

Modal logics: overview

- Part I: Introduction to modal and multimodal logics
 1. Motivation and introduction
 2. The basic multimodal logic K
 3. The basic monomodal logics
 4. Completeness of $G(k, l, m, n)$ logics, and decidability of the basic modal logics
 5. Basic multimodal logics
 6. Other modal logics
- Part II: Applications
 7. Knowledge and announcements
 8. Belief
 9. Common knowledge and common belief
 10. **Action and propositional dynamic logic**
 11. Goals and intentions
 12. Ability, agency and branching time
- Part II: Proof methods
 13. Translation method
 14. Tableau method

Chapter 10.

Propositional dynamic logic

Nov.5, 2008

Propositional dynamic logic: overview

- programs and actions
- propositional dynamic logic (*PDL*)
 - ▶ semantics
 - ▶ axiomatics
 - ★ induction axiom
- reasoning about actions

Actions: introduction and language

- program = relation between states = possible worlds (cf. automata)
 - ▶ intermediate states are not considered (only result counts)
 - ▶ program \neq formula
- action \approx program
- set of atomic actions $Acts = \{a_1, a_2, \dots\}$
- BNF: parameter set and formulas defined by mutual recursion:

$$\alpha = a \mid skip \mid \alpha; \alpha \mid \alpha \cup \alpha \mid \alpha^* \mid \varphi?$$

$$\varphi = p \mid \neg\varphi \mid \varphi \wedge \varphi \mid [\alpha]\varphi$$

where a ranges over $Acts$, and p ranges over $Atms$

Actions: introduction and language, ctd.

- complex actions:

- ▶ $skip$ = “do nothing”
- ▶ $\alpha; \beta$ = “do first α , then β ”
- ▶ $\alpha \cup \beta$ = “nondeterministically choose between α and β , and perform chosen action”
- ▶ α^* = “do α an arbitrary number of times”
- ▶ $\varphi?$ = “test φ ” = “if φ is true then continue, else fail”

Actions: introduction and language, ctd.

- complex actions:

- ▶ $skip$ = “do nothing”
- ▶ $\alpha; \beta$ = “do first α , then β ”
- ▶ $\alpha \cup \beta$ = “nondeterministically choose between α and β , and perform chosen action”
- ▶ α^* = “do α an arbitrary number of times”
- ▶ $\varphi?$ = “test φ ” = “if φ is true then continue, else fail”
- ▶ exercises:
 - ★ express *fail* (action which always fails)
 - ★ express if-then-else
 - ★ express if-then
 - ★ express while-loop
 - ★ express repeat-loop
- ▶ missing: assignments $x := t$
 - ★ should replace the abstract atomic actions of *Acts*
 - ★ uninterpreted vs. interpreted version of *PDL*

- abbreviation:

- ▶ $\langle \alpha \rangle \varphi \stackrel{\text{def}}{=} \neg[\alpha]\neg\varphi$

- complex formulas:

- ▶ $[\alpha]\varphi =$ “ φ holds after every possible execution of α ”

- ▶ $\langle \alpha \rangle \varphi = \neg[\alpha]\neg\varphi$

- ▶ $\langle \alpha \rangle \varphi =$ “there is a possible execution of α after which φ is true”

Actions: introduction and language, ctd.

- abbreviation:
 - ▶ $\langle \alpha \rangle \varphi \stackrel{\text{def}}{=} \neg[\alpha]\neg\varphi$
- complex formulas:
 - ▶ $[\alpha]\varphi$ = “ φ holds after every possible execution of α ”
 - ▶ $\langle \alpha \rangle \varphi = \neg[\alpha]\neg\varphi$
 - ▶ $\langle \alpha \rangle \varphi$ = “there is a possible execution of α after which φ is true”
- nondeterministic actions/programs:
 - ▶ $[toss](Heads \vee Tails)$
 - ▶ $\langle toss \rangle Heads$
 - ▶ $\langle toss \rangle Tails$
 - ▶ $\neg[toss]Heads$

Actions: exercises

- express that α is executable
- express that α is inexecutable
- express that φ is the precondition of α
- express that φ is the postcondition of α
- express the effect of toggling a switch
- express: ‘if switch-on is executable then switch-off is not’
- express: ‘exactly one among switch-on and switch-off is executable’
- express: ‘toggling a switch twice does not change the state of the lamp’
- find the preconditions of the action of starting a car
- find the postconditions of the action of starting a car

Propositional dynamic logic (PDL): semantics

[Fisher&Ladner 76, Pratt 77, Parikh, Harel]; Polish predecessor:
algorithmic logic

- option 1: directly define R and truth conditions in mutual recursion
 - ▶ model $M = \langle W, R, V \rangle$ such that W nonempty, $V : Atms \rightarrow 2^W$, and:
 - ★ $R_{skip} = \{ \langle w, w \rangle : w \in W \}$
 - ★ $R_{\alpha;\beta} = R_\alpha \circ R_\beta$
 - ★ $R_{\alpha \cup \beta} = R_\alpha \cup R_\beta$
 - ★ $R_{\alpha^*} = (R_\alpha)^*$
 - ★ $R_{\varphi?} = \{ \langle w, w \rangle : M, w \Vdash \varphi \}$
 - ▶ truth condition:
 - ★ $M, w \Vdash [\alpha]\varphi$ iff $M, v \Vdash \varphi$ for every $v' \in R_\alpha(w)$
 - ★ $M, w \Vdash \langle \alpha \rangle \varphi$ iff $M, v \Vdash \varphi$ for some $v' \in R_\alpha(w)$

Propositional dynamic logic (PDL): semantics

[Fisher&Ladner 76, Pratt 77, Parikh, Harel]; Polish predecessor:
algorithmic logic

- option 1: directly define R and truth conditions in mutual recursion
 - ▶ model $M = \langle W, R, V \rangle$ such that W nonempty, $V : Atms \rightarrow 2^W$, and:
 - ★ $R_{skip} = \{ \langle w, w \rangle : w \in W \}$
 - ★ $R_{\alpha;\beta} = R_\alpha \circ R_\beta$
 - ★ $R_{\alpha \cup \beta} = R_\alpha \cup R_\beta$
 - ★ $R_{\alpha^*} = (R_\alpha)^*$
 - ★ $R_{\varphi?} = \{ \langle w, w \rangle : M, w \Vdash \varphi \}$
 - ▶ truth condition:
 - ★ $M, w \Vdash [\alpha]\varphi$ iff $M, v \Vdash \varphi$ for every $v' \in R_\alpha(w)$
 - ★ $M, w \Vdash \langle \alpha \rangle \varphi$ iff $M, v \Vdash \varphi$ for some $v' \in R_\alpha(w)$
- option 2: define frames where R interprets only atomic actions:
 - ▶ frame $F = \langle W, R \rangle$ such that $R_a \subseteq W$, for $a \in Acts$then define standard models interpreting complex formulas and complex actions in the 'right' way:
 - ▶ $R_{\alpha;\beta} = R_\alpha \circ R_\beta, \dots$
 - ▶ $V_{\varphi \wedge \psi} = V_\varphi \cap V_\psi, \dots$

Propositional dynamic logic (PDL): axiomatization

- axiomatics of $K([\alpha])$ for all α

- ▶ equivalences:

- ★ $[skip]\varphi \leftrightarrow \varphi$
- ★ $[\alpha; \beta]\varphi \leftrightarrow [\alpha][\beta]\varphi$
- ★ $[\alpha \cup \beta]\varphi \leftrightarrow [\alpha]\varphi \wedge [\beta]\varphi$
- ★ $[\psi?]\varphi \leftrightarrow (\psi \rightarrow \varphi)$

(can't be abbreviations!)

- ▶ fixpoint axiom:

- ★ $[\alpha^*]\varphi \leftrightarrow (\varphi \wedge [\alpha][\alpha^*]\varphi)$

- ▶ least fixpoint axiom (alias induction axiom):

- ★ $(\varphi \wedge [\alpha^*](\varphi \rightarrow [\alpha]\varphi)) \rightarrow [\alpha^*]\varphi$

Propositional dynamic logic (PDL): axiomatization

- axiomatics of $K([\alpha])$ for all α
 - ▶ equivalences: (can't be abbreviations!)
 - ★ $[skip]\varphi \leftrightarrow \varphi$
 - ★ $[\alpha; \beta]\varphi \leftrightarrow [\alpha][\beta]\varphi$
 - ★ $[\alpha \cup \beta]\varphi \leftrightarrow [\alpha]\varphi \wedge [\beta]\varphi$
 - ★ $[\psi?]\varphi \leftrightarrow (\psi \rightarrow \varphi)$
 - ▶ fixpoint axiom:
 - ★ $[\alpha^*]\varphi \leftrightarrow (\varphi \wedge [\alpha][\alpha^*]\varphi)$
 - ▶ least fixpoint axiom (alias induction axiom):
 - ★ $(\varphi \wedge [\alpha^*](\varphi \rightarrow [\alpha]\varphi)) \rightarrow [\alpha^*]\varphi$
- deduction, theorems, ...
 - ▶ N.B.: version of deduction theorem:
 - ★ $\varphi_1, \dots, \varphi_n \vdash \psi$ iff $[(\bigcup_{\{a : a \text{ occurs in } \psi\}} a)^*](\varphi_1 \wedge \dots \wedge \varphi_n) \rightarrow \psi$
 - ★ proof uses *generated submodel property*: modal formulas cannot distinguish between a pointed model M, w and the pointed submodel of M generated from w
- sound, complete and decidable
 - ▶ only weakly complete, but not strongly

Propositional dynamic logic (PDL): axiomatization

- axiomatics of $K([\alpha])$ for all α
 - ▶ equivalences: (can't be abbreviations!)
 - ★ $[skip]\varphi \leftrightarrow \varphi$
 - ★ $[\alpha; \beta]\varphi \leftrightarrow [\alpha][\beta]\varphi$
 - ★ $[\alpha \cup \beta]\varphi \leftrightarrow [\alpha]\varphi \wedge [\beta]\varphi$
 - ★ $[\psi?]\varphi \leftrightarrow (\psi \rightarrow \varphi)$
 - ▶ fixpoint axiom:
 - ★ $[\alpha^*]\varphi \leftrightarrow (\varphi \wedge [\alpha][\alpha^*]\varphi)$
 - ▶ least fixpoint axiom (alias induction axiom):
 - ★ $(\varphi \wedge [\alpha^*](\varphi \rightarrow [\alpha]\varphi)) \rightarrow [\alpha^*]\varphi$
- deduction, theorems, ...
 - ▶ N.B.: version of deduction theorem:
 - ★ $\varphi_1, \dots, \varphi_n \vdash \psi$ iff $[(\bigcup_{\{a : a \text{ occurs in } \psi\}} a)^*](\varphi_1 \wedge \dots \wedge \varphi_n) \rightarrow \psi$
 - ★ proof uses *generated submodel property*: modal formulas cannot distinguish between a pointed model M, w and the pointed submodel of M generated from w
- sound, complete and decidable
 - ▶ only weakly complete, but not strongly
- complexity of satisfiability: EXPTIME complete

Propositional dynamic logic (PDL): axiomatization

- axiomatics of $K([\alpha])$ for all α
 - ▶ equivalences: (can't be abbreviations!)
 - ★ $[skip]\varphi \leftrightarrow \varphi$
 - ★ $[\alpha; \beta]\varphi \leftrightarrow [\alpha][\beta]\varphi$
 - ★ $[\alpha \cup \beta]\varphi \leftrightarrow [\alpha]\varphi \wedge [\beta]\varphi$
 - ★ $[\psi?]\varphi \leftrightarrow (\psi \rightarrow \varphi)$
 - ▶ fixpoint axiom:
 - ★ $[\alpha^*]\varphi \leftrightarrow (\varphi \wedge [\alpha][\alpha^*]\varphi)$
 - ▶ least fixpoint axiom (alias induction axiom):
 - ★ $(\varphi \wedge [\alpha^*](\varphi \rightarrow [\alpha]\varphi)) \rightarrow [\alpha^*]\varphi$
- deduction, theorems, ...
 - ▶ N.B.: version of deduction theorem:
 - ★ $\varphi_1, \dots, \varphi_n \vdash \psi$ iff $[(\bigcup_{\{a : a \text{ occurs in } \psi\}} a)^*](\varphi_1 \wedge \dots \wedge \varphi_n) \rightarrow \psi$
 - ★ proof uses *generated submodel property*: modal formulas cannot distinguish between a pointed model M, w and the pointed submodel of M generated from w
- sound, complete and decidable
 - ▶ only weakly complete, but not strongly
- complexity of satisfiability: EXPTIME complete
- N.B.: also: $\alpha \cap \beta$ (intersection of programs \approx parallel execution)

Exercises

- $\vdash_{PDL} \langle p? \cup \neg p? \rangle \top$
- $\vdash_{PDL} p \rightarrow \langle a^* \rangle p$
- $\vdash_{PDL} \langle a \rangle p \rightarrow \langle a^* \rangle p$
- model blocksworld with 2 blocks in *PDL*:
 - ▶ $Atms = On_{1,2}, On_{2,1}, Holds_1, Holds_2$
 - ▶ $Acts = pickUp_1, pickUp_2, putOn_{1,2}, putOn_{2,1}$
 - ▶ preconditions:
 - ★ robot can only hold one block at a time
 - ★ to pick up a block it must be free
 - ▶ postconditions: ...
- frame problem, qualification problem, ramification problem ...