Formal Methods in Software Engineering

Dynamic Logic

Bernhard Beckert

UNIVERSITÄT KOBLENZ-LANDAU

WHILE: A Simple Programming Language

Logical basis

Typed first-order predicate logic

(Types, variables, terms, formulas, . . .)

WHILE: A Simple Programming Language

Logical basis

Typed first-order predicate logic

(Types, variables, terms, formulas, . . .)

Assumption for examples

The signature contains a type Nat and appropriate symbols:

- **function symbols $0, s, +, *$ (terms $s(0), s(s(0)), \ldots$ written as $1, 2, \ldots$)**
- **predicate symbols $\doteq$, $<$, $\leq$, $>$, $\geq$**

NOTE: This is a "convenient assumption" not a definition

WHILE: A Simple Programming Language

Programs

- **Assignments:** $X := t$ — X: **variable,** t: **term**
- **Test:** **<u>if</u>** B **<u>then</u>** α **<u>else</u>** β **<u>fi</u>** — B: **quantifier-free formula,** α, β: **programs**
- **Loop:** **<u>while</u>** B **<u>do</u>** α **<u>od</u>** — B: **quantifier-free formula,** α: **program**
- **Composition:** $\alpha;\ \beta$ — α, β **programs**

WHILE is computationally complete

WHILE: Examples

Compute the square of X and store it in Y

$$Y := X * X$$

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

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

WHILE: Example – Square of a Number

Compute the square of X (the complicated way)

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

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

WHILE: Example – Multiplication

Russian multiplication

$$
\left.
\begin{array}{l}
Z := 0; \\
\underline{\textbf{while}}\ \neg\ (B \doteq 0)\ \underline{\textbf{do}} \\
\quad \underline{\textbf{if}}\ ((B/2) * 2 \doteq B)\ \underline{\textbf{then}} \\
\quad\quad A := 2 * A; \\
\quad\quad B := B/2 \\
\quad \underline{\textbf{else}} \\
\quad\quad Z := Z + A; \\
\quad\quad A := 2 * A; \\
\quad\quad B := B/2 \\
\quad \underline{\textbf{fi}} \\
\underline{\textbf{od}}
\end{array}
\right\} \alpha_{mult}
$$

WHILE: Operational Semantics

Given

A (fixed) first-order structure $\mathcal{A}$ interpreting the function and predicate symbols in the signature

State

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

where

β a variable assignment (i.e. function interpreting the variables)

WHILE: Operational Semantics

State update

$$s[X/e] \quad = \quad (\mathcal{A}\,, \beta[X/e])$$

with

$$\beta[X/e](Y) \quad = \quad \begin{cases} e & \textbf{if } Y = X \\ \beta(Y) & \textbf{otherwise} \end{cases}$$

WHILE: Operational Semantics

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

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

  $s \models B$ **and** $s[\![\alpha]\!]s'$ **or** $s \models \neg B$ **and** $s[\![\beta]\!]s'$
- $s[\![\underline{\textbf{while}}\ B\ \underline{\textbf{do}}\ \alpha\ \underline{\textbf{od}}]\!]s'$ **iff** **there are states** $s = s_0, \ldots, s_t = s'$ **s.t.**

  $s_i \models B$ **for** $0 \leq i \leq t-1$ **and** $s_t \models \neg B$ **and**

  $s_0[\![\alpha]\!]s_1$**,** $s_1[\![\alpha]\!]s_2$**, . . . ,** $s_{t-1}[\![\alpha]\!]s_t$
- $s[\![\alpha;\ \beta]\!]s'$ **iff** **there is a state** s'' **such that**

  $s[\![\alpha]\!]s''$ **and** $s''[\![\beta]\!]s'$

$[\![\alpha]\!]$ **is a partial function**

A Different Approach to WHILE

Programs

- $X := t$ **(atomic program)**
- $\alpha;\ \beta$ **(sequential composition)**
- $\alpha \cup \beta$ **(non-deterministic choice)**
- α^* **(non-deterministic iteration, n times for some $n \geq 0$)**
- $F?$ **(test)**
  remains in initial state if F is true, does not terminate if F is false

A Different Approach to WHILE

Restriction to deterministic programs

Non-deterministic program constructors may only be used in

$$\underline{\textbf{if}}\ B\ \underline{\textbf{then}}\ \alpha\ \underline{\textbf{else}}\ \beta\ \underline{\textbf{fi}} \quad \equiv \quad (B?;\ \alpha) \cup ((\neg\ B)?;\ \beta)$$

$$\underline{\textbf{while}}\ B\ \underline{\textbf{do}}\ \alpha\ \underline{\textbf{od}} \quad \equiv \quad (B?;\ \alpha)^*;\ (\neg\ B)?$$

Expressing Program Properties

Logic for expressing properties

Full first-order logic (usually with arithmetic)

Expressing Program Properties

Logic for expressing properties

Full first-order logic (usually with 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 $s \models P$ and $s[\![\alpha]\!]s'$, then $s' \models Q$

Expressing Program Properties: Examples

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

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

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

An Annotated Program

$Z := 0;$
assert $X \doteq A \wedge Y \doteq B;$
while $\neg\, (B \doteq 0)$ **do**
    assert $A * B + Z \doteq X * Y;$
    if $((B/2) * 2 \doteq B)$ **then**
        $A := 2 * A;$
        $B := B/2$
    else
        $Z := Z + A;$
        $A := 2 * A;$
        $B := B/2$
    fi
od
assert $B \doteq 0$
assert $Z \doteq X * Y$

Note

X, Y **are "external" variables**

Dynamic Logic

The idea of dynamic logic

- **Annotated programs use formulas within programs**
- **Dynamic Logic uses programs within formulas**
- **Instead of "assert F" after program segment α, write:** $[\alpha]F$

Dynamic Logic

The idea of dynamic logic

- **Annotated programs use formulas within programs**
- **Dynamic Logic uses programs within formulas**
- **Instead of "assert F" after program segment α, write:** $[\alpha]F$

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 $s[\![\alpha]\!]s'$**

Dynamic Logic: Semantics

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

Example formulas **(validity depends on α, β)**

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

$\exists\, X\, \langle\alpha\rangle$ **true**

Dynamic Logic: Examples

Example formulas **(validity depends on α,β)**

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

$\exists\, X\, \langle\alpha\rangle$ **true**

Valid formulas

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

[**<u>while</u> true <u>do</u>** $X := X$ **<u>od</u>**] **false**

$\langle\alpha^*\rangle F \longrightarrow (F \vee \langle\alpha^*\rangle(\neg\, F \wedge \langle\alpha\rangle F))$

Dynamic Logic: Examples

Example formulas **(validity depends on α, β)**

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

$\exists\, X\, \langle\alpha\rangle$ **true**

Valid formulas

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

[**while** **true** **do** $X := X$ **od**] **false**

$\langle\alpha^*\rangle F \longrightarrow (F \vee \langle\alpha^*\rangle(\neg\, F \wedge \langle\alpha\rangle F))$

Multiplication example

$\forall\, A, B, X, Y, Z(X \doteq A \wedge Y \doteq B \longrightarrow [\alpha_{mult}]\, Z \doteq X * Y)$

Dynamic Logic: More Examples

Hoare formulas

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

Dynamic Logic: More Examples

Hoare formulas

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

Duality of the modal operators

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

Some DL-Tautologies

Assumption: X **does not occur in** π

$$(\exists X \langle\pi\rangle F) \quad \leftrightarrow \quad (\langle\pi\rangle \exists X F)$$

$$(\forall X [\pi] F) \quad \leftrightarrow \quad ([\pi] \forall X F)$$

$$(\exists X [\pi] F) \quad \longrightarrow \quad ([\pi] \exists X F)$$

$$([\pi] \exists X F) \quad \longrightarrow \quad (\exists X [\pi] F)$$ **provided** π **is deterministic**

$$(\langle\pi\rangle \forall X F) \quad \longrightarrow \quad (\forall X \langle\pi\rangle F)$$

$$(\forall X \langle\pi\rangle F) \quad \longrightarrow \quad (\langle\pi\rangle \forall X F)$$ **provided** π **is deterministic**

$$(\langle\pi\rangle(F \wedge G)) \quad \longrightarrow \quad ((\langle\pi\rangle F) \wedge \langle\pi\rangle G)$$

$$((\langle\pi\rangle F) \wedge \langle\pi\rangle G) \quad \longrightarrow \quad (\langle\pi\rangle(F \wedge G))$$ **provided** π **is deterministic**

A Sequent Calculus for Dynamic Logic

Sequent

$$\Gamma \rightarrow \Delta$$

Meaning

$\bigwedge \Gamma$ **logically implies** $\bigvee \Delta$

(for all variable assignments, i.e.,
free variables in the sequent are implicitly universally quantified)

Sequent Rules

Form of sequent rules

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

(rules can also have more than two premisses)

Sequent Rules

Form of sequent rules

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

(rules can also have more than two premisses)

Meaning

The conclusion is true in a state
whenever all premisses are true in that state

In particular:

The conclusion is valid whenever all premisses are valid

Sequent Calculus for First-order Logic

Axioms

$$\frac{}{F,\ \Gamma \ \rightarrow\ F,\ \Delta} \qquad \frac{}{\textbf{false},\ \Gamma \ \rightarrow\ \Delta} \qquad \frac{}{\Gamma \ \rightarrow\ \textbf{true},\ \Delta}$$

Sequent Calculus for First-order Logic

Axioms

$$\frac{}{F, \Gamma \rightarrow F, \Delta} \qquad \frac{}{\textbf{false}, \Gamma \rightarrow \Delta} \qquad \frac{}{\Gamma \rightarrow \textbf{true}, \Delta}$$

Negation

$$\frac{\Gamma \rightarrow F, \Delta}{\Gamma, \neg F \rightarrow \Delta} \qquad \frac{\Gamma, F \rightarrow \Delta}{\Gamma \rightarrow \neg F, \Delta}$$

Sequent Calculus for First-order Logic

Axioms

$$\frac{}{F, \Gamma \rightarrow F, \Delta} \qquad \frac{}{\textbf{false}, \Gamma \rightarrow \Delta} \qquad \frac{}{\Gamma \rightarrow \textbf{true}, \Delta}$$

Negation

$$\frac{\Gamma \rightarrow F, \Delta}{\Gamma, \neg F \rightarrow \Delta} \qquad \frac{\Gamma, F \rightarrow \Delta}{\Gamma \rightarrow \neg F, \Delta}$$

Implication

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

Sequent Calculus for First-order Logic

Conjunction

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

Sequent Calculus for First-order Logic

Conjunction

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

Disjunction

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

Sequent Calculus for First-order Logic

Universal quantification

$$\frac{\Gamma,\ \forall X F,\ F\{X \leftarrow t\} \ \rightarrow\ \Delta}{\Gamma,\ \forall X F \ \rightarrow\ \Delta}$$

t **an arbitrary term,**

$\{X \leftarrow t\}$ **admissible for** F

$$\frac{\Gamma \ \rightarrow\ F\{X \leftarrow Z\},\ \Delta}{\Gamma \ \rightarrow\ \forall X F,\ \Delta}$$

Z **a new variable**

Sequent Calculus for First-order Logic

Universal quantification

$$\frac{\Gamma, \forall X F, F\{X \leftarrow t\} \rightarrow \Delta}{\Gamma, \forall X F \rightarrow \Delta}$$

t **an arbitrary term,**

$\{X \leftarrow t\}$ **admissible for** F

$$\frac{\Gamma \rightarrow F\{X \leftarrow Z\}, \Delta}{\Gamma \rightarrow \forall X F, \Delta}$$

Z **a new variable**

Existential quantification

$$\frac{\Gamma \rightarrow \exists X F, F\{X \leftarrow t\}, \Delta}{\Gamma \rightarrow \exists X F, \Delta}$$

t **an arbitrary term,**

$\{X \leftarrow t\}$ **admissible for** F

$$\frac{\Gamma, F\{X \leftarrow Z\} \rightarrow \Delta}{\Gamma, \exists X F \rightarrow \Delta}$$

Z **a new variable**

Example Proof

axiom

6 $p(V),\ p(U) \rightarrow p(U),\ \forall Y\, p(Y)$

impl-right

5 $p(V) \rightarrow p(U),\ p(U) \rightarrow \forall Y\, p(Y)$

ex-right

4 $p(V) \rightarrow p(U),\ \exists X (p(X) \rightarrow \forall Y\, p(Y))$

all-right

3 $p(V) \rightarrow \forall Y\, p(Y),\ \exists X\, (p(X) \rightarrow \forall Y\, p(Y))$

impl-right

2 $\rightarrow p(V) \rightarrow \forall Y\, p(Y),\ \exists X\, (p(X) \rightarrow \forall Y\, p(Y))$

ex-right

1 $\rightarrow \exists X\, (p(X) \rightarrow \forall Y\, p(Y))$

Admissibility of Substitutions

Motivation

We want to have that

$$\forall X F \quad \longrightarrow \quad F\sigma$$

$$F\sigma \quad \longrightarrow \quad \exists X F$$

is valid for all formulas F and substitutions σ

Admissibility of Substitutions

Definition

A substitution

$$\{ X \leftarrow t \}$$

is admissible for a formula F iff

there is no variable Y such that

- Y **occurs in** t
- **there is a quantification $\forall Y$ or $\exists Y$ in F**
- **there is a free occurrencence of X in the scope of that quantification**

Sequent Calculus for Dynamic Logic

Cut rule

$$\frac{\Gamma \rightarrow F, \Delta \qquad F, \Gamma \rightarrow \Delta}{\Gamma \rightarrow \Delta}$$

Sequent Calculus for Dynamic Logic

Cut rule

$$\frac{\Gamma \rightarrow F, \Delta \qquad F, \Gamma \rightarrow \Delta}{\Gamma \rightarrow \Delta}$$

Equality rules

$$\frac{}{\Gamma \rightarrow t \doteq t,\ \Delta}$$

$$\frac{s \doteq t,\ \Gamma\{s \leftarrow t\} \rightarrow \Delta\{s \leftarrow t\}}{s \doteq t,\ \Gamma \rightarrow \Delta} \qquad \frac{t \doteq s,\ \Gamma\{s \leftarrow t\} \rightarrow \Delta\{s \leftarrow t\}}{t \doteq s,\ \Gamma \rightarrow \Delta}$$

Sequent Calculus for Dynamic Logic

Oracle for first-order logic

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

if no programs occur in Γ, Δ **and** $\mathcal{A} \models \bigwedge \Gamma \rightarrow \bigvee \Delta$

Only of theoretical use! Not computable!

A Sequent Calculus for Dynamic Logic

Composition rule

$$\frac{\Gamma \rightarrow [\alpha][\beta]F,\ \Delta}{\Gamma \rightarrow [\alpha;\ \beta]F,\ \Delta}$$

A Sequent Calculus for Dynamic Logic

Composition rule

$$\frac{\Gamma \rightarrow [\alpha][\beta]F,\ \Delta}{\Gamma \rightarrow [\alpha;\ \beta]F,\ \Delta}$$

Assignment rule

$$\frac{\Gamma\{X \leftarrow X'\},\ X \doteq t\{X \leftarrow X'\} \rightarrow F,\ \Delta\{X \leftarrow X'\}}{\Gamma \rightarrow [X := t]F,\ \Delta}$$

X' **a new variable**

Example:

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

A Sequent Calculus for Dynamic Logic

Conditional rule

$$\frac{\Gamma, B \rightarrow [\alpha]F, \Delta \qquad \Gamma, \neg B \rightarrow [\beta]F, \Delta}{\Gamma \rightarrow [\underline{\textbf{if}}\ B\ \underline{\textbf{then}}\ \alpha\ \underline{\textbf{else}}\ \beta\ \underline{\textbf{fi}}]F, \Delta}$$

Reasoning about Loops

To prove

$$[\underline{\textbf{while}}\ B\ \underline{\textbf{do}}\ body\ \underline{\textbf{od}}]\,F$$

find an (arbitrary) formula Inv such that

1. Inv **is true before execution of the loop**
2. $Inv \wedge B \longrightarrow [body]Inv$ **is true**
3. $Inv \wedge \neg\, B \longrightarrow F$ **is true**

Note

Inv **is a loop invariant**

Sequent Calculus for Dynamic Logic

Loop rule

$$\frac{\Gamma \rightarrow Inv, \Delta \qquad Inv, B \rightarrow [\alpha]\, Inv \qquad Inv, \neg B \rightarrow F}{\Gamma \rightarrow [\underline{\textbf{while}}\ B\ \underline{\textbf{do}}\ \alpha\ \underline{\textbf{od}}]\, F, \Delta}$$

Example

➔ $[\alpha_{square}]\, Y \doteq X * X$

B**:** $I < X$

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

Example

➔ $[I := 0; \quad Y := 0; \quad$ **while** B **do** α **od**$]\, Y \doteq X * X$

➔ $[\alpha_{square}]\, Y \doteq X * X$

B**:** $I < X$

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

Example

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

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

➙ $[\alpha_{square}]\,Y \doteq X * X$

B: $I < X$

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

Example

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

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

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

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

B**:** $I < X$

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

Example

$$I \doteq 0,\ Y \doteq 0 \quad \rightarrow \quad [\underline{\textbf{while}}\ B\ \underline{\textbf{do}}\ \alpha\ \underline{\textbf{od}}]\, Y \doteq X * X$$

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

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

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

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

B**:** $I < X$

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

Example

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

$$I \doteq 0,\ Y \doteq 0 \rightarrow Inv \qquad Inv,\ B \rightarrow [\alpha]\,Inv \qquad Inv,\ \neg B \rightarrow Y \doteq X * X$$

$$I \doteq 0,\ Y \doteq 0 \quad \rightarrow \quad [\underline{\textbf{while}}\ B\ \underline{\textbf{do}}\ \alpha\ \underline{\textbf{od}}]\,Y \doteq X * X$$

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

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

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

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

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

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

Example

Left branch (pre-condition implies invariant)

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

Example

Left branch (pre-condition implies invariant)

$$I \doteq 0,\ Y \doteq 0 \quad \rightarrow \quad 0 \leq X \ \land \ Y \doteq 0 * 0$$

$$I \doteq 0,\ Y \doteq 0 \quad \rightarrow \quad I \leq X \ \land \ Y \doteq I * I$$

Example

Left branch (pre-condition implies invariant)

$$I \doteq 0,\ Y \doteq 0 \quad \rightarrow \quad 0 \leq X \qquad\qquad I \doteq 0,\ Y \doteq 0 \quad \rightarrow \quad Y \doteq 0 * 0$$

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

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

Example

Middle branch (invariant is indeed invariant)

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

Example

Middle branch (invariant is indeed invariant)

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

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

Example

Middle branch (invariant is indeed invariant)

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

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

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

Example

Middle branch (invariant is indeed invariant)

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

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

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

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

Example

Middle branch (invariant is indeed invariant)

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

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

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

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

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

Example

Middle branch (invariant is indeed invariant)

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

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

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

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

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

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

$$Inv,\ B \ \rightarrow\ [\alpha]\, Inv$$

Example

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

$$Inv \wedge \neg B \quad \rightarrow \quad Q$$

Example

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

$$I \leq X,\ Y \doteq I * I,\ \neg\,(I < X) \quad \rightarrow \quad Y \doteq X * X$$

$$Inv \wedge \neg\, B \quad \rightarrow \quad Q$$

Example

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

$$I \leq X,\ Y \doteq I * I,\ \neg\,(I < X) \quad \rightarrow \quad I \doteq X,\ Y \doteq X * X$$

$$I \leq X,\ Y \doteq I * I,\ \neg\,(I < X) \quad \rightarrow \quad Y \doteq X * X$$

$$Inv \wedge \neg\, B \quad \rightarrow \quad Q$$

Example

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

$$I \leq X,\ Y \doteq I * I,\ \neg\,(I < X) \quad \rightarrow \quad I \doteq X,\ Y \doteq I * I$$

$$I \leq X,\ Y \doteq I * I,\ \neg\,(I < X) \quad \rightarrow \quad I \doteq X,\ Y \doteq X * X$$

$$I \leq X,\ Y \doteq I * I,\ \neg\,(I < X) \quad \rightarrow \quad Y \doteq X * X$$

$$Inv \wedge \neg\, B \quad \rightarrow \quad Q$$

Example II: Multiplication

$$X \doteq A,\ Y \doteq B \quad \rightarrow \quad [\alpha_{mult}]\, Z \doteq X * Y$$

Example II: Multiplication

$$X \doteq A,\ Y \doteq B \quad \rightarrow \quad [Z := 0;\ \ \alpha_{while}]\, Z \doteq X * Y$$

$$X \doteq A,\ Y \doteq B \quad \rightarrow \quad [\alpha_{mult}]\, Z \doteq X * Y$$

Example II: Multiplication

$$X \doteq A,\ Y \doteq B,\ Z \doteq 0 \quad \rightarrow \quad [\alpha_{while}]\, Z \doteq X * Y$$

$$X \doteq A,\ Y \doteq B \quad \rightarrow \quad [Z := 0;\ \ \alpha_{while}]\, Z \doteq X * Y$$

$$X \doteq A,\ Y \doteq B \quad \rightarrow \quad [\alpha_{mult}]\, Z \doteq X * Y$$

Example II: Multiplication

Invariant Inv: $A * B + Z \doteq X * Y$

$$X \doteq A,\ Y \doteq B,\ Z \doteq 0 \quad \rightarrow \quad Inv$$

$$Inv,\ \neg B \doteq 0 \quad \rightarrow \quad [\alpha_{body}]\, Inv$$

$$Inv,\ B \doteq 0 \quad \rightarrow \quad Z \doteq X$$

$$X \doteq A,\ Y \doteq B,\ Z \doteq 0 \quad \rightarrow \quad [\alpha_{while}]\, Z \doteq X * Y$$

$$X \doteq A,\ Y \doteq B \quad \rightarrow \quad [Z := 0;\ \ \alpha_{while}]\, Z \doteq X * Y$$

$$X \doteq A,\ Y \doteq B \quad \rightarrow \quad [\alpha_{mult}]\, Z \doteq X * Y$$

Example II: Multiplication

Left branch (pre-condition implies invariant)

$$X \doteq A,\ Y \doteq B,\ Z \doteq 0 \quad \rightarrow \quad A * B + Z \doteq X * Y$$

Example II: Multiplication

Left branch (pre-condition implies invariant)

$$X \doteq A,\ Y \doteq B,\ Z \doteq 0 \quad \rightarrow \quad A * B + Z \doteq X * Y$$

Middle branch (invariant is indeed invariant)

$$A * B + Z \doteq X * Y,\ \neg B \doteq 0 \quad \rightarrow \quad [\alpha_{body}] A * B + Z \doteq X * Y$$

Example II: Multiplication

Left branch (pre-condition implies invariant)

$$X \doteq A,\ Y \doteq B,\ Z \doteq 0 \quad \rightarrow \quad A * B + Z \doteq X * Y$$

Middle branch (invariant is indeed invariant)

$$A * B + Z \doteq X * Y,\ \neg B \doteq 0 \quad \rightarrow \quad [\alpha_{body}] A * B + Z \doteq X * Y$$

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

$$A * B + Z \doteq X * Y,\ B \doteq 0 \quad \rightarrow \quad Z \doteq X * Y$$

Induction Rule

Purpose

- Needed to prove first-order theorems on natural numbers (oracle not available in practice)
- Handling loops in $\langle\cdot\rangle$ modality

Induction Rule

Purpose

- **Needed to prove first-order theorems on natural numbers (oracle not available in practice)**
- **Handling loops in $\langle\cdot\rangle$ modality**

$$\frac{\Gamma \rightarrow F\{N \leftarrow 0\},\ \Delta \qquad \Gamma,\ F \rightarrow F\{N \leftarrow N+1\},\ \Delta \qquad \Gamma,\ \forall N F \rightarrow \Delta}{\Gamma \rightarrow \Delta}$$

N **not occurring in** Γ, Δ

N **not occurring in any program in** F

Induction Rule: Example

$$\begin{array}{rcl} & \rightarrow & even(2*0),\ even(2*3) \\ even(2*N) & \rightarrow & even(2*(N+1)),\ even(2*3) \\ \forall N\,(even(2*N)) & \rightarrow & even(2*3) \\ \hline & \rightarrow & even(2*3) \end{array}$$

Loop Unwind Rule

Rule

$$\frac{\Gamma, \neg B \;\rightarrow\; F, \Delta \qquad\qquad \Gamma, B \;\rightarrow\; \langle \alpha \rangle \langle \underline{\textbf{while}}\ B\ \underline{\textbf{do}}\ \alpha\ \underline{\textbf{od}} \rangle F, \Delta}{\Gamma \;\rightarrow\; \langle \underline{\textbf{while}}\ B\ \underline{\textbf{do}}\ \alpha\ \underline{\textbf{od}} \rangle F, \Delta}$$

Loop Unwind Rule

Rule

$$\frac{\Gamma, \neg B \ \rightarrow\ F, \Delta \qquad\qquad \Gamma, B \ \rightarrow\ \langle \alpha \rangle \langle \underline{\textbf{while}}\ B\ \underline{\textbf{do}}\ \alpha\ \underline{\textbf{od}} \rangle F, \Delta}{\Gamma \ \rightarrow\ \langle \underline{\textbf{while}}\ B\ \underline{\textbf{do}}\ \alpha\ \underline{\textbf{od}} \rangle F, \Delta}$$

Note

Only useful

- **in connection with induction rule, or**
- **if number of loop iterations has a (small) known upper bound**

Loop Unwind Rule / Induction Rule: Example

Proof goal

- $\langle \underline{\textbf{while}}\ I > 0\ \underline{\textbf{do}}\ I := I - 1\ \underline{\textbf{od}} \rangle\, I \doteq 0$

Induction hypothesis

$$F(N) \quad \equiv \quad \forall I\, (I \leq N \;\rightarrow\; \langle \underline{\textbf{while}}\ I > 0\ \underline{\textbf{do}}\ I := I - 1\ \underline{\textbf{od}} \rangle\, I \doteq 0)$$

Admissibility of Substitutions Revisited

Problem

Previous definition of admissibility
is not sufficient if formulas contain programs

Example

$$F \quad \equiv \quad J \doteq K \rightarrow [I := 0]\,(J \doteq K) \qquad \textbf{valid}$$

Admissibility of Substitutions Revisited

Problem

Previous definition of admissibility is not sufficient if formulas contain programs

Example

$$F \equiv J \doteq K \rightarrow [I := 0](J \doteq K) \qquad \textbf{valid}$$

$$F\{I \leftarrow J\} \equiv J \doteq K \rightarrow [J := 0](J \doteq K) \qquad \textbf{not valid}$$

Admissibility of Substitutions Revisited

Problem

Previous definition of admissibility is not sufficient if formulas contain programs

Example

$$F \equiv J \doteq K \rightarrow [I := 0]\,(J \doteq K) \qquad \textbf{valid}$$

$$F\{I \leftarrow J\} \equiv J \doteq K \rightarrow [J := 0]\,(J \doteq K) \qquad \textbf{not valid}$$

$$F\{J \leftarrow I\} \equiv I \doteq K \rightarrow [I := 0]\,(I \doteq K) \qquad \textbf{not valid}$$

Admissibility of Substitutions Revisited

Problem

Previous definition of admissibility is not sufficient if formulas contain programs

Example

$$F \equiv J \doteq K \rightarrow [I := 0](J \doteq K)$$ **valid**

$$F\{I \leftarrow J\} \equiv J \doteq K \rightarrow [J := 0](J \doteq K)$$ **not valid**

$$F\{J \leftarrow I\} \equiv I \doteq K \rightarrow [I := 0](I \doteq K)$$ **not valid**

$$F\{I \leftarrow 1\} \equiv J \doteq K \rightarrow [1 := 0](J \doteq K)$$ **not a formula**

Admissibility of Substitutions Revisited

Revised definition

A substitution $\{X \leftarrow t\}$ **is admissible for a formula** F **iff**

1. $t = X$, **or**

2. t **is a variable not occurring in** F, **or**

3. **there is no variable** Y **in** t **such that a free occurrence of** X **in** F **is in the scope of**

   (a) a quantification $\forall Y$ **or** $\exists Y$, **or**

   (b) a modality containing an assignment of the form $Y := s$