Propositional Dynamic Logic

Zihan Hu(2019012330)

November 15, 2020

1. The syntax and the semantics of PDL

1.1 The syntax of PDL

In the language of PDL, there are two components, say programs P and propositions F. They are defined as follows:

$$\begin{aligned} F &:= \text{atomic propositions}|\neg F|F \wedge F|\,\langle P\rangle\, F \\ P &:= \text{atomic programs}|\,(P;P)\,|\,(P \cup P)\,|P^*|\,(F)? \end{aligned}$$

Note that test $(F)?$ takes a proposition into a program. $\langle\ \rangle$ takes a program and a proposition into a proposition.

1.2 The semantics of PDL

Proposition can be seen as a set of the states where it holds true. Program can be seen as a binary relation (u, v) over states where a successful execution of the program can lead u to v. To be more specific, $M, s \models F$ says that F is true at the state s. $M, s_1, s_2 \models P$ says that there is successful execution of the program P that lead s_1 to s_2.

The notation ';' is a sequence of two programs, '$\cup$' is choice of two programs ,'*' is an iteration and '?' just check whether the proposition holds true ta that state. The notation '$\neg$' and '$\wedge$' are used in the usual way. $\langle P\rangle\, F$ just says that there is a successful execution of the program P that ends in a position where F holds true. $[\pi]\varphi$ is shorthand for $\neg\,\langle\pi\rangle\neg\varphi$.

Its formal form is listed in [1] as follows,

$$\begin{aligned} &M, s \models p && \text{iff } s \in V(p) \\ &M, s \models\neg\psi && \text{iff not} M, s \models \psi \\ &M, s \models \psi_1 \wedge \psi_2 && \text{iff } M, s \models \psi_1 \text{ and } M, s \models \psi_2 \\ &M, s \models \langle\pi\rangle\, \psi && \text{iff } \exists s \text{ such that } M, s, s' \models \pi \text{ and } M, s' \models \psi \end{aligned}$$

$M, s_1, s_2 \models a$	iff $(s_1, s_2) \in R_a$
$M, s_1, s_2 \models \pi_1; \pi_2$	iff $\exists s_3$ such that $M, s_1, s_3 \models \pi_1$ and $M, s_3, s_2 \models \pi_2$
$M, s_1, s_2 \models \pi_1 \cup \pi_2$	iff $M, s_1, s_2 \models \pi_1$ or $M, s_1, s_2 \models \pi_2$
$M, s_1, s_2 \models \pi^*$	iff some finite sequence of π-transitions in M connects the state s_1 with the state s_2
$M, s_1, s_2 \models (\phi)?$	iff $s_1 = s_2$ and $M, s_1 \models \phi$

In [2], the author gives axioms and rules of inference for PDL and proves the completeness of it.

2 Connections between PDL and PAL

In PDL, a program can be considered as binary relations between states. After a public announcement, the accessibility relation of an agent will change. So it is natural to consider the connections between PDL and PAL.

2.1 Reduce PAL to PDL

In [3], the author shows how to reduce general dynamic epistemic logic to PDL via program transformation. As a special case, we can reduce public announcement logic to PDL by program transformation.

Using the notation in [3], the action model of public announcement $[!\varphi]$ is $P_\varphi = ([s_0], \text{pre}, T)$ where $\text{pre}(s_0) = \varphi$, $T(a) = (s_0, s_0)$ for all agents a. Basically, it just says that only when the precondition $\text{pre}(s_0) = \varphi$ is satisfied, the action s_0 (public announcement $[!\varphi]$) can be taken. The map T from the set of agents to binary relations of the action states tells that all the agents a can not distinguish the action s_0 from itself. For agent a, let a be a program. If we execute it, we can go from a world w to a world w' where $(w, w') \in \{(w, w') | \text{agent } a \text{ can not distinguish } w \text{ from } w'\}$.

Then $K_a\psi$ can be converted into $[a]\psi$. $[!\varphi]$ is equivalent to execute the pointed action model (P_φ, s_0). Therefore, we can convert $[!\varphi]\, p$, $[!\varphi]\neg\psi$, $[!\varphi]\,(\psi_1 \wedge \psi_2)$ and $[!\varphi]\, K_a\psi$ into $[P_\varphi, s_0]p$, $[P_\varphi, s_0]\neg\psi$, $[P_\varphi, s_0]\,(\psi_1 \wedge \psi_2)$ and $[P_\varphi, s_0][a]\psi$ respectively.

The program transformation for P_φ has only one program transformer $T_{00}^{P_\varphi}$. $T_{00}^{P_\varphi}(\pi)$ is a program. If we execute it, we can go from a world w to a world w' if and only if there is a π path from (w, s_0) to (w', s_0) in the model after we take the action (P_φ, s_0) (the public announement that φ). Using the reduction axioms shown in [3], we can get that for proposition letters p and agent a,

$$
\begin{aligned}
[P_\varphi, s_0]p &\leftrightarrow (\text{pre}(s_0) \rightarrow p) \leftrightarrow (\varphi \rightarrow p) \\
[P_\varphi, s_0]\neg\psi &\leftrightarrow (\text{pre}(s_0) \rightarrow \neg[P_\varphi, s_0]\psi) \leftrightarrow (\varphi \rightarrow \neg[P_\varphi, s_0]\psi) \\
[P_\varphi, s_0]\,(\psi_1 \wedge \psi_2) &\leftrightarrow ([P_\varphi, s_0]\psi_1 \wedge [P_\varphi, s_0]\psi_2) \\
[P_\varphi, s_0][a]\psi &\leftrightarrow \left[T_{00}^{P_\varphi}\,(a)\right][P_\varphi, s_0]\psi \leftrightarrow [\text{pre}(s_0)?; a]\,[P_\varphi, s_0]\psi \leftrightarrow [\varphi?; a]\,[P_\varphi, s_0]\psi
\end{aligned}
$$

In this way, with the help of program transformation, we can reduce PAL to PDL.

2.2 Derive recursion axioms for PAL via PDL

It is easy to derive some recursion axioms for PAL.

From section 2.1, we can convert $[!\varphi]\, p$, $[!\varphi]\neg\psi$, $[!\varphi]\,(\psi_1 \wedge \psi_2)$, $[!\varphi]\, K_a\psi$ into $[P_\varphi, s_0]p$, $[P_\varphi, s_0]\neg\psi$, $[P_\varphi, s_0]\,(\psi_1 \wedge \psi_2)$ and $[P_\varphi, s_0][a]\psi$ respectively.

Then from $[P_\varphi, s_0]p \leftrightarrow (\varphi \to p)$, we can get $[!\varphi]p \leftrightarrow (\varphi \to p)$.

From $[P_\varphi, s_0]\neg\psi \leftrightarrow (\varphi \to \neg[P_\varphi, s_0]\psi)$, we can get $[!\varphi]\neg\psi \leftrightarrow (\varphi \to\neg[!\varphi]\psi)$.

From $[P_\varphi, s_0]\,(\psi_1 \wedge \psi_2) \leftrightarrow ([P_\varphi, s_0]\psi_1 \wedge [P_\varphi, s_0]\psi_2)$, we can get $[!\varphi]\,(\psi_1 \wedge \psi_2) \leftrightarrow ([!\varphi]\psi_1 \wedge [!\varphi]\psi_2)$.

From $[P_\varphi, s_0][a]\psi \leftrightarrow [\varphi?; a]\,[P_\varphi, s_0]\psi$, we can get $[!\varphi]K_a\psi \leftrightarrow [\varphi?; a]\,[!\varphi]\psi \leftrightarrow [\varphi?][a]\,[!\varphi]\psi \leftrightarrow (\varphi \to [a][!\varphi]\psi) \leftrightarrow (\varphi \to K_a[!\varphi]\psi)$. Note that it is simpler than the axiom shown in the slides.

Note that the fifth axiom $[!\varphi][!\psi]\alpha \leftrightarrow [!\,(\varphi \wedge [!\varphi]\psi)]\alpha$ shown in the slides is unneccessary, because we can use the above axioms to reduce $[!\psi]\alpha$ first.

2.3 About Kleene star

The Kleene star is an important component of PDL. In some cases, we need Kleene star to convert some extensions of PAL into PDL.

For example, we need Kleene star to convert common knowledge among group into PDL. $[!\varphi]\, C_G\psi$ can be converted into $[P_\varphi, s_0][\left(\bigcup_{g\in G} g\right)^*]\psi$. As shown in [3], it is equivalent to $[\left(\varphi?; \bigcup_{g\in G} g\right)^*][P_\varphi, s_0]\psi$. In fact, in [4], the authors point out that every sentence of the extension of PAL with the common knowledge is effectively equivalent to a sentence of PDL.

We also need Kleene star to express $[!\varphi^*]\,\psi$ (after repeated public announcement that φ, ψ holds true). But note that in [4], the authors show that the iterated modal relativation is undecidable while PDL is decidable. So $[!\varphi^*]\,\psi$ can not be converted into PDL directly. Maybe some extensions of PDL is needed.

3 Connections between PDL and games

3.1 The syntax and the semantics of dynamic game logic

In the language of DGL, there are two components: formulas F and game expressions G. They are defined as follows:

$$F := \text{atomic formulas}|\neg F|F \vee F|\{G\}F$$
$$G := \text{atomic game expressions}|G \cup G|G^d|G;G|?F$$

It is similar to the language of the propositional dynamic logic. Formulas are similar to propositions and game expressions are similar to programs. ';' is for sequential composition, '$\cup$' is for a choice of the first player E and '?' taking formulas into game expressions is for a test game. Dual '^{d}' is for role switch. '$\neg$', '$\vee$' are used in the usual way. $\{G\}F$ just says that the first player E has a strategy to force F in the game G.

Note that if the two players play the game G starting from a point, the player E can only force a set of outcomes in general. (By taking different strategies, he may be able to force multiple sets of outcomes.) So the game G can be considered as a binary relation of point and sets of outcomes that E can force starting from the point, which is different from the program that can be considered as a binary relation between states. This is the key of $P; (P' \cup P'') = (P; P') \cup (P; P'')$ in PDL but $G; (G' \cup G'') \neq (G; G') \cup (G; G'')$ in DGL.

3.2 Define strategies with PDL

It is easy to define any specific strategies σ in a finite game with PDL. In a finite game, σ is a finite set of the form $\{(s,t) | \text{in the strategy } \sigma\text{, if the player is at the point } s\text{, he may go to } t\}$. Denote the set as S. For any point s, let f_s denote the proposition such that it is true only at the point s. For any pair (s,t) such that we can reach t from s in a move, let $A_{(s,t)}$ denote the corresponding move. As shown in [5], we can define σ as $\bigcup_{(s,t)\in S} \left(f_s?; A_{(s,t)}; f_t?\right)$. But in the infinite games, the strategies may be infinite set of pairs, so we can not define it in the way above.

In a finite game, we can also define power in PDL. For example, in a finite two-person game of perfect information, the first player E can force power ψ can be defined in PDL as $\langle \text{move}_E \rangle [\text{move}_A] \langle \text{move}_E \rangle [\text{move}_A] \ldots \psi$. That either E can force ψ or A can force $\urcorner\psi$ can be expressed in $\langle \text{move}_E \rangle [\text{move}_A] \langle \text{move}_E \rangle [\text{move}_A] \ldots \psi \vee [\text{move}_E] \langle \text{move}_A \rangle [\text{move}_E] \langle \text{move}_A \rangle \ldots \urcorner\psi$.

In real games, usually the players not only care about whether they 'win' or 'lose', but only care about what they get after the game. That is, the players have preferences over all the possible outcomes. Under the common knowledge that both players are rational, the two players will play the game according to Backward Induction strategies. As shown in [5], Backward Induction strategy is also definable in PDL, say it is the relation σ satisfying $(\text{turn}_i \wedge \langle \sigma^* \rangle (\text{end} \wedge p)) \to [\text{move}_i] \langle \sigma^* \rangle (\text{end} \wedge \langle \text{pref}_i \rangle p)$ for every player i and every propositions p where pref_i can be considered as a program which can let us go from the current state to any state such that the player i perfers to the current state.

As discussed in the lecture, we can add public announcement to the game in order to cooperate with the other player. Both PAL and DGL are related to PDL. Maybe it is possible to describe strategies in games with public announcement with PDL.

4 Conclusions

By the discussion above, propositional dynamic logic can be used in many scenarios. The test and the iteration makes it extremely expressive.

Acknowledgements: Thanks the professor Johan van Benthem and the teaching assistant Li Lei for their tons of useful advice on the mini paper.

References

[1]Van Benthem, Johan. 2010. Modal Logic for Open Minds, lecture notes, Center for the Study of Language and Information.

[2]Parikh R. The completeness of propositional dynamic logic[C]//International Symposium on Mathematical Foundations of Computer Science. Springer, Berlin, Heidelberg, 1978: 403-415.

[3]Van Eijck J. Reducing dynamic epistemic logic to PDL by program transformation[M]. CWI. Software Engineering [SEN], 2004.

[4]Miller J S, Moss L S. The undecidability of iterated modal relativization[J]. Studia Logica, 2005, 79(3): 373-407.

[5]van Benthem J. In praise of strategies[M]//Games, Actions and Social Software. Springer, Berlin, Heidelberg, 2012: 96-116.

[6]Van Benthem, Johan. 1999. Logic in games, lecture notes. ILLC Amsterdam & Department of Philosophy, Stanford University

[7]Troquard, Nicolas and Philippe Balbiani, Propositional Dynamic Logic The Stanford Encyclopedia of Philosophy (Spring 2019 Edition), Edward N. Zalta (ed.), URL = <https://plato.stanford.edu/archives/spr2019 /entries/logic dynamic/>.