A Quick Introduction to Dynamic Logic

Can BAŞKENT

cbaskent@gc.cuny.edu www.canbaskent.net

Istanbul Kultur University
Graduate Summer School in Modal Logic and Its Applications

Istanbul - Turkey

2010

TC
İSTANBUL
KÜLTÜR
ÜNİVERSİTESİ

Contents

1 Introduction **4**
1.1 Motivation . . . 4
1.2 Syntax and Semantics . . . 4
1.3 Axioms . . . 6
1.4 Formalizing Programs . . . 7
1.5 More on Basics . . . 8

2 Completeness **9**
2.1 Completeness . . . 9
2.2 Decidability and Complexity . . . 10

3 Game Logic **12**
3.1 Syntax and Semantics . . . 12
3.2 Axioms . . . 14
3.3 Satisfiability . . . 14
3.4 Completeness: A Homework . . . 15

4 Dynamic Epistemic Logic **16**
4.1 Syntax and Semantics . . . 16
4.2 Axioms . . . 17
4.3 Completeness and Decidability and Complexity . . . 18
4.4 An Example: Muddy Children . . . 19

Bibliography **21**

Prologue

One of the recent trends in the field of applied logic is dynamic logic. As opposed to static logical paradigms, dynamic logics focus on the notion of *change*. How our knowledge changes, how our actions creates certain outcomes are some of the issues that dynamic logic addresses.

Dynamic logic, in this regards, has a wide variety of applications. Apart from the mathematical merit of its own, it has significant applications in computer science, game theory, computational linguistics, artificial intelligence, and of course, in formal philosophy.

The lectures have been organized in two parts. In the first part, we will discuss the basics of propositional dynamic logic. In the second part, we will study two major applications of propositional dynamic logic to observe how it is applied and what it provides in such applications.

These lecture notes were prepared for the *Istanbul Kultur University Graduate Summer School in Modal Logic and Its Applications*. The intended course is a quick and dirty one, thus we *assume* a working knowledge of modal logic. There will be many exercise questions spread in the text and it is highly recommended that the reader spend enough time on them. We also included several challenging exercise questions which we called "Hard Exercises". The purpose of these exercises is to encourage the reader to go and check the literature and familiarize herself with some fundamental work that may go beyond the scope of this course. During the course, we will try to work on some of those exercises depending on our pace. Furthermore, our basic references for these notes are as expected (Blackburn *et al.*, 2001; van Ditmarsch *et al.*, 2007; Harel *et al.*, 2000; Parikh & Pauly, 2003).

One major difference between this lecture notes and any other mathematics text is that I did not include most proofs. There are several reasons for that. First is to invite the reader to attend the classes to learn the proofs with discussions. Second, once the reader gets stuck, is to encourage him to go check the basic articles and textbooks which would be considered as a small-scale research.

These notes were written in Summer 2010 and come with no guarantee. If you spot a typo, or an unconvincing argument along these lines, please let me know. You can contact me by e-mail at cbaskent@gc.cuny.edu or at www.canbaskent.net. You can also find the lecture notes at the latter address.

Enjoy!

Can Başkent
Department of Computer Science
The Graduate Center, The City University of New York

Lecture 1

Introduction: Syntax and Semantics

1.1 Motivation

Dynamic logic is a logic of actions. Then one can ask the following question: "What do we mean by actions?".

Present lectures will aim at answering this question at different levels. We will define actions first as computations, then as game strategies, then as epistemic updates. In short, actions are what makes a logic *dynamic*. In dynamic logic, as a distinguishing property, truth values change. Therefore, it would not be exaggeration to call dynamic logics as logics of change. Notice that in so-called classical logics, truth values are *static*, namely they do not change (Harel *et al.*, 2000).

Historically, propositional dynamic logic (PDL, henceforth) was first presented as a formal tool to verify the correctness of computer programs. The reason for that was the fact that PDL provided researchers with sufficient tools to represent variety of programs; conditionals, loops and their variants are representable in PDL. Moreover, program verification has always been a problem in theoretical computer science and even in recursion theory for the reasons based on the "halting problem".

Exercise 1.1. *What is the halting problem? State the theorem and prove it.*

1.2 Syntax and Semantics

Reader who is familiar with automata theory or recursion theory will immediately realize that PDL has a recursion theoretical flavor in order to be able to reflect the basic idea behind the computation by resorting to regular expressions. Recall that regular expressions are obtained by taking an alphabet with a symbol for empty string and closing it under concatenation, union and iteration. Therefore, PDL can be expected to reflect this construction. Modal flavor of PDL then will be clear in due course.

Exercise 1.2. *What is a regular expression? What is a regular language?*

The language of PDL is obtained in a rather unusual way compared to basic modal logic. In PDL, we have two countable disjoint sets for atoms. The set $\mathbf{P}$ will be the set of propositional formulae

(that we will denote with lowercase Roman letters such as $p, q, r \dots$) whereas the set $\mathbf{\Pi}$ will be the set of atomic programs (that we will denote with lowercase Greek letters such as $\alpha, \beta, \gamma \dots$)

The language of PDL will have the expected propositional operators. We will put the unary operator $\neg$ for negation and the binary operator $\rightarrow$ for material implication. We will also have the constants $\bot$ for falsehood and $\top$ for truth. The program operators of PDL, on the other hand, will be given as follows. The composition operator ; will express concatenation, the nondeterministic choice operator $\cup$ will denote the union and the Kleene star * will denote the arbitrary iteration.

Moreover, we will have mixed operators $[\cdot]$ and $?$. The modal necessitation operator is the familiar one. The test operator then $?$ will be used to test programs.

As we have indicated, the language of PDL has two distinct set of atoms: $\mathbf{P}$ and $\mathbf{\Pi}$. Therefore, we will call PDL a *two-sorted logic* as the formulae of PDL will be obtained from these two distinct sets of atoms in a special way.

The language of PDL, that we will denote as $\mathcal{L}_{PDL}$, is obtained recursively as follows.

$$\begin{array}{rcl} F & := & \bot \mid p \mid \neg F \mid F \rightarrow G \mid [\alpha]F \\ \alpha & := & a \mid \alpha;\beta \mid \alpha \cup \beta \mid \alpha^* \mid ?F \end{array}$$

In this language, we take $\top$, $\wedge$, $\vee$ as abbreviations in the usual sense. It is important to notice that the definitions of programs and propositions are interrelated and cannot be separated. The definition of a proposition F depends on a program α which in turn depends on a proposition in $?F$.

We already have hinted out what these constructions mean. Let us now make it clearer. The modally motivated formula $[\alpha]F$ will mean that "It is necessary that after executing α, the formula F is true". The formula $\alpha;\beta$ will mean that "First execute α, then execute β" reflecting the basic idea behind concatenation. Similarly, the formula $\alpha \cup \beta$ will mean that "choose either α or β nondeterministically and execute it" reflecting the basic idea behind union. Then, α^* will mean that "execute α nondeterministically chosen finite number of times ($n < \omega$)" reflection the basic idea behind iteration. Finally, $?F$ will mean that "test F; proceed if it is true, fail if its is false".

Notationally, in order to be able to avoid using parentheses, we will assign precedence to the operators following the traditional approach. Unary operators $\neg$, $[\cdot]$ and * will bind tighter than binary ones, and ; will bind tighter than $\cup$. In the literature, $\alpha;\beta$ is commonly written as $\alpha\beta$, and we will follow this tradition as well.

Exercise 1.3. *Parse the statement* $[\alpha^*\neg;\beta;\gamma \cup \delta^*;\epsilon]\neg F \rightarrow G$.

Furthermore, the possibility modality will be defined in the usual sense: $\langle\alpha\rangle F := \neg[\alpha]\neg F$ with the meaning that "there is a computation of α that terminates in a state that satisfies F". Nevertheless, there is a key difference which also reflects a basic modal logical fact. The formula $\langle\alpha\rangle F$ implies that α terminates whereas $[\alpha]F$ does not.

Exercise 1.4. *The formula* $\langle\alpha\rangle F$ *implies that* α *terminates whereas* $[\alpha]F$ *does not. Why?*

Hard Exercise 1.5. *Look for another logical system with a two-sorted language.*

For the semantics of PDL, we will use the traditional Kripke models which is familiar from basic modal logic. A PDL model $M = \langle W, \{R_\alpha\}, V\rangle$ is a tuple with a non-empty set W, a family of binary relations R_α defined on W and a valuation function $V : \mathbf{P} \rightarrow \wp(W)$ returning a subset of W for each propositional variable, namely the set of points where a given propositional variable is true.

Exercise 1.6. *Why do we impose such conditions on the carrier set (non-emptiness), the relation (binaryness) and the valuation (functionality)?*

We will define the truth of the formulae in $\mathcal{L}_{PDL}$ *inductively*. Recall that in the preceding paragraphs, we have defined the formulae recursively. Now, we will define how the truth is effected under the logical connectives in $\mathcal{L}_{PDL}$. We will start from the base case - namely, propositional variables. Then, we will consider the truth of the complex statements based on earlier constructions.

Satisfaction is a ternary relation that relates a model and a point in that model to a formula and denoted as $\models$.

$$\begin{array}{lll} M, w \models \bot & \text{iff} & \text{never} \\ M, w \models p & \text{iff} & w \in V(p) \\ M, w \models \neg\varphi & \text{iff} & \text{not-}(M, w \models \varphi) \text{ (notation: } M, w \not\models \varphi) \\ M, w \models \varphi \wedge \psi & \text{iff} & M, w \models \varphi \text{ and } M, w \models \psi \\ M, w \models [\alpha]F & \text{iff} & \forall v \in W(wR_\alpha v \rightarrow M, v \models F) \end{array}$$

The action modality $[a]$ has the expected dual which is denoted as $\langle a \rangle$, and is defined in the usual sense $\langle a \rangle := \neg[\neg a]$. Then, the semantics of $\langle a \rangle$ should be straightforward.

Exercise 1.7. *What is the semantics of $\langle a \rangle \varphi$?*

For a given formula F, we will denote the *extension* of F in the model M with $|F|^M$. Then, $|F|^M = \{w \in W : M, w \models F\}$. We will drop the superscript when it is obvious in which model we are working. Similarly, for a given state w, we will use $[[w]]_R$ to denote the set of accessible states via R. Thus, $[[w]]_R = \{v : (w, v) \in R\}$. We will drop the subscript when it is obvious.

Exercise 1.8. *Let M be given. What are the extensions of $\bot$, $\top$, $\varphi \wedge \psi$, $\neg\varphi$, $\varphi \rightarrow \psi$, $\langle\varphi\rangle$ in terms of $|\varphi|$ or $|\psi|$?*

As we have observed, the action a can be anything! It can be a composition of two actions, or it can be a test action etc. Therefore, we will analyze the relations that depend on those actions inductively. The following chart does that.

$$\begin{array}{lcl} R_{\alpha;\beta} & = & R_\alpha \circ R_\beta \\ R_{\alpha \cup \beta} & = & R_\alpha \cup R_\beta \\ R_{\alpha^*} & = & R_\alpha^* \\ R(?F) & = & \{(w, w) : M, w \models F\} \end{array}$$

1.3 Axioms

Now, we will give the axioms of PDL as follows.

1. Axioms for propositional logic
2. $[\alpha](F \rightarrow G) \rightarrow ([\alpha]F \rightarrow [\alpha]G)$ — Kripke Axiom
3. $[\alpha \cup \beta]F \leftrightarrow [\alpha]F \wedge [\beta]F$ — Union Axiom
4. $[\alpha; \beta]F \leftrightarrow [\alpha][\beta]F$ — Composition Axiom

5. $[?F]G \leftrightarrow (F \to G)$ Text Axiom
6. $F \wedge [\alpha][\alpha^*]F \leftrightarrow [\alpha^*]F$ Fixed Point Axiom
7. $F \wedge [\alpha^*](F \to [\alpha]F) \to [\alpha^*]F$ Induction Axiom

We have two rules of inference. The first is coming from propositional logic and called *modus ponens*: $F, F \to G \therefore G$. The second one is from modal logic, and called *necessitation rule*: $F \therefore [\alpha]F$.

These axioms and rules of inference do deserve some further comments. The second axiom is what is usually called the *normativity* axiom, or with the more famous name, the Kripke Axiom. The third axiom seems a bit counter-intuitive as we normally associate union operator with disjunction. Nevertheless, the nondeterministic choice operator $\cup$ means that both α and β action will result in F no matter which one you choose. In other words, in order to be able to have a nondeterministic choice, the options should not matter, hence they both should work. Induction axiom, is indeed very similar to the Peano's induction axiom. What it utters is the following. Assume F is true in the current state, and assume further that after any number of iterations of α, if F is still true, then it will be true that after one more iteration of α.

We will write $\vdash F$, if F is a theorem of PDL.

Hard Exercise 1.9. *Show that the following formulae are theorems of PDL.*

- $\langle\alpha\rangle(F \vee G) \leftrightarrow (\langle\alpha\rangle F \vee \langle\alpha\rangle G)$
- $\langle\alpha\rangle F \wedge [\beta]G \to \langle\alpha\rangle(F \wedge G)$
- $[\alpha](F \wedge G) \leftrightarrow ([\alpha]F \wedge [\alpha]G)$.
- $\langle\alpha\rangle\bot \leftrightarrow \bot$
- $[\alpha]\top$

Notice that these are valid in any normal modal logic.

Soundness of the given axiomatization is an easy exercise, and the proof is by induction (on what?).

Exercise 1.10. *Show that the given axioms are sound.*

Hard Exercise 1.11. *Show that monotonicity is a valid rule of inference: $F \to G \therefore [\alpha]F \to [\alpha]G$.*

1.4 Formalizing Programs

As we already emphasized, PDL has first been developed and studied for program verification. In this section, we will first define the correspondence between simple program operators and PDL syntax, then discuss them in some simple examples.

We define the programs in PDL as follows.

$$
\begin{array}{rcl}
\texttt{skip} & := & ?\top \\
\texttt{fail} & := & ?\bot \\
\texttt{if } F_1 \to \alpha_1 \mid \ldots \mid F_n \to \alpha_n \texttt{ fi} & := & ?F_1;\alpha_1 \cup \cdots \cup\ ?F_n;\alpha_n \\
\texttt{do } F_1 \to \alpha_1 \mid \ldots \mid F_n \to \alpha_n \texttt{ od} & := & (?F_1;\alpha_1 \cup \cdots \cup\ ?F_n;\alpha_n)^*;?(\neg F_1 \wedge \cdots \wedge \neg F_n) \\
\texttt{if } F \texttt{ then } \alpha \texttt{ else } \beta & := & ?F;\alpha\ \cup\ ?\neg F;\beta \\
\texttt{while } F \texttt{ do } \alpha & := & (?F;\alpha)^*;?\neg F \\
\texttt{repeat } \alpha \texttt{ until } F & := & \alpha;(?\neg F;\alpha)^*;?F \\
\{F\}\alpha\{G\} & := & F \to [\alpha]G
\end{array}
$$

Hard Exercise 1.12. *What do the programs* `skip` *and* `fail` *do?*

Hard Exercise 1.13. *Investigate the connections between PDL and Hoare logic.*

Hard Exercise 1.14. *Take natural numbers as your domain with the successor relation, assume that you are at the state* 0*. Consider the program expressed by the following formula.*

$$\varphi = (?\top;\texttt{move.to.next.state})^*;?\bot$$

What does this program do, does it terminate if it is executed at state 0*?*
Hint: Check $0 \models \varphi$*.*

1.5 More on Basics

In this section, we will observe several valid formulae of PDL.

Exercise 1.15. *Show that* $\langle\alpha;\beta\rangle F \leftrightarrow \langle\alpha\rangle\langle\beta\rangle F$ *and* $[\alpha;\beta]F \leftrightarrow [\alpha][\beta]F$ *are valid formulae of PDL.*

Notice that in PDL, atomic programs are simply letters $a, b, c, \ldots$ from some alphabet which is an abstraction from any specific and discrete domain of computation and programming. Therefore, atomic programs are the ones which are basic and they are executed in one single step.

Exercise 1.16. *Show that* $\langle ?F\rangle G \leftrightarrow F \wedge G$ *and* $[?F]G \leftrightarrow F \to G$ *are valid formulae of PDL.*

Hard Exercise 1.17. *Show that the following are valid formulae of PDL.*

- $[\alpha^*]F \to F$
- $[\alpha^*]F \to [\alpha]F$
- $\langle\alpha\rangle F \to \langle\alpha^*\rangle F$
- $[\alpha^*]F \leftrightarrow [\alpha^*\alpha^*]F$
- $[\alpha^*]F \leftrightarrow [\alpha^{**}]F$
- $[\alpha^*]F \leftrightarrow F \wedge [\alpha][\alpha^*]F$

Lecture 2

Completeness, Decidability and Complexity

PDL is complete and decidable. Nevertheless, satisfiability of a PDL formula is decided in EXPTIME making it a rather complex logical system.

2.1 Completeness

PDL is complete. All valid formuale are theorems. The way we show it is by the standard method (Blackburn *et al.*, 2001). We construct a modal for maximal consistent set (MCS) of formulae of PDL.

Exercise 2.1. *Given a formula F, how can we construct a maximally consistent set?*

Hard Exercise 2.2. *Observe that MCSs are not unique.*

Hard Exercise 2.3. *What is the Lindenbaum Lemma?*

Let us start off by making some observations about MCSs.

Lemma 2.4. *Let Σ be a MCS of formulae of PDL. Then, we have the following properties.*

- *Σ is consistent iff either $\Sigma \cup \{F\}$ or $\Sigma \cup \{\neg F\}$*
- *$F \wedge G \in \Sigma$ iff $F \in \Sigma$ and $G \in \Sigma$*
- *$F \in \Sigma$ iff $\neg F \notin \Sigma$*
- *$\bot \notin \Sigma$*

Proof. Exercise. ■

Similarly, we have the following observation about PDL formulae.

Lemma 2.5. *Let Σ and Σ' be two MCSs of formulae of PDL, and α a program. Then, the following are equivalent.*

- *For all formulae F, if $F \in \Sigma$ then $\langle\alpha\rangle F \in \Sigma'$*
- *For all formulae F, if $[\alpha]F \in \Sigma'$ then $F \in \Sigma$*

Proof. In class. ■

Now, we will construct a Kripke model whose states are MCSs. Therefore, we will define the accessibility relation accordingly.

Formally, our new model $N = \langle \mathcal{W}, \{\mathcal{R}_\alpha\}, \mathcal{V}\rangle$ where $\mathcal{W}$ is the set of MCSs of formulae and $\mathcal{R}$ is the accessibility relation between MCSs defined as follows. We put $\Sigma\mathcal{R}_\alpha\Gamma$ if for all formulae F, $F \in \Gamma$ implies $\langle\gamma\rangle F \in \Sigma$ for $\Sigma, \Gamma \in \mathcal{W}$. Moreover, we put $\mathcal{V}(p) = \{\Sigma \in \mathcal{W} : p \in \Sigma\}$.

Exercise 2.6. *Equivalently, we can define $\mathcal{R}_\alpha$ as follows. We put, $\Sigma\mathcal{R}_\alpha\Gamma$ if for all formulae F, $[\alpha]F \in \Sigma$ implies $F \in \Gamma$. Prove that the two definitions are equivalent.*

The reason why we consider MCS is this. In MCSs, truth is identified with *membership*. Therefore, $|F|^N = \{\Sigma \in \mathcal{W} : F \in \Sigma\}$. For programs, we then have $|\alpha|^N = \{(\Sigma, \Gamma) :$ for all $F, F \in \Gamma$ implies that $\langle\alpha\rangle F \in \Sigma\}$.

Exercise 2.7. *Show that $|\bot|^N = \emptyset$*

Now we can show the extension of $\langle\alpha\rangle F$.

Hard Exercise 2.8. *Show that $|\langle\alpha\rangle F| = |\alpha| \circ |F|$ where $\circ$ denotes composition.*

This result can be extended to the following observations.

$$\begin{aligned} |\alpha \cup \beta| &= |\alpha| \cup |\beta| \\ |\alpha;\beta| &= |\alpha| \circ |\beta| \\ |?F| &= \{(\Sigma, \Sigma) : \Sigma \in |F|\} \end{aligned}$$

Now we have almost everything to show the completeness of PDL.

Theorem 2.9. *PDL is complete with respect to the given axiomatization. In other words $\models F$ implies $\vdash F$.*

Proof. In class. ■

2.2 Decidability and Complexity

Decidability of PDL uses the finite model property obtained in a special method, called Fischer - Ladner closure. Finite model property states that any satisfiable formula is satisfiable in a finite model.

We will give a theorem without a proof here.

Theorem 2.10. *Let F be a satisfiable formula of PDL. Then F is satisfied in a Kripke frame with no more than $2^{length(F)}$ states.*

The proof uses the Fischer - Ladner closure of F. Here, we will only state how the Fischer - Ladner closure is obtained.

We define two functions $\mathbf{FL}$ and $\mathbf{FL}^{\Box}$. The first one takes propositional formulae and returns a set of propositional formulae whereas the latter one does the same thing for modal formuale. The functions FL and $FL^{\Box}$ are defined inductively.

$$
\begin{aligned}
\mathbf{FL}(p) &:= \{p\} \\
\mathbf{FL}(\bot) &:= \{\bot\} \\
\mathbf{FL}(F \to G) &:= \mathbf{FL}(F) \cup \mathbf{FL}(G) \cup \{F \to G\} \\
\mathbf{FL}([\alpha]F) &:= \mathbf{FL}^{\Box}([\alpha]F) \cup \mathbf{FL}(F) \\
\mathbf{FL}^{\Box}([\alpha]F) &:= \{[\alpha]F\} \\
\mathbf{FL}^{\Box}([\alpha \cup \beta]F) &:= \mathbf{FL}^{\Box}([\alpha]F) \cup \mathbf{FL}^{\Box}([\beta]F) \cup \{[\alpha \cup \beta]F\} \\
\mathbf{FL}^{\Box}([\alpha;\beta]F) &:= \mathbf{FL}^{\Box}([\alpha][\beta]F) \cup \mathbf{FL}^{\Box}([\beta]F) \cup \{[\alpha;\beta]F\} \\
\mathbf{FL}^{\Box}([\alpha^*]F) &:= \mathbf{FL}^{\Box}([\alpha][\alpha^*]F) \cup \{[\alpha^*]F\} \\
\mathbf{FL}^{\Box}([?F]G) &:= \mathbf{FL}(F) \cup \{[?F]G\}
\end{aligned}
$$

Exercise 2.11. *What is the Fischer Ladner closure of a negated propositional and a negated modal formula.*

Exercise 2.12. *If $G \in \mathbf{FL}(F)$, then $\mathbf{FL}(G) \subseteq \mathbf{FL}(F)$.*

It is now intuitive to see that the cardinality of $\mathbf{FL}$ is bound by a function of the length of F, namely, an upper bound exists. We will skip the tedious details here and jump to the main theorem.

This upper-bound is helpful for SATISFIABILITY problem and as a result we conclude the following

Theorem 2.13. *Satisfiability problem for PDL is EXPTIME-complete.*

Lecture 3

Applications: Game Logic

Game theory is an interesting and exciting field. Even if it is mostly studied by economists, mathematicians have an increasing interest towards the field (Nash and Aumann are mathematicians and were awarded the Nobel Memorial Prize in Economics by Bank of Sweden).

In this chapter, we will study an abstraction of games. The framework we will use depends on PDL with some tweaks. Therefore, it presents a good example as a mathematical and practical application of PDL (Parikh & Pauly, 2003; Parikh, 1985). Furthermore, there are enormous body of work based on game logic.

Hard Exercise 3.1. *Assume that two people are given a (circular) cake and asked to share it* fairly. *Assume that they are not allowed to use any kind of measurement device, describe an algorithm that* guarantees *a fair division among two people. Can you generalize your solution to 3 people?*

Hard Exercise 3.2. *Describe Banach-Knaster procedure for fair division.*

3.1 Syntax and Semantics

Game logic (GL) is important from two aspects. First, it is a very nice extension of PDL, and second it does not use Kripke semantics, but *neighborhood semantics* making it one of the most interesting logical systems that use neighborhood semantics. Also, note that GL is a logic to "reason about determined 2-player games" (Parikh & Pauly, 2003). To follow the tradition, we will call the players $\forall$ and $\exists$.

The syntax of GL is an extension of that of PDL. Then, the formulae are obtained in the following fashion.

$$\begin{array}{rcl} F & := & \bot \mid p \mid \neg F \mid F \to G \mid \langle\gamma\rangle F \\ \gamma & := & g \mid \gamma;\eta \mid \gamma \cup \eta \mid \alpha^* \mid ?F \mid \gamma^d \end{array}$$

For $p \in \mathbf{P}$ and $\gamma \in \mathbf{\Gamma}$ where $\mathbf{\Gamma}$ is the set of atomic games. We define $[\gamma]F := \neg\langle\gamma\rangle\neg F$.

In this setting, $\cup$ and * indicate the choice and iteration operations for the Player $\exists$ only. One can then dually define the choice and iteration operations $\cap$ and $^\times$ for the Player $\forall$ as well.

The crucial part of GL is the reading of the modal formula $\langle\gamma\rangle F$. In GL, $\langle\gamma\rangle F$ means that the Player $\exists$ has a F-strategy in game γ. On the other hand, $[\gamma]F$ expresses that $\exists$ does not have a $\neg F$ strategy, which means that $\forall$ has a F-strategy.

In a similar fashion, $\gamma_1 \cup \gamma_2$ denotes the game where $\exists$ chooses which of the two subgames to keep playing. The sequantial composition of $\gamma_1; \gamma_2$ consists of first playing γ_1 and then playing γ_2. In the iterated game, γ^*, $\exists$ chooses how often to play γ (possibly zero).

The additional construct γ^d is somehow unique to GL. Playing γ^d is the same as playing γ with player's roles reversed. In other words, any move made by $\exists$ in γ will now be made by $\forall$ in γ^d, and vice versa.

Exercise 3.3. *What is the meaning of playing $\gamma_1 \cap \gamma_2$ and $\gamma^\times$?*

The semantics of GL is not straightforward. Since GL is not normal, Kripke structures will not be sufficient. For this reason, we will use neighborhood structures.

Hard Exercise 3.4. *Argue as to why GL is not normal.*

A game model M is given as follows: $M = \langle W, \{E_g : g \in \mathbf{\Gamma}\}, V\rangle$. Here, the only different construction is the *effectivity functions*. Effectivity functions are defined from W to the set of subsets of W, i.e. $E_\gamma : W \to \wp(\wp(W))$, and are monotonic, i.e. $X \in E_\gamma(s)$ and $X \subseteq X'$ imply that $X' \in E_\gamma(s)$.

The idea behind the effectivity functions is this. We stipulate that $X \in E_\gamma(s)$ holds whenever Player $\exists$ has a strategy at the game state s that can guarantee an outcome in X.

Hard Exercise 3.5. *Why are the effectivity functions assumed to be monotonic? What is the game theoretical explanation of it?*

The semantics of GL uses effectivity functions. Boolean cases are as usual. So, we will now only define the semantics of the dynamic modality.

$$M, w \models \langle\gamma\rangle F \quad \text{iff} \quad wE_\gamma|F|$$

where $|F|$ is the extension of F.

Notice that in the semantic definition, E_γ take *any* game. However, in the GL model, we took only the effectivity functions with atomic games. Thus, we need to state how we can combine effectivity functions to obtain effectivity functions for more complex games.

Let us first introduce a piece of notation. Define $E_\gamma(Y) := \{w \in W : wE_\gamma Y\}$. Now, we inductively define the following.

$$\begin{aligned}
E_{\gamma;\eta}(Y) &:= E_\gamma(E_\eta(Y)) \\
E_{\gamma\cup\eta}(Y) &:= E_\gamma(Y) \cup E_\eta(Y) \\
E_{?F}(Y) &:= |F| \cap Y \\
E_{\gamma^d}(Y) &:= (E_\gamma(Y^c))^c \\
E_{\gamma*}(Y) &:= \mu X.(Y \cup E_\gamma(X))
\end{aligned}$$

Hard Exercise 3.6. *For which partial orders or for which lattices, do the fixed points exist?*

Hard Exercise 3.7. *What is the Tarski-Knaster Theorem about fixed-points?*

Hard Exercise 3.8. *Consider the statement $E_{\gamma*}(Y) = \mu X.(Y \cup E_\gamma(X))$. Why does the least fixed point exist?*

3.2 Axioms

Even if we started out with neighborhood semantic for GL, it is also possible to associate Kripke models with GL. In that case, for each atomic game g, one needs to associate an accessibility relation R_g to g. Therefore, given a game model M, one can always not construct a Kripke model that M corresponds to.

Exercise 3.9. *Why one cannot always construct a Kripke model given a game model?*

However, given a Kripke model $M = \langle W, \{R_g : g \in \mathbf{\Gamma}\}, V\rangle$, one can construct a game model $M' = \langle W, \{E_g : g \in \mathbf{\Gamma}\}, V\rangle$ by putting wE_gX iff there is some $v \in X$ such that wR_gv hold.

Notice that the only distinction of GL is the dual operator. Nevertheless, if we omit it, we end up with a nice system.

Hard Exercise 3.10. *Convince yourselves that dual-free GL is PDL.*

Such semantical considerations immediately give rise to some notion of equivalence between dynamic GL models. Since bisimulations are the common tools for such truth preserving equivalence, one can easily define them within the context of GL.

Hard Exercise 3.11. *How can you define bisimulations in GL?*

Following axiomatization has been suggested for game logic. It is similar to that of PDL with one additional axiom for dual operator, and it misses the normality axiom.

1. Axioms for propositional logic
2. $\langle \alpha \cup \beta\rangle F \leftrightarrow \langle \alpha\rangle F \vee \langle \beta\rangle F$
3. $\langle \alpha; \beta\rangle F \leftrightarrow \langle \alpha\rangle\langle \beta\rangle F$
4. $\langle ?F\rangle G \leftrightarrow (F \to G)$
5. $F \vee \langle \alpha\rangle\langle \alpha^*\rangle F \leftrightarrow \langle \alpha^*\rangle F$
6. $\langle \gamma^d\rangle F \leftrightarrow \neg\langle \gamma\rangle\neg F$

The rule of inferences are modus ponens, monotonicity and fixed point rule. Modus ponens is as expected. Monotonicity is given as follows: $F \to G \therefore \langle \gamma\rangle F \to \langle \gamma\rangle G$. The fixed point rule is as follows: $(F \vee \langle \gamma\rangle G) \to G) \to \therefore \langle \gamma^*\rangle F \to G$.

Exercise 3.12. *Explain why the union axiom looks different than that of PDL.*

3.3 Satisfiability

The satisfiability of GL is known to be EXPTIME. The very complicated argument translates the formulae of GL to those of modal-μ calculus. The translation is two-fold. The easy one is the translation of formulae from GL into μ-calculus. A bit more complicated translation is the transformation of GL models into the fixed-point models.

3.4 Completeness: A Homework

Hard Exercise 3.13. *Prove that GL is complete with respect to the given semantics.*

Well, it has been conjectured that GL is complete with respect to the given axiomatization. Completeness of some fragments of GL has been shown, however, the completeness of (full) GL is still an open problem (as of June 2010).

Lecture 4

Applications: Dynamic Epistemic Logic

Dynamic logics is not limited to express programs or games. They can also express change in *knowledge*. What do we mean by "change in knowledge" is a rather epistemological concept.

We will abstract variety of knowers and their respective knowledge (a.k.a *multi-agent system* in terms of computer science and artificial intelligence) and formalize their interaction. For this purpose of us, we will use *modal epistemic logic* which is nothing but a basic modal logic with a bit different interpretation. The basic system of modal epistemic logic was suggested by Hintikka in 1960s (Hintikka, 1962). Nevertheless, we will not stop at that static level, and incorporate a dynamic operator to express *knowledge update* (Plaza, 1989).

The way we will update our epistemic states is as follows. We will assume that knowers are exposed a truthful public announcement by an external agent. Then, they *update* their epistemic states in such a way that they get rid off the states and epistemic situations that do not agree with the announcement. For instance, consider the following story. Assume that you got to the bus station to take a bus to İzmir and consequently saw a bus in distance leaving. Now, you are puzzled that whether the bus which has just left was the one you were supposed to take. Then, you hear an announcement in the bus station saying that the bus which has just left was going to Ankara. Then, you eliminate the option that the bus you saw from a distance could be the bus to İzmir. Now, you know that you did not miss your bus.

PAL is not the only dynamic paradigm, there are many other paradigms in formal philosophy that tries to model knowledge and belief update (van Ditmarsch *et al.*, 2007).

Hard Exercise 4.1. *What is belief? How can it be expressed in modal logic? What is the difference between belief and knowledge?*

4.1 Syntax and Semantics

The syntax of *public announcement logic* (PAL, henceforth) is an extension of basic epistemic logic. In basic epistemic logic, we take K as our primitive modality and index it with respect to an agent i. Therefore, the formula $\mathsf{K}_i F$ reads "The knower (agent) i knows that F".

The syntax of basic modal logic is that of basic modal logic, and constructed recursively from $\mathbf{P}$ in the usual sense.

The semantics of PAL is given with respect to Kripke models. Given a model $M = \langle W, \{R_i\}_{i\in I}, V\rangle$ where $W \neq \emptyset$, each R_i corresponds to the epistemic accessibility of each agent i, and $V : \mathbf{P} \to \wp(W)$ is a valuation function, we define the semantics of epistemic modality as expected.

$$M, w \models \mathsf{K}_i F \text{ iff } \forall v \in W(wR_iv \to M, v \models F)$$

The dual of K_i is denoted as L_i, and defined as $\mathsf{L}_i F := \neg\mathsf{K}_i\neg F$.

Exercise 4.2. *What is the semantics of L_i modality?*

Now, we can add the dynamic modality. The formula $[F]G$ will read that "after the announcement of F, G will be true".

The extended language of PAL is then constructed recursively from the language of modal epistemic logic in the usual fashion. Notice that announcements and knowledge thus can be announced. So, we can have formulae such as $[\mathsf{K}_i F]G$ or $[[F] \wedge G]\mathsf{K}_i F$.

However, public announcements update the epistemic model. Therefore, we obtain a submodel by omitting the states that do not validate the announcement.

Given a model $M = \langle W, \{R_i\}_{i\in I}, V\rangle$ updated model after the announcement F is $M_F = \langle W', \{R'_i\}_{i\in I}, V'\rangle$ where $W' = |F|^M$, $R'_i = R_i \cap (W' \times W')$ and $V'(p) = V(p) \cap W'$.

Notice that M_F depends on F.

4.2 Axioms

Knowledge modality is reflexive, transitive and symmetric. Therefore, we assume S5 logic for knowledge.

Exercise 4.3. *Express reflexivity, transitivity and symmetry by using the given R_i.*

Hard Exercise 4.4. *What are the defining modal properties for reflexivity, transitivity and symmetry?*

Hard Exercise 4.5. *Convince yourself that philosophically, knowledge modality satisfies reflexivity, transitivity and symmetry.*

Hard Exercise 4.6. *Confuse yourself that philosophically, knowledge modality may not satisfy transitivity and symmetry.*

Let us first give the axioms of PAL.

1. Axioms of propositional logic
2. $\mathsf{K}_i(F \to G) \to (\mathsf{K}_i F \to \mathsf{K}_i G)$
3. $\mathsf{K}_i F \to F$
4. $\mathsf{K}_i F \to \mathsf{K}_i\mathsf{K}_i F$
5. $F \to \mathsf{K}_i\mathsf{L}_i F$
6. $[F](G \to H) \leftrightarrow ([F]G \to [F]H)$

7. $[F]p \leftrightarrow (F \rightarrow p)$
8. $[F](G \wedge H) \leftrightarrow ([F]G \wedge F[H])$
9. $[F]\neg G \leftrightarrow (F \rightarrow \neg[F]G)$
10. $[F]\mathsf{K}_i G \leftrightarrow (F \rightarrow \mathsf{K}_i[F]G)$
11. $[F][G]H \leftrightarrow [F \wedge [F]G]H$

It can be seen as an overkill to call the statements about PAL as axioms. Notice that what the axioms about PAL do is to reduce the complexity of the formula. In the 10. Axiom, for instance, knowledge after an announcement is reduced to a situation when the announcement becomes known. Since, we already have a very well working epistemic logic, reducing the PAL formulae to epistemic logic is helpful.

Hard Exercise 4.7. *Show that* $[F](G \rightarrow H) \leftrightarrow ([F]G \rightarrow [F]H)$ *is derivable from the given axioms.*

Now we will show the validity of 10. Axiom as an example and leave the rest to the reader. Given an arbitrary model M and a state w in M.

$$
\begin{array}{lll}
M, w \models F \rightarrow \mathsf{K}_i[F]G & \text{iff} & M, w \models F \text{ implies } M, s \models \mathsf{K}_i[F]G \\
& \text{iff} & M, w \models F \text{ implies } (\forall t \in M . s R_i t \text{ implies } M, t \models [F]G) \\
& \text{iff} & M, w \models F \text{ implies} \\
& & (\forall t \in M . s R_i t \text{ implies } (M, t \models F \text{ implies } M_F, t \models G)) \\
& \text{iff} & M, w \models F \text{ implies} \\
& & (\forall t \in M . M, t \models F \text{ and } s R_i t \text{ implies } M_F, t \models G \\
& \text{iff} & M, w \models F \text{ implies } (\forall t \in M_F . s R_i t \text{ implies } M_F, t \models G \\
& \text{iff} & M, w \models F \text{ implies } M_F, s \models \mathsf{K}_i G \\
& \text{iff} & M, w \models [F]\mathsf{K}_i G
\end{array}
$$

Hard Exercise 4.8. *Show the soundness of all other axioms.*

4.3 Completeness and Decidability and Complexity

The completeness of PAL is immediate. Since, every formualae in the language of PAL can be reduced to a formula in the basic modal epistemic logic, and since the basic modal logic is complete we deduce that PAL is complete. Recall that we already have made sure that we have reducibility axioms for all possible types of formulae.

Based on the above observations, we can show that PAL is decidable.

Hard Exercise 4.9. *Show that PAL is decidable.*

Similarly, based on the above observations, we can show that the complexity of the satisfiability problem for PAL is PSPACE-complete.

Hard Exercise 4.10. *Show that the complexity of the satisfiability problem for PAL is PSPACE-complete.*

4.4 An Example: Muddy Children

A famous example for an application of PAL is the muddy children puzzle (van Benthem *et al.*, 2005; van Benthem, 2006; van Ditmarsch *et al.*, 2007). The literature is full some other puzzles that PAL can easily be applied.

What is the muddy children puzzle then? There are n children playing together. During their play some of the children, say k of them, get mud on their foreheads. Each can see the mud on others but not on his own forehead. Along comes a father, who says, At least one of you has mud on your head. He then asks the following question, over and over: Can any of you prove that you have mud on your head? Assuming that all the children are perceptive, intelligent, truthful, and that they answer simultaneously, what will happen? There is a proof that the first $k?1$ times the father asks the question, the children will all say no but that the k-th time the children that are dirty will answer yes.

Let us model this puzzle for 3 players and take three children named 1, 2 and 3. Let us state them being dirty and clean by **D** and **C** respectively. The vector **DDC** then will mean that the children 1 and 2 are dirty while 3 is clean. Moreover, the current state, let us assume, will be denoted by a star. We are at the current real state **DDC**, but do not know it.

The Picture 4.1 models and illustrates the situation before any announcemeny by the father (van Benthem, 2006). First announcement of the father eliminates the **CCC** state and yields the Picture 4.2. Then, when no child knows his status yet, the state **DDD** disappears, and what we have then is Picture 4.3. Finally, the second announcement eliminates the last option and yields the following model in Picture 4.4.

For many researchers, such puzzles are genuine examples of dynamic epistemologies. Moreover, recalling the material we have covered in the previous section, it is also possible to approach such puzzles from PDL point of view.

Hard Exercise 4.11. *Model the muddy children puzzle in GL, and state some satisfiable formulae in your model.*

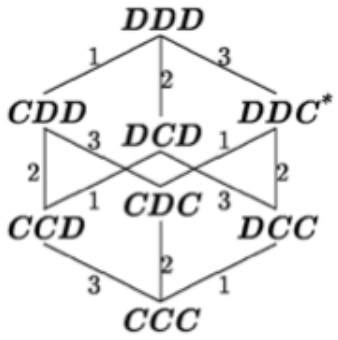

Figure 4.1: Model before the start of the game

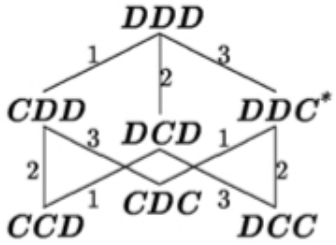

Figure 4.2: Model after the first announcement

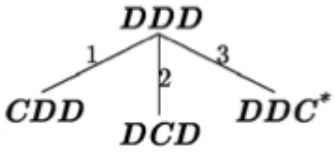

Figure 4.3: Model after children realized that none knows his status yet

Figure 4.4: Model after the last announcement

Bibliography

BLACKBURN, PATRICK, RIJKE, MAARTIJN DE, & VENEMA, YDE. 2001. *Modal Logic*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press.

HAREL, DAVID, KOZEN, DEXTER, & TIURYN, JERZY. 2000. *Dynamic Logic*. Cambridge, MA: MIT Press.

HINTIKKA, JAAKKO. 1962. *Knowledge and Belief*. Cornell University Press.

PARIKH, ROHIT. 1985. The logic of games and its applications. *Annals of Discrete Mathematics*, **24**, 111–140.

PARIKH, ROHIT, & PAULY, MARC. 2003. Game Logic - An Overview. *Studia Logica*, **75**(2), 165–182.

PLAZA, JAN A. 1989. Logic of Public Communication. *Pages 201–216 of:* EMRICH, M. L., PFEIFER, M. S., HADZIKADIC, M., & RAS, Z. W. (eds), *4th International Symposium on Methodologies for Intelligent Systems*.

VAN BENTHEM, JOHAN. 2006. "One is a Lonely Number": Logic and Communication. *In:* CHATZIDAKIS, Z., KOEPKE, P., & POHLERS, W. (eds), *Logic Colloquium '02*. Lecture Notes in Logic, vol. 27. Association for Symbolic Logic.

VAN BENTHEM, JOHAN, VAN EIJCK, JAN, & KOOI, BARTELD. 2005. *Logics of Communication and Change*. Tech. rept. Institute for Logic, Language and Computation.

VAN DITMARSCH, HANS, VAN DER HOEK, WIEBE, & KOOI, BARTELD. 2007. *Dynamic Epistemic Logic*. Springer.