Formale Systeme II: Theorie

Dynamic Logic:
Uninterpreted and Interpreted First Order DL

SS 2016

Prof. Dr. Bernhard Beckert · Dr. Mattias Ulbrich
Slides partially by Prof. Dr. Peter H. Schmitt
Overview – a family of logics

- Modal Logics
- Propositional Dynamic Logic
- Dynamic Logic

- Hybrid DL
- Java DL
Motivation

First Order Dynamic Logic

Atomic programs are refined to assignments.

Example Formula

\[ x_0 = x \land y_0 = y \rightarrow [x := x + y; y := x - y; x := x - y] \varphi \]
**First Order Dynamic Logic**

Inherit from FOL:
- Terms over function symbols and variables
- Predicate symbols
- Quantification over variables

Inherit from PDL
- Modalities
- Composite program constructors

Refine PDL
Unspecified atomic programs replaced by assignments $\text{var} := \text{term}$
Syntax

Syntactical material

$\Sigma = (F, P, \alpha)$ ... signature
- $F$ ... function symbols
- $P$ ... predicate symbols
- $\alpha : F \cup P \rightarrow \mathbb{N}$ ... arity function

Var ... set of variables

- No atomic programs like in PDL
- Same as for FOL
Syntax

As abstract grammar:

\[
\begin{align*}
term & ::= \quad \text{var} \mid f(\text{term}_1, \ldots, \text{term}_{\alpha(f)}) \\
fml & ::= \quad \text{true} \mid \text{false} \mid p(\text{term}_1, \ldots, \text{term}_{\alpha(p)}) \mid \text{term}_1 = \text{term}_2 \\
      & \quad \mid \neg fml \mid fml_1 \land fml_2 \mid fml_1 \lor fml_2 \mid fml_1 \rightarrow fml_2 \\
      & \quad \mid \exists \text{var. } fml \mid \forall \text{var. } fml \\
      & \quad \mid [\text{prog}]fml \mid \langle \text{prog} \rangle fml \\
prog & ::= \quad \text{var} ::= \text{term} \\
      & \quad \mid \text{var} ::= * \\
      & \quad \mid \text{prog}_1 ; \text{prog}_2 \mid \text{prog}_1 \cup \text{prog}_2 \mid \text{prog}^* \mid \\
\end{align*}
\]

for \( \text{var} \in \text{Var}, \text{f} \in \text{F}, \text{p} \in \text{P} \)
Semantics – Kripke Structures

First Order Structure \((D, I)\)

- \(D\) ... set of objects (domain)
- \(I\) ... Interpretation
- \(I(f) : D^\alpha(f) \rightarrow D\) for function symbol \(f \in F\)
- \(I(P) \subseteq D^\alpha(p)\) for predicate symbol \(p \in P\)

Kripke Structure \((S, \rho)\)

- \(S\) ... set of states
- \(\rho : \text{prog} \rightarrow S \times S\) ... accessibility relation

FODL: Fixed Kripke Frame \(\mathcal{K}_D = (S_D, \rho_D)\)

which depends on the domain \(D\)
The set of states $\mathcal{K}_D$ is the set of assignments of elements in the universe $D$ to variables in $\text{Var}$:

$$ S = \text{Var} \rightarrow D $$

For every $t \in \text{Term}_\Sigma$ we denote by

$$ \text{val}_{D,I,s}(t) $$

the usual first-order evaluation of $t$ in $(D, I)$; variables are interpreted via $s$. 
Function Update Notation

Notation: for $s \in S_D$, $x \in Var$, $a \in D$

$$s[x/a](y) = \begin{cases} 
  a & \text{if } y = x \\
  s(y) & \text{otherwise}
\end{cases}$$
Semantics of Programs

**Binary Relation**

\[ \rho : \text{prog} \rightarrow S_D \times S_D \text{ assigns accessibility to programs} \]

\[
\begin{align*}
\rho(x := t) &= \{(s, t) \mid t = s[x/\text{val}_{D,I}, s(t)]\} \\
\rho(x := *) &= \{(s, t) \mid \text{ex. } a \text{ with } t = s[x/a]\} \\
\rho(\pi_1 \cup \pi_2) &= \rho(\pi_1) \cup \rho(\pi_2) \\
\rho(\pi_1 ; \pi_2) &= \rho(\pi_1) ; \rho(\pi_2) \quad \text{; is forward composition} \\
&= \{(s, t) \mid \text{ex. } u \in S_D \text{ with } (s, u) \in \rho(\pi_1), (u, t) \in \rho(\pi_2)\} \\
\rho(\pi^*) &= \rho(\pi)^* \quad * \text{ is refl. transitive closure} \\
&= \{(s_0, s_n) \mid \text{ex. } n \geq 0 \text{ with } (s_i, s_{i+1}) \in \rho(\pi) \text{ f.a. } i < n\} \\
\rho(\varphi) &= \{(s, s) \mid I, s \models \varphi\}
\end{align*}
\]
Semantics of Formulae

\[ l, s \models p(t_1, \ldots, t_n) \iff (\text{val}_{l,s}(t_1), \ldots, \text{val}_{l,s}(t_n)) \in l(p) \]

\[ l, s \models t_1 = t_2 \iff \text{val}_{l,s}(t_1) = \text{val}_{l,s}(t_2) \]

\[ l, s \models [\pi]F \iff l, s' \models F \text{ for all } s' \text{ with } (s, s') \in \rho(\pi) \]

\[ l, s \models \langle \pi \rangle F \iff l, s' \models F \text{ for some } s' \text{ with } (s, s') \in \rho(\pi) \]

\[ \models \text{ is homomorphic for } \neg, \land, \lor, \rightarrow, \forall x, \exists x. \]

We write \( l \models \varphi \) iff \( l, s \models \varphi \) for all \( s \in S \).
Basic Observation

$$\pi \in \text{prog} \ 	ext{a program}$$

$$FV(\pi) = \{ x \in \text{Var} \mid \text{ex. } t \text{ such that } x := t \text{ occurs in } \pi \}$$

$$V(\pi) = \{ x \in \text{Var} \mid x \text{ occurs in } \pi \}$$

1. If $$(s, s_1) \in \rho(\pi)$$ then $$s(x) = s_1(x)$$ for all $$x \not\in FV(\pi)$$. i.e., program $$\pi$$ only changes variables in $$FV(\pi)$$;

2. If $$(s, s_1) \in \rho(\pi)$$ then $$(s[x/a], s_1[x/a]) \in \rho(\pi)$$ for $$a \in D$$, $$x \not\in V(\pi)$$. i.e., variables outside $$V(\pi)$$ do not influence the program $$\pi$$;

3. more general: If $$(s, s_1) \in \rho(\pi)$$ and $$s' \in S_D$$ such that $$s'(y) = s(y)$$ for all $$y \in V^{\pi}$$ then there is $$s'_1$$ such that
   1. $$(s', s'_1) \in \rho(\pi)$$ and
   2. $$s'_1(x) = s'(x)$$ for all $$x \not\in V(\pi)$$
   3. $$s'_1(y) = s_1(y)$$ for all $$y \in V(\pi)$$. 
Basic Observation

\[(s, s_1) \in \rho(\pi) \text{ and } s' \text{ with } s'(y) = s(y) \text{ for all } y \in V(\pi) \text{ then there is } s'_1 \text{ with} \]

\[(s', s'_1) \in \rho(\pi), \quad s'_1(x) = \begin{cases} 
  s'(x) & \text{for all } x \not\in V(\pi) \\
  s_1(x) & \text{for all } x \in V(\pi) 
\end{cases}. \]
Interesting Tautologies

All PDL tautologies
e.g. $[\pi; \tau] \varphi \leftrightarrow [\pi][\tau] \varphi$

$[x := t] \varphi \leftrightarrow \langle x := t \rangle \varphi$

$[x := *] \varphi \leftrightarrow \forall x. \varphi$

$\langle x := * \rangle \varphi \leftrightarrow \exists x. \varphi$

$\varphi$ a FO formula w/o quantification over $x$:

$[x := t] \varphi \leftrightarrow \varphi[x/t]$

$\forall x.[\pi] \varphi \leftrightarrow [\pi]\forall x. \varphi$ \quad \text{if } x \not\in V(\pi)$
Example

\[ z = y \land \forall x. \ f(g(x)) = x \]
\[ \rightarrow \ [(y := g(y))^*](y := f(y))^*y = z \]

\[ z = y \land \forall x. \ f(g(x)) = x \]
\[ \rightarrow \ [\textbf{while } p(y) \textbf{ do } y := g(y)]\langle \textbf{while } y \neq z \textbf{ do } y := f(y) \rangle \textbf{true} \]
Indeterminism

DL programs can be indeterministic

Sources of indeterminism

- Non-deterministic choice $\cup$
- Non-deterministic iteration $\ast$
- Non-deterministic assignment $v := \ast$

Example for $v := \ast$:

choose $x$ such that $p(x) \iff x := \ast ; ?p(x)$
Deterministic programs

Definition

A DL program $\pi \in \text{prog}$ is called a while-program if:

1. $\cup$ occurs only within the patterns of if,
2. $*$ occurs only within the patterns of while,
3. $\text{var} := *$ does not occur for any variable $\text{var} \in \text{Var}$

Reminder

$\text{if } \varphi \text{ then } \alpha \text{ else } \beta := (\varphi ; \alpha) \cup (\neg \varphi ; \beta)$
$\text{while } \varphi \text{ do } \alpha := (\varphi ; \alpha)^* ; \neg \varphi$
Deterministic programs

Semantic Definition
A program $\pi \in \text{prog}$ is called deterministic if its accessibility relation is a partial function.

i.e., if $(s, t_1), (s, t_2) \in \rho(\pi) \implies t_1 = t_2$

Characterisation of deterministic programs
A program $\pi \in \text{prog}$ is deterministic iff $\langle \pi \rangle \varphi \rightarrow [\pi] \varphi$ is a tautology for every formula $\varphi \in \text{fml}$.

Observation
While programs are deterministic.
Deterministic programs

For deterministic programs:

\[ [\pi] \varphi \] means “\( \pi \) is \textbf{partially} correct with respect to postcondition \( \varphi \)”

\[ \langle \pi \rangle \varphi \] means “\( \pi \) is \textbf{totally} correct with respect to postcondition \( \varphi \)”

(i.e. \( \pi \) partially correct \textbf{and} \( \pi \) terminates)

Moreover:

Total correctness is partial correctness plus termination:

\[ \models \langle \pi \rangle \varphi \iff [\pi] \varphi \land \langle \pi \rangle \text{true} \]
Expressiveness

Expressiveness of uninterpreted FODL
First order dynamic logic is more expressive than first order logic.

Arithmetic **cannot** be axiomatised in FOL
a direct implication of Gödel’s Incompleteness Theorem

Arithmetic **can** be axiomatised in FODL
... we shall see how ...
Axiomatisation of natural arithmetic

**Signature:** Let $\Sigma$ contain:
- constant $o$ (the “zero”)
- unary function $s$ (the “successor”)

**Goal**
Define a FODL formula $\varphi_N$ over $\Sigma$ s.t.
$D, I \models \varphi_N$ iff $(D, I(n), I(s)) \sim (\mathbb{N}, 0, +1)$

**Idea:**
*Formalise:* “Every element can be reached by a number of loop iterations from zero.”

**Solution:**
$$\varphi_N := \forall y. \langle x := o; (x := s(x))^* \rangle x = y$$
$$\land \forall x, y. ((s(x) = s(y) \rightarrow x = y) \land \neg s(x) = o)$$
Interpreted Dynamic Logic

Fix the first order structure and domain.

In particular: consider

\[ \Sigma_\mathcal{N} = (\{0, 1, -1, \ldots, +, \ast \}, \{<\}) \text{ and } \mathcal{N} = (\mathbb{N}, I_\mathcal{N}) \]

s.t. \( I_\mathcal{N} \) interprets the symbols “as expected”.
Examples

Valid formulas:

- $3 < 5, x < x + 2, 0 \times x = 0$

- $(p(0) \land \forall x.(p(x) \to p(x + 1))) \to \forall x.p(x)$

- $\neg \exists x(0 < x \land x < 1)$

- $[y := x ; (a := \ast ; x := x + a)^\ast]x \geq y$

- $x_0 = x \land y_0 = y$
  \[ \to [x := x + y ; y := x - y ; x := x - y]x = y_0 \land y = x_0 \]
Relative Completeness and Calculi
Encoding sequences (Gödel, ~1930)

There exists a first-order definable function \( \beta : \mathbb{N} \times \mathbb{N} \to \mathbb{N} \) with:

For every \( n \in \mathbb{N} \) and every sequence \( c_1, \ldots, c_n \in \mathbb{N}^* \) there exists some \( c \) such that \( \beta(c, i) = c_i \) for \( i = 0, \ldots, n \).

\( c \) is called the Gödel number for \( c_1, \ldots, c_n \).
Notation: \( c = \left[ c_1, \ldots, c_n \right] \)

**Example encoding:**
\[
\left[ c_1, \ldots, c_n \right] := 2^{c_1+1} \cdot 3^{c_2+1} \cdot 5^{c_3+1} \cdot \ldots \cdot p_n^{1+c_n}
\]

\( \beta(c, i) = k \iff p_i^{k+1} \mid c \land p_i^{k+2} \nmid c \)

**Example:** \( \left[ 2, 0, 1 \right] = 2^3 \cdot 3^1 \cdot 5^2 = 600 \)
Comparing logics

- **Uninterpreted FODL is more expressive than FOL.** There exists a FODL formula such that no FOL formula has the same models.

- **Is FODL over \( \mathcal{N} \) more expressive than FOL over \( \mathcal{N} \)?** How can the compare expressiveness with a fixed interpretation?
Relative Completeness

Let $L$ be a logic.
Let $T \subseteq Fml_L$ be a set of formulas (a theory).

Oracle

Function $O_T : Fml_L \to \{\text{true, false}\}$ with $\varphi \in T \iff O(\varphi) = \text{true}$ is called an oracle for $T$.

Relative Completeness (Cook, 1978)

A logic is called complete relative to $T$ if there exists a correct and complete calculus which may make use of oracle $O_T$.

Note: $T$ (resp. $O_T$) may not be computable!
Relative Completeness of FODL

Let $T_{\mathbb{N}} = \{ \varphi \mid \mathbb{N} \models \varphi \}$ be the set of valid statements over $\mathbb{N}$.

**Theorem**

FODL is complete relative to $T_{\mathbb{N}}$. 
Programs as Formulas

Programs representable

Every DL program $\pi$ can be represented as a formula $\kappa(\pi) \in Fml_{FOL,N}$.

Here: only one-variable-programs

$V(\pi) = \{x\}$

(general case $\Rightarrow$ exercise)

Predicate $\kappa(\pi)(x, x')$ has two free variables:

1. $x$ for the pre-state,
2. $x'$ for the post-state.

Modelling goal:

$s[x'/s'(x)] \models \kappa(\pi)(x, x') \iff (s, s') \in \rho(\pi)$
Programs as Formulas (II)

\[ \kappa(x := t)(x, x') := x' = t \]

\[ \kappa(\pi_1 \cup \pi_2)(x, x') := \kappa(\pi_1)(x, x') \lor \kappa(\pi_2)(x, x') \]

\[ \kappa(\pi_1 ; \pi_2)(x, x') := \exists u. \ \kappa(\pi_1)(x, u) \land \kappa(\pi_2)(u, x') \]

\[ \kappa(\varphi)(x, x') := \varphi(x) \land x = x' \]

\[ \kappa(\pi^*) := \exists n. \exists x_1, \ldots, x_n \forall. \ x = x_1 \land x' = x_n \land \forall i < n. \ \kappa(\pi)(x_i, x_{i+1}) \]
Reduction of $\text{FODL}_N$ to $\text{FOL}_N$

**Theorem**

There is a function $\kappa : \text{Fml}_{\text{FODL}_N} \rightarrow \text{Fml}_{\text{FOL}_N}$ such that

- $\mathcal{N} \models \varphi \leftrightarrow \kappa(\varphi)$ and
- $\kappa$ is computable.

**Proof**

by structural induction.

Interesting case:

$$\kappa([\pi] \varphi(x)) \leftrightarrow \forall x'. \kappa(\pi)(x, x') \rightarrow \kappa(\varphi(x'))$$

(Remainder left as exercise)
A practical calculus

Let $\varphi$ be a FOL formula and $\pi$ a program with only FOL tests.

<table>
<thead>
<tr>
<th>Calculus</th>
</tr>
</thead>
<tbody>
<tr>
<td>$[x := t] \varphi \rightsquigarrow \varphi[x/t]$</td>
</tr>
<tr>
<td>$[\pi_1 ; \pi_2] \varphi \rightsquigarrow [\pi_1][\pi_2] \varphi$</td>
</tr>
<tr>
<td>$[\pi_1 \cup \pi_2] \varphi \rightsquigarrow [\pi_1] \varphi \land [\pi_2] \varphi$</td>
</tr>
<tr>
<td>$[?\psi] \varphi \rightsquigarrow \psi \rightarrow \varphi$</td>
</tr>
<tr>
<td>$[\pi^*] \varphi \rightsquigarrow INV$</td>
</tr>
</tbody>
</table>

for an arbitrary formula $INV \in Fml_{FOL}$. $\bar{x} = FV(\pi)$

The calculus allows reduction of FODL formulae to FOL formulae
Weakest Precondition Calculus

Let $\varphi$ be a FOL formula and $\pi$ a while program (with FOL tests).

Calculus

$[x := t] \varphi \leadsto \varphi[x/t]$

$[\pi_1 ; \pi_2] \varphi \leadsto [\pi_1][\pi_2] \varphi$

$[\text{if } \psi \text{ then } \pi_1 \text{ else } \pi_2] \varphi \leadsto (\psi \rightarrow [\pi_1] \varphi) \land (\neg \psi \rightarrow [\pi_2] \varphi)$

$[\text{while } \psi \text{ do } \pi] \varphi \leadsto INV$

$\land (\forall \bar{x}. INV \land \psi \rightarrow [\pi]INV)$

$\land (\forall \bar{x}. INV \land \neg \psi \rightarrow \varphi)$

for an arbitrary formula $INV \in Fml_{FOL}$. $\bar{x} = FV(\pi)$

This is the weakest-precondition calculus (Dijkstra, 1975)

Notation: $wlp(\pi, \varphi) = [\pi] \varphi$, $wp(\pi, \varphi) = \langle \pi \rangle \varphi$
Properties

Let $[\pi] \varphi \rightsquigarrow^* \psi$ be the result of applying the calculus.

1. $\models \psi \rightarrow [\pi] \varphi$
   $\psi$ is a precondition such that $\varphi$ is guaranteed to hold after $\pi$.

2. There exist loop invariants such that $\models \psi \leftrightarrow [\pi] \varphi$
   earlier defined $\kappa(\cdot)$ formulates strongest loop invariants
   Then $\psi$ is the weakest precondition

3. If $\models \text{pre} \rightarrow \psi$, then also $\models \text{pre} \rightarrow [\pi] \varphi$
   Prove pre/post-condition contracts by applying calculus to program and
   postcondition and then showing implication from precondition.
Arithmetic Completeness

Axioms
All first-order formulas valid in \( \mathcal{N} \)
Axioms for PDL
\[ \langle x := t \rangle \varphi \iff \varphi[x/t] \]

Rules
\[
\frac{F, F \rightarrow G}{G} \\
\frac{F}{[\pi]F} \\
\frac{F}{\forall x F}
\]

\[
\forall n(F(n + 1) \rightarrow \langle \pi \rangle F(n))
\]

\[
\forall n(F(n) \rightarrow \langle \pi^* \rangle F(0))
\]

Theorem
For any formula \( \varphi \in Fml_{FODL} \):
\[ \mathbb{N} \models \varphi \iff \vdash_{\mathbb{N}} \varphi \]

Beckert, Ulbrich – Formale Systeme II: Theorie