of these reasons, we can conclude that LCCI is an inquisitive version of LCC, even when it is not a conservative extension of LCC.

5.2 AMLI and IEL

We will first compare IE-PDL with IEL[2]. As was already shown in Section 3.7, every IEL formula can be translated into an equivalent IE-PDL formula. However, this equivalence only holds as long as we restrict ourselves to models with a finite number of worlds, since IE-PDL is only defined for models with a finite number of worlds. While this is a limitation in the current version of IE-PDL, we believe this to be a technical limitation, and not a conceptual one. Therefore, we can conclude that IE-PDL is a conservative extension of IEL.

However, not every IE-PDL formula can be translated to an IEL formula, not even if we look at a generalized version of IEL where there are no constraints on the state maps for the agents, i.e., no factivity or introspection. This means that IE-PDL is strictly more expressive than IEL, which raises the question: What can we express in IE-PDL that we cannot express in IEL? The first class of formulas all contain group knowledge, which we cannot express using the common knowledge and public issue operators from IEL. It is however possible to add these to IEL, and then those formulas can be expressed.

There is, however, a large group of formulas that cannot be expressed in IEL, even if we loosen its constraints a bit. These are formulas that use iteration in interesting ways, such as $[(a;b)^*]\varphi$, which expresses that a knows that b knows φ, and that a knows that b knows that a knows that b knows φ, etc. Since we keep alternating between the knowledge for a and b, this is different from Common Knowledge between a and b. For example, if we are only dealing with belief, so b's beliefs might not be truthful, a does not have to believe whether φ, even when b does. Another example of this is a formula like $[\![(a \cup b \cup ?\varphi)^*]\!]\psi$, which states that ψ is a public issue between a and b assuming that φ is not supported. However, like in the case of LCC versus standard epistemic logic, we have found no practical use for these kinds of formulas. So while IE-PDL is more expressive than IEL, we do not believe that this additional expressivity actually adds much to the language itself.

Then, we will move on to the comparison between LCCI and AMLI [22]. Using Section 3.7 and the fact that AMLI action models are LCCI action models without substitutions, we can also find a translation from AMLI formulas to LCCI formulas. This translation extends Definition 3.11 with the following clause:

$$([\mathsf{U}, \mathsf{s}]\, \varphi)' = [\mathsf{U}', \mathsf{s}]\, \varphi'$$

where U' is like U, but it also assigns the empty substitution to every event in U. Therefore, we can also say that every AMLI formula, even if it includes K_* or E_*, has an equivalent LCCI formula. Since we have already shown in Section 4.7.1 that the reduction equivalences from AMLI still hold in LCCI, we can conclude that LCCI is a conservative extension of AMLI.

5.3 InqPDL

While the work on this thesis was ongoing, Punčochář and Sedlár created a different inquisitive version of PDL [16]. We would also like to compare this logic, called Inquisitive Propositional Dynamic Logic (InqPDL), with IE-PDL.

While both logics are meant to be inquisitive versions of PDL, the two do have different goals. While IE-PDL was designed to be used as an epistemic language, InqPDL is meant to be used as a proper inquisitive version of PDL. This means that the intuitive interpretation of the modal operators is slightly different. So while in IE-PDL $[a](\alpha \vee\!\!\vee \beta)$ means that agent a knows whether $\alpha \vee\!\!\vee \beta$, the corresponding InqPDL interpretation is "after the execution of action a, is α or β supported?".

Because of this difference in interpretation, IE-PDL has an extra operator, namely the entertain operator, $[\![\pi]\!]\varphi$, which has no corresponding operator in InqPDL. This also makes sense, since in the interpretation of InqPDL there are no agents that can have issues or know things, so we don't need to be able to express this difference. However, it should be noted here that the semantics for IE-PDL's $[\![\pi]\!]\varphi$ and InqPDL's $[\pi]\varphi$ are the same.

Beside this difference in interpretation, there are also a few other big differences between the logics. The first of these is that InqPDL is based on the algebraic semantics of [15]. However, as has already been pointed out in [16], this difference is not substantial. This is because sets of possible worlds, such as are used in this thesis, are the primitive elements of a complete atomic Boolean algebra of power sets, making it one of the possibilities for defining InqPDL models.

A bigger difference is that, unlike in IE-PDL, in InqPDL $[\pi]\varphi$ is interrogative if φ is interrogative. This has to do with the fact that in InqPDL, unlike in IE-PDL, there are no restrictions on the atomic relations between states. This means that, while it is possible to give an epistemic reading to InqPDL, this has to be done in a different manner than it happens in IE-PDL. More details about this can be found at the end of Section 4 in [16].

A last big difference is that InqPDL has an extra program operator. Being meant as a logic to interpret programs, it also has an inquisitive choice operator, $\shortmid\!\cup$. While its standard choice "executes" actions on the level of worlds, its inquisitive choice "executes" actions on the level of states. In its epistemic interpretation, this would not create a group of agents, but choose between the different agents. IE-PDL does not have a corresponding operator for this, since we do not want to choose between agents, but we want to create groups of agents. However, adding such an operator might lead to more interesting notions of "commonly believing whether" than IE-PDL currently has, where all agents have to believe the same resolution of a formula to commonly belief whether that formula holds.

One last interesting note to end on, is that both logics have the same semantics for the test operator. While this semantics differs in interpretation from the default interpretation, the currently employed solution might then be the most natural alternative in inquisitive settings.

All in all, while both are based on PDL, the two logics IE-PDL and InqPDL try to solve different problems, and therefore are also quite different from one another. Our recommendation would be to use IE-PDL only if you want to reason about the knowledge of agents, but for anything else, such as reasoning about program execution and the issues that they raise and resolve, it is better to use InqPDL.

Chapter 6

Conclusion and further research

6.1 Conclusion

In this thesis, we set out to build a logical framework that extends both the Logic of Communication and Change (LCC) with questions and issues, and Action Model Logic with Issues (AMLI) with common knowledge, public issues, and factual change. We have been able to successfully combine these systems with respect to finite models. We found that most of the ideas from LCC were easily adapted into an inquisitive framework, which means that the Logic of Communication, Change, and Issues (LCCI) is a rather natural extension of both LCC and AMLI.

Since this process required the combination of multiple logics in different ways, it might be useful to refer back to Figure 2.4, which is also reprinted here as Figure 6.1. This figure gives the relations between the various logics that were used as basis or inspiration, or designed in this thesis.

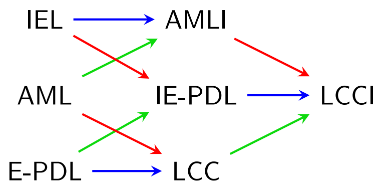


Figure 6.1: The relations between the different Logics in this thesis. A blue arrow means that the logic on the left is used as the static language for the logic on the right. A green arrow means that the logic on the right is an inquisitive variant of the logic on the left. A red arrow means that the logic on the right borrows ideas from the logic on the left.

We started with the definition of the new logic, Inquisitive Epistemic Propositional Dynamic Logic (IE-PDL), which is meant to be used as an Inquisitive version of Epistemic Propositional Dynamic Logic (E-PDL). Here we ran into problems with the reflexive nature of some E-PDL program operators, like Test. These problems were solved by interpreting the operators a bit differently, but still in such a way that they are suitable for our purposes. Because of the interaction between Test and unbounded iteration, we also ran into a problem when giving the resolutions for $[\![\pi^*]\!]\mu$, which means that we were limited to models with a finite number of worlds. This flexibility did give us an inquisitive epistemic language with common knowledge and public issues for arbitrary finite groups of agents.

During this process, we have established that IE-PDL is a conservative extension of Inquisitive Epistemic Logic (IEL). We also provided a sound and complete axiomatisation of IE-PDL with respect to finite models. This makes IE-PDL the first inquisitive epistemic logic with common knowledge and public issues, that we know of, that has a sound and complete axiomatisation.

After that, we defined the action models for LCCI by extending the action models from AMLI with a substitution for each event. This leads to a natural extension that can be reduced into its underlying static language in a similar way to AMLI, with the exception of the modal operators, for which we could apply the tricks used for LCC. We then used this reduction to give a sound and complete axiomatisation of LCCI.

By combining features from both LCC and AMLI, LCCI extends its predecessors in the following ways:

- It adds the ability to encode what an agent wants to know into the framework of LCC in a natural way;
- It allows for the asking of questions as an action into the framework of LCC;
- It extends AMLI with the addition of common knowledge and public issue operators while keeping the reduction;
- And it adds the ability to model factual change into AMLI.

Besides the technical results in this thesis, this project is accompanied by a computational tool, about which more details can be found in Appendix A.

6.2 Further work

We will end this thesis by discussing some ideas for further work. First we will discuss some results we expect for IEL, then we will discuss some ways in which IE-PDL can be improved, and we will end with some uses and extensions for LCCI.

6.2.1 Implications for IEL

Since IEL and IE-PDL have similar goals, there are some results for IE-PDL that we expect will carry over to IEL. The first of these is the soundness and completeness proof obtained in this thesis. Using a similar approach as the one employed in this thesis, it should be possible to also give a sound and complete axiomatisation for IEL with common knowledge and public issues.[1] We also expect that the soundness and completeness proof that this would give for IEL can be simpler than the one given for IE-PDL, since one would not have to tie every proof to a maximum number of worlds.

We also expect that it will be possible to use the IE-PDL semantics to build an extension for IEL that has common knowledge and public issues for arbitrary groups of agents. The soundness and completeness result discussed before should also carry over to this logic.

If this approach were to be used to give a completeness proof for IEL, then that would mean that IEL would have the finite model property. In turn this would mean that IEL would be decidable, which would be a good first step towards a complexity analysis for Inquisitive Modal logics, which has, to our knowledge, not been started.

[1]An earlier version of this thesis used a completeness proof based on the work in [10, 17], which worked for models with countably many worlds. While this approach did not work for IE-PDL, we expect that the problems encountered do not carry over to IEL, meaning that it should be possible to give a sound and complete axiomatisation for IEL with models with countably many worlds.

6.2.2 Extensions to IE-PDL

There is still work to be done on IE-PDL itself. The first is to check whether there is a fragment of IE-PDL that can be used as a basis for LCCI instead. Of particular interest for this are the declarative programs. If we can find a subset of the declarative programs that we can use to express everything that we want to express and that allows us to still have the full reduction from LCCI into IE-PDL, then that would make working with the logic easier, since then we would not have to specify a maximum number of worlds in the model.

This would probably require extending the declarative programs a bit more, since there are currently programs in the reductions that cannot be expressed using only the declarative programs. Therefore, a good start to tackle this problem would be to see what is required for the reductions, and check whether those programs are declarative.

A different solution to the problem of specifying the maximum number of worlds every time would require figuring out the maximum number of worlds necessary to falsify a certain formula. If this number was known, then that could be used to calculate the resolutions of that formula. For declaratives, this is possible to do, using the construction of the counter-model. However, for interrogatives, the construction of the counter-model depends on the resolutions of that formula, which depend on the maximum number of worlds again. We currently see no way to untie this knot, but if it can be done, this would also solve the problem of calculating the resolutions for IE-PDL, possibly making the proof system easier to use, and would lead to a more elegant completeness proof.

Another open problem is to see if there are more fitting definitions for the test operator. A few were tried over the course of this thesis, such as test being reflexive on the set itself, which simply lifts the normal interpretation to the level of information states, or only testing at the level of worlds, but none of these were found to be satisfactory. However, the current definition might be the most useful one, since it is also used by InqPDL[16], see also Section 5.3 for a discussion.

There is still the issue of the usual reflexivity of the * operator that we chose not to investigate. Both studying the semantics that result from our proposed solution, which was to make sure that if $w \in s$, then $sR_{\pi^*}w$ for all programs π, and finding other possible solutions would be of interest.

It is also possible to extend IE-PDL itself further. For example, while IE-PDL can be used as a language for belief, this is done without a preference order or any form of belief revision. Therefore, it would be interesting to see if we can extend to a real language for belief revision. We can see two approaches for this.

The first of these is to take the approach from [7] and extend the models for IE-PDL with an ordering over information states. This approach would allow us to model different types of belief revision into the logic and reason about the beliefs of agents that way. Results from this approach will probably carry over to the corresponding version of IEL.

Another approach would be to extend the language of IE-PDL in a similar manner to how van Eijck and Wang extended E-PDL in [21], i.e. by adding more program operators. The benefit of this would be that this would not just allow us to model both knowledge and common belief in one model, but also notions like conditional and safe belief. This approach will also include common knowledge and belief. However, since we are extending the language with more program operators, this will not carry over to other logics.

6.2.3 Uses for LCCI

Similar to how we can extend IE-PDL in many ways, we can apply the same kinds of reasoning to LCCI. LCC is used in a lot of applications and has many extensions, some of which could also be applied to LCCI. For example, [21] also has a method for extending LCC, which can probably be applied to LCCI so it can be used for belief revision as well. Another potential extension could be created by incorporating the ideas from [20], and create an inquisitive logic for lying agents.

Another interesting area to investigate concerns the effects of actions on common knowledge and public issues, which is now feasible thanks to the reductions in LCCI. This could lead to semantics and reductions for other Inquisitive Dynamic Epistemic Logics, such as extensions of IDEL and AMLI where they have common knowledge and public issues. Especially finding the reductions might not be trivial otherwise.

Besides these, there are many situations that can now be modelled using LCCI that could not be modelled before. For example, now we can look at how the issues of agents change during a game of Clue, or check whether the extra information that an agent might want to know a specific card can help when playing Hanabi. However, LCCI might also be able to help in the modelling of situations where one agent tries to figure out what an agent wants to know, which might occur during a question session of a course, or during the questions after a seminar. Other situations that involve factual change would be the effects of observed, but unknown actions, such as something falling in another room but making noise, or as shown in the examples in this thesis, drawing a closed card in a game.

Appendix A

Implementation

This thesis comes accompanied with an implementation, which is a Haskell library called `LCCI.hs`. This library can be used as a model checker for LCCI and it can calculate the reduction for a formula using the equivalences given in Section 4.5. The program is open source and available online, so others can use and build upon it, and it can be found at the following url: `github.com/rmellema/LCCI.hs`. Its intended use is as a library for programs that need to use LCCI models in some way, or directly from an interactive read-evaluate-print-loop like ghci.

Since the library can be used as a model checker, it can also calculate product updates for static and update models and how programs, i.e. agents, relate different states to each other. Because it has to be able to give reductions, it can also calculate the resolutions for a formula for a given maximum number of worlds. Besides these things, it also has various quality-of-life utilities, such as a recognizable syntax for formulas and various functions for defining common action models.

The rest of this chapter consists of two parts. In the first part we will show an example application in which we will model some situations from the Hexa game [19]. In the second part, we will prove that all the simplifications used by the `simplify` function are equivalences in the language.

A.1 The Hexa Game

In this example, we will walk through the modeling of the Hexa game in LCCI, using the implementation to aid us in answering questions about the model. The Hexa game was first presented in [19], and is used more often as a testbed for Dynamic Epistemic Logics. We decided to use this as an example instead of Citadels since the models are smaller and easier to understand.

Since the implementation is written in Haskell, the example will also be given in Haskell. While we will try to keep the example easy to follow for those who have no experience with Haskell, if you want to use `LCCI.hs` yourself, it might be useful to read up on using the language. For this we would recommend using *The Haskell Road to Logic, Maths and Programming, Second Edition* [8]. If you are also interested in using Haskell as a general programming language, then *Learn you a Haskell* [13] is also a good introduction.

A.1.1 The Game

Before we can start to model the game in LCCI, we will first have to understand the rules of the game. In the Hexa game, there are three players, which we will call Ann (a), Bill (b), and Carol

(c). Each of these players will get a unique card out of a deck of three. We will number the cards 0, 1, and 2. The goal of the game is to know the full deal of the cards by asking questions and getting answers.

The only types of questions that we will allow are asking for a specific card. To make the example more interesting and informative, we will also allow for different actions, in particular the swapping of cards between agents.

A.1.2 Setting up

Before we can start to model the example, we will first have to set up some Haskell specific things. In Haskell code is normally organized in modules, which are named after where they are in the file system. Since this files lives in the `LCCI/Examples` folder and is named `Hexa.lhs`, the module name will be `LCCI.Examples.Hexa`.

```
module LCCI.Examples.Hexa where
```

Since we did not want to reinvent the wheel, we will also need to import some other packages, in particular packages for sets and maps. These come from the collections module from Hackage.

```
import qualified Data.Map as Map
import qualified Data.Set as Set
```

Now we can import the LCCI specific packages that we need. The first of these is the `LCCI.Issue` package, which contains code for representing information states, issues, and state maps.

```
import LCCI.Issue
```

We will also need to represent the models. The `LCCI.Model` package contains code for working with and defining both the static models from IE-PDL and the update models from LCCI.

```
import LCCI.Model
```

Besides needing to represent the models, we will also need to represent formulas of the language, so we can ask questions about them to the implementation. For this, we will need to import the `LCCI.Syntax` package.

```
import LCCI.Syntax
```

Because `LCCI.Syntax` defines the syntax in a way that is not as easy to read or write as one might want, we will import the `LCCI.Syntax.Pretty` package to use functions that are a bit easier on the eyes.

```
import LCCI.Syntax.Pretty
```

We will also want to be able to evaluate formulas in models, which we can do using the `supports` function from the `LCCI.Evaluation` module.

```
import LCCI.Evaluation
```

There is also a module that has functions for the creation of action models for common scenarios, which we will use in our discussion of action models.

```
import LCCI.Announcements
```

When we want to define our own action models, we will also need to define the substitutions for events, for which we will also need to import a package.

```
import LCCI.Substitution as Substitution
```

This are all the packages that we have to import, meaning we can now start to define the actual model.

A.1.3 The Model

The Worlds and Valuation

The first thing that we need to define for the model itself, are the worlds. While it is usual to number worlds in implementations, `LCCI.hs` is a bit more flexible. Because of this, we can also use a different representation in the implementation. The only requirement that the implementation has it that the worlds can be ordered. Thanks to this, we can use the representation that is usual for the Hexa model, where each world is represented by the deal of cards.[1]

We will represent a deal of cards by a triple of numbers, where the number is the card, so one of 0, 1, or 2, and the position in the triple signifies the player, so a 0 being in position one means that Ann has card 0, etc. Since triples can be lexicographically ordered, all we have to do is tell Haskell that we want to use this to represent worlds. We do this by declaring a triple as an instance of the type `World`.

```
instance (Ord a, Ord b, Ord c) => World (a, b, c)
```

We will also want to be able to show the worlds in a readable manner. This means we will have to make our triples an instance of the class `PrettyShow`, which is what is used in `LCCI.hs` to prettyprint objects.

```
instance (Show a, Show b, Show c) => PrettyShow (a, b, c) where
    prettyShow (a, b, c) = show a ++ show b ++ show c
```

Now we can define our worlds. Since all the cards are unique and there are only three cards, there is a total of 6 worlds.

```
-- | The worlds in the Hex model
w012, w021, w102, w120, w201, w210 :: (Int, Int, Int)
w012 = (0, 1, 2)
w021 = (0, 2, 1)
w102 = (1, 0, 2)
w120 = (1, 2, 0)
w201 = (2, 0, 1)
w210 = (2, 1, 0)
```

For the definition of the model later on, we will also need to have a set of all the worlds, which we will already define here.

```
-- | The set of worlds
ws :: Set.Set (Int, Int, Int)
ws = Set.fromList [w012, w021, w102, w120, w201, w210]
```

[1]If we wanted to, we could have also used the `IntWorld` data type that is already defined by `LCCI.hs`. Then we wouldn't have to define an instance of `World` or `PrettyShow` but we would give up the flexibility we now have.

Now we have our worlds defined we can start to define our valuation function. In `LCCI.hs` the valuation is a function from a proposition to a world to a boolean. This function can be defined directly, or built up from a `Map` from worlds to the propositions that are true in that world, using the function `valuationFromMap`. Since we have a large amount of structure in our definition of the worlds, we will go with the first option. This means that we will first have to define our propositional atoms.

We will keep our propositional atoms simple, and add one for each combination of player and card. In `LCCI.hs` propositional atoms start with a capital letter, in order to easily differentiate them from agents.

```
-- | The propositions in the Hex model
a0, a1, a2, b0, b1, b2, c0, c1, c2 :: Proposition
a0 = proposition "A0" -- Ann has card 0
a1 = proposition "A1" -- Ann has card 1
a2 = proposition "A2" -- Ann has card 2
b0 = proposition "B0" -- Bill has card 0
b1 = proposition "B1" -- Bill has card 1
b2 = proposition "B2" -- Bill has card 2
c0 = proposition "C0" -- Carol has card 0
c1 = proposition "C1" -- Carol has card 1
c2 = proposition "C2" -- Carol has card 2
```

For the valuation, it is also very useful to split up this player card pair so we can compare it with the card in the triple again. For this we will define two functions that extract the card or player name from a proposition. Both of these depend upon the pretty string representation of a proposition, which is just the string we gave it before.

```
-- | Get the number of the card for the given proposition.
card :: Proposition -> String
card = tail . prettyShow

-- | Get the identifier for the player in the given proposition.
player :: Proposition -> Char
player = head . prettyShow
```

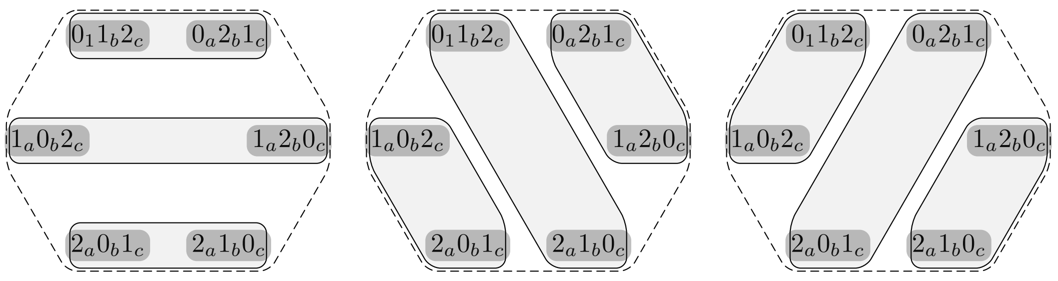
So now we can define our valuation. The valuation compares the (string representation of) the card that the player of a proposition `p` has in the world to the card that they have according to the proposition. If they are equal, it returns `True`, and otherwise it returns `False`.

```
-- | The valuation of the propositional atoms
v :: Valuation (Int, Int, Int)
v p (ca, cb, cc)
    | player p == 'A' = show ca == card p
    | player p == 'B' = show cb == card p
    | player p == 'C' = show cc == card p
    | otherwise       = False
```

This ends the definition of the valuation, which means that we can now focus on the last part of the static model, which is the state map.

The State Map

For the state map we will first have to consider exactly which scenario we want to model. Since this is most interesting for our purposes, we will pick a situation where all of the agents have



(a) The state map for Ann (b) The state map for Bill (c) The state map for Carol

Figure A.1: The three different state maps

been dealt a card, but where they have not looked at it yet. This means that in our initial situation the agents have no information yet, beside the information on how the game works. For simplicity, we will at first assume that the agents are only interested in knowing their own card. This gives the state maps in Figure A.1.

Before we can define the state maps in `LCCI.hs`, we will also have to define the agents. This is done by using the function `atom`, that defines an atomic program. In `LCCI.hs`, agents start with a lower case letter, in order to differentiate them from propositional atoms.

```
-- | The "actions" in the Hex model (The knowledge relations)
a, b, c :: Atomic
a = atom "a"
b = atom "b"
c = atom "c"
```

In `LCCI.hs`, a state map for one agent is defined as a `Map` from worlds to issues, just as in LCCI. We can define issues using the function `issue`, which takes a list of states and returns the issue represented by those states. Similarly, we can create a state using the function `state`, that takes a list of worlds, and returns the information state that contains only those worlds. This means that we can define the state map for Ann as follows:

```
ann_map :: StateMap (Int, Int, Int)
ann_map = Map.fromList
    [ (w012, issue [state [w012, w021], state [w102, w120], state [w201, w210]])
    , (w021, issue [state [w012, w021], state [w102, w120], state [w201, w210]])
    , (w102, issue [state [w012, w021], state [w102, w120], state [w201, w210]])
    , (w120, issue [state [w012, w021], state [w102, w120], state [w201, w210]])
    , (w201, issue [state [w012, w021], state [w102, w120], state [w201, w210]])
    , (w210, issue [state [w012, w021], state [w102, w120], state [w201, w210]])
    ]
```

The full state map then becomes a mapping from agents to state maps like `ann_map`, as follows:

```
-- | The state maps for all the various agents.
s :: Map.Map Atomic (StateMap (Int, Int, Int))
s = Map.fromList
    [ (a, ann_map)
    , (b, Map.fromList
        [ (w012, issue [state [w012, w210], state [w102, w201], state [w021, w120]])
        , (w021, issue [state [w012, w210], state [w102, w201], state [w021, w120]])
        , (w102, issue [state [w012, w210], state [w102, w201], state [w021, w120]])
        , (w120, issue [state [w012, w210], state [w102, w201], state [w021, w120]])
        , (w201, issue [state [w012, w210], state [w102, w201], state [w021, w120]])
        , (w210, issue [state [w012, w210], state [w102, w201], state [w021, w120]])
        ])
    , (c, Map.fromList
        [ (w012, issue [state [w012, w102], state [w021, w201], state [w120, w210]])
        , (w021, issue [state [w012, w102], state [w021, w201], state [w120, w210]])
        , (w102, issue [state [w012, w102], state [w021, w201], state [w120, w210]])
        , (w120, issue [state [w012, w102], state [w021, w201], state [w120, w210]])
        , (w201, issue [state [w012, w102], state [w021, w201], state [w120, w210]])
        , (w210, issue [state [w012, w102], state [w021, w201], state [w120, w210]])
        ])
    ]
```

Now we have all the ingredients to define the static model, which we will call `hexa`.

```
-- | The actual model
hexa :: StaticModel (Int, Int, Int)
hexa = StaticModel ws v s
```

Now we can use the model to test some formulas. Formulas are made with the constructors from the `LCCI.Syntax` package, or the functions from the `LCCI.Syntax.Pretty` package. First we will check if in the entire model, so each world in the model, Ann wonders what her card is. We can test this by loading up `ghci` with the current file and executing the following line:

```
supports hexa ws (wonder a (a0 \\/ a1 \\/ a2))
```

This line returns `True`, meaning that Ann wonders what her card is. We will now spent some time explaining what this line does.

The function `supports` is used to check if a certain model and state support a given formula. Here we want to know something about the `hexa` model, so we pass that as the model. Since we want to know if the formula is supported in the whole model, we then pass the set of all worlds (`ws`) as the second parameter. If the formula is supported in the set of all worlds, then, by persistence, it is also supported in all subsets of the set of all worlds, and therefore in the whole model. Then we need to pass the formula that we want to check to the function. For this we use functions from the `LCCI.Syntax.Pretty` package. We use the function `wonder` to create a formula akin to the wonder modality. The function takes two arguments, the first of which is a program representing the agents that wonder something, in our case just `a`, and secondly a formula that we want to know if the agent wants to know. For creating this formula we can use the `\\/` operator, which is inquisitive disjunction. Conjunction and classical disjunction can similarly be written out using slashes. We have to make an inquisitive disjunction with all the cards Ann might have, so we have to use all the propositional atoms we defined for Ann before.

We can now do something similar for Bill:

```
supports hexa ws (wonder b (a0 \\/ a1 \\/ a2))
```

As one would expect, since Bill is not yet interested in which card Ann has, this call returns `False`. However, Bill should know that Ann does not yet know what her card is. We can check for higher order knowledge like this by making more “complex” formulas.

```
supports hexa ws (knows b (neg (knows a (a0 \\/ a1 \\/ a2))))
```

As we would expect, this returns `True`, since it is common knowledge that no-one has looked at their card yet. The function `knows` in LCCI's knowledge operator, and `neg` creates the negation of a formula.

We can also check if some of the rules of the game are common knowledge. For example, we can check if it is common knowledge that when Ann has card 0, Bill or Carol cannot have card 0:

```
supports hexa ws (knows (iter (a .+ b .+ c)) (a0 --> neg (b0 \/ c0)))
```

Which, as one would expect, returns `True`. Here we use the function `iter` to create an unbounded iteration version of a program, and use `.+`, the choice operator, to combine the various agents into one group. `-->` is implication.

A.1.4 It's Time for Action

Having the static model is nice, but we should also try to introduce some actions into our model. For this we can create action models in a similar manner as we did for defining the `hexa` model. Before we will dive into defining action models however, we will first show how to use them, using the `LCCI.Announcements` package.

Announcements

The `LCCI.Announcements` package exports functions for the creation of public, private, and secret announcements, based on a list of formulas and (one or more) lists of agents. We will use the function `privateInform` to model the action of Ann looking at her card. This function takes 5 arguments. The first is the name of the model as a string, which is used when pretty printing the model in a formula. The second is the group of agents that learn the information, the third is the group of agents that see the information being shared and that are interested in what is being shared, and the fourth is the group of agents that see the information being shared, but don't care. The last argument is a list of possibilities for the material that is being shared.

We will call the model read$_a$, since Ann reads her card. She is the only one with which the information is being shared, so the first list is just Ann. Bill and Carol are both interested in which card Ann saw, so they want to know what she learned. Since these are all the agents, the next list is empty. The last list contains the possible formulas that Ann might have learned, but now wrapped in the function `Prop`. The reason for this is that `LCCI.Syntax.Pretty` is a bit more liberal in the types of arguments that it accepts then that `LCCI.Announcements` is. When we defined the propositions, we did not define them as formulas, but as propositions, and here we use the function `Prop` to cast them to formulas.

```
ann_look = privateInform "read_a" [a] [b, c] [] [Prop a0, Prop a1, Prop a2]
ann_events = Set.toList $ events $ snd ann_look
```

Besides defining the model, we also extract the events from the model for later use.

Now we can ask the system about the properties that the model has after updating. There are two ways in which we can do this. We can use formulas with an update operator, or we can update `hexa` with `ann_look` to get a new model. For now, we will go with the former, but later on we will show the latter.

Lets say that we want to express that after Ann looks at her card, that she knows which card she has. For this, we will have to use the constructor `Update`, which takes an action model, a list of events, and a formula to test in the new model.

```
supports hexa ws (Update ann_look ann_events (knows a (a0 \\/ a1 \\/ a2)))
```

As one would expect, this returns `True`. We can also test if this is common knowledge among all the agents. Since this line is a little long, we will split it across multiple lines. In ghci we can do this by eclosing the line with `:{` and `:}` on their own separate lines.

```
:{
supports hexa ws (knows (iter (a .+ b .+ c))
                        (Update ann_look ann_events (knows a (a0 \\/ a1 \\/ a2))))
:}
```

Which yet again returns `True`. Now, it is also important to check that Bill does not learn Ann's card from the announcement.

```
supports hexa ws (Update ann_look ann_events (knows b (a0 \\/ a1 \\/ a2)))
```

Luckily, this returns `False`. It is important that Bill now wonders which card Ann has.

```
supports hexa ws (Update ann_look ann_events (wonder b (a0 \\/ a1 \\/ a2)))
```

Which returns `True`, as expected. If fact, this is even a public issue between Bill and Carol:

```
supports hexa ws (Update ann_look ann_events (wonder (iter (b .+ c)) (a0 \\/ a1 \\/ a2)))
```

As mentioned before, we can also create a new model using an action model and a static model. For this we can use the `productUpdate` function from the `LCCI.Evaluation` module. There is similar function for making an updated state, which is the `updatedState` function. For example, if we wanted to transform our model into a model in which every agent knew their card, we could create that model by updating it with action models created by the `privateInform` function. Here we would need to use the function `snd` to extract the action model and throw away the name.

```
hexa1 = productUpdate hexa $ snd ann_look
ws1 = worlds hexa1
hexa2 = productUpdate hexa1 $
            snd $ privateInform "read_b" [b] [a, c] [] [Prop b0, Prop b1, Prop b2]
ws2 = worlds hexa2
hexa3 = productUpdate hexa2 $
            snd $ privateInform "read_c" [c] [a, b] [] [Prop c0, Prop c1, Prop c2]
ws3 = worlds hexa3
```

Now we can use this new model to for example check if Bill knows his card.

```
supports hexa3 ws3 (knows b (b0 \\/ b1 \\/ b2))
```

Which, as one would expect, gives us back `True`.

Now we have seen how the update operator works in `LCCI.hs`, we will take a look at defining action models.

Swapping cards

Bill and Carol are a bit mischievous and decided to swap their cards. This action can also be modelled within `LCCI.hs`, since it also implements the substitutions from LCCI as well. Defining an action model is similar to defining a static model, so here we will have to define the events first.

The swapping of cards sounds like it would only need one event, but it actually will need a bit more. The method we will use will require 3 events, one for Bill and Carol swapping cards 0 and 1, one for them swapping 0 and 2, and one for them swapping 1 and 2. Who holds which card is not important for this model. We will assume that all agents are disinterested in which event actually happens. The reasoning for this is that, if the agents were interested in which card one of the two had before the action, they will be interested after the action has been executed, but if they were not interested, the swapping cards will have no effect on their state maps. This gives us the following events and state maps.

```
e1, e2, e3 :: Event
e1 = Event 1
e2 = Event 2
e3 = Event 3

ei :: StateMap Event
ei = Map.fromList [(e1, issue [state [e1, e2, e3]]),
                   (e2, issue [state [e1, e2, e3]]),
                   (e3, issue [state [e1, e2, e3]])]

em :: Map.Map Atomic (StateMap Event)
em = Map.fromList [(a, ei), (b, ei), (c, ei)]
```

Besides needing state maps for the event, we will also need preconditions for the events. The precondition for swapping card 0 and card 1 is that the one agent has card 0 and the other has card 1. Since the cards are all unique, we won't have to worry about the agents swapping the same card. This gives us the following preconditions.

```
pre :: Map.Map Event Formula
pre = Map.fromList
        [ (e1, (b0 /\ c1) \/ (b1 /\ c0))
        , (e2, (b0 /\ c2) \/ (b2 /\ c0))
        , (e3, (b1 /\ c2) \/ (b2 /\ c1))
        ]
```

That only leaves the substitutions, which work similarly to the preconditions, but have a more complex structure. Instead of mapping every event to a formula, they map every event to a substitution, which maps some propositional atoms to a formula. This is also why `LCCI.hs` has such a clear divide between propositional atoms and formulas. We can make substitutions using the `Substitution.fromList` function in a natural manner.

```
sub :: Map.Map Event Substitution
sub = Map.fromList
        [ (e1, Substitution.fromList [(b0, Prop c0), (b1, Prop c1),
                                      (c0, Prop b0), (c1, Prop b1)])
        , (e2, Substitution.fromList [(b0, Prop c0), (b2, Prop c2),
                                      (c0, Prop b0), (c2, Prop b2)])
        , (e3, Substitution.fromList [(b1, Prop c1), (b2, Prop c2),
                                      (c1, Prop b1), (c2, Prop b2)])
        ]
```

Now we have everything to put together our action model.

```
swap_bill_carol :: UpdateModel
swap_bill_carol = UpdateModel (Set.fromList [e1, e2, e3]) em pre sub
swap_events :: [Event]
swap_events = Set.toList $ events $ swap_bill_carol
```

We can use this action model to create an updated model, but if we want to use it in formulas, we will also need to give it a name. We can do this by wrapping it in a pair with a string.

```
swap_bill_carol' :: (String, UpdateModel)
swap_bill_carol' = ("swap(b,c)", swap_bill_carol)
```

Now we can use this version in a formula as we did before. We can for example say that, after Bill has looked at his card and then swapped, that he knows which card he does not have.

```
:{
supports hexa ws
        (Update (privateInform "read_b" [b] [a, c] [] [Prop b0, Prop b1, Prop b2])
               [Event 1, Event 2, Event 3]
               (Update swap_bill_carol' swap_events
                      (knows b (neg b0 \\/ neg b1 \\/ neg b2))))
:}
```

Here the system gives us `True`, as expected.

A.1.5 Other Uses of Formulas

Besides checking is a formula is supported, there are also more interesting things that `LCCI.hs` can do with formulas. The first of these is `isDeclarative`, which, as the name suggests, tests if a certain formula is declarative. It only takes the formula to test for. So for example, we can test if "Ann wonders what her card is" is declarative.

```
isDeclarative (wonder a (a0 \\/ a1 \\/ a2))
```

Since this formula is declarative, the system also correctly answers with `True`. We can also try it out on a question, such as "Does Ann, Bill, or Carol have card number 0?".

```
isDeclarative (a0 \\/ b0 \\/ c0)
```

To which the system correctly responds with `False`. If this example would have been more interesting, then one might have wondered what the resolutions for this formula were. Luckily, `LCCI.hs` can also calculate this for us, using the function `resolutions`. This function takes two arguments. The first is the maximum number of worlds for which we need to calculate the resolution, and the second is the formula that you want the resolutions for. Since this formula does not have a non-declarative iteration, the first argument is ignored, so we can pick any number.

```
resolutions 0 (a0 \\/ b0 \\/ c0)
```

This correctly gives us back the three propositional atoms in the formula. Besides being able to calculate the resolutions for a formula, it can also simplify (shorten) formulas using the function `simplify`, and convert them into a LaTeX version using the `toTex` function from the `LCCI.Util.Tex` module. More importantly, it can also calculate the IE-PDL equivalent of every LCCI formula using the functions in the `LCCI.Reduce` module.

The `LCCI.Reduce` module has as goal to implement the algorithm as laid out in Theorem 4.24, including the T and K functions from Definitions 4.8 and 4.9. This is also the module that was used in Section 4.7 to calculate these reductions. The main functions in this package are the `reduce` and `reduceStep` functions. Both of these reduce a formula, the first does it fully, and the latter does one step from the proof, but works from the outside in instead of the inside out. Both functions take two arguments, a number and a formula, where the number represents the maximum number of worlds that the formula needs to be reduced for, and is used in the calculation of the resolutions of a formula. The second argument is the formula that needs to be reduced.

As an example of this, this call is similar to the one used for the public announcement in Section 4.7:

```
alpha = Prop $ proposition "alpha"
beta = Prop $ proposition "beta"
:{
reduceStep 100 (Update (publicRaise "" [a, b, c] [alpha]) [Event 1]
                       (knows (iter (a .+ b .+ c)) beta))
:}
```

We will probably want to save this result. For that, `ghci` has the special variable `it`, which we will reassign to a different variable.

```
reduced = it
prettyPrint reduced
```

This formula is a lot more complicated then the one given in Section 4.7. This is because the formula in Section 4.7 is simplified, so we will also inspect the simplified formula.

```
prettyPrint $ simplify reduced
```

This ends our usage of the implementation. We will end this chapter with a discussion of what `simplify` does and show that this is correct.

A.2 Simplify

The function `simplify`, and its helper function `simplifyStep` can rewrite formulas in such a way that the formulas become (hopefully) easier to understand. We wanted to spend some time on explaining what this function does, and showing that it is indeed correct. To show this, we will go over what it does for each type of formula.

When `simplifyStep` gets a formula, it first checks to see if it is a negation, $\neg\varphi$. If φ is either $\top$ or $\bot$, then it returns the other (since they are each others negation). If φ is itself a negation, so the formula as a whole is a double negation, then it tries to apply the double negation elimination rule if φ is declarative. In all other cases, it will try to simplify φ and wrap the result in a negation. All of these rules either come from the proof system, or are abbreviations, so for negations, `simplifyStep` will give an equivalent formula.

If the formula is a conjunction, it will test if one of the conjuncts is $\bot$. If this is the case, then the function will return $\bot$, since the conjunction is false in any case. Then it will test if $\top$ is one of the conjuncts, and if it is, it will be removed from the conjunction. If neither $\top$ nor $\bot$ is a conjunct, then it will simplify all the conjuncts and remove duplicates. So this will also always lead to an equivalent formula.

If the formula is a disjunction or an inquisitive disjunction, `simplifyStep` will do the same thing as for a conjunction, but with the roles for $\top$ and $\bot$ reversed.

If the formula is a conditional, then `simplifyStep` will first test if the consequent is $\bot$, and if so, return the negation of the antecedent, following the definition for negation. If the consequent is not $\bot$, it will simplify both the antecedent and the consequent. The function executes this second step also if the formula is a bi-implication.

For the modal operators, the function becomes a lot more complex. The first thing that it tests for, is if the formula after the modal operator is $\top$, in which case it returns $\top$, since the formula is supported in all cases. Then it tests if the program is $?\bot$ or a sequence that ends with $?\bot$. In both those cases it also returns $\top$, since any program ending with $?\bot$ ends in the empty state, and thus every formula after it is supported there. If neither of these two cases are true, then it will try to simplify the program that is inside the modal operator.

For atomic programs, `simplifyStep` simply returns the program, since it cannot be further simplified. When it encounters a test, it will try to simplify the formula that is being tested. When it encounters a sequence, it will first try to cut it off after the first $?\bot$. If there are no $?\bot$, it will try to remove all the $?\top$ except for the last one, since testing for $\top$ always returns the current state and its subsets. If neither of these actions can be done, it will simplify the elements of the sequence itself.

The section for the choice operator in `simplifyStep` is the most complicated one. First, it tests if every element is either $?\bot$ or a sequence that ends with $?\bot$. If this is the case, then the last program executed will always be $?\bot$, so the function returns that. Then it will test if all the options are the same, and if so, return that one option. If that is not the case, then it tries to remove any (sequence that ends with) $?\bot$. This is safe to do, since it first already tested if there are other elements in the program. After that, it checks if for some element π, π^* is also in the program. If so, it will remove the element from the options. This is safe from the fact that $R_\pi \subseteq R_{\pi^*}$, so any state that is in $R_{\pi\cup\pi^*}$ is also in R_{π^*}. If that is also not the case, it tries to use the equivalence from Lemma A.1, $[\pi_1; \bigcup_{\pi_i\in B}\pi_i; \pi_n]\varphi \dashv\vdash_m [\bigcup_{\pi_i\in B}(\pi_1;\pi_i;\pi_n)]\varphi$ to push the choice into a sequence. If all of these options failed, it will simply try to simplify all of the elements in the choice, and return that.

Lastly, when it encounters unbounded iteration, it will test if the program being iterated is a test, and if so, return that, since $R_{?\varphi^*} = R_{?\varphi}$ for all formulas φ. If that is not the case, it will return an iterated version of the simplified inner program.

That only leaves us with the update operator. When it encounters the update operator it checks if the set of events is empty. If it is, the function returns $\top$, since that formula is then always supported. If there are events being executed in the operator, it will return the same update operator, but it will simplify the formula in its scope.

For most of the actions that `simplifyStep` undertakes it is easy to see or proof that they are sound, with the exception of the equivalence for choice. Therefore, we will prove that one equivalence.

Lemma A.1. *For any number of worlds m,*

- $[\pi; \bigcup_{\pi_i\in B}\pi_i; \pi']\varphi \dashv\vdash_m [\bigcup_{\pi_i\in B}(\pi;\pi_i;\pi')]\varphi$, *and*
- $[\![\pi; \bigcup_{\pi_i\in B}\pi_i; \pi']\!]\varphi \dashv\vdash_m [\![\bigcup_{\pi_i\in B}(\pi;\pi_i;\pi')]\!]\varphi$

where B is a set of programs.

Proof. We will use the proof system to show this. Since we will only be using axioms that are available to both operators, we will show the two at the same time, using $[\![]\!]$ as an example. We will also fix some arbitrary number of worlds m.

First of all, by sequence we know that $[\![\pi; \bigcup_{\pi_i\in B}\pi_i; \pi']\!]\varphi \dashv\vdash_m [\![\pi]\!][\![\bigcup_{\pi_i\in B}\pi_i]\!][\![\pi']\!]\varphi$. And by choice we get that $[\![\pi]\!][\![\bigcup_{\pi_i\in B}\pi_i]\!][\![\pi']\!]\varphi \dashv\vdash_m [\![\pi]\!]\left(\bigwedge_{\pi_i\in B}[\![\pi_i]\!][\![\pi']\!]\varphi\right)$. By the distribution of $[\![\pi]\!]$ over $\wedge$ we get $[\![\pi]\!]\left(\bigwedge_{\pi_i\in B}[\![\pi_i]\!][\![\pi']\!]\varphi\right) \dashv\vdash_m \bigwedge_{\pi_i\in B}[\![\pi]\!][\![\pi_i]\!][\![\pi']\!]\varphi$. Then, by using sequence and choice again, we get $\bigwedge_{\pi_i\in B}[\![\pi]\!][\![\pi_i]\!][\![\pi']\!]\varphi \dashv\vdash_m [\![\bigcup_{\pi_i\in B}(\pi;\pi_i;\pi')]\!]\varphi$.

Since we have shown that this holds for an arbitrary number of worlds, we can conclude that it holds for all $m \in \mathbb{N}$. □

Since all `simplify` does is call `simplifyStep` until it has no effect, this function is also sound with respect to the semantics of LCCI.

This concludes our discussion of the simplification functions.

A.3 Conclusion

This chapter showed how to use `LCCI.hs`, the software implementation that accompanies this thesis. It also discussed the simplification algorithm used, and showed that it is sound with respect to the semantics.

Bibliography

[1] A. Baltag, L. S. Moss and S. Solecki. ‘The logic of public announcements, common knowledge, and private suspicions’. In: *Readings in Formal Epistemology*. Cham: Springer, Cham, 2016, pp. 773–812.

[2] I. Ciardelli. ‘Modalities in the realm of questions: Axiomatizing inquisitive epistemic logic’. In: *Advances in Modal Logic 10, invited and contributed papers from the tenth conference on "Advances in Modal Logic," held in Groningen, The Netherlands, August 5-8, 2014*. 2014, pp. 94–113.

[3] I. Ciardelli. ‘The dynamic logic of stating and asking: A study of inquisitive dynamic modalities’. In: *Logics in Artificial Intelligence*. Berlin, Heidelberg: Springer Berlin Heidelberg, Aug. 2017, pp. 240–255.

[4] I. Ciardelli, J. Groenendijk and F. Roelofsen. *Inquisitive Semantics*. Oxford Surveys in Semantics and Pragmatics. Oxford: Oxford University Press, 2018.

[5] I. Ciardelli, J. Groenendijk and F. Roelofsen. ‘On the semantics and logic of declaratives and interrogatives’. In: *Synthese* 192.6 (2013), pp. 1689–1728.

[6] I. Ciardelli and F. Roelofsen. ‘Inquisitive dynamic epistemic logic’. In: *Synthese* 192.6 (Mar. 2014), pp. 1643–1687.

[7] I. Ciardelli and F. Roelofsen. ‘Issues in epistemic change’. In: *European Epistemology Network Meeting*. Madrid: John Wiley & Sons, Ltd, July 2014.

[8] H. C. Doets and D. J. N. van Eijck. *The Haskell Road to Logic, Maths and Programming, Second Edition*. College Publications, Dec. 2012.

[9] R. Fagin et al. *Reasoning about Knowledge*. Cambridge, Massachussetts: The MIT Press, 1995.

[10] R. Goldblatt. ‘An abstract setting for Henkin proofs’. In: *Mathematics of Modality*. Stanford: CSLI Publications, 1993, pp. 191–212.

[11] R. Goldblatt. *Logics of Time and Computation*. Menlo Park, Calif.: CSLI Press, 1992.

[12] D. Harel, D. Kozen and J. Tiuryn. ‘Dynamic Logic’. In: *Handbook of Philosophical Logic*. Ed. by D. M. Gabbay and F. Guenthner. Dordrecht: Springer, Dordrecht, 2001, pp. 99–217.

[13] M. Lipovača. *Learn you a Haskell*. 2011. URL: `http://learnyouahaskell.com`.

[14] R. Parikh and P. Krasucki. ‘Levels of knowledge in distributed systems’. In: *Sadhana* 17.1 (Mar. 1992), pp. 167–191.

[15] V. Punčochář. ‘A generalization of inquisitive semantics’. In: *Journal of Philosophical Logic* 45.4 (July 2015), pp. 399–428.

[16] V. Punčochář and I. Sedlár. ‘Inquisitive propositional dynamic logic’. Manuscript.

[17] G. Renardel de Lavalette, B. P. Kooi and R. Verbrugge. 'Strong completeness and limited canonicity for PDL'. In: *Journal of Logic, Language and Information* 17.1 (Sept. 2007), pp. 69–87.

[18] J. van Benthem, J. van Eijck and B. P. Kooi. 'Logics of communication and change'. In: *Information and Computation* 204.11 (Nov. 2006), pp. 1620–1662.

[19] H. P. van Ditmarsch. 'Knowledge Games'. PhD thesis. Groningen: ILLC Dissertation Series 2000-06., Nov. 2000.

[20] H. P. van Ditmarsch et al. 'On the logic of lying'. In: *Games, Actions and Social Software.* Ed. by J. van Eijck and R. Verbrugge. Berlin, Heidelberg: Springer, Berlin, Heidelberg, 2012, pp. 41–72.

[21] J. van Eijck and Y. Wang. 'Propositional dynamic logic as a logic of belief revision'. In: *Logic, Language, Information and Computation.* Ed. by W. Hodges and R. de Queiroz. Berlin, Heidelberg: Springer, Berlin, Heidelberg, July 2008, pp. 136–148.

[22] T. van Gessel. 'Action models in inquisitive logic'. In: *Synthese* 21.3–4 (Aug. 2018), pp. 1–41.