Introduction to Dynamic Logic

Peter H. Schmitt

Universität Karlsruhe, Germany

Presentation at the ANU Logic & Automated Reasoning Summer School,

December, 2002

Contents

- Reasoning about Programs: the DL Approach

Contents

- Reasoning about Programs: the DL Approach
- Syntax and Semantics of Dynamic Logic

Contents

- Reasoning about Programs: the DL Approach
- Syntax and Semantics of Dynamic Logic
- Axioms for Dynamic Logic

Contents

- Reasoning about Programs: the DL Approach
- Syntax and Semantics of Dynamic Logic
- Axioms for Dynamic Logic
- Propositional Dynamic Logic

Contents

- Reasoning about Programs: the DL Approach
- Syntax and Semantics of Dynamic Logic
- Axioms for Dynamic Logic
- Propositional Dynamic Logic
- Summary

Reasoning About Programs

An Example Program $\alpha_{\mathbf{RM}}$

$$
\begin{array}{l}
\text{int } \ a, b, z; \\
z = 0; \\
\text{while } (b! = 0) \\
\qquad \{ \ \text{ if } ((b/2) * 2 == b) \\
\qquad\qquad \{a = 2 * a; \\
\qquad\qquad b = b/2; \} \\
\qquad \text{else} \\
\qquad\qquad \{z = z + a; \\
\qquad\qquad a = 2 * a; \\
\qquad\qquad b = b/2; \} \\
\qquad \}
\end{array}
$$

Annotated Programs

The Program $\alpha_{\mathbf{RM}}$ with annotations

$$
\begin{array}{l}
\text{int } a, b, z; \\
z = 0; \\
\text{assert } x \doteq a \wedge y \doteq b \wedge z \doteq 0; \\
\text{while } (b! = 0) \\
\quad \{ \quad \text{if } ((b/2) * 2 == b) \\
\qquad\qquad \{a = 2 * a; \\
\qquad\qquad b = b/2; \} \\
\quad\ \text{else} \\
\qquad\qquad \{z = z + a; \\
\qquad\qquad a = 2 * a; \\
\qquad\qquad b = b/2; \} \\
\text{assert } a * b + z \doteq x * y \ ; \\
\quad \} \\
\text{assert b} \doteq 0; \\
\text{assert z} \doteq \text{x} * \text{y};
\end{array}
$$

external variable

The DL Approach

- Annotated programs use formulas within programs.

The DL Approach

- Annotated programs use formulas within programs.
- Dynamic Logic uses programs within formulas.

The DL Approach

- Annotated programs use formulas within programs.
- Dynamic Logic uses programs within formulas.
- Instead of placing annotation F after program segment α,

The DL Approach

- Annotated programs use formulas within programs.
- Dynamic Logic uses programs within formulas.
- Instead of placing annotation F after program segment α,
- we write $[\alpha]F$ in DL.

The DL Approach

- Annotated programs use formulas within programs.
- Dynamic Logic uses programs within formulas.
- Instead of placing annotation F after program segment α,
- we write $[\alpha]F$ in DL.
- Example

$$\forall a, b, z, x, y \quad (x \doteq a \land y \doteq b \to \ [\alpha_{RM}] \ \ z \doteq x * y \ )$$

The First Proof Steps

$$\forall a, b, z, x, y \quad (x \doteq a \land y \doteq b \rightarrow \ [\alpha_{RM}]\ \ z \doteq x * y\ )$$

The First Proof Steps

$$\forall a, b, z, x, y \quad (x \doteq a \land y \doteq b \longrightarrow \ [\alpha_{RM}] \ \ z \doteq x * y \ )$$

$$\forall a, b, z, x, y \quad (x \doteq a \land y \doteq b \longrightarrow \ [z = 0; \alpha_{RMwhile}] \ \ z \doteq x * y \ )$$

The First Proof Steps

$$\forall a, b, z, x, y \quad (x \doteq a \land y \doteq b \longrightarrow \ [\alpha_{RM}] \ \ z \doteq x * y \ )$$

$$\forall a, b, z, x, y \quad (x \doteq a \land y \doteq b \longrightarrow \ [z = 0; \alpha_{RMwhile}] \ \ z \doteq x * y \ )$$

$$\forall a, b, z, x, y \quad (x \doteq a \land y \doteq b \longrightarrow \ [z = 0][\alpha_{RMwhile}] \ \ z \doteq x * y \ )$$

The First Proof Steps

$$\forall a, b, z, x, y \quad (x \doteq a \land y \doteq b \to \ [\alpha_{RM}]\ \ z \doteq x * y\ )$$

$$\forall a, b, z, x, y \quad (x \doteq a \land y \doteq b \to \ [z = 0; \alpha_{RMwhile}]\ \ z \doteq x * y\ )$$

$$\forall a, b, z, x, y \quad (x \doteq a \land y \doteq b \to \ [z = 0][\alpha_{RMwhile}]\ \ z \doteq x * y\ )$$

$$\forall a, b, z, x, y \quad (x \doteq a \land y \doteq b \land z \doteq 0 \to [\alpha_{RMwhile}]\ \ z \doteq x * y\ )$$

The First Proof Steps

$$\forall a, b, z, x, y \quad (x \doteq a \land y \doteq b \to \; [\alpha_{RM}] \; z \doteq x * y \;)$$

$$\forall a, b, z, x, y \quad (x \doteq a \land y \doteq b \to \; [z = 0; \alpha_{RMwhile}] \; z \doteq x * y \;)$$

$$\forall a, b, z, x, y \quad (x \doteq a \land y \doteq b \to \; [z = 0][\alpha_{RMwhile}] \; z \doteq x * y \;)$$

$$\forall a, b, z, x, y \quad (x \doteq a \land y \doteq b \land z \doteq 0 \to [\alpha_{RMwhile}] \; z \doteq x * y \;)$$

$$\forall a \ldots (x \doteq a \land y \doteq b \land z \doteq 0 \to [\alpha_{RMwhile}] \, (b \doteq 0 \land a * b + z \doteq x * y) \;)$$

A Few More Proof Steps

$\forall a \ldots (x \doteq a \wedge y \doteq b \wedge z \doteq 0 \rightarrow [\alpha_{RMwhile}]\,(b \doteq 0 \wedge a * b + z \doteq x * y)\,)$

A Few More Proof Steps

$\forall a \ldots (x \doteq a \wedge y \doteq b \wedge z \doteq 0 \rightarrow [\alpha_{RMwhile}]\,(b \doteq 0 \wedge a * b + z \doteq x * y)\,)$

$\forall a \ldots (x \doteq a \wedge y \doteq b \wedge z \doteq 0 \rightarrow [while(b! = 0)\{body\}]\; b \doteq 0\;)$

and

$\forall a \ldots (x \doteq a \wedge y \doteq b \wedge z \doteq 0 \rightarrow [while(b! = 0)\{body\}]\; a * b + z \doteq x$

A Few More Proof Steps

$$\forall a \ldots (x \doteq a \wedge y \doteq b \wedge z \doteq 0 \to [\alpha_{RMwhile}]\,(b \doteq 0 \wedge a * b + z \doteq x * y)\,)$$

$$\forall a \ldots (x \doteq a \wedge y \doteq b \wedge z \doteq 0 \to [while(b! = 0)\{body\}]\; b \doteq 0\;) \quad \textit{okay}$$

and

$$\forall a \ldots (x \doteq a \wedge y \doteq b \wedge z \doteq 0 \to [while(b! = 0)\{body\}]\; a * b + z \doteq x$$

A Few More Proof Steps

$\forall a \ldots (x \doteq a \wedge y \doteq b \wedge z \doteq 0 \rightarrow [\alpha_{RMwhile}]\ (b \doteq 0 \wedge a * b + z \doteq x * y)\ )$

$\forall a \ldots (x \doteq a \wedge y \doteq b \wedge z \doteq 0 \rightarrow [while(b! = 0)\{body\}]\ b \doteq 0\ )$ *okay*

and

$\forall a \ldots (x \doteq a \wedge y \doteq b \wedge z \doteq 0 \rightarrow [while(b! = 0)\{body\}]\ a * b + z \doteq x$

$\forall a \ldots (x \doteq a \wedge y \doteq b \wedge z \doteq 0 \rightarrow\ a * b + z \doteq x * y\ )$

and

$\forall a \ldots (a * b + z \doteq x * y \rightarrow [body]\ a * b + z \doteq x * y\ )$

A Few More Proof Steps

$\forall a \ldots (x \doteq a \wedge y \doteq b \wedge z \doteq 0 \rightarrow [\alpha_{RMwhile}]\,(b \doteq 0 \wedge a * b + z \doteq x * y)\,)$

$\forall a \ldots (x \doteq a \wedge y \doteq b \wedge z \doteq 0 \rightarrow [while(b! = 0)\{body\}]\; b \doteq 0\;)$ *okay*
and
$\forall a \ldots (x \doteq a \wedge y \doteq b \wedge z \doteq 0 \rightarrow [while(b! = 0)\{body\}]\; a * b + z \doteq x$

$\forall a \ldots (x \doteq a \wedge y \doteq b \wedge z \doteq 0 \rightarrow \; a * b + z \doteq x * y\;)$ *okay*
and
$\forall a \ldots (a * b + z \doteq x * y \rightarrow [body]\; a * b + z \doteq x * y\;)$

A Few More Proof Steps

$\forall a \ldots (x \doteq a \land y \doteq b \land z \doteq 0 \rightarrow [\alpha_{RMwhile}]\,(b \doteq 0 \land a * b + z \doteq x * y)\,)$

$\forall a \ldots (x \doteq a \land y \doteq b \land z \doteq 0 \rightarrow [while(b! = 0)\{body\}]\ b \doteq 0\ )$ *okay*

and

$\forall a \ldots (x \doteq a \land y \doteq b \land z \doteq 0 \rightarrow [while(b! = 0)\{body\}]\ a * b + z \doteq x$

$\forall a \ldots (x \doteq a \land y \doteq b \land z \doteq 0 \rightarrow\ a * b + z \doteq x * y\ )$ *okay*

and

$\forall a \ldots (a * b + z \doteq x * y \rightarrow [body]\ a * b + z \doteq x * y\ )$

$\forall a \ldots (a * b + z \doteq x * y \land (b/2) * 2 \doteq b$

$\quad \rightarrow [a = 2 * a; b = b/2]\ a * b + z \doteq x * y\ )$

and

$\forall a \ldots (a * b + z \doteq x * y \land (b/2) * 2 \not\doteq b$

$\quad \rightarrow [z = z + a; a = 2 * a; b = b/2]\ a * b + z \doteq x * y\ )$

The Last Proof Steps

$$\forall a \ldots (a * b + z \doteq x * y \land (b/2) * 2 \doteq b \\ \quad \to [a = 2 * a; b = b/2]\; a * b + z \doteq x * y\;)$$

and

$$\forall a \ldots (a * b + z \doteq x * y \land (b/2) * 2 \not\doteq b \\ \quad \to [z = z + a; a = 2 * a; b = b/2]\; a * b + z \doteq x * y\;)$$

The Last Proof Steps

$$\forall a \ldots (a * b + z \doteq x * y \wedge (b/2) * 2 \doteq b \\ \rightarrow [a = 2 * a; b = b/2]\; a * b + z \doteq x * y\ )$$

and

$$\forall a \ldots (a * b + z \doteq x * y \wedge (b/2) * 2 \not\doteq b \\ \rightarrow [z = z + a; a = 2 * a; b = b/2]\; a * b + z \doteq x * y\ )$$

$$\forall a \ldots ((2 * a) * (b/2) + z \doteq x * y \wedge (b/2) * 2 \doteq b \\ \rightarrow\ a * b + z \doteq x * y\ )$$

and

$$\forall a \ldots ((2 * a) * (b/2) + z + a \doteq x * y \wedge (b/2) * 2 \not\doteq b \\ \rightarrow\ a * b + z \doteq x * y\ )$$

The Last Proof Steps

$$\forall a \ldots (a * b + z \doteq x * y \land (b/2) * 2 \doteq b$$
$$\quad \rightarrow [a = 2 * a; b = b/2]\ a * b + z \doteq x * y\ )$$

and

$$\forall a \ldots (a * b + z \doteq x * y \land (b/2) * 2 \not\doteq b$$
$$\quad \rightarrow [z = z + a; a = 2 * a; b = b/2]\ a * b + z \doteq x * y\ )$$

$$\forall a \ldots ((2 * a) * (b/2) + z \doteq x * y \land (b/2) * 2 \doteq b$$
$$\quad \rightarrow \ a * b + z \doteq x * y\ ) \qquad \mathit{okay}$$

and

$$\forall a \ldots ((2 * a) * (b/2) + z + a \doteq x * y \land (b/2) * 2 \not\doteq b$$
$$\quad \rightarrow \ a * b + z \doteq x * y\ )$$

The Last Proof Steps

$$\forall a \ldots (a * b + z \doteq x * y \land (b/2) * 2 \doteq b$$
$$\quad \rightarrow [a = 2 * a; b = b/2]\ a * b + z \doteq x * y\ )$$

and

$$\forall a \ldots (a * b + z \doteq x * y \land (b/2) * 2 \not\doteq b$$
$$\quad \rightarrow [z = z + a; a = 2 * a; b = b/2]\ a * b + z \doteq x * y\ )$$

$$\forall a \ldots ((2 * a) * (b/2) + z \doteq x * y \land (b/2) * 2 \doteq b$$
$$\quad \rightarrow\ a * b + z \doteq x * y\ ) \qquad \mathit{okay}$$

and

$$\forall a \ldots ((2 * a) * (b/2) + z + a \doteq x * y \land (b/2) * 2 \not\doteq b$$
$$\quad \rightarrow\ a * b + z \doteq x * y\ ) \qquad \mathit{okay} \qquad \text{here } b/2 \doteq (b - 1)$$

Formal Introduction

of

Dynamic Logic

Syntax: Terms

Let Σ be a vocabulary (set of function - and relation symbols).

The set Term_Σ is defined as usual:

Syntax: Terms

Let Σ be a vocabulary (set of function - and relation symbols).

The set Term_Σ is defined as usual:

Every variable x is in Term_Σ.

Syntax: Terms

Let Σ be a vocabulary (set of function - and relation symbols).

The set Term_Σ is defined as usual:

Every variable x is in Term_Σ.

If f is an n-place function symbol in Σ and t_i are terms in Term_Σ then

$f(t_1, \ldots, t_n)$ is in Term_Σ.

Syntax: Terms

Let Σ be a vocabulary (set of function - and relation symbols).

The set Term_Σ is defined as usual:

Every variable x is in Term_Σ.

If f is an n-place function symbol in Σ and t_i are terms in Term_Σ then

$f(t_1, \ldots, t_n)$ is in Term_Σ.

We ignore types for simplicity.

Syntax: Formulas

The sets Fml_Σ of formulas and Π_Σ of programs are defined by mutual recursion

- If $r \in \Sigma$ is an n-place relation symbol and $t_i \in \mathrm{Term}_\Sigma$ then $r(t_1, \ldots, t_n)$ is in Fml_Σ.

Syntax: Formulas

The sets Fml_Σ of formulas and Π_Σ of programs are defined by mutual recursion

- If $r \in \Sigma$ is an n-place relation symbol and $t_i \in \mathrm{Term}_\Sigma$ then $r(t_1, \ldots, t_n)$ is in Fml_Σ.
- If t_1, t_2 are terms then $t_1 \doteq t_2$ is in Fml_Σ .

Syntax: Formulas

The sets Fml_Σ of formulas and Π_Σ of programs are defined by mutual recursion

- If $r \in \Sigma$ is an n-place relation symbol and $t_i \in \mathrm{Term}_\Sigma$ then $r(t_1, \ldots, t_n)$ is in Fml_Σ.
- If t_1, t_2 are terms then $t_1 \doteq t_2$ is in Fml_Σ .
- If $F_1, F_2 \in \mathrm{Fml}_\Sigma$ then also $F_1 \vee F_2$, $F_1 \wedge F_2$, $F_1 \to F_2$, $\neg F_1$, $\forall x F_1$ and $\exists x F_1$.

Syntax: Formulas

The sets Fml_Σ of formulas and Π_Σ of programs are defined by mutual recursion

- If $r \in \Sigma$ is an n-place relation symbol and $t_i \in \mathrm{Term}_\Sigma$ then $r(t_1, \ldots, t_n)$ is in Fml_Σ.
- If t_1, t_2 are terms then $t_1 \doteq t_2$ is in Fml_Σ .
- If $F_1, F_2 \in \mathrm{Fml}_\Sigma$ then also
  $F_1 \vee F_2$, $F_1 \wedge F_2$, $F_1 \rightarrow F_2$, $\neg F_1$, $\forall x F_1$ and $\exists x F_1$.
- If F is a formula in Fml_Σ and $\pi \in \Pi_\Sigma$ then
  $$[\pi]F \text{ and } \langle\pi\rangle F$$
  are in Fml_Σ .

Syntax: Programs

- If $t \in \mathrm{Term}_\Sigma$ and x a variable then $x = t$ is in Π_Σ.

Syntax: Programs

- If $t \in \mathrm{Term}_\Sigma$ and x a variable then $x = t$ is in Π_Σ.
- If π_1,$\pi_2 \in \Pi_{RM}$ then also $\pi_1; \pi_2$ is in Π_{RM}.

Syntax: Programs

- If $t \in \mathrm{Term}_\Sigma$ and x a variable then $x = t$ is in Π_Σ.
- If π_1,$\pi_2 \in \Pi_{RM}$ then also $\pi_1; \pi_2$ is in Π_{RM}.
- If con is a quantifierfree formula in Fml_Σ and $\pi \in \Pi_{RM}$ then

$$\mathrm{while}\ (con)\ \{\pi\}$$

  is in Π_Σ.

Syntax: Programs

- If $t \in \mathrm{Term}_\Sigma$ and x a variable then $x = t$ is in Π_Σ.
- If π_1,$\pi_2 \in \Pi_{RM}$ then also $\pi_1; \pi_2$ is in Π_{RM}.
- If con is a quantifierfree formula in Fml_Σ and $\pi \in \Pi_{RM}$ then

$$\text{while } (con)\ \{\pi\}$$

  is in Π_Σ.
- If con is a quantifierfree formula in Fml_Σ and $\pi_1, \pi_2 \in \Pi_\Sigma$ then

$$\text{if } (con)\ \{\pi_1\}\ \text{else}\ \{\pi_2\}$$

  is in Π_Σ.

Kripke Structures

A DL-Kripke structure

$$\mathcal{K} = (S, \rho)$$

consists of

- a set S of states (or worlds) and
- a function ρ that maps every program π to a binary relation $\rho(\pi)$ on S.

The $\rho(\pi)$ are called called the accessibility relations.

The State Space

For a Dynamic Logic with vocabulary Σ a state $s \in S$ of any Kripke structure $\mathcal{K}$ is a pair

$$s = (\mathcal{A}, \beta)$$

where

- $\mathcal{A}$ is a usual first-order structure for Σ that is fixed for all of S.
- A variable assignment $\beta : Var \to A$ that determines the values of the program variables in state s

$A =$ universe of $\mathcal{A}$.

Semantics of DL Formulas (I)

For any state $(\mathcal{A}, \beta)$ of a Kripke structure $\mathcal{K}$ define:

- $t^{(\mathcal{A},\beta)}$ for a term t is as usual.

Semantics of DL Formulas (I)

For any state $(\mathcal{A}, \beta)$ of a Kripke structure $\mathcal{K}$ define:

- $t^{(\mathcal{A},\beta)}$ for a term t is as usual.
- $(\mathcal{A}, \beta) \models r(t_1, \ldots, t_k)$ iff $(t_1^{(\mathcal{A},\beta)}, \ldots, t_k^{(\mathcal{A},\beta)}) \in r^{\mathcal{A}}$.

Semantics of DL Formulas (I)

For any state $(\mathcal{A}, \beta)$ of a Kripke structure $\mathcal{K}$ define:

- $t^{(\mathcal{A},\beta)}$ for a term t is as usual.
- $(\mathcal{A}, \beta) \models r(t_1, \ldots, t_k)$ iff $(t_1^{(\mathcal{A},\beta)}, \ldots, t_k^{(\mathcal{A},\beta)}) \in r^{\mathcal{A}}$.
- $(\mathcal{A}, \beta) \models t_1 = t_2$ iff $t_1^{(\mathcal{A},\beta)} = t_2^{(\mathcal{A},\beta)}$

Semantics of DL Formulas (I)

For any state $(\mathcal{A}, \beta)$ of a Kripke structure $\mathcal{K}$ define:

- $t^{(\mathcal{A},\beta)}$ for a term t is as usual.
- $(\mathcal{A}, \beta) \models r(t_1, \ldots, t_k)$ iff $(t_1^{(\mathcal{A},\beta)}, \ldots, t_k^{(\mathcal{A},\beta)}) \in r^{\mathcal{A}}$.
- $(\mathcal{A}, \beta) \models t_1 = t_2$ iff $t_1^{(\mathcal{A},\beta)} = t_2^{(\mathcal{A},\beta)}$
- $(\mathcal{A}, \beta) \models F$ is defined as usual for connectives for first order logic.

Semantics of DL Formulas (I)

For any state $(\mathcal{A}, \beta)$ of a Kripke structure $\mathcal{K}$ define:

- $t^{(\mathcal{A},\beta)}$ for a term t is as usual.
- $(\mathcal{A}, \beta) \models r(t_1, \ldots, t_k)$ iff $(t_1^{(\mathcal{A},\beta)}, \ldots, t_k^{(\mathcal{A},\beta)}) \in r^{\mathcal{A}}$.
- $(\mathcal{A}, \beta) \models t_1 = t_2$ iff $t_1^{(\mathcal{A},\beta)} = t_2^{(\mathcal{A},\beta)}$
- $(\mathcal{A}, \beta) \models F$ is defined as usual for connectives for first order logic.
- $(\mathcal{A}, \beta) \models \langle p \rangle F$ iff $(\mathcal{A}, \gamma) \models F$ for at least one pair $((\mathcal{A}, \beta), (\mathcal{A}, \gamma))$ of states in $\rho(p)$

Semantics of DL Formulas (I)

For any state $(\mathcal{A}, \beta)$ of a Kripke structure $\mathcal{K}$ define:

- $t^{(\mathcal{A},\beta)}$ for a term t is as usual.
- $(\mathcal{A}, \beta) \models r(t_1, \ldots, t_k)$ iff $(t_1^{(\mathcal{A},\beta)}, \ldots, t_k^{(\mathcal{A},\beta)}) \in r^{\mathcal{A}}$.
- $(\mathcal{A}, \beta) \models t_1 = t_2$ iff $t_1^{(\mathcal{A},\beta)} = t_2^{(\mathcal{A},\beta)}$
- $(\mathcal{A}, \beta) \models F$ is defined as usual for connectives for first order logic.
- $(\mathcal{A}, \beta) \models \langle p \rangle F$ iff $(\mathcal{A}, \gamma) \models F$ for at least one pair $((\mathcal{A}, \beta), (\mathcal{A}, \gamma))$ of states in $\rho(p)$
- $(\mathcal{A}, \beta) \models [p] F$ iff $(\mathcal{A}, \gamma) \models F$ for all pairs $((\mathcal{A}, \beta), (\mathcal{A}, \gamma))$ of states in $\rho(p)$.

Semantics of DL Formulas (II)

- $\rho(x := s) = \{((\mathcal{A}, \beta), (\mathcal{A}, \beta[x/s^{(\mathcal{A},\beta)}])) \mid (\mathcal{A}, \beta) \in S\}$.

Semantics of DL Formulas (II)

- $\rho(x := s) = \{((\mathcal{A}, \beta), (\mathcal{A}, \beta[x/s^{(\mathcal{A},\beta)}])) \mid (\mathcal{A}, \beta) \in S\}$.
- $\rho(\pi_1; \pi_2)$ consists of all pairs (β, γ) such that $(\beta, \delta) \in \rho(\pi_1)$ and $(\delta, \gamma) \in \rho(\pi_2)$ for some δ.

Semantics of DL Formulas (II)

- $\rho(x := s) = \{((\mathcal{A}, \beta), (\mathcal{A}, \beta[x/s^{(\mathcal{A},\beta)}])) \mid (\mathcal{A}, \beta) \in S\}$.
- $\rho(\pi_1; \pi_2)$ consists of all pairs (β, γ)
  such that $(\beta, \delta) \in \rho(\pi_1)$ and $(\delta, \gamma) \in \rho(\pi_2)$ for some δ.
- $((\beta, \gamma) \in \rho(\mathrm{if}(F_0)\{\pi_1\}\,\mathrm{else}\{\pi_2\})$
  iff
  $\beta \models F_0$ and $(\beta, \gamma) \in \rho(\pi_1)$
  or
  $\beta \models \neg F_0$ and $(\beta, \gamma) \in \rho(\pi_2)$

Semantics of DL Formulas (II)

- $\rho(x := s) = \{((\mathcal{A}, \beta), (\mathcal{A}, \beta[x/s^{(\mathcal{A},\beta)}])) \mid (\mathcal{A}, \beta) \in S\}$.
- $\rho(\pi_1; \pi_2)$ consists of all pairs (β, γ) such that $(\beta, \delta) \in \rho(\pi_1)$ and $(\delta, \gamma) \in \rho(\pi_2)$ for some δ.
- $((\beta, \gamma) \in \rho(\mathrm{if}(F_0)\{\pi_1\}\,\mathrm{else}\{\pi_2\})$
  iff
  $\beta \models F_0$ and $(\beta, \gamma) \in \rho(\pi_1)$
  or
  $\beta \models \neg F_0$ and $(\beta, \gamma) \in \rho(\pi_2)$

We have simplified notation: β instead of $(\mathcal{A}, \beta)$.

Semantics of DL Formulas (III)

$$(\beta, \gamma) \in \rho(\mathrm{while}(F_0)\{\pi\})$$

iff

there is an $n \in \mathbb{N}$ and there are states β_i for $0 \leq i \leq n$ such that

1. $\beta_0 = \beta$,
2. $\beta_n = \gamma$,
3. $\beta_i \models F_0$ for $0 \leq i < n$
4. $\beta_n \models \neg F_0$
5. $(\beta_i, \beta_{i+1}) \in \rho(\pi)$ for $0 \leq i < n$

Some DL-Tautologies

General assumption: x does not occuir in π.

- $(\exists x \; \langle\pi\rangle F) \leftrightarrow (\langle\pi\rangle\exists x \; F)$

Some DL-Tautologies

General assumption: x does not occuir in π.

- $(\exists x \ \langle\pi\rangle F) \leftrightarrow (\langle\pi\rangle \exists x \ F)$
- $(\forall x \ [\pi] F) \leftrightarrow ([\pi] \forall x \ F)$

Some DL-Tautologies

General assumption: x does not occuir in π.

- $(\exists x \; \langle\pi\rangle F) \leftrightarrow (\langle\pi\rangle \exists x \; F)$
- $(\forall x \; [\pi] F) \leftrightarrow ([\pi] \forall x \; F)$
- $(\exists x \; [\pi] F) \rightarrow ([\pi] \exists x \; F)$

Some DL-Tautologies

General assumption: x does not occuir in π.

- $(\exists x \ \langle \pi \rangle F) \leftrightarrow (\langle \pi \rangle \exists x \ F)$
- $(\forall x \ [\pi] F) \leftrightarrow ([\pi] \forall x \ F)$
- $(\exists x \ [\pi] F) \rightarrow ([\pi] \exists x \ F)$
- $([\pi] \exists x \ F) \rightarrow (\exists x \ [\pi] F)$ if π is deterministic

Some DL-Tautologies

General assumption: x does not occuir in π.

- $(\exists x \ \langle \pi \rangle F) \leftrightarrow (\langle \pi \rangle \exists x \ F)$
- $(\forall x \ [\pi] F) \leftrightarrow ([\pi] \forall x \ F)$
- $(\exists x \ [\pi] F) \rightarrow ([\pi] \exists x \ F)$
- $([\pi] \exists x \ F) \rightarrow (\exists x \ [\pi] F)$ if π is deterministic
- $(\langle \pi \rangle \forall x \ F) \rightarrow (\forall x \ \langle \pi \rangle F)$

Some DL-Tautologies

General assumption: x does not occuir in π.

- $(\exists x \ \langle\pi\rangle F) \leftrightarrow (\langle\pi\rangle\exists x \ F)$
- $(\forall x \ [\pi]F) \leftrightarrow ([\pi]\forall x \ F)$
- $(\exists x \ [\pi]F) \rightarrow ([\pi]\exists x \ F)$
- $([\pi]\exists x \ F) \rightarrow (\exists x \ [\pi]F)$ if π is deterministic
- $(\langle\pi\rangle\forall x \ F) \rightarrow (\forall x \ \langle\pi\rangle F)$
- $(\forall x \ \langle\pi\rangle F) \rightarrow (\langle\pi\rangle\forall x \ F)$ if π is deterministic

Some DL-Tautologies

General assumption: x does not occuir in π.

- $(\exists x \ \langle\pi\rangle F) \leftrightarrow (\langle\pi\rangle\exists x \ F)$
- $(\forall x \ [\pi]F) \leftrightarrow ([\pi]\forall x \ F)$
- $(\exists x \ [\pi]F) \rightarrow ([\pi]\exists x \ F)$
- $([\pi]\exists x \ F) \rightarrow (\exists x \ [\pi]F)$ if π is deterministic
- $(\langle\pi\rangle\forall x \ F) \rightarrow (\forall x \ \langle\pi\rangle F)$
- $(\forall x \ \langle\pi\rangle F) \rightarrow (\langle\pi\rangle\forall x \ F)$ if π is deterministic
- $(\langle\pi\rangle(F \wedge G)) \rightarrow ((\langle\pi\rangle F) \wedge \langle\pi\rangle G)$

Some DL-Tautologies

General assumption: x does not occuir in π.

- $(\exists x \; \langle \pi \rangle F) \leftrightarrow (\langle \pi \rangle \exists x \; F)$
- $(\forall x \; [\pi] F) \leftrightarrow ([\pi] \forall x \; F)$
- $(\exists x \; [\pi] F) \rightarrow ([\pi] \exists x \; F)$
- $([\pi] \exists x \; F) \rightarrow (\exists x \; [\pi] F)$ if π is deterministic
- $(\langle \pi \rangle \forall x \; F) \rightarrow (\forall x \; \langle \pi \rangle F)$
- $(\forall x \; \langle \pi \rangle F) \rightarrow (\langle \pi \rangle \forall x \; F)$ if π is deterministic
- $(\langle \pi \rangle (F \wedge G)) \rightarrow ((\langle \pi \rangle F) \wedge \langle \pi \rangle G)$
- $(\langle \pi \rangle (F \wedge G)) \leftrightarrow ((\langle \pi \rangle F) \wedge \langle \pi \rangle G)$
  if no free variable of F occurs in π

Further Examples

- $\langle x = 1 \rangle \; x \doteq 1$ always true

Further Examples

- $\langle x = 1 \rangle \; x \doteq 1$ always true

- $[\mathrm{while}(true)\{\pi\}] \; false$ always true

Further Examples

- $\langle x = 1 \rangle \; x \doteq 1$ always true
- $[\mathrm{while}(true)\{\pi\}] \; false$ always true
- $\langle \pi \rangle true$ true if π terminates

Further Examples

- $\langle x = 1 \rangle\ x \doteq 1$ always true
- $[\mathrm{while}(true)\{\pi\}]\ false$ always true
- $\langle \pi \rangle true$ true if π terminates
- $\langle \pi_1 \rangle true \rightarrow \langle \pi_2 \rangle true$ says: if π_1 terminates then π_2 terminates.

Axioms for

of

Dynamic Logic

Sequents

1. A *sequent* is of the form

$$\Gamma \twoheadrightarrow \Delta$$

   where Γ and Δ are sequences of formulas.
2. Let $\mathcal{K} = (S, \rho)$ be a DL-Kripke structure, $(\mathcal{A}, \beta)$ a state in S.

$$(\mathcal{A}, \beta) \models \Gamma \twoheadrightarrow \Delta \text{ iff } (\mathcal{A}, \beta) \models \bigwedge \Gamma \to \bigvee \Delta$$

3. $$\mathcal{K} \models \Gamma \twoheadrightarrow \Delta \quad \text{iff} \quad (\mathcal{A}, \beta) \models \bigwedge \Gamma \to \bigvee \Delta$$
$$\text{for all } (\mathcal{A}, \beta) \in S.$$

4. A sequent $\Gamma \twoheadrightarrow \Delta$ is called *universally valid* if $\mathcal{K} \models \Gamma \twoheadrightarrow \Delta$ holds for all Kripke structures $\mathcal{K}$ in the signature of the sequent.

Sequent Rules

A sequent rule is of the form

$$\frac{\Gamma_1 \rightarrow \Delta_1}{\Gamma_2 \rightarrow \Delta_2} \quad \text{or} \quad \frac{\Gamma_1 \rightarrow \Delta_1 \quad \Gamma_1' \rightarrow \Delta_1'}{\Gamma_2 \rightarrow \Delta_2}$$

A sequent rule

$$\frac{\Gamma_1 \rightarrow \Delta_1 \quad \Gamma_1' \rightarrow \Delta_1'}{\Gamma_2 \rightarrow \Delta_2}$$

is *sound* if $\Gamma_2 \rightarrow \Delta_2$ is universally valid whenever $\Gamma_1 \rightarrow \Delta_1$ and $\Gamma_1' \rightarrow \Delta_1'$ are universally valid.

Some Sound Sequent Rules

1. $$\frac{}{\Gamma^1, A, \Gamma^1 \rightarrow \Delta^1, A, \Delta^2}$$

2. $$\frac{\Gamma, \Gamma' \rightarrow \Delta, A, \Delta'}{\Gamma, \neg A, \Gamma' \rightarrow \Delta, \Delta'}$$

3. $$\frac{\Gamma \rightarrow \Delta, A(y/x), \Delta'}{\Gamma \rightarrow \Delta, \forall x A, \Delta'}$$ where y is not free in Γ, Δ, Δ'

4. $$\frac{\Gamma, A, \Gamma' \rightarrow \Delta \qquad \Gamma, \Gamma' \rightarrow \Delta, A, \Delta'}{\Gamma, \Gamma' \rightarrow \Delta, \Delta'}$$

The Axiom System S_0 (1)

axiom

$$\frac{}{\Gamma, F \Rightarrow F, \Delta}$$

not-left

$$\frac{\Gamma, \Rightarrow F, \Delta}{\Gamma, \neg F \Rightarrow \Delta}$$

not-right

$$\frac{\Gamma, F \Rightarrow \Delta}{\Gamma \Rightarrow \neg F, \Delta}$$

impl-left

$$\frac{\Gamma \Rightarrow F, \Delta \qquad \Gamma, G \Rightarrow \Delta}{\Gamma, F \to G \Rightarrow \Delta}$$

impl-right

$$\frac{\Gamma, F \Rightarrow G, \Delta}{\Gamma \Rightarrow F \to G, \Delta}$$

and-left

$$\frac{\Gamma, F, G \Rightarrow \Delta}{\Gamma, F \wedge G \Rightarrow \Delta}$$

and-right

$$\frac{\Gamma \Rightarrow F, \Delta \qquad \Gamma \Rightarrow G, \Delta}{\Gamma \Rightarrow F \wedge G, \Delta}$$

The Axiom System S_0 (2)

or-left

$$\frac{\Gamma, F \rightarrow \Delta \quad \Gamma, G \rightarrow \Delta}{\Gamma, F \vee G \rightarrow \Delta}$$

or-right

$$\frac{\Gamma \rightarrow F, G, \Delta}{\Gamma \rightarrow F \vee G, \Delta}$$

all-left

$$\frac{\Gamma, \forall x F, F(t/x) \rightarrow \Delta}{\Gamma, \forall x F \rightarrow \Delta}$$

where t is a ground term.

all-right

$$\frac{\Gamma \rightarrow F(c/x), \Delta}{\Gamma \rightarrow \forall x F, \Delta}$$

where c is a new constant symbol.

The Axiom System S_0 (3)

ex-right

$$\frac{\Gamma \rightarrow \exists x F, F(t/x), \Delta}{\Gamma, \rightarrow \exists x F, \Delta}$$

where t is ground term.

ex-left

$$\frac{\Gamma \rightarrow F(c/x), \Delta}{\Gamma, \exists x F \rightarrow \Delta}$$

where c is a new constant symbol.

The Axiom System S_0^{fv}

all-left

$$\frac{\Gamma, \forall x F, F(X/x) \rightarrow \Delta}{\Gamma, \forall x F \rightarrow \Delta}$$

where X is a new variable.

all-right

$$\frac{\Gamma \rightarrow F(f(x_1, \ldots, x_n)/x), \Delta}{\Gamma \rightarrow \forall x F, \Delta}$$

where f is a new functions symbol and $x_1, \ldots, x_n$ are all free variables in $\forall x F$.

ex-right

$$\frac{\Gamma \rightarrow \exists x F, F(X/x), \Delta}{\Gamma, \rightarrow \exists x F, \Delta}$$

where X is a new variable.

ex-left

$$\frac{\Gamma \rightarrow F(f(x_1, \ldots, x_n)/x), \Delta}{\Gamma, \exists x F \rightarrow \Delta}$$

where f is a new functions symbol and $x_1, \ldots, x_n$ are all free variables in $\forall x F$.

Proof in S_0

$$
\begin{array}{lrcl}
 & & \text{axiom} & \\
6 & p(d), p(c) & \Longrightarrow & p(c), \forall y p(y) \\
 & & \big| \ \text{impl-right} & \\
5 & p(d) & \Longrightarrow & p(c), p(c) \to \forall y p(y) \\
 & & \big| \ \text{ex-right} & \\
4 & p(d) & \Longrightarrow & p(c), \exists x (p(x) \to \forall y p(y)) \\
 & & \big| \ \text{all-right} & \\
3 & p(d) & \Longrightarrow & \forall y p(y), \exists x (p(x) \to \forall y p(y)) \\
 & & \big| \ \text{impl-right} & \\
2 & & \Longrightarrow & p(d) \to \forall y p(y), \exists x (p(x) \to \forall y p(y)) \\
 & & \big| \ \text{ex-right} & \\
1 & & \Longrightarrow & \exists x (p(x) \to \forall y p(y))
\end{array}
$$

Proof in S_0^{fv}

closed by $\tau(X) = c$

4 $\quad p(X) \Rightarrow p(c), \exists x(p(x) \to \forall y p(y))$

all-right

3 $\quad p(X) \Rightarrow \forall y p(y), \exists x(p(x) \to \forall y p(y))$

impl-right

2 $\quad \Rightarrow p(X) \to \forall y p(y), \exists x(p(x) \to \forall y p(y))$

ex-right

1 $\quad \Rightarrow \exists x(p(x) \to \forall y p(y))$

The Assignment Rule

$$\frac{\Gamma(c/a), a \doteq t(c/a) \twoheadrightarrow F, \Delta(c/a)}{\Gamma \twoheadrightarrow \langle a = t \rangle F, \Delta}$$

where a is a variable and t a term, c a new variable.

The Assignment Rule

$$\frac{\Gamma(c/a), a \doteq t(c/a) \Rightarrow F, \Delta(c/a)}{\Gamma \Rightarrow \langle a = t \rangle F, \Delta}$$

where a is a variable and t a term, c a new variable.

Example

The Assignment Rule

$$\frac{\Gamma(c/a), a \doteq t(c/a) \rightarrow F, \Delta(c/a)}{\Gamma \rightarrow \langle a = t \rangle F, \Delta}$$

where a is a variable and t a term, c a new variable.

Example

$$\frac{}{\begin{array}{c} a * b + z \doteq x * y \wedge (b/2) * 2 \doteq b \\ \rightarrow \langle a = 2 * a; b = b/2 \rangle \; a * b + z \doteq x * y \end{array}}$$

The Assignment Rule

$$\frac{\Gamma(c/a), a \doteq t(c/a) \Rightarrow F, \Delta(c/a)}{\Gamma \Rightarrow \langle a = t \rangle F, \Delta}$$

where a is a variable and t a term, c a new variable.

Example

$$\frac{\dfrac{}{c * b + z \doteq x * y \land (b/2) * 2 \doteq b \land a \doteq 2 * c \Rightarrow \langle b = b/2 \rangle\; a * b + z \doteq x * y}}{a * b + z \doteq x * y \land (b/2) * 2 \doteq b \Rightarrow \langle a = 2 * a; b = b/2 \rangle\; a * b + z \doteq x * y}$$

The Assignment Rule

$$\frac{\Gamma(c/a), a \doteq t(c/a) \rightarrow F, \Delta(c/a)}{\Gamma \rightarrow \langle a = t \rangle F, \Delta}$$

where a is a variable and t a term, c a new variable.

Example

$$\frac{\dfrac{\begin{array}{c} c * d + z \doteq x * y \wedge (d/2) * 2 \doteq d \wedge a \doteq 2 * c \wedge b \doteq d/2 \\ \rightarrow \; a * b + z \doteq x * y \end{array}}{\begin{array}{c} c * b + z \doteq x * y \wedge (b/2) * 2 \doteq b \wedge a \doteq 2 * c \\ \rightarrow \langle b = b/2 \rangle \; a * b + z \doteq x * y \end{array}}}{\begin{array}{c} a * b + z \doteq x * y \wedge (b/2) * 2 \doteq b \\ \rightarrow \langle a = 2 * a; b = b/2 \rangle \; a * b + z \doteq x * y \end{array}}$$

A Branching Rule

$$\frac{\Gamma, F_0 \Rightarrow \langle \pi_1 \rangle F, \Delta \qquad \Gamma, \neg F_0 \Rightarrow \langle \pi_2 \rangle F, \Delta}{\Gamma \Rightarrow \langle \mathrm{if}(F_0)\{\pi_1\}\,\mathrm{else}\{\pi_2\}\rangle F, \Delta}$$

A While Rule

$$\frac{\Gamma \Rightarrow I \quad I, F_0 \Rightarrow \langle \pi \rangle I \quad \Gamma, I, \neg F_0 \Rightarrow F, \Delta}{\Gamma \Rightarrow \langle \mathrm{while}(F_0)\{\pi\}\rangle F, \Delta}$$

Propositional

Dynamic Logic

PDL

PDL Formulas

The sets Fml_{PDL} of formulas and Π_{PDL} of programs are defined by:

- If p is a propositional variable then

$$p$$

is in Fml_{PDL}

PDL Formulas

The sets Fml_{PDL} of formulas and Π_{PDL} of programs are defined by:

- If p is a propositional variable then

  $$p$$

  is in Fml_{PDL}
- If $F_1, F_2 \in \mathrm{Fml}_{PDL}$ then also
  $F_1 \vee F_2$, $F_1 \wedge F_2$, $F_1 \rightarrow F_2$, $\neg F_1$,

PDL Formulas

The sets Fml_{PDL} of formulas and Π_{PDL} of programs are defined by:

- If p is a propositional variable then
$$p$$
is in Fml_{PDL}
- If $F_1, F_2 \in \mathrm{Fml}_{PDL}$ then also
$F_1 \vee F_2$, $F_1 \wedge F_2$, $F_1 \rightarrow F_2$, $\neg F_1$,
- If F is a formula in Fml_{PDL} and $\pi \in \Pi_{PDL}$ then
$$[\pi]F \text{ and } \langle \pi \rangle F$$
are in Fml_{PDL} .

PDL Programs

- If π_0 is an atomic program then

  $$\pi_0$$

  is in Π_{PDL}.

PDL Programs

- If π_0 is an atomic program then

$$\pi_0$$

  is in Π_{PDL}.

- If π_1,$\pi_2 \in \Pi_{PDL}$ then also

$$\pi_1; \pi_2$$

PDL Programs

- If π_0 is an atomic program then

$$\pi_0$$

is in Π_{PDL}.

- If π_1,$\pi_2 \in \Pi_{PDL}$ then also

$$\pi_1;\pi_2$$

- If $\pi \in \Pi_{PDL}$ then

$$\pi^*$$

PDL Programs

- If π_0 is an atomic program then

$$\pi_0$$

  is in Π_{PDL}.

- If π_1,$\pi_2 \in \Pi_{PDL}$ then also

$$\pi_1 ; \pi_2$$

- If $\pi \in \Pi_{PDL}$ then

$$\pi^*$$

- If π_1,$\pi_2 \in \Pi_{PDL}$ then also

$$\pi_1 \cup \pi_2$$

PDL Programs

- If π_0 is an atomic program then

$$\pi_0$$

  is in Π_{PDL}.

- If π_1,$\pi_2 \in \Pi_{PDL}$ then also

$$\pi_1 ; \pi_2$$

- If $\pi \in \Pi_{PDL}$ then

$$\pi^*$$

- If π_1,$\pi_2 \in \Pi_{PDL}$ then also

$$\pi_1 \cup \pi_2$$

- If $F \in \mathrm{Fml}_{PDL}$ then

$$(F?)$$

Literature

- David Harel *First-Order Dynamic Logic*
  Lecture Notes in Computer Science, Vol. 68, 1979

Literature

- David Harel *First-Order Dynamic Logic*
  Lecture Notes in Computer Science, Vol. 68, 1979
- David Harel *Chapter on Dynamic Logic*
  in: Handbook of Philosophical Logic, Vol II, *pages 497 - 604*
  D.Reidel, 1984

Literature

- David Harel *First-Order Dynamic Logic*
  Lecture Notes in Computer Science, Vol. 68, 1979
- David Harel *Chapter on Dynamic Logic*
  in: Handbook of Philosophical Logic, Vol II, *pages 497 - 604*
  D.Reidel, 1984
- Dexter Kozen and Jerzy Tiuryn *Chapter on Logics of Programs*
  in: Handbook of Theoretical Computer Science, *pages 791–840*
  Elsevier, Amsterdam, 1990

Literature

- David Harel *First-Order Dynamic Logic*
  Lecture Notes in Computer Science, Vol. 68, 1979
- David Harel *Chapter on Dynamic Logic*
  in: Handbook of Philosophical Logic, Vol II, *pages 497 - 604*
  D.Reidel, 1984
- Dexter Kozen and Jerzy Tiuryn *Chapter on Logics of Programs*
  in: Handbook of Theoretical Computer Science, *pages 791–840*
  Elsevier, Amsterdam, 1990
- David Harel, Dexter Kozen, and Jerzy Tiuryn
  Dynamic Logic The MIT Press, 2000