POPULATIONAL ANNOUNCEMENT LOGIC (PPAL)

Vitor Pereira Machado

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientador: Mario Roberto Folhadela Benevides

Rio de Janeiro
Julho de 2016

POPULATIONAL ANNOUNCEMENT LOGIC (PPAL)

Vitor Pereira Machado

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

Prof. Mario Roberto Folhadela Benevides, Ph.D.

Prof. Valmir Carneiro Barbosa, Ph.D.

Prof. Carla Amor Divino Moreira Delgado, D.Sc.

Prof. Bruno Lopes Vieira, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
JULHO DE 2016

Machado, Vitor Pereira

Populational Announcement Logic (PPAL)/Vitor Pereira Machado. – Rio de Janeiro: UFRJ/COPPE, 2016.

X, 52 p.: il.; 29, 7cm.

Orientador: Mario Roberto Folhadela Benevides

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2016.

Referências bibliográficas: p. 39 – 42.

1. knowledge. 2. fuzzy. 3. logic. I. Benevides, Mario Roberto Folhadela. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

For my mother and father, whose
love and support made everything,
including this work, possible.

Acknowledgments

I would like to express the deepest appreciation to my supervisor and friend Professor Mario Roberto Folhadela Benevides, for his good spirits, guidance and enthusiasm in regards to research. Without his support this work would not have been possible.

I also wish to thank Professor Hans van Ditmarsch for his inspirational work in Dynamic Epistemic Logic. His research served as a solid foundation for this work.

A very special thank you goes out to my girlfriend Kalisia Autuori, for her patience, support and encouragement. She gives me strength to keep going.

I must also extend my gratitude to my friends André Albuquerque, Gabriel Almeida and Luan Garrido for their support and companionship along the way.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

LÓGICA DE ANÚNCIOS POPULACIONAIS (PPAL)

Vitor Pereira Machado

Julho/2016

Orientador: Mario Roberto Folhadela Benevides

Programa: Engenharia de Sistemas e Computação

Apresenta-se nesta dissertação a Lógica de Anúncios Populacionais (Populational Announcement Logic - PPAL), uma variante da Lógica de Anúncios Públicos (Public Announcement Logic - PAL) com semântica fuzzy, onde ao invés de agentes específicos temos populações e grupos. A semântica da lógica de anúncios é definida e exemplos são dados. Além disso, uma biblioteca e verificador de modelos em Java implementando essa lógica são discutidos.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

POPULATIONAL ANNOUNCEMENT LOGIC (PPAL)

Vitor Pereira Machado

July/2016

Advisor: Mario Roberto Folhadela Benevides

Department: Systems Engineering and Computer Science

Populational Announcement Logic (PPAL), is a variant of the standard Public Announcement Logic (PAL) with a fuzzy semantics, where instead of specific agents we have populations and groups. The semantics and the announcement logic are defined, and examples are provided. Also, a Java open-source library and a model checker implementing this language are discussed.

Contents

List of Figures

Chapter 1

Introduction

1.1 Previous developments

Epistemic logics are formal logical systems that deal with the representation of knowledge.

Sowa, inventor of conceptual graphs and an important figure in artificial intelligence, gave us in [1] one possible definition for knowledge representation, which is "the application of logic and ontology to the task of constructing computable models for some domain". And it goes beyond the realm of artificial intelligence and computing, also posing some important questions in the area of philosophy [2]. In fact, some view certain aspects of it as "applied philosophy" [3].

As noted in [4] however, knowledge representation serves little purpose without ways to reason about said knowledge, and epistemic logics are attempts to do just that.

While some of the basic ideas date back to ancient Greece times [5], C. I. Lewis published back in 1918 the book "Survey of Symbolic Logic", possibly the first systematic approach to the matter [6].

Later in 1951, Von Wright's "An Essay in Modal Logic" introduced a concept called "modality" to better represent the semantics of knowledge [7], providing operators such as "usually" and "always" to qualify propositional statements. It is considered the first important work regarding modal logic.

A further development is Dynamic Logic, which aims to reason about actions and their effects [8]. Dynamic Epistemic Logic (DEL) is conceived to reason about actions that change agents or groups of agents' knowledge and beliefs. It is not interested in justifying whether something is knowledge or not, but in inferring something from said knowledge.

The first Dynamic Epistemic Logic was proposed independently by [9] and [10], and is called Public Announcement Logic (PAL), a logic which allows simple public

actions which alter the state of knowledge for its agents. Later Baltag et al. proposed in [11, 12] the Action Model Logic approach, allowing for more complex reasoning.

1.2 Motivation and objectives

This work aims to specify a variation of PAL where knowledge is represented across populations and groups of populations, instead of discrete agents as is usually done. The motivation behind this is to provide a framework that is capable of dealing with applications where one intends to reason about evolution of knowledge across populations instead of individual agents. It is specially designed for model checking, because it is not necessary to specify agents (or populations in this case) beforehand, and instead these populations can be defined and evolved during the model checker's execution, using the announcement operator of the language.

To achieve this we have introduced and defined a "fuzzyfied" variant of PAL that allows partial announcements to its agents (called "populations" and "groups"). We formalize the model, language and semantics, and give an example where an initial model is given and some announcements are made which modify it, in order to illustrate its applicability in a scenario where knowledge evolves across populations as announcements are made.

Consider this situation: a famous politician is under suspicion of money laundering and bribery. Elections are coming up, and this politician expects to be re-elected, but people are less willing to vote for him if they are aware of both charges against him. Not everyone in the population is aware of the charges, however. We will expand this example on chapter 4 to show how announcements to fractions of populations can be modelled and how these models evolve, and also how announcements can result in different knowledge for certain parts of groups.

We also show a proof that the satisfiability problem for any formula of the language is decidable for finite models, which is an important result for model checking, as it ensures that it is not possible to input a formula which could result in a infinite loop in the program.

We then discuss the library and model checker implementation, and also provide an input model and usage examples.

We finish with the conclusion that the main advantage of the Populational Announcement Logic (PPAL) over Dynamic Epistemic Logic (DEL) and Public Announcement Logic (PAL) is the flexibility to work with non-priorly defined agents. It is possible to define an initial population and to evolve it in a more natural way that does not require a formal initial definition of every agent on the system. Therefore we believe it can be a more appropriate logic for practical use in situations where we want to reason about the knowledge among populations, as it would be

much easier to control and evolve a knowledge model.

A paper [13] derived from this work was accepted for the Encontro Nacional de Inteligência Artificial e Computacional (ENIAC), a conference held in Natal, RN - Brazil. It is published in the Biblioteca Digital Brasileira de Computação (BDBComp) [1].

1.3 Roadmap

This work is organized as follows: In Chapter 2 we recall classical propositional logic with a bit of history and a brief explanation aided by an example; We then talk about fuzzy and modal logics, both extensions to classical propositional logic.

After that, we present Multi-Agent Epistemic Logic, where we really start to deal with the topic of knowledge. After that we continue into Public Announcement Logic (PAL), showing a classical example to illustrate its use.

In Chapter 3, we introduce Populational Announcement Logic (PPAL), explain its purpose and differences with respect to PAL and define its language, semantics and model.

Chapter 4 provides two basic examples of PPAL, showing how the model evolves after an announcement. We also prove it is decidable, an important result for model checking as previously mentioned.

Chapter 5 is about the model checker, its implementation and usage examples.

Chapter 6 provides a conclusion, some final remarks and future works.

[1] `http://www.lbd.dcc.ufmg.br/bdbcomp/servlet/Trabalho?id=23755`

Chapter 2

A background of logic

We begin this Chapter with a reminder of classical propositional logic and a brief introduction to many-valued logic (fuzzy logic). Then we give a brief presentation of a logic with modalities (modal logic). After that, we provide an overview of epistemic logic, outlining its early development in different fields such as philosophy, artificial intelligence and computer science. Finally, we give a brief introduction to Public Announcement Logic (PAL) with an example to illustrate its usage.

2.1 Classical propositional logic

The development of propositional logic has first begun in Egypt, but has only been turned into a formal logic by the Greek philosopher Chrysippus [14].

The language for Propositional Logic consists of a set of countably many propositional symbols, and the Boolean connectives $\neg$ (negation operator), $\wedge$ (conjunction), $\vee$ (disjunction) and $\rightarrow$ (implication). Its semantics is defined by truth tables for each of the connectives, which evaluates each possible input combination to a *true* or $false$ result.

The Boolean algebra introduced by George Boole in 1847 [15] is closely related to propositional logic, and is commonly used today in mathematics and computer science, and is very well known and developed.

Consider the implication truth table in table 2.1 for the following example. Take the statement "If it is summer then it is hot". If we call "summer" proposition S, and "hot" proposition H, the statement is represented in propositional logic as the expression $S \rightarrow H$. If we know, for instance, that it is indeed summer which means $S = true$, the expression can only be true if it is hot ($H = true$). If it were summer and not hot, it would mean the statement itself is false. Note that, if it is not summer, nothing can be said about whether it is hot or not as neither case would violate the statement, and therefore the expression $S \rightarrow H$ remains *true* whether H is *true* or $false$.

P	**Q**	**P → Q**
true	*true*	*true*
true	*false*	*false*
false	*true*	*true*
false	*false*	*true*

Table 2.1: Implication truth table.

2.2 Fuzzy logic

The term "fuzzy logic" was first coined by [16], and is also known as infinite-valued logic or many-valued logic. It is a kind of logic whose propositions can assume more than two values, usually between 0 and 1 inclusive.

Fuzzy logic has been widely used in control systems for many practical applications, such as [17]. The advantages of these controllers as opposed to common "PID controllers" (proportional-integral-derivative controllers) are the easy way of translating expert knowledge into controller rules, even when an actual mathematical model for the process does not exist, and scalability by adding new rules [18].

For instance, if proposition "water_level" has value 0.5 you might consider it is "somewhat full". Linguistic values and hedges ("full" and "somewhat" respectively in the previous example) can be used to discretize fuzzy variables into a human-readable format.

There are many different formal systems of fuzzy logics, we will take Łukasiewicz logic for the following definitions.

2.2.1 Language and semantics

Definition 1 *The language for the Łukasiewicz logic consists of a set Φ of countably many proposition symbols, the connectives $\neg$, $\wedge$, $\vee$ and $\rightarrow$, and "strong" variations for the conjunction and disjunction operators, $\otimes$ and $\oplus$, respectively. The formulae are defined as follows in BNF (Backus-Naur Form) notation:*

$$\varphi ::= p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \rightarrow \varphi_2 \mid \varphi_1 \otimes \varphi_2 \mid \varphi_1 \oplus \varphi_2$$

where $p \in \Phi$.

In this logic, for instance, the implication is defined as

$$f_{\rightarrow}(x, y) = \min\{1, 1 - x + y\} \tag{2.1}$$

Take the statement "If the tank is full then open the valve". We will call the proposition representing the tank being full as F and the valve being open as O,

therefore the statement becomes the expression $F \to O$. Now, suppose we are receiving data from a sensor in the tank saying that it is 0.5 full. From figure 2.1, we can see that for the statement to be completely true, the valve must be at least 0.5 open.

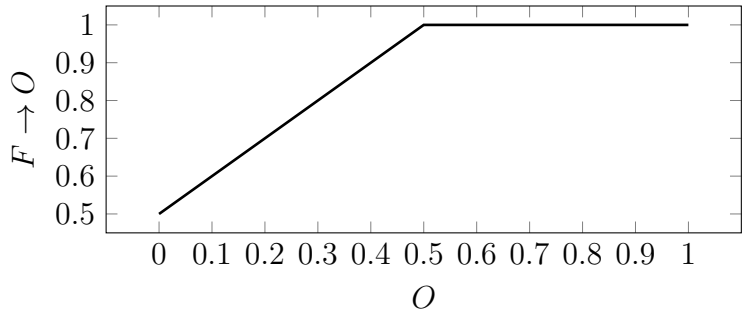

Figure 2.1: Values assumed by the expression $F \to O$ according to the Łukasiewicz implication shown in equation 2.1, given $x = 0.5$ for all possible values of y.

The other operators in the Łukasiewicz logic are defined as follows:

- Negation: $f_{\neg}(x) = 1 - x$;
- Weak conjunction: $f_{\wedge}(x, y) = \min\{x, y\}$;
- Weak disjunction: $f_{\vee}(x, y) = \max\{x, y\}$;
- Strong conjunction: $f_{\otimes}(x, y) = \max\{0, x + y - 1\}$;
- Strong disjunction: $f_{\oplus}(x, y) = \min\{1, x + y\}$.

2.3 Modal logic

Modern formal modal logics date back to 1932 when Lewis introduced in [19] five different logic systems named $S1$ through $S5$. Modal logics are a family of logics consisting of extensions to the classical propositional logic that contains operators of modality, that is operators that qualify a given statement.

Classical modal logic contains the operators "necessarily" and "possibly", which are usually written $\Box$ and $\Diamond$ respectively. For instance, $\Box p$ reads "Necessarily p".

2.3.1 Language and semantics

This section presents the usual language and semantics of modal logic [20]. The model is usually called a "frame", which we will denote $\mathscr{F}$.

Definition 2 *A modal logic frame is a tuple $\mathscr{F} = (W, \sim)$ where*

- *W is a non-empty set of states;*
- *$\sim$ is a binary relation over W;*

Definition 3 *A modal logic model is a pair $\mathscr{M} = (\mathscr{F}, V)$, where $\mathscr{F}$ is a frame, V is a valuation function $V : \Phi \to 2^W$ and Φ is a set of countably many proposition symbols.*

The binary relation $\sim$ is called the "accessibility relation". For example, $w \sim u$ means that the state u is accessible from state w. That is to say, someone "living" in state w would see u as a plausible state as well.

Then, we can define the truth of formulas in a state of a frame (we use "iff" as shorthand for "if and only if"):

Definition 4 *The language for modal logic consists of a set Φ of countably many proposition symbols, the Boolean connectives $\neg$ and $\wedge$, and modalities $\Box$ and $\Diamond$. The formulae are defined as follows in BNF notation:*

$$\varphi ::= p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \Box\varphi \mid \Diamond\varphi$$

where $p \in \Phi$.

Definition 5 *Given a modal logic model $\mathscr{M} = (\mathscr{F}, V)$, the notion of satisfaction $w \models \varphi$ is defined as follows:*

1. *$w \models p$ iff $w \in V(p)$*
2. *$w \models \neg\varphi$ iff $w \not\models \varphi$*
3. *$w \models \varphi_1 \wedge \varphi_2$ iff $w \models \varphi_1$ and $w \models \varphi_2$*
4. *$w \models \Diamond\varphi$ iff for some state u of W, it holds that $w \sim u$ and $u \models \varphi$*
5. *$w \models \Box\varphi$ iff for every state u of W, if $w \sim u$ it holds that $u \models \varphi$*

It is also possible to easily interchange the $\Box$ and $\Diamond$ operators with the following axioms:

$$\Diamond\varphi \leftrightarrow \neg\Box\neg\varphi \tag{2.2}$$

$$\Box\varphi \leftrightarrow \neg\Diamond\neg\varphi \tag{2.3}$$

2.3.2 S4 and S5 logics

The modal logic systems differ by the properties of their accessibility relations. From the logic systems Lewis defined, $S4$ and $S5$ are usually the most studied ones.

Definition 6 *The $S4$ logic is defined by the properties of*

- *Reflectivity:* $w \sim w, \forall w \in W$*;*
- *Transitivity:* $w \sim u \wedge u \sim q \rightarrow w \sim q, \forall w, u, q \in W$*;*

That is, the $S4$ logic is defined by the properties that any state is accessible from itself (Reflectivity), and that if a state is accessible via an intermediate state, then the former is also directly accessible (Transitivity. Figure 2.2).

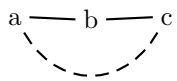

Figure 2.2: The transitive property. The relations are bidirectional.

Definition 7 *The $S5$ logic is defined by the properties of*

- *Reflectivity:* $w \sim w, \forall w \in W$*;*
- *Euclidean relation:* $w \sim u \wedge w \sim q \rightarrow u \sim q, \forall w, u, q \in W$*;*

In other words, $S5$ logic is defined by the property of reflectivity as in $S4$, and by the Euclidean relation: all accessible states are also accessible from themselves (figure 2.3). Note that the Euclidean relation implies transitivity as well.

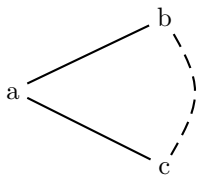

Figure 2.3: The Euclidean property. The relations are bidirectional.

2.4 Multi-agent epistemic logic

The works of von Wright [21] and Hintikka [22] are widely regarded as the first formal approaches to epistemic logics, that is, logics seeking to represent knowledge and belief (the latter sometimes referred to as "doxastic" logic).

Philosophers are also interested in epistemic logic, as a part of the broader field of epistemology [23]. One of the most studied themes in the field is that of "Closure", that is, whether or not an agent is always capable of deducing logical consequences [24]. In other words, if p and $p \rightarrow q$, is an agent always capable of deducing q? This problem also relates to the "problem of logical omniscience", which states that if an agent knows every formula in a set, and a statement follows from this set, then the agent is always capable of deducing the statement [25]. Some other important themes are: the "Moore sentence" [26], which consists of the paradoxical sentence "p is true but I do not believe p", a logically consistent sentence but an apparent contradiction; and the "Fitch paradox" [27], a result that states that there must be some truths which are not knowable [28].

Epistemic logic's main characteristic is the operator K, the "knowledge operator". It is important to note that this "knowledge" refers not to one's knowledge on how to perform certain tasks, or simply being aware of someone or something. It is about propositional knowledge, that is, knowing that something is true [29].

We will focus here on Multi-Agent Epistemic Logic, which has been investigated in Computer Science [30] to represent and reason about agents or groups of agents' knowledge and beliefs.

2.4.1 Language and semantics

This section presents the Multi-Agent Epistemic Logic $S5'$. It is inspired on the work presented in [30].

Definition 8 *The language for $S5'$ consists of a set Φ of countably many proposition symbols, a finite set $\mathcal{A}$ of agents, the Boolean connectives $\neg$ and $\wedge$, and a modality K_a for each agent a. The formulas are defined as follows in BNF notation:*

$$\varphi ::= p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \rightarrow \varphi_2 \mid K_a\varphi$$

where $p \in \Phi$, $a \in \mathcal{A}$.

The intended meaning of the modal operator $K_a\varphi$ is "agent a knows φ".

Definition 9 *A multi-agent epistemic frame is a tuple $\mathcal{F} = (W, \sim_a)$ where*

- *W is a non-empty set of states;*

- $\sim_a$ *is a binary relation over* W*, for each agent* $a \in \mathcal{A}$*;*

Definition 10 *A multi-agent epistemic model is a pair* $\mathcal{M} = (\mathcal{F}, V)$*, where* $\mathcal{F}$ *is a frame and* V *is a valuation function* $V : \Phi \to 2^W$*. We call a rooted multi-agent epistemic model* $(\mathcal{M}, s)$ *an epistemic state, where* $s \in W$*.*

In most applications of multi-agent epistemic logic the relations $\sim_a$ are equivalence relations. In this work we only deal with the case where $\sim_a$ are equivalence relations, for each agent a.

Definition 11 *Given a multi-agent epistemic model* $\mathcal{M} = \langle S, \sim_a, V \rangle$*, the notion of satisfaction* $\mathcal{M}, s \models \varphi$ *is defined as follows:*

1. $\mathcal{M}, s \models p$ *iff* $s \in V(p)$
2. $\mathcal{M}, s \models \neg\phi$ *iff* $\mathcal{M}, s \not\models \phi$
3. $\mathcal{M}, s \models \phi \wedge \psi$ *iff* $\mathcal{M}, s \models \phi$ *and* $\mathcal{M}, s \models \psi$
4. $\mathcal{M}, s \models \phi \vee \psi$ *iff* $\mathcal{M}, s \models \phi$ *or* $\mathcal{M}, s \models \psi$
5. $\mathcal{M}, s \models \phi \to \psi$ *iff* $\mathcal{M}, s \not\models \phi$ *or* $\mathcal{M}, s \models \psi$
6. $\mathcal{M}, s \models K_a\phi$ *iff* $\forall s' \in S : s \sim_a s' \Rightarrow \mathcal{M}, s' \models \phi$

For an agent a, the modal operator $K_a\varphi$ has the intuitive meaning: "agent a knows φ". $K_a\varphi$ is true iff φ is true in every state agent a considers as possible.

2.4.2 Example

This example is from [29]. Suppose we have a card game with three cards: 0, 1 and 2, and three players a, b and c. Each player receives a card and does not know the other players' cards. We use propositional symbols $0_x, 1_x, 2_x$ for $x \in \{a, b, c\}$ meaning "player x has card 0, 1 or 2". We name each state by the cards that each player has in that state, for instance 012 is the state where player a has card 0, player b has card 1 and player c has card 2. The following epistemic model represents the epistemic state of each agent (Figure 2.4).

$Hexa1 = \langle S, \sim, V \rangle$:

- $S = \{012, 021, 102, 120, 201, 210\}$

  $\sim_a = \{(012, 021), (102, 120), (201, 210)\}$
- $\sim_b = \{(102, 201), (012, 210), (021, 120)\}$

  $\sim_c = \{(102, 012), (201, 021), (210, 120)\}$

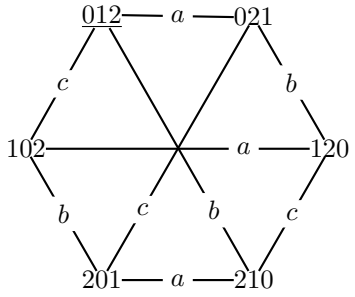

Figure 2.4: Epistemic Model $Hexa_1$

- $V(0_a) = \{012, 021\}, V(1_a) = \{102, 120\}, V(2_a) = \{201, 210\},$
- $V(0_b) = \{102, 201\}, V(1_b) = \{012, 210\}, V(2_b) = \{021, 120\},$
- $V(0_c) = \{120, 210\}, V(1_c) = \{021, 201\}, V(2_c) = \{012, 102\}$

We could, for instance, assert that agent a knows that agent b does not know which card a has. That is:

$$Hexa_1, 012 \models K_a \neg K_b 0_a \tag{2.4}$$

Because in every state that a conceives (012 and 021), agent b is not certain on which card a has. In state 012 agent b conceives a state where agent a has card 0, and one where he has card 2. And in state 021, b conceives a state where agent a has card 0 and one where he has card 1. Which is of course expected, since agent b (like the other agents) is only aware of his own card.

2.5 Public announcement logic (PAL)

The logics we have seen so far share a common trait: they are all static. The main development that gave rise to dynamic logic as currently known was dynamic modal logic, and it was conceived mainly to reason about the behavior of computer programs [31]. We will focus here however on reasoning about knowledge. Plaza introduced a public announcement logic in [9], which is the main inspiration for this work, and we talk about it in detail in this section. Later some more complex actions were made possible, such as "epistemic actions" [32] and "action models" [12], both of which we briefly mention in sections 2.6 and 2.7, respectively.

Note that DEL is mostly concerned with information known by the agents of the world, not the actual facts of the world itself. That is, it models changes in the accessibility relations connecting the states, not changes to the states themselves.

Now, we will briefly introduce the Public Announcement Logic, which extends the

previous $\mathbf{S5_a}$ logical system to the first actual Dynamic Epistemic Logic, developed by Plaza in [9].

2.5.1 Language and semantics

The formulas are essentially the same as before, with the addition of the public announcement operator:

$$\varphi ::= p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \rightarrow \varphi_2 \mid K_a\varphi \mid [\phi]\varphi$$

where $p \in \Phi$ and $a \in \mathcal{A}$.

The modal operator of public announcement $[\phi]\varphi$ has the intended meaning: "after the announcement of ϕ, φ is holds". The effect of announcing ϕ publicly is a restriction in the model to contain only states where ϕ is true. The definitions of frame and models remain the same as in the logic $\mathbf{S5_a}$. For the notion of satisfaction we have to add the following item to definition 11:

7. $\mathcal{M}, s \models [\phi]\varphi$ iff $\mathcal{M}, s \models \phi \Rightarrow \mathcal{M}|\phi, s \models \varphi$

where $\mathcal{M}|\phi$ is a new model obtained by restricting $\mathcal{M}$ to have only states where ϕ holds. It is defined as follows:

Definition 12 *Givem as epistemic model* $\mathcal{M} := \langle S, \sim_a, V\rangle$. *The model* $\mathcal{M}|\phi := \langle S', \sim'_a, V'\rangle$ *is defined as:*

- $S' := [\![\phi]\!]_{\mathcal{M}} = \{s \in S | \mathcal{M}, s \models \phi\}$
- $\sim'_a := \sim_a \cap([\![\phi]\!]_{\mathcal{M}} \times [\![\phi]\!]_{\mathcal{M}})$
- $V'(p) := V(p) \cap [\![\phi]\!]_{\mathcal{M}}$

There is also an axiomatisation for PAL, that is, a set of tautologies (formulas that are always true in the logic), called axioms, that can be used to derive all other formulas in the language. It is also possible to show that this axiomatisation is sound and complete.

Definition 13 *An axiomatisation for PAL, according to [29], is:*

- $K_a(\varphi \rightarrow \psi) \rightarrow (K_a\varphi \rightarrow K_a\psi)$ (*distribution of* K_a *over* $\rightarrow$);
- $K_a\varphi \rightarrow \varphi$ (*truth*);
- $K_a\varphi \rightarrow K_aK_a\varphi$ (*positive introspection*);
- $\neg K_a\varphi \rightarrow K_a\neg K_a\varphi$ (*negative introspection*);

- $[\varphi]p \leftrightarrow (\varphi \rightarrow p)$ (*atomic permanence*);
- $[\varphi]\neg\psi \leftrightarrow (\varphi \rightarrow \neg[\varphi]\psi)$ (*announcement and negation*);
- $[\varphi](\psi \wedge \chi) \leftrightarrow ([\varphi]\psi \wedge [\varphi]\chi)$ (*announcement and conjunction*);
- $[\varphi]K_a\psi \leftrightarrow (\varphi \rightarrow K_a[\varphi]\psi)$ (*announcement and knowledge*);
- $[\varphi][\psi]\chi \leftrightarrow [\varphi \wedge [\varphi]\psi]\chi$ (*announcement composition*);
- *From* φ *and* $\varphi \rightarrow \psi$, *infer* ψ (*modus ponens*);
- *From* φ, *infer* $K_a\varphi$ (*necessitation of* K_a).

2.5.2 Example

Continuing on the previous card game example, we are now going to make a public announcement and check the knowledge of an agent given the announcement. Consider an announcement that agent a does not have card 1 (which we represent as $\neg 1_a$), and after that we want to check if agent c knows if agent a has the card 0 ($K_c 0_a$). This can be expressed as:

$$Hexa_1, 012 \models [\neg 1_a]K_c 0_a \tag{2.5}$$

The epistemic model $Hexa_2 = Hexa_1|\neg 1_a$ representing the epistemic state of each agent after the announcement is (Figure 2.5):

$Hexa2 = \langle S', \sim'_a, V' \rangle$:

- $S' = \{012, 021, 201, 210\}$

  $\sim'_a = \{(012, 021), (201, 210)\}$
- $\sim'_b = \{(012, 210)\}$

  $\sim'_c = \{(201, 021)\}$

  $V'(0_a) = \{012, 021\}, V'(1_a) = \{\}, V'(2_a) = \{201, 210\},$
- $V'(0_b) = \{201\}, V'(1_b) = \{012, 210\}, V'(2_b) = \{021\},$

  $V'(0_c) = \{210\}, V'(1_c) = \{021, 201\}, V'(2_c) = \{012\}$

We can check if $K_c 0_a$ holds in the new model $Hexa_2$:

$$Hexa_2, 012 \models K_c 0_a \tag{2.6}$$

Now, we have to check if 0_a is true in every state connected to state 012 via the relation $\sim_c$. Since there are no other states other than 012 itself, we check only this state:

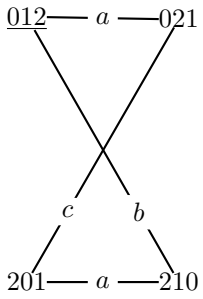

Figure 2.5: Epistemic Model $Hexa_2 = Hexa_1|\neg 1_a$

$$Hexa_2, 012 \models 0_a \tag{2.7}$$

Which is true because $012 \in V'(0_a)$, and therefore $Hexa_2, 012 \models K_c 0_a$ is also true.

In the following sections we talk a bit about alternative, and more complex, approaches to model knowledge updates.

2.6 Epistemic actions

It is possible to convey more complex changes to the model, which are called collectively "epistemic actions". Public announcements provide only a subset of what is possible with epistemic actions, and would for instance be represented as:

$$[L_{\mathcal{A}}?\varphi]\psi \tag{2.8}$$

Which should be read as "after agents $\mathcal{A}$ (all agents of the model) learn φ, it holds that ψ."

2.6.1 Language and semantics

The language $\mathcal{L}_!(\mathcal{A}, \Phi)$ can be defined as the union of the "formulas" $\mathcal{L}_!^{stat}(\mathcal{A}, \Phi)$ and the "actions" $\mathcal{L}_!^{act}(\mathcal{A}, \Phi)$ defined by, respectively:

$$\varphi ::= p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \rightarrow \varphi_2 \mid K_a\varphi \mid C_{\mathcal{B}}\varphi \mid [\alpha]\varphi$$

$$\alpha ::= ?\varphi \mid L_{\mathcal{B}}\beta \mid (\alpha \,!\, \alpha) \mid (\alpha \,¡\, \alpha) \mid (\alpha \,;\, \beta') \mid (\alpha \cup \alpha)$$

where $p \in \Phi$, $a \in \mathcal{A}$, $\mathcal{B} \subseteq \mathcal{A}$, $\psi \in \mathcal{L}_!^{stat}(gr(\alpha), \Phi)$, $\beta \in \mathcal{L}_!^{act}(\mathcal{B}, \Phi)$ and $\beta' \in \mathcal{L}_!^{act}(gr(\alpha), \Phi)$. The group $gr(\alpha)$ of an action α is defined as: $gr(?\varphi) = \emptyset$, $gr(L_{\mathcal{B}}\alpha) = \mathcal{B}$,

$gr(\alpha \,!\, \alpha') = gr(\alpha)$, $gr(\alpha \,¡\, \alpha') = gr(\alpha')$, $gr(\alpha \,;\, \alpha') = gr(\alpha')$ and $gr(\alpha \cup \alpha') = gr(\alpha) \cap gr(\alpha')$.

We will not go into details here, and the reader is referred to [29], probably the most comprehensive material on this subject and the basis for this section and section 2.7. There the reader will find the semantics definitions and examples.

Now, suppose an initial model (figure 2.6) where 0 stands for "agent a does not know whether p or $\neg p$" and 1 stands for "agent a knows whether p or $\neg p$".

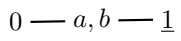

Figure 2.6: Before the epistemic action. State 1 is the real state.

It is possible to write complex expressions such as $L_{ab}(L_a?p \cup L_a?\neg p \cup !?\top)$ which stands for "agents a and b learn that a may learn the truth about fact p, although actually nothing happens (expressed by the exclamation mark)". This action increases the number of states in the system, something standard public announcements can't do, as it introduces doubt for agent b of whether agent a learned about p or not (figure 2.7).

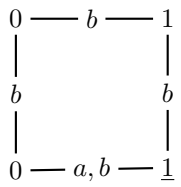

Figure 2.7: After the epistemic action. There are now four states instead of two, and the upper states represent the new possibility that agent a may have learned the truth about fact p, although agent b cannot be sure of that.

2.7 Action models

Action models are a different approach to describe epistemic actions. Their structure resembles a Kripke model, which gives them their name. Again we refer the reader to [29] for more details, and we go straight to an example also from the aforementioned work. Suppose an initial model as in figure 2.8 where 0 represents the state where $\neg p$ and 1 represents the state where p.

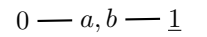

Figure 2.8: Initial epistemic model

Now consider a letter that contains information about whether p or $\neg p$ and both agents a and b are aware of it, however only agent a is going to read it. We give these two possibilities two so-called "action points": p with precondition $pre(p) = p$ and np with precondition $pre(np) = \neg p$. These preconditions are not enough to fully specify these actions, we need the relation between p and np as well. This can be seen in figure 2.9.

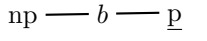

Figure 2.9: Action model for reading the letter

We can now define the computation of the cross-product between an epistemic state and an epistemic action. The resulting "factual states" of this cross-product are given by pairs resulting of the combinations of the states in the epistemic model and the action points from the action model, respecting the preconditions. In this case, the only possible pairs are $(0, np)$ and $(1, p)$, as the pairs $(0, p)$ and $(1, np)$ are excluded due to preconditions $pre(p) = p$ and $pre(np) = \neg p$, respectively.

Regarding the relations between the factual states, they are given by

$$(s_1, p_1) \sim_a (s_2, p_2) \text{ iff } s_1 \sim_a s_2 \text{ and } p_1 \sim_a p_2, \tag{2.9}$$

where a is an arbitrary agent, s_1 and s_2 are states from the epistemic model and p_1 and p_2 are action points from action model.

We can then compute the cross-product of the epistemic model with the action model, which results in figure 2.10.

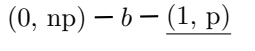

Figure 2.10: Cross-product of the epistemic model with the action model

The resulting epistemic model can be extracted from the cross-product, resulting

in figure 2.11. Now everything is distinguishable for agent a (who read the letter), and agent b is still unaware of whether p or $\neg p$. Note however, that agent b does know that agent a knows this. That is, it is true that $K_b(K_a p \vee K_a \neg p)$.

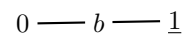

Figure 2.11: Epistemic model after the epistemic action is executed

Chapter 3

Populational announcement logic (PPAL)

Now we introduce PPAL, where we have populations and groups instead of agents. It works like a "fuzzyfied" version of PAL, where announcements act upon fractions of populations, so it does not make sense to talk explicitly of private or public announcements. It should be noticed that in the limit case where announcements are made to the whole population, the model will evolve equivalently to a PAL model with public announcements.

This allows us much more flexibility, as we do not have to define every agent in a system beforehand. Also, it allows expressing partial knowledge in a much more natural way.

As a motivation, consider the following example: A famous politician is under suspicion of money laundering and bribery. Elections are coming up, and this politician expects to be re-elected, but people are less willing to vote for him if they are aware of both charges against him. Not everyone in the population is aware of the charges, however. With PPAL we can model announcements made by a TV program which only a certain amount of the population watches, and then we can assert the overall knowledge of the population in regards to the politician. For instance, the formula

$$[l]_G^{0.3} K_G(l \wedge b) \tag{3.1}$$

represents an announcement made to a fraction of 0.3 of a group G, that the politician is in fact guilty of money laundering (l), and a subsequent knowledge assertion of whether G knows that the politician is corrupt or not (guilty of money laundering and bribery).

We will return to this example in section 4.2.

3.1 Populations and groups

Definition 14 *A population represents a collection of individuals. A population P has size $|P| \in \mathbb{Q}_{>0}$.*

It is defined as a rational number because announcements split populations into a group with two populations, and their sizes will be then fractions of the population's size.

Note that while a population represents many individuals, actual individuals are never directly expressed or referenced (as in DEL). Individuals are represented solely by the size of the population. The motivation for this comes down to the fact that we are mostly interested in representing and working with the knowledge of populations or fractions of populations, thus there is no purpose in modelling actual individuals. Next we will define the notion of groups.

Definition 15 *A group G can be empty, a population or a disjoint set of groups:*

$$G := \emptyset \mid P \mid \{G_0, G_1, \ldots, G_n\} \tag{3.2}$$

The size of a group G is defined as:

$$|G| = \begin{cases} 0 & \text{if } G = \emptyset, \\ |P| & \text{if } G = P, \\ \sum_{G_i \in G} |G_i| & \text{if } G = \{G_0, G_1, \ldots, G_n\}. \end{cases}$$

The reason for having two definitions for "groups" and "populations" is the fact that announcements operate splitting groups into populations, as will be seen in section 3.4 and in chapter 4. Having groups and populations allows for a clearer definition of announcements, because if we only had populations there would be no easy way to reference, as a whole, the same individuals that we could reference before the announcement was made and they were a single population.

3.2 Model

Since the announcements apply to fractions of a population, they work similarly to private announcements. Due to this, the many worlds approach is used. Each population has its own model, representing its current knowledge of the world:

Definition 16 *A model for a population P, $M_P = \langle T, \overset{M_P,G}{\sim}, V \rangle$ is composed of a set of states T, a valuation function that maps propositional symbols in Φ into subsets of T, $V : \Phi \to 2^T$, and a family of binary relations over T, $\overset{M_P,G}{\sim}$, for each group G known (represented) by this population.*

For example, an edge representing the doubts of population P in regards to group G would be in the relation $\overset{M_P,G}{\sim}$.

Definition 17 *A model M_G, for a group $G = \{G_0, G_1, \ldots, G_n\}$ is defined as the set of models of each of its groups. That is, $M_G = \{M_{G_0}, M_{G_1}, \ldots, M_{G_n}\}$.*

Since announcements split groups into populations, groups appearing on a branch after a split are not known to groups on other branches. That is, they will not appear on these groups' models.

3.3 Language

This sections presents the language of PPAL.

The language is as follows in BNF notation:

$$\varphi ::= p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \rightarrow \varphi_2 \mid K_G\varphi \mid B_G\varphi \mid [\varphi_1]^r_G\varphi_2$$

where $r \in U = [0, 1]$, G denotes a group and $p \in \Phi$.

The modal operators have the following intended meaning:

- $K_G\varphi$ and $B_G\varphi$: "the group G knows/believes that φ is true";
- $[\varphi_1]^r_G\varphi_2$: "φ_2 is true on group G's model after the announcement of φ_1 to fraction r of the individuals in G", where $G = \{P_1, P_2\}$ and $|P_1| = |G|r$. This expression describes a partial announcement which defines two new populations, one of them with additional knowledge announced and other that did not received the new information announced.

3.4 Semantics

This section presents the semantics of PPAL. We start by redefining (fuzzifying) the valuation functions V as definitions 16 and 17. In order to achieve that, we define an evaluation function $e : p \times (M_G, s) \rightarrow U$ where $p \in \Phi$, M_G is a model for G, s is a state and U is the unit interval $[0, 1]$. We can define the semantics as follows:

$$E_{M_G,s}(p) = e(p, (M_G, s)) \tag{3.3}$$

$$E_{M_G,s}(\neg\varphi) = NOT(E_{M_G,s}(\varphi)) \tag{3.4}$$

$$E_{M_G,s}(\varphi \wedge \psi) = AND(E_{M_G,s}(\varphi), E_{M_G,s}(\psi)) \tag{3.5}$$

$$E_{M_G,s}(\varphi \vee \psi) = OR(E_{M_G,s}(\varphi), E_{M_G,s}(\psi)) \tag{3.6}$$

$$E_{M_G,s}(\varphi \to \psi) = IMP(E_{M_G,s}(\varphi), E_{M_G,s}(\psi)) \tag{3.7}$$

$$E_{M_G,s}(K_{G'}\varphi) = K(\varphi, (M_G, s), G') \tag{3.8}$$

$$E_{M_G,s}(B_{G'}\varphi) = B(\varphi, (M_G, s), G') \tag{3.9}$$

$$E_{M_G,s}([\varphi]^r_{G_{new}}\psi) = E_{M_{G_{new}},s}(\psi) \tag{3.10}$$

where $\exists G^1, G^2$ such that $G_{new} = \{G^1, G^2\}$, and $M_{G_{new}} = \{M_{G^1}, M_{G^2}\}$ such that $M_{G^1} = M_G|^r_\varphi$ and $M_{G^2} = M_G|^{1-r}_\top$. $\top$ is a symbol for $true$.

$M_G|^r_\varphi$ is defined as a recursive call for every $G' \in G$ (that is, $M_{G'}|^r_\varphi$), and $M_P|^r_\varphi$ as in definition 12, but substituting agent a for P, and multiplying the size of P by r. Intuitively, $M_G|^r_\varphi$ is the new model obtained from M_G by removing all states where $\neg\varphi$ holds, and multiplying the sizes of every group contained in G by r.

Depending on the choice of the evaluation function we can have different semantics. One possible evaluation function is the crisp evaluation. Suppose $M_G = \langle T, \overset{M_G,S'}{\sim}, V\rangle$

$$e(p, (M_G, s)) = 1 \text{ if } p \in V(p) \text{ , and 0 otherwise} \tag{3.11}$$

Below, we give some properties about the fuzzy functions that are used in the definition of satisfaction. At this point we choose not to define a particular fuzzy semantics and instead define classes for each operation. One can choose any semantics which satisfies the following properties, and we provide an example for each of the classes.

It is also important to note that this work is concerned with model checking and for that the notion of satisfaction is enough. For this reason we did not define the consequence relation.

3.5 Fuzzy negation

A unary operation $NOT : U \to U$ is a fuzzy negation if

- $NOT(0) = 1$;
- $NOT(1) = 0$;
- $x \leq y \to NOT(y) \leq NOT(x)$.

For instance, a possible negation function is $NOT(x) = 1 - x$.

Examples:

- $NOT(0.2) = 0.8$;
- $NOT(0.5) = 0.5$;

3.6 Fuzzy conjunction

A binary operation $AND : U \times U \to U$ is a fuzzy conjunction if

- $AND(x, y) = AND(y, x)$ (Symmetry);
- $AND(x, AND(y, z)) = AND(AND(x, y), z)$ (Associativity);
- $y \leq z \to AND(x, y) \leq AND(x, z)$ (Right monotonicity);
- $AND(x, 1) = x$ (1-Identity).

These also define precisely the class of t-norm functions, which are widely used in fuzzy logics as they are considered "well-behaved" functions [33]. For instance, a possible conjunction function is $AND(x, y) = \min\{x, y\}$.

Examples:

- $AND(1, 0) = 0$;
- $AND(0.2, 0.8) = 0.2$;

3.7 Fuzzy disjunction

A binary operation $OR : U \times U \to U$ is a fuzzy disjunction if

- $OR(x, y) = OR(y, x)$ (Symmetry);
- $OR(x, OR(y, z)) = OR(OR(x, y), z)$ (Associativity);
- $y \leq z \to OR(x, y) \leq OR(x, z)$ (Right monotonicity);
- $OR(x, 0) = x$ (0-Identity).

These also define precisely the class of t-conorm functions, analogous to the t-norm class. For instance, a possible disjunction function is $OR(x, y) = \max\{x, y\}$.

Examples:

- $OR(1, 0) = 1$;
- $OR(0.2, 0.8) = 0.8$;

3.8 Fuzzy implication

A binary operation $IMP : U \times U \to U$ is a fuzzy implication if

- $x \leq y \to IMP(x, z) \geq IMP(y, z)$;
- $y \geq z \to IMP(x, y) \geq IMP(x, z)$;
- $IMP(0, x) = 1$;
- $IMP(x, 1) = 1$;
- $IMP(1, 0) = 0$.

For instance, a possible implication function is $IMP(x, y) = \min\{1, 1 - x + y\}$, also known as the Łukasiewicz implication.

Examples:

- $IMP(0.2, 0.8) = 1$;
- $IMP(0.8, 0.2) = 0.4$;

3.9 Fuzzy knowledge assertion

A ternary operation $K : \phi \times (M_S, s) \times G \to U$ is a fuzzy knowledge assertion if

- if $G = P$ then $\forall s' \in T \mid s \overset{M_S,P}{\sim} s'[E_{M_S,s'}(\varphi) = 1 \to K(\varphi, (M_S, s), P) = 1]$;

  Which means that for a population P, if φ is true on every state connected to the current state via $\overset{M_S,P}{\sim}$ edges, then the population knows φ (result is 1).

- if $G = P$ then $\forall s' \in T \mid s \overset{M_S,P}{\sim} s'[E_{M_S,s'}(\neg\varphi) = 1 \to K(\varphi, (M_S, s), P) = 0]$;

  Which means that for a population P, if φ is not true on every state connected to the current state via $\overset{M_S,P}{\sim}$ edges, then the population does not know φ (result is 0).

- $K(\varphi, (M_S, s), G) \leq \max_{G' \in G}\{K(\varphi, (M_S, s), G')\}$;

  Which means that for a group G, the knowledge about φ should be less than or equal to the knowledge about φ in its contained group with the highest knowledge about it.

- $K(\varphi, (M_S, s), G) \geq \min_{G' \in G}\{K(\varphi, (M_S, s), G')\}$.

  Which means that for a group G, the knowledge about φ should be greater than or equal to the knowledge about φ in its contained group with the lowest knowledge about it.

For instance, a possible knowledge assertion function is

$$K(\varphi,(M_S,s),G)=\begin{cases}\sum_{G'\in G}\frac{|G'|}{|G|}K(\varphi,(M_S,s),G') & \text{if } G\neq P,\\ 1 & \text{if } \forall s'\in T \mid s \overset{M_S,P}{\sim} s' \rightarrow \\ & E_{M_S,s'}(\varphi),\\ 0 & \text{otherwise.}\end{cases}$$

Note here that we slightly abuse notation for binary relations when they appear referring to models of groups, such as the expression $\overset{M_G,P}{\sim}$, which represents the binary relation for the model of population P contained inside M_G.

Intuitively, it means that the knowledge of a group is equal to the weighted average knowledge of its contained groups. If the group is a single population, it is equal to 1 when φ is true in every state connected to state s via $\overset{M_S,G}{\sim}$, and 0 otherwise. A single population knows something only when it is true in every conceivable world this population contemplates.

3.10 Fuzzy belief assertion

A ternary operation $B:\phi\times(M_S,s)\times G\rightarrow U$ is a fuzzy belief assertion if

- if $G=P$ then $\exists s'\in T \mid s \overset{M_S,P}{\sim} s'[E_{M_S,s'}(\varphi)=1\rightarrow B(\varphi,(M_S,s),P)>0]$;

  Which means that for a population P, if φ is true on a state connected to the current state via $\overset{M_S,P}{\sim}$ edges, then the population's belief about φ is greater than zero.

- if $G=P$ then $\exists s'\in T \mid s \overset{M_S,P}{\sim} s'[E_{M_S,s'}(\neg\varphi)=1\rightarrow B(\varphi,(M_S,s),P)<1]$;

  Which means that for a population P, if φ is not true on a state connected to the current state via $\overset{M_S,P}{\sim}$ edges, then the population's belief about φ is less than one.

- if $G=P$ then $\forall s'\in T \mid s \overset{M_S,P}{\sim} s'[E_{M_S,s'}(\varphi)=1\rightarrow$
  $B(\varphi,(M_S,s),P)=K(\varphi,(M_S,s),P)]$;

  Which means that for a population P, if φ is true on every state connected to the current state via $\overset{M_S,P}{\sim}$ edges, then the population's belief about φ is equal it its knowledge about φ.

- $B(\varphi,(M_S,s),G)\leq\max_{G'\in G}\{B(\varphi,(M_S,s),G')\}$;

  Which means that for a group G, the belief about φ should be less than or equal to the belief about φ in its contained group with the highest belief about it.

- $B(\varphi, (M_S, s), G) \geq \min_{G' \in G}\{B(\varphi, (M_S, s), G')\}$.

  Which means that for a group G, the belief about φ should be greater than or equal to the belief about φ in its contained group with the lowest belief about it.

For instance, a possible population belief assertion function is

$$B(\varphi, (M_S, s), G) = \begin{cases} \sum_{G' \in G} \frac{|G'|}{|G|} B(\varphi, (M_S, s), G') & \text{if } G \neq P, \\ \sum_{s' \in N_s} \frac{E_{M_S, s'}(\varphi)}{|N_s|} & \text{if } G \neq \emptyset, \\ 0 & \text{otherwise,} \end{cases}$$

where $N_s = \bigcup_{s' \in T | s \overset{M_S, G}{\sim} s'} s'$ are the neighbours of s via relation $\overset{M_S, G}{\sim}$.

Which is functionally similar to K, but allowing any number in the interval $[0, 1]$. For example, if in half of the states connected to state s via $\overset{M_S, P}{\sim}$, φ evaluates to 1 and in the other half to 0, then $B(\varphi, (M_S, s), P) = 0,5$. And also similarly to K, the belief of a group is equal to the weighted average beliefs of its contained groups.

3.11 Proof of decidability

An important aspect of model checking a logical system is that of decidability, which relates to the halting problem, that is, whether or not there exists some formula in the language that could potentially cause the model checker to get stuck in a loop. Some multi-agent logics are undecidable, and some of their public announcement extensions are as well [34]. In this section we show that, when using the specified operator functions, any valid finite formula of the PPAL language over a finite model is in fact decidable. We use induction over the size s of the formula for this proof.

Definition 18 *The size s for each expression of the language is defined as follows:*

- $s(p) = 1$;
- $s(\varphi_1 \wedge \varphi_2) = s(\varphi_1) + s(\varphi_2) + 1$;
- $s(\varphi_1 \vee \varphi_2) = s(\varphi_1) + s(\varphi_2) + 1$;
- $s(\varphi_1 \rightarrow \varphi_2) = s(\varphi_1) + s(\varphi_2) + 1$;
- $s(K_G \varphi) = s(\varphi) + 1 + |G|$;
- $s(B_G \varphi) = s(\varphi) + 1 + |G|$;
- $s([\varphi_1]^r_G \varphi_2) = s(\varphi_1) + s(\varphi_2) + 1 + |G|$.

Next, we can proceed to the induction basis step where $n = 1$, that is, the case of a formula consisting of a single propositional symbol. $E_{M_G,s}(p) = e(p, (M_G, s))$ evaluates to 1 or 0 in a finite model, and therefore is decidable.

We then consider the induction hypothesis to be that any formula with size s is decidable. Now it remains necessary to take the inductive step, assuming the induction hypothesis and proving that formulas with size $s + \delta$ are also decidable, where $\delta \in \mathbb{R}_{>0}$. Note that δ is in the real interval because the size $|G|$ of a group G can be a real value.

Inductive step:

1. $E_{M_G,s}(\neg\varphi) = NOT(E_{M_G,s}(\varphi))$: NOT is a simple mathematical function which evaluates to a value in U for any input in U. Since $E_{M_G,s}(\varphi)$ is decidable by hypothesis, this formula is decidable. $E_{M_G,s}(\varphi \wedge \psi)$, $E_{M_G,s}(\varphi \vee \psi)$ and $E_{M_G,s}(\varphi \to \psi)$ are all decidable due to the same arguments;

2. $E_{M_G,s}(K_{G'}\varphi) = K(\varphi, (M_G, s), G')$:

   (a) It is trivially decidable when $G' = \emptyset$, because its evaluation will always be equal to 0;

   (b) It is decidable when $G' = P$, as the formula is resolved via calculations of $E_{M_G,s'}(\varphi)$ which is decidable by hypothesis;

   (c) In the case where $G' = \{G'_0, G'_1, \ldots, G'_n\}$, the sum will range recursively over every $G'_i \in G'$. The sizes $|G'_i|$ are strictly smaller than $|G|$ because by definition the group G' is composed of a disjoint set of groups. Due to this, $s(K_{G'_i}\varphi) < s(K_{G'}\varphi)$ and $K_{G'_i}\varphi$ is decidable by hypothesis. $K_{G'}\varphi$ is resolved via calculations of $K_{G'_i}\varphi$, therefore, the formula is decidable.

3. $E_{M_G,s}(B_{G'}\varphi) = B(\varphi, (M_G, s), G')$:

   (a) It is trivially decidable when $G' = \emptyset$, because its evaluation will always be equal to 0;

   (b) When $G' = P$, the sum ranges over all neighbours s' of state s, but since the model is finite and $E_{M_G,s'}(\varphi)$ is decidable by hypothesis, it is decidable;

   (c) In the case where $G' = \{G'_0, G'_1, \ldots, G'_n\}$, the same argument used in 2b shows this formula is decidable.

4. $E_{M_G,s}([\varphi]^r_{G'}\psi) = E_{M_{G'},s}(\psi)$: M_G is finite and therefore $M_{G'}$ is also finite as it contains exactly twice the number of elements in M_G. To define $M_{G'}$ we need to calculate $E_{M_G,s}(\varphi)$ for every $s \in T$, which are decidable by hypothesis. $E_{M_{G'},s}(\psi)$ is also decidable by hypothesis. Also note that it is not possible to

write a formula that divides the group indefinitely, as that would require us to write down an infinite number of announcements. Therefore, the formula is decidable.

Hence we have covered every possible case and shown that they are all decidable, therefore successfully showing that any valid finite formula of the PPAL language over a finite model is decidable using the specified operator functions.

Notice however, that recursive formulas such as $\varphi := [\varphi]^r_{G'}\psi$ or $\varphi := [\psi]^r_{G'}\varphi$ are not included in the language definition, and they would break the language's property of being decidable.

Chapter 4

Usage examples of PPAL

In this section two examples are provided, where simple models are established and some assertions are made to probe the knowledge represented by the models. For the sake of clarity, we name the states with the set of propositional symbols, or their negation, that hold in it. For instance, state $\{a, \neg b\}$ is the state where a and $\neg b$ are true.

4.1 Basic example

We begin with a simple example where we are able to easily demonstrate how the model evolves, and give step-by-step resolutions of the evaluations. Later we provide a more complex example with an actual motivation.

4.1.1 Initial model

We suppose an initial model composed of a single population P, two propositions a and b, and a real state $\{a, b\}$. The evaluation function is the same as defined before. Initially the population is entirely unaware of its real state, which means there are edges between every state of this model (it is a complete graph. Figure 4.1).

The following assertions hold:

- $E_{M_P,\{a,b\}}(a) = e(a, (M_P, \{a, b\})) = 1.0$, because the state $\{a, b\}$ has proposition a;
- $E_{M_P,\{a,b\}}(K_P a) = 0.0$, because states $\{\neg a, b\}$ and $\{\neg a, \neg b\}$ are connected to $\{a, b\}$ via $\overset{M_S,P}{\sim}$ edges, but do not have proposition a.

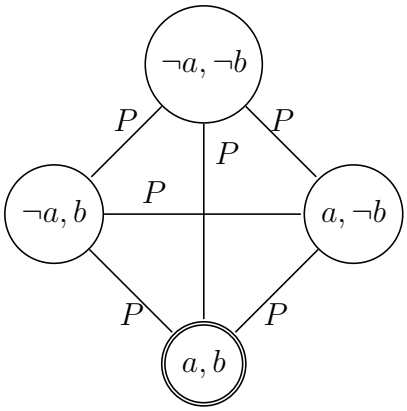

Figure 4.1: Initial model.

4.1.2 First announcement

Now we announce a for fraction r of the initial population P, obtaining group G, and we want to know if G knows a. That is:

$$E_{M_P,\{a,b\}}([a]^r_G K_G a) \tag{4.1}$$

The announcement is processed over model $M_G = \{P^1, P^2\}$ (Figure 4.2), where P^1's model is the same as P's, but without the edges connecting to states where a does not hold, and size $r|P|$. P^2's model is just like P's, but has size $(1-r)|P|$. Notice that P^2 did not receive the announcement, and is unaware that P^1 did. In this case we're effectively assessing:

$$E_{M_G,\{a,b\}}(K_G a) \tag{4.2}$$

This equation is the same as $K(a, M_G, \{a, b\}, G)$ according to the rules in the previous section. Expanding:

$$\begin{aligned} &K(a, M_G, \{a, b\}, G) = \\ &\frac{|P^1|K(a, M_{P^1}, \{a, b\}, P^1) + |P^2|K(a, M_{P^2}, \{a, b\}, P^2)}{|P^1| + |P^2|} = \\ &\frac{|P^1| \cdot 1 + |P^2| \cdot 0}{|P^1| + |P^2|} = \frac{|P^1|}{|P^1| + |P^2|} = \frac{r|P|}{|P|} = r \end{aligned} \tag{4.3}$$

Which of course makes sense, as the announcement was made for fraction r of the population.

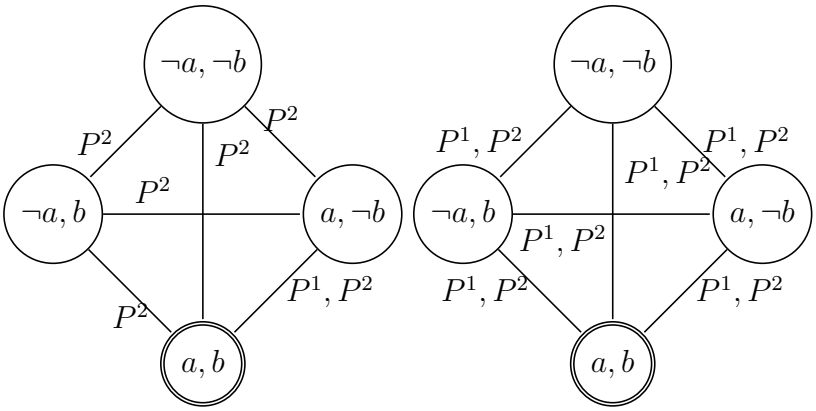

Figure 4.2: Model G (P^1's to the left, P^2's to the right).

4.2 The corrupt politician

In this section we attempt to provide a more motivational example, with a scenario that is closer to real life. Consider a famous politician who is under suspicion of money laundering and bribery. Elections are coming up, and this politician expects to be re-elected, but people are less willing to vote for him if they are aware of both charges against him. Not everyone in the population is aware of the charges, however.

4.2.1 Initial model

Again we suppose an initial model composed of a single population P, two propositions l (money laundering) and b (bribery), and a real state $\{l, b\}$, which means the politician in fact is guilty of the charges against him. The evaluation function is the same as defined before. This model is as shown in figure 4.3.

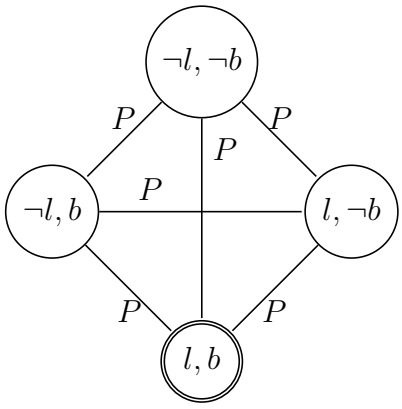

Figure 4.3: Initial model.

As expected if we assert how much of the population would not vote on the

politician, that is, if we ask:

$$E_{M_P,\{l,b\}}(K_P(l \wedge b)) \tag{4.4}$$

It will evaluate to 0, as there are P edges connecting the state $\{l, b\}$ to other states.

4.2.2 First announcement

A TV program aired revealing the politician in fact did money laundering, but given that only 30% of the population watched the program, only a fraction of the population is now aware of this fact. This is equivalent to the announcement $[l]_G^{0.3}$, where $G = \{P^1, P^2\}$ is a group containing the two populations resulting from the announcement: P^1 who received the announcement, and P^2 who did not. Figure 4.4 shows the resultant model G.

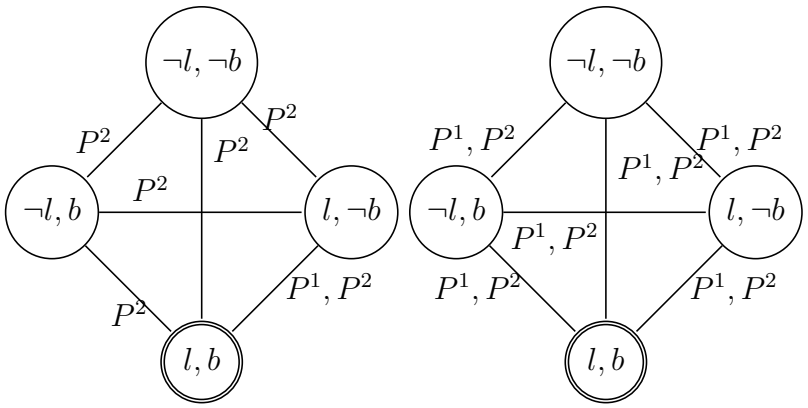

Figure 4.4: Model G (P^1's to the left, P^2's to the right).

If we were now to assert how much of the population would not vote on the politician, we would be asking:

$$E_{M_P,\{l,b\}}([l]_G^{0.3} K_G(l \wedge b)) \tag{4.5}$$

At this moment, this would still evaluate to 0, as there are still p_1 and p_2 edges connecting the state $\{l, b\}$ to other states. That is, both populations remain in doubt about whether $\{l, b\}$ is the real state or not.

We could also make an assertion in regards to the belief of the population, that is:

$$E_{M_P,\{l,b\}}([l]_G^{0.3} B_G(l \wedge b)) \tag{4.6}$$

To evaluate the belief we must evaluate $E_{M_G,r}(B_G(\varphi))$, which expands into

$$\begin{aligned} B(\varphi,(M_G,r),G) &= \sum_{G'\in G}\frac{|G'|}{|G|}B(\varphi,(M_G,r),G') \\ &= 0.3\sum_{r'\in N_r}\frac{E_{M_{P^1},r}(\varphi)}{|N_r|} + 0.7\sum_{r'\in N_r}\frac{E_{M_{P^2},r}(\varphi)}{|N_r|} \\ &= 0.3\cdot 1/2 + 0.7\cdot 1/4 = 0.325 \end{aligned} \tag{4.7}$$

The first sum evaluates to $1/2$ because out of two neighbour states to r ($\{l,b\}$ itself and $\{l,\neg b\}$), φ is only true in r. The second has all four states as neighbours, but φ is only true in r again. The result 0.325 demonstrates a small belief at this moment that the politician is in fact corrupt.

4.2.3 Second announcement

Now, let us consider another TV program aired, and this time covering 40% of the same original population, and it was told that the politician is in fact guilty of bribery. This corresponds to the announcement $[b]^{0.4}_{G'}$, where $G' = \{G^1, G^2\}$ is a group containing the two groups resulting from the announcement: G^1 who received the announcement, and G^2 who did not. The model resulting from both announcements is shown in figure 4.5.

If we were now to assert how much of the population would not vote on the politician, we would be asking:

$$E_{M_P,\{l,b\}}([l]^{0.3}_G[b]^{0.4}_{G'}K_{G'}(l\wedge b)) \tag{4.8}$$

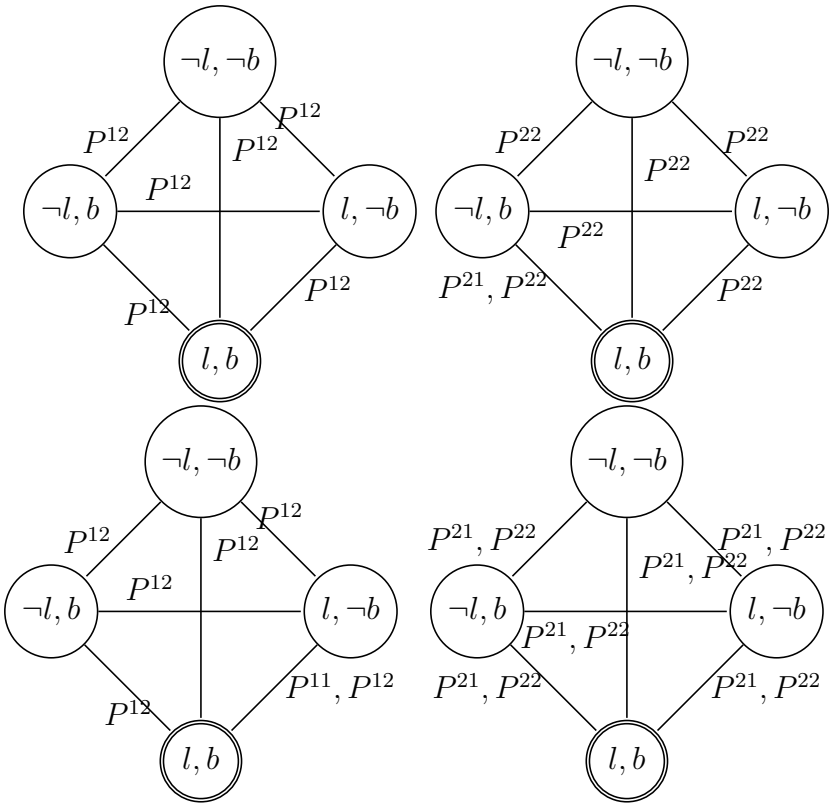

Figure 4.5: Model G'. G^1's model to the left, population above received both announcements, below received only the first one; G^2's to the right, population above received only the second announcement, below received none.

We omit the complete math here, but at this moment the assertion would evaluate to 0.12 (0.3×0.4, intuitively in this case), since there are no more edges connecting the state $\{l, b\}$ to anything else for population P^{11} in G^1.

It is a simple but interesting example, in that it shows how "partial" knowledge can build up to create useful information for certain fractions of the populations, due to them having being exposed enough to these partial knowledge pieces in the past, and how it might not be intuitive to find out how much exactly of a population knows a certain fact.

Chapter 5

Model checker

A general-purpose Java library implementing the PPAL language and semantics, and a CLI (Command Line Interface) model checker have been developed. Both programs are extensively covered by unit and integration testing to ensure consistency with the PPAL language and semantics, and the source-code of both library and model checker are available on GitHub[1] under the BSD license. Important parts of the model checker's source-code are shown in Appendix A.

The main current implementations of model checkers available for epistemic logics are MCK [35][2], MCMAS [36][3] and DEMO [37][4]. A recent development called SMCDEL [38][5] formalizes a new structure to represent DEL models which allows for symbolic model checking, a model which uses binary decision diagrams [39] to represent models instead of having every state directly represented in the system. Due to this, SMCDEL also provides better performance over the other model checkers. Note that in this work we do not yet attempt symbolic model checking, and this is left as a future work.

There are also model checkers for other types of logic, such as Propositional Dynamic Logic (PDL) [40], whose main objective is to model the execution of programs, by representing them as binary relations connecting states, similarly to the relations in the modal logics shown here. One such tool is mcpdl[6].

5.1 Model format

One of the recurring problems of these model checkers is the difficulty in specifying input models to be worked on. Difficulty arises not only because one usually has

[1] `https://github.com/vittau/PPAL`
[2] `http://cgi.cse.unsw.edu.au/~mck/pmck/`
[3] `http://vas.doc.ic.ac.uk/software/mcmas/`
[4] `http://homepages.cwi.nl/~jve/#Software`
[5] `https://github.com/jrclogic/SMCDEL`
[6] `http://www2.tcs.ifi.lmu.de/~axelsson/veri_non_reg/pdlig_mc.html`

to define every state and every edge of the model, but because these specifications use to be somewhat unintuitive and hard to read as well. We define an XML-based format, which aims to make the process of declaring models easier than what the currently available software permits, and that is also scalable for larger models.

In this model specification format we provide tags to generate states based on combinatorics where one can, for example, group propositions by population and then define that every state must contain at least one proposition from a set for each population. There are restrictions for mutual exclusion that can be applied as well, which can be thought of as "trimming" the combinatorics-generated states. It is also possible to define states manually by declaring all its propositions.

Appendix B for example models a populational version of the card game shown in section 2.4. Its first section declares the populations using the tag <societies>, with their names and sizes. The second section declares propositions with the tags <propositions> and <propdef>, with their names and the populations aware of them. A population is aware of a proposition if it can distinguish between states containing and not containing the proposition. The third section declares the states using the tag <states>, via combinations (<comb>) and restrictions (<restrictions>). Restrictions of type <atleast> mean that at least one of the declared propositions must appear in every state generated by this combination, and restrictions of type <mutex> (mutually exclusive) mean that at most one of the declared propositions can appear in these states.

States can also be declared in full by means of specifying every proposition in the state using the tag <state>. Restrictions of states can be declared the same way.

5.2 Sample run

After the model (the model shown in appendix B is used for these examples) is loaded into the model checker, we can now give some commands to it. For instance, we can print the simulation state (print all):

```
ppalmc> print a
Real state: {ah0, ch2, bh1}
Populations:
a (size = 3.0)
b (size = 5.0)
c (size = 7.0)
```

We can then, for instance, announce *ch2* to population *a* in its own model and in the real state:

```
ppalmc> announce a 0.7 a ch2
```

```
"ch2" announced to "a" with ratio 0.7 successfully.
```

And then if we ask if *a* knows *ch2*, it returns the expected result:

```
ppalmc> knows a ah0&bh1&ch2 a ch2
0,7
```

Not as intuitively perhaps, if we ask if *a* knows *bh1*, it returns the same result:

```
ppalmc> knows a ah0&bh1&ch2 a bh1
0,7
```

Which is true given our model's restrictions, because *a* knows *ah0* since it is its own card, and the fraction of *a* that now knows *ch2* because it was announced, now knows the only remaining possibility is *bh1* because *b* can't have any other card other than *1*.

Chapter 6

Final remarks and future works

In this work we introduced and defined a "fuzzyfied" variant of PAL that allows partial announcements to its agents (called "populations" and "groups").

The main advantage of PPAL over DEL and PAL is the flexibility to work with non-priorly defined agents. It is possible to define an initial population and have it evolving in a more natural way that doesn't require a formal initial definition of every agent on the system, which is particularly useful for model checking. Therefore we believe it can be a more appropriate logic for practical use, as it would be much easier to control and evolve a knowledge model.

We also believe this work expands horizons for the definition of a "dynamic" logic, in that not only accessibility relations are dynamic, but the agents of the world themselves are changing and evolving. Therefore it is a worthwhile topic of theoretical discussion as well, and not only of interest for model checking applications.

6.1 Future works

There is room for further development on a few directions still, and in the following sections we present some of these directions we believe are worth pursuing.

- Axiomatisation: One important formalism for logics is that of axiomatisation, a set of tautologies (formulas that are always true in the logic), called axioms, that can be used to derive all other formulas in the language.

- Symbolic model checking: We did not attempt symbolic model checking. A model checker is considered "symbolic" if it does not represent all reachable states individually, and instead relies on representations of states sets and formulas for transition relations, which can greatly improve performance. One such approach is based on an structure known as "binary decision diagram" (BDD) [39]. These structures provide a compact way to represent and check

x	y	z	f
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Table 6.1: Truth table of the function $f(x, y, z) = \neg x \neg y \neg z \vee xy \vee yz$.

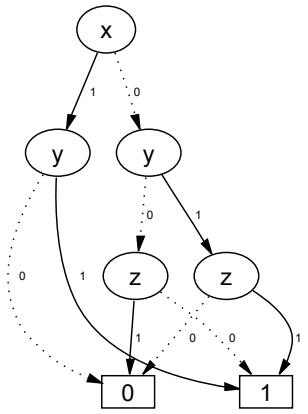

Figure 6.1: Binary decision diagram for function f.

valuations in Boolean functions, and equivalent Boolean sub-expressions are uniquely represented (see table 6.1 and figure 6.1 for an example). This property can provide an exponentially more compact representation compared to a naive truth table one, which in turn can provide better model checking performance for larger models.

Bibliography

[1] SOWA, J. F. *Knowledge Representation: Logical, Philosophical and Computational Foundations*. Pacific Grove, CA, USA, Brooks/Cole Publishing Co., 2000. ISBN: 0-534-94965-7.

[2] WAY, E. C. *Knowledge Representation and Metaphor*, v. 7, *Studies in Cognitive Systems*. Dordrecht, Boston, Kluwer, 1991. ISBN: 0-7923-1005-5.

[3] SHOHAM, Y. "Why Knowledge Representation Matters", *Communications of the ACM*, v. 59, n. 1, pp. 47–49, 2015. ISSN: 0001-0782. doi: 10.1145/2803170.

[4] SHAPIRO, S. C. "Knowledge Representation", *Encyclopedia of Cognitive Science*, v. 2, pp. 671–680, 2003.

[5] KNEALE, W., KNEALE, M. "The Development of Logic", 1962.

[6] LEWIS, C. I. "A Survey of Symbolic Logic", 1918.

[7] VON WRIGHT, G. H. "An Essay in Modal Logic", 1951. doi: 10.1017/S0031819100026176.

[8] HAREL, D. "Dynamic Logic", v. 165, 1984. doi: 10.1007/978-94-009-6259-0_10.

[9] PLAZA, J. A. "Logics of public communications", *Synthese*, v. 158, n. 2, pp. 165–179, 1989. doi: 10.1007/s11229-007-9168-7.

[10] GERBRANDY, J., GROENEVELD, P. W. "Reasoning about Information Change", *Journal of Logic, Language and Information*, , n. 6, pp. 147–169, 1997.

[11] BALTAG, A., MOSS, L. S. "Logics for Epistemic Programs", *Synthese*, v. 139, n. 2, pp. 165–224, 2004. doi: 10.1023/B:SYNT.0000024912.56773.5e.

[12] BALTAG, A., MOSS, L. S., SOLECKI, S. "The Logic of Public Announcements, Common Knowledge and Private Suspicions", *Proceedings of TARK'98*, pp. 43–56, 1998. doi: 10.1007/s11229-007-9168-7.

[13] MACHADO, V. P., BENEVIDES, M. R. F. "Populational Dynamic Epistemic Logic (PPDEL)", 2015.

[14] BOBZIEN, S. "Ancient Logic". In: Zalta, E. N. (Ed.), *The Stanford Encyclopedia of Philosophy*, spring 2014 ed., 2014.

[15] BOOLE, G. *The Mathematical Analysis of Logic*. Contemporary philosophy. Philosophical Library, 1847.

[16] ZADEH, L. A. "Fuzzy Sets", *Information and Control*, v. 8, pp. 338–353, 1965. doi: 10.1016/S0019-9958(65)90241-X.

[17] MAMDANI, E. H. "Application of fuzzy algorithms for control of simple dynamic plant", *Proceedings of the Institution of Electrical Engineers*, v. 121, pp. 1585–1588, 1974. doi: 10.1049/piee.1974.0328.

[18] GODJEVAC, J. "Comparison between PID and fuzzy control", 1993.

[19] LEWIS, C. I., LANGFORD, C. H. *Symbolic Logic*. Dover Publications Inc., 1959.

[20] FITTING, M., MENDELSOHN, R. L. *First-Order Modal Logic*. Synthese Library Studies in Epistemology Logic, Methodology, and Philosophy of Science Volume 277. Springer, 1998. ISBN: 9780792353355.

[21] VON WRIGHT, G. H. "Deontic Logic", *Mind*, v. 60, n. 237, pp. 1–15, 1951. ISSN: 00264423, 14602113.

[22] HINTIKKA, J. *Knowledge and belief: an introduction to the logic of the two notions*. Contemporary philosophy. Cornell University Press, 1962.

[23] STEUP, M. "Epistemology". In: Zalta, E. N. (Ed.), *The Stanford Encyclopedia of Philosophy*, spring 2014 ed., 2014.

[24] KVANVIG, J. L. "Closure Principles", *Philosophy Compass*, v. 1, n. 3, pp. 256–267, 2006. ISSN: 1747-9991. doi: 10.1111/j.1747-9991.2006.00027.x.

[25] HENDRICKS, V. F., SYMONS, J. "Where's the Bridge? Epistemology and Epistemic Logic", *Philosophical Studies: An International Journal for Philosophy in the Analytic Tradition*, v. 128, n. 1, pp. 137–167, 2006. ISSN: 00318116, 15730883.

[26] MOORE, G. E. "A Reply to My Critics". In: Schilpp, P. A. (Ed.), *The Philosophy of G. E. Moore*, Open Court, 1942.

[27] FITCH, F. B. "A logical analysis of some value concepts", *The Journal of Symbolic Logic*, v. 28, pp. 135–142, 6 1963. ISSN: 1943-5886. doi: 10.2307/2271594.

[28] BROGAARD, B., SALERNO, J. "Fitch's Paradox of Knowability". In: Zalta, E. N. (Ed.), *The Stanford Encyclopedia of Philosophy*, winter 2013 ed., 2013.

[29] VAN DITMARSCH, H., VAN DER HOEK, W., KOOI, B. "Dynamic Epistemic Logic", *Synthese Library*, v. 337, 2007. doi: 10.1007/978-1-4020-5839-4.

[30] FAGIN, R., HALPERN, J. Y., MOSES, Y., et al. "Reasoning about Knowledge", 1995.

[31] HAREL, D., KOZEN, D., TIURYN, J. *Dynamic Logic*. Foundations of Computing. MIT Press, 2000. ISBN: 9780262263023.

[32] BALTAG, A., COECKE, B., SADRZADEH, M. "Epistemic actions as resources", 2006.

[33] ESTEVA, F., GODO, L., NOGUERA, C. "First-order t-norm based fuzzy logics with truth-constants: distinguished semantics and completeness properties", *Annals of Pure and Applied Logic*, v. 161, pp. 185–202, 2009.

[34] FRENCH, T., VAN DITMARSCH, H. "Undecidability for Arbitrary Public Announcement Logic", *Advances in Modal Logic*, v. 7, pp. 23–42, 2008.

[35] GAMMIE, P., VAN DER MEYDEN, R. "MCK: Model Checking the Logic of Knowledge", *Computer Aided Verification*, v. 3114, pp. 479–483, 2004. doi: 10.1007/978-3-540-27813-9_41.

[36] LOMUSCIO, A., RAIMONDI, F. "mcmas: A Model Checker for Multi-agent Systems", *Tools and Algorithms for the Construction and Analysis of Systems*, v. 3920, pp. 450–454, 2006. doi: 10.1007/11691372_31.

[37] VAN DITMARSCH, H., VAN EIJCK, J., HERNÁNDEZ-ANTÓN, I., et al. "Modelling Cryptographic Keys in Dynamic Epistemic Logic with DEMO", *10th International Conference on Practical Applications of Agents and Multi-Agent Systems*, v. 156, pp. 155–162, 2012. doi: 10.1007/978-3-642-28762-6_19.

[38] VAN BENTHEM, J., VAN EIJCK, J., GATTINGER, M., et al. "Symbolic Model Checking for Dynamic Epistemic Logic". In: van der Hoek, W., Holliday, W. H., Wang, W.-f. (Eds.), *Logic, Rationality, and Interaction*, v. 9394,

Lecture Notes in Computer Science, Springer Berlin Heidelberg, pp. 366–378, 2015. ISBN: 978-3-662-48560-6. doi: 10.1007/978-3-662-48561-3_30.

[39] AKERS, S. B. "Binary Decision Diagrams", *IEEE Trans. Comput.*, v. 27, n. 6, pp. 509–516, 1978. ISSN: 0018-9340. doi: 10.1109/TC.1978.1675141.

[40] LANGE, M. "Model Checking Propositional Dynamic Logic with All Extras", *Journal of Applied Logic*, v. 4, n. 1, pp. 39–49, 2006. ISSN: 1570-8683. doi: 10.1016/j.jal.2005.08.002.

Appendix A

Model checker's source-code

This appendix contains excerpts from the key points of the model checker's source-code, to facilitate the reader in asserting the internal functioning of the checker. The complete source-code is available on GitHub[1].

Listing A.1: Method responsible for splitting a population in an announcement

```
/**
   * Splits a population into a group with two
      populations, to be used by the announcement
      operator.
   *  @param pop Population to be split.
   * @param pre Pre-condition of the announcement. Used
      to trim the model.
   * @param ratio Ratio of the population which will
      receive the announcement.
   * @return Group containing two populations, one that
      received the announcement, and one that did not.
   * @throws IllegalArgumentException If the ratio is
      not between 0.0 and 1.0 (both inclusive).
   */
  public static Group announce(Population pop, Evaluable
     pre, double ratio) throws IllegalArgumentException {

    if(ratio < 0D || ratio > 1D) {
      throw new IllegalArgumentException("Ratio must be
         between 0.0 and 1.0 (both inclusive).");
    }
```

[1] https://github.com/vittau/PPAL

```
    SocietyModel clonedSmOld =
       pop.getSocietyModel().clone();
    Population p_old = new
       BasicPopulation(pop.getName(), pop.getId() +
       "_o", clonedSmOld, pop.getSize() * (1D - ratio));
    clonedSmOld.replaceSociety(pop, p_old);

    SocietyModel clonedSmNew =
       pop.getSocietyModel().clone();
    Population p_new = new
       BasicPopulation(pop.getName(), pop.getId() +
       "_n", clonedSmNew, pop.getSize() * ratio);
    clonedSmNew.replaceSociety(pop, p_new);
    clonedSmNew.trimEdges(p_new, pre);

    return new BasicGroup(pop.getName(), pop.getId(),
       p_old, p_new);
  }
```

Listing A.2: Evaluation function as defined in section 3.4

```
/**
   * Evaluates a proposition in a model given a state.
   * @param p Proposition to evaluate.
   * @param societyModel Model to evaluate onto.
   * @param state State where the proposition is to be
      evaluated.
   * @return 1.0 if the proposition is in the state, 0.0
      otherwise.
   * @throws IllegalArgumentException
   */
  @Override public double eval(Proposition p,
     SocietyModel societyModel, State state) throws
     IllegalArgumentException {
    if(!societyModel.getStates().contains(state)) {
      throw new IllegalArgumentException("State not
         present in the given model.");
    }
    if(state.getPropositions().contains(p)) {
      return 1.0;
```

```
    }
    else {
      return 0.0;
    }
  }
```

Listing A.3: Interface "Evaluable" and binary operators implementing its subclasses, as defined in section 3.4

```
public interface Evaluable {
  /**
   * Evaluates the structure.
   * @param soc Society to evaluate on.
   * @param s State to evaluate on.
   * @return Evaluation. A real value ranging from 0.0
     to 1.0.
   */
  double eval(Society soc, State s);
}

/**
 * This class implements a fuzzy negation unary operator
    of the PPAL logic.
 */
public class BasicNegationOperator implements
   UnaryOperator {
  @Override public double eval(Society soc, State s) {
    return (1D - ev.eval(soc, s));
  }
}

/**
 * This class implements a fuzzy conjunction binary
    operator of the PPAL logic.
 */
public class BasicConjunctionOperator implements
   BinaryOperator {
  @Override public double eval(Society soc, State s) {
    double evD = ev.eval(soc, s);
    double ev2D = ev2.eval(soc, s);
```

```
    return Math.min(evD, ev2D);
  }
}

/**
 * This class implements a fuzzy disjunction binary
   operator of the PPAL logic.
 */
public class BasicDisjunctionOperator implements
   BinaryOperator {
@Override public double eval(Society soc, State s) {
    double evD = ev.eval(soc, s);
    double ev2D = ev2.eval(soc, s);
    return Math.max(evD, ev2D);
  }
}
```

Listing A.4: Knowledge operator evaluation methods: evalKG for groups, evalKP for populations

```
/**
   * Evaluation for a group.
   * @param soc Society being evaluated.
   * @param s State to evaluate on.
   * @return Evaluation. A real value ranging from 0.0
      to 1.0.
   */
private double evalKG(Society soc, State s) {
  double result = 0.0;
  for(Society sc : soc.getSocieties()) {
    BasicKnowledgeOperator bko = new
       BasicKnowledgeOperator(sm, ev);
    result += sc.getSize() * bko.eval(sc, s);
  }
  return result / soc.getSize();
}

/**
   * Evaluation for a population.
   * @param soc Society being evaluated.
```

```
    * @param s State to evaluate on.
    * @return Evaluation. A real value ranging from 0.0
       to 1.0.
    */
private double evalKP(Society soc, State s) {
  Set<State> neighbourStates =
     sm.getNeighbourStates(soc, s);
  for(State state : neighbourStates) {
    if(ev.eval(soc, state) < 1.0)
      return 0.0;
  }
  return ev.eval(soc, s);
}
```

Listing A.5: Belief operator evaluation methods: evalBG for groups, evalBP for populations

```
/**
 * Evaluation for a group.
 * @param soc Society being evaluated.
 * @param s State to evaluate on.
 * @return Evaluation. A real value ranging from 0.0 to
    1.0.
 */
private double evalBG(Society soc, State s) {
  double result = 0.0;
  for(Society sc : soc.getSocieties()) {
    BasicBeliefOperator bbo = new
       BasicBeliefOperator(sm, ev);
    result += sc.getSize() * bbo.eval(sc, s);
  }
  return result / soc.getSize();
}

/**
 * Evaluation for a population.
 * @param soc Society being evaluated.
 * @param s State to evaluate on.
 * @return Evaluation. A real value ranging from 0.0 to
    1.0.
```

```
 */
private double evalBP(Society soc, State s) {
  Set<State> neighbourStates =
     sm.getNeighbourStates(soc, s);
  double result = 0.0;
  int count = 1;
  for(State state : neighbourStates) {
    count++;
    //System.out.println(soc.getName() + ": " +
       ev.eval(soc, state));
    result += ev.eval(soc, state);
  }
  result += ev.eval(soc, s);
  return result/count;
}
```

Listing A.6: Document Type Definition (DTD) defining a valid XML input

```
<!ELEMENT model (societies, propositions, states,
   realstate?)>
<!ATTLIST model
  version CDATA #REQUIRED
  name CDATA #IMPLIED>
<!ELEMENT societies (socdef)+>
<!ELEMENT socdef (#PCDATA)>
<!ATTLIST socdef
  id CDATA #REQUIRED
  name CDATA #IMPLIED
  size CDATA #REQUIRED>
<!ELEMENT propositions (propdef)+>
<!ELEMENT propdef (soc)*>
<!ATTLIST propdef
  id CDATA #REQUIRED
  name CDATA #IMPLIED>
<!ELEMENT soc (#PCDATA)>
<!ATTLIST soc
  id CDATA #REQUIRED>
<!ELEMENT states (comb|state)+>
<!ELEMENT comb (restrictions*)>
<!ELEMENT prop (#PCDATA)>
```

```
<!ATTLIST prop
  id CDATA #REQUIRED
  always CDATA #IMPLIED
  mutex CDATA #IMPLIED>
<!ELEMENT state (prop+)>
<!ELEMENT restrictions (atleast|mutex|state)+>
<!ELEMENT atleast (prop)+>
<!ELEMENT mutex (prop)+>
<!ELEMENT realstate (prop)+>
```

Appendix B

Card game (XML model)

This appendix contains a model specification of a version of the Card Game modified for the PPAL logic.

Listing B.1: The Card Game

```
<!-- Cards game example using the PPDEL DTD. -->
<model version = "0.1" name = "Cards game">

  <populations>
    <popdef id = "a" name = "Alderaaneans" size="3"/>
    <popdef id = "b" name = "Bespians" size="5"/>
    <popdef id = "c" name = "Coruscanti" size="7"/>
  </populations>

  <propositions>
    <propdef id = "ah0" name = "a has 0">
      <pop id = "a"/>
    </propdef>

    ...

    <propdef id = "ch2" name = "c has 2">
      <pop id = "c"/>
    </propdef>
  </propositions>

  <states>
    <comb>
      <restrictions>
```

```
<!-- Populations must have at least one card -->
<atleast>
  <prop id = "ah0"/>
  <prop id = "bh0"/>
  <prop id = "ch0"/>
</atleast>
<atleast>
  <prop id = "ah1"/>
  <prop id = "bh1"/>
  <prop id = "ch1"/>
</atleast>
<atleast>
  <prop id = "ah2"/>
  <prop id = "bh2"/>
  <prop id = "ch2"/>
</atleast>
<!-- Different populations can't have the same card -->
<mutex>
  <prop id = "ah0"/>
  <prop id = "bh0"/>
  <prop id = "ch0"/>
</mutex>
<mutex>
  <prop id = "ah1"/>
  <prop id = "bh1"/>
  <prop id = "ch1"/>
</mutex>
<mutex>
  <prop id = "ah2"/>
  <prop id = "bh2"/>
  <prop id = "ch2"/>
</mutex>
<!-- Populations must have at most one card -->
<mutex>
  <prop id = "ah0"/>
  <prop id = "ah1"/>
  <prop id = "ah2"/>
</mutex>
<mutex>
```

```
            <prop id = "bh0"/>
            <prop id = "bh1"/>
            <prop id = "bh2"/>
          </mutex>
          <mutex>
            <prop id = "ch0"/>
            <prop id = "ch1"/>
            <prop id = "ch2"/>
          </mutex>
        </restrictions>
      </comb>
    </states>

    <realstate>
      <prop id = "ah0"/>
      <prop id = "bh1"/>
      <prop id = "ch2"/>
    </realstate>

</model>
```