Dynamic Logic:
A Logic for Software Verification

Bernhard Beckert

UNIVERSITÄT KOBLENZ-LANDAU

November 2002

WHILE: A Simple Programming Language

Logical Basis

- **Variables (infinitely many)**
- **Built-in arithmetic:**

  function symbols $0, s, +.*$ **(terms** $s(0), s(s(0)), \ldots$ **written as** $1, 2, \ldots$**)**

  predicate symbols $\doteq$**,** $<$**,** $\leq$**,** $>$**,** $\geq$
- **Built-in propositional logic: operators** $\wedge$**,** $\vee$**,** $\rightarrow$

WHILE: A Simple Programming Language

Programs

- **Assignments:** $X := t$ — X: variable, t: term
- **Test:** **if** B **then** α **else** β **end** — B: formula, α, β: programs
- **Loop:** **while** B **do** α **end** — B: formula, α: program
- **Composition:** $\alpha_1; \ldots; \alpha_n$ — α_i programs

WHILE is computationally complete

WHILE: Examples

Compute the square of X and store it in Y

$$Y := X * X$$

If X is positive then add one else subtract one

$$\underline{\textbf{if}}\ X > 0\ \underline{\textbf{then}}\ X := X + 1\ \underline{\textbf{else}}\ X := X - 1\ \underline{\textbf{end}}$$

WHILE: Examples

Compute the square of X (the complicated way)

Making use of: $n^2 = 1 + 3 + 5 + \cdots + (2n-1)$

$$\left.\begin{array}{l}
I := 0\textbf{;} \\
Y := 0\textbf{;} \\
\underline{\textbf{while}}\ I < X\ \underline{\textbf{do}} \\
\quad Y := Y + 2 * I + 1\textbf{;} \\
\quad I := I + 1 \\
\underline{\textbf{end}}
\end{array}\right\}\ \alpha\textbf{square}$$

WHILE: Operational Semantics

State

Function from the variables into $\mathbb{N}$

Extended to terms and (propositional) formulas

using the canonical interpretations of $0, s, +, *, \doteq, <, \leq, >, \geq$

State update

s[X/n] is the state defined by

$$s[X/n](Y) = \begin{cases} n & \textbf{if } Y = X \\ s(Y) & \textbf{otherwise} \end{cases}$$

WHILE: Operational Semantics

Define the relation $s[\![\alpha]\!]s'$ as follows

- $s[\![X := t]\!]s'$ **iff** $s' = s[X/s(t)]$
- $s[\![\underline{\textbf{if}}\ B\ \underline{\textbf{then}}\ \alpha\ \underline{\textbf{else}}\ \beta\ \underline{\textbf{end}}]\!]s'$ **iff**

  $s(B) =$ **true and** $s[\![\alpha]\!]s'$ **or** $s(B) =$ **false and** $s[\![\beta]\!]s'$
- $s[\![\underline{\textbf{while}}\ B\ \underline{\textbf{do}}\ \alpha\ \underline{\textbf{end}}]\!]s'$ **iff there are states** $s = s_0, \ldots, s_t = s'$ **s.t.**

  $s_0(B) = s_1(B) = \cdots = s_{t-1}(B) =$ **true and** $s_t(B) =$ **false and**

  $s_0[\![\alpha]\!]s_1$, $s_1[\![\alpha]\!]s_2$, ..., $s_{t-1}[\![\alpha]\!]s_t$
- $s[\![\alpha_1; \ldots; \alpha_n]\!]s'$ **iff there are states** $s = s_0, \ldots, s_n = s'$ **such that**

  $s_0[\![\alpha_1]\!]s_1$, $s_1[\![\alpha_2]\!]s_2$, ..., $s_{n-1}[\![\alpha_n]\!]s_n$,

$[\![\alpha]\!]$ **is right-unique, but not right-total**

Expressing Program Properties

Logic for expressing properties

Full first-order logic with Peano arithmetic

Partial correctness assertion (Hoare formula)

$$\{P\}\ \alpha\ \{Q\}$$

Meaning:

If α is started in a state satisfying P and terminates, then its final state satisfies Q

Formally:

$\{P\}\ \alpha\ \{Q\}$ is valid iff
for all states s, s', if $s(P) =$ true and $s[\![\alpha]\!]s'$, then $s'(Q) =$ true

Expressing Program Properties: Examples

$\{$**true**$\}\ X := X + 1\ \{X > 1\}$

$\{even(X)\}\ X := X + 2\ \{even(X)\}$ **where** $even(X) \equiv \exists Z (X \doteq 2 * Z)$

$\{$**true**$\}\ \alpha_{\textbf{square}}\ \{Y = X * X\}$

The Idea of Dynamic Logic

A multi-modal logic

- the states are the possible worlds
- two modalities $[\alpha]$ and $\langle\alpha\rangle$ for each program α
- state s' is α-reachable from state s iff $s[\![\alpha]\!]s'$

Semantics

- $[\alpha]\,F$ true in a state s iff
  F is true in all states that are α-reachable from s
  (partial correctness)
- $\langle\alpha\rangle\,F$ true in a state s iff
  F is true in some state that is α-reachable from s
  (total correctness)
- A formula is valid iff it is valid in all states.

Dynamic Logic: Examples

Some examples

$[X := 1]\, X \doteq 1$

$[\underline{\textbf{while}}\ \textbf{true}\ \underline{\textbf{do}}\ X := X\ \underline{\textbf{end}}]\ \textbf{false}$

$(\langle\alpha\rangle\, X \doteq Y) \leftrightarrow (\langle\beta\rangle\, X \doteq Y)$

$\exists X\, \langle\alpha\rangle\, \textbf{true}$

Hoare formulas

$\{P\}\ \alpha\ \{Q\}$ **the same as** $P \rightarrow [\alpha]\, Q$

Duality of the modal operators

$[\alpha]\, P \ \leftrightarrow\ \neg\langle\alpha\rangle\neg P$

A Sequent Calculus for Dynamic Logic

Sequent $\Gamma \ \vdash \ \Delta$

Meaning: $\bigvee \Delta$ **is a consequence of** $\bigwedge \Gamma$

Axioms:

$$\frac{}{\phi, \Gamma \ \vdash \ \phi, \Delta} \qquad \frac{}{\textbf{false}, \Gamma \ \vdash \ \Delta} \qquad \frac{}{\Gamma \ \vdash \ \textbf{true}, \Delta}$$

Rules for propositional logic

$$\frac{\Gamma \ \vdash \ \phi, \Delta}{\neg\phi, \Gamma \ \vdash \ \Delta} \qquad \frac{\phi, \Gamma \ \vdash \ \Delta}{\Gamma \ \vdash \ \neg\phi, \Delta}$$

$$\frac{\phi, \psi, \Gamma \ \vdash \ \Delta}{\phi \wedge \psi, \Gamma \ \vdash \ \Delta} \qquad \frac{\Gamma \ \vdash \ \phi, \Delta \qquad \Gamma \ \vdash \ \psi, \Delta}{\Gamma \ \vdash \ \phi \wedge \psi, \Delta}$$

etc.

A Sequent Calculus for Dynamic Logic

Rules for quantifiers

$$\frac{\phi\{X \leftarrow t\}, \forall X\, \phi, \Gamma \ \vdash \ \Delta}{\forall X\, \phi, \Gamma \ \vdash \ \Delta} \qquad \frac{\Gamma \ \vdash \ \phi\{X \leftarrow t\}, \exists X\, \phi, \Delta}{\Gamma \ \vdash \ \exists X\, \phi, \Delta}$$

t **a term**

$$\frac{\phi\{X \leftarrow Y\}, \Gamma \ \vdash \ \Delta}{\exists X\, \phi, \Gamma \ \vdash \ \Delta} \qquad \frac{\Gamma \ \vdash \ \phi\{X \leftarrow Y\}, \Delta}{\Gamma \ \vdash \ \forall X\, \phi, \Delta}$$

Y **a new variable**

Cut rule

$$\frac{\Gamma \ \vdash \ \phi, \Delta \qquad \phi, \Gamma \ \vdash \ \Delta}{\Gamma \ \vdash \ \Delta}$$

A Sequent Calculus for Dynamic Logic

Equality rules

$$\frac{}{\vdash\ t \doteq t} \qquad \frac{s \doteq t, \Gamma^t_s\ \vdash\ \Delta^t_s}{s \doteq t, \Gamma\ \vdash\ \Delta} \qquad \frac{t \doteq s, \Gamma^t_s\ \vdash\ \Delta^t_s}{t \doteq s, \Gamma\ \vdash\ \Delta}$$

Peano arithmetic rule

$$\frac{}{\Gamma\ \vdash\ \Delta}$$

if no programs occur in Γ, Δ and $\bigwedge \Gamma \models_{\textbf{Peano}} \bigvee \Delta$

not “computable”!

A Sequent Calculus for Dynamic Logic

Composition rule

$$\frac{\Gamma \;\vdash\; [\alpha_1][\alpha_2;\ldots;\alpha_n]\,\phi, \Delta}{\Gamma \;\vdash\; [\alpha_1;\alpha_2;\ldots;\alpha_n]\,\phi, \Delta}$$

Assignment rule

$$\frac{\Gamma\{X \leftarrow X'\}, X \doteq t\{X \leftarrow X'\} \;\vdash\; \phi, \Delta\{X \leftarrow X'\}}{\Gamma \;\vdash\; [X := t]\,\phi, \Delta}$$

X' **a new variable**

Example:

$$\frac{even(X'), X \doteq X' + 2 \;\vdash\; even(X)}{even(X) \;\vdash\; [X := X + 2]\,even(X)}$$

A Sequent Calculus for Dynamic Logic

Test rule

$$\frac{\Gamma, B \;\vdash\; [\alpha]\,\phi, \Delta \qquad\qquad \Gamma, \neg B \;\vdash\; [\beta]\,\phi, \Delta}{\Gamma \;\vdash\; [\underline{\textbf{if}}\ B\ \underline{\textbf{then}}\ \alpha\ \underline{\textbf{else}}\ \beta\ \underline{\textbf{end}}]\,\phi, \Delta}$$

Loop rule

$$\frac{\Gamma \;\vdash\; Inv, \Delta \qquad\qquad Inv, B \;\vdash\; [\alpha]\,Inv \qquad\qquad Inv, \neg B \;\vdash\; \phi}{\Gamma \;\vdash\; [\underline{\textbf{while}}\ B\ \underline{\textbf{do}}\ \alpha\ \underline{\textbf{end}}]\,\phi, \Delta}$$

Inv an arbitrary formula – the invariant of the loop

Example

Invariant Inv: $I \leq X \ \wedge \ Y \doteq I * I$

$$\frac{I \doteq 0,\, Y \doteq 0 \vdash Inv \qquad Inv,\, B \vdash [\alpha]\, Inv \qquad Inv,\, \neg B \vdash Y \doteq X * X}{I \doteq 0,\, Y \doteq 0 \quad \vdash \quad [\underline{\textbf{while}}\ B\ \underline{\textbf{do}}\ \alpha\ \underline{\textbf{end}}]\, Y \doteq X * X}$$

$$I \doteq 0 \quad \vdash \quad [Y := 0]\, [\underline{\textbf{while}}\ B\ \underline{\textbf{do}}\ \alpha\ \underline{\textbf{end}}]\, Y \doteq X * X$$

$$\vdash \quad [I := 0]\, [Y := 0]\, [\underline{\textbf{while}}\ B\ \underline{\textbf{do}}\ \alpha\ \underline{\textbf{end}}]\, Y \doteq X * X$$

$$\vdash \quad [I := 0;\ \ Y := 0;\ \ \underline{\textbf{while}}\ B\ \underline{\textbf{do}}\ \alpha\ \underline{\textbf{end}}]\, Y \doteq X * X$$

$$\vdash \quad [\alpha_{\textbf{square}}]\, Y \doteq X * X$$

B**:** $\quad I < X$

α**:** $\quad Y := Y + 2 * I + 1;\ \ I := I + 1$

Example

Left branch (pre-condition implies invariant)

$$I \doteq 0,\ Y \doteq 0\ \vdash\ 0 \leq X \qquad\qquad I \doteq 0,\ Y \doteq 0\ \vdash\ Y \doteq 0 * 0$$

$$I \doteq 0,\ Y \doteq 0 \quad \vdash \quad 0 \leq X\ \wedge\ Y \doteq 0 * 0$$

$$I \doteq 0,\ Y \doteq 0 \quad \vdash \quad I \leq X\ \wedge\ Y \doteq I * I$$

Example

Middle branch (invariant is indeed invariant)

$$I' < X,\ I \doteq I' + 1 \ \vdash\ I \leq X$$

$$Y' \doteq I' * I',\ Y \doteq Y' + 2 * I' + 1,\ I \doteq I' + 1 \ \vdash\ Y \doteq I * I$$

$$I' \leq X,\ Y' \doteq I' * I',\ I' < X,\ Y \doteq Y' + 2 * I' + 1,\ I \doteq I' + 1 \quad \vdash \quad Inv$$

$$I \leq X,\ Y' \doteq I * I,\ I < X,\ Y := Y' + 2 * I + 1 \quad \vdash \quad [I := I + 1]\, Inv$$

$$I \leq X,\ Y \doteq I * I,\ I < X \quad \vdash \quad [Y := Y + 2 * I + 1]\, [I := I + 1]\, Inv$$

$$I \leq X,\ Y \doteq I * I,\ I < X \quad \vdash \quad [Y := Y + 2 * I + 1;\ \ I := I + 1]\, Inv$$

$$Inv,\ B \quad \vdash \quad [\alpha]\, Inv$$

Example

Right branch
(invariant and negated loop condition imply post-condition)

$$
\begin{array}{rcl}
I \leq X,\ Y \doteq I * I,\ \neg(I < X) & \vdash & I \doteq X\ \ Y \doteq I * I \\
I \leq X,\ Y \doteq I * I,\ \neg(I < X) & \vdash & I \doteq X,\ Y \doteq X * X \\
I \leq X,\ Y \doteq I * I,\ \neg(I < X) & \vdash & Y \doteq X * X \\
Inv \wedge \neg B & \vdash & Q
\end{array}
$$