
Formal Verification of Software

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 $s(0), s(s(0)), \dots$ written as $1, 2, \dots$)
- predicate symbols $\dot{=}, <, \leq, >, \geq$

NOTE: This is a “convenient assumption” not a definition

WHILE: A Simple Programming Language

Programs

- **Assignments:** $X := t$ X : variable, t : term
- **Test:** if B then α else β fi B : quantifier-free formula,
 α, β : programs
- **Loop:** while B do α od 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

if $X > 0$ then $X := X + 1$ else $X := X - 1$ fi

WHILE: Example – Square of a Number

Compute the square of X (the complicated way)

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

$I := 0;$

$Y := 0;$

while $I < X$ **do**

$Y := Y + 2 * I + 1;$

$I := I + 1$

od

} α square

WHILE: Example – Multiplication

Russian multiplication

$Z := 0;$

while $\neg (B \doteq 0)$ **do**

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

α_{mult}

WHILE: Operational Semantics

Given

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

State

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

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

WHILE: Operational Semantics

State update

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

with

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

WHILE: Operational Semantics

Define the relation $s \llbracket \alpha \rrbracket s'$ as follows

- $s \llbracket X := t \rrbracket s'$ iff $s' = s[X/s(t)]$
- $s \llbracket \underline{\text{if}} B \underline{\text{then}} \alpha \underline{\text{else}} \beta \underline{\text{fi}} \rrbracket s'$ iff
 $s \models B$ and $s \llbracket \alpha \rrbracket s'$ or $s \models \neg B$ and $s \llbracket \beta \rrbracket s'$
- $s \llbracket \underline{\text{while}} B \underline{\text{do}} \alpha \underline{\text{od}} \rrbracket s'$ iff there are states $s = s_0, \dots, s_t = s'$ s.t.
 $s_i \models B$ for $0 \leq i \leq t-1$ and $s_t \models \neg B$ and
 $s_0 \llbracket \alpha \rrbracket s_1, s_1 \llbracket \alpha \rrbracket s_2, \dots, s_{t-1} \llbracket \alpha \rrbracket s_t$
- $s \llbracket \alpha; \beta \rrbracket s'$ iff there is a state s'' such that
 $s \llbracket \alpha \rrbracket s''$ and $s'' \llbracket \beta \rrbracket s'$

$\llbracket \alpha \rrbracket$ 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

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

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

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 $s \llbracket \alpha \rrbracket s'$, then $s' \models Q$

Expressing Program Properties: Examples

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

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

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

An Annotated Program

```
Z := 0;
assert X ≐ A ∧ Y ≐ B;
while ¬ (B ≐ 0) do
    assert A * B + Z ≐ X * Y;
    if ((B/2) * 2 ≐ B) then
        A := 2 * A;
        B := B/2
    else
        Z := Z + A;
        A := 2 * A;
        B := B/2
    fi
od
assert B ≐ 0
assert Z ≐ 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)

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

$$\exists X \langle \alpha \rangle \mathbf{true}$$

Dynamic Logic: Examples

Example formulas (validity depends on α, β)

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

$$\exists X \langle \alpha \rangle \mathbf{true}$$

Valid formulas

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

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

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

Dynamic Logic: Examples

Example formulas (validity depends on α, β)

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

$$\exists X \langle \alpha \rangle \mathbf{true}$$

Valid formulas

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

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

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

Multiplication example

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

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** $P \rightarrow [\alpha] Q$

Duality of the modal operators

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

Some DL-Tautologies

Assumption: X does not occur in π

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

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

$$(\exists X [\pi] F) \rightarrow ([\pi] \exists X F)$$

$$([\pi] \exists X F) \rightarrow (\exists X [\pi] F)$$

provided π is deterministic

$$(\langle \pi \rangle \forall X F) \rightarrow (\forall X \langle \pi \rangle F)$$

$$(\forall X \langle \pi \rangle F) \rightarrow (\langle \pi \rangle \forall X F)$$

provided π is deterministic

$$(\langle \pi \rangle (F \wedge G)) \rightarrow ((\langle \pi \rangle F) \wedge \langle \pi \rangle G)$$

$$(((\langle \pi \rangle F) \wedge \langle \pi \rangle G) \rightarrow (\langle \pi \rangle (F \wedge G)))$$

provided π is deterministic

A Sequent Calculus for Dynamic Logic

Sequent

$$\Gamma \rightarrow \Delta$$

Meaning

$\wedge \Gamma$ logically implies $\vee \Delta$

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

Sequent Rules

Form of sequent rules

$$\frac{\Gamma_1 \rightarrow \Delta_1}{\Gamma_2 \rightarrow \Delta_2} \quad \text{or} \quad \frac{\Gamma_1 \rightarrow \Delta_1 \quad \Gamma'_1 \rightarrow \Delta'_1}{\Gamma_2 \rightarrow \Delta_2}$$

(rules can also have more than two premisses)

Sequent Rules

Form of sequent rules

$$\frac{\Gamma_1 \rightarrow \Delta_1}{\Gamma_2 \rightarrow \Delta_2} \quad \text{or} \quad \frac{\Gamma_1 \rightarrow \Delta_1 \quad \Gamma'_1 \rightarrow \Delta'_1}{\Gamma_2 \rightarrow \Delta_2}$$

(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

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

$$\frac{}{\mathbf{false}, \Gamma \rightarrow \Delta}$$

$$\frac{}{\Gamma \rightarrow \mathbf{true}, \Delta}$$

Sequent Calculus for First-order Logic

Axioms

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

$$\frac{}{\mathbf{false}, \Gamma \rightarrow \Delta}$$

$$\frac{}{\Gamma \rightarrow \mathbf{true}, \Delta}$$

Negation

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

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

Sequent Calculus for First-order Logic

Axioms

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

$$\frac{}{\text{false}, \Gamma \rightarrow \Delta}$$

$$\frac{}{\Gamma \rightarrow \text{true}, \Delta}$$

Negation

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

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

Implication

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

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

Sequent Calculus for First-order Logic

Conjunction

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

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

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

t an arbitrary term,
 $\{X \leftarrow t\}$ admissible for F

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

Z a **new** variable

Sequent Calculus for First-order Logic

Universal quantification

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

t an arbitrary term,
 $\{X \leftarrow t\}$ admissible for F

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

Z a **new** variable

Existential quantification

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

t an arbitrary term,
 $\{X \leftarrow t\}$ admissible for F

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

Z a **new** variable

Example Proof

6	$p(V), p(U)$	$\xrightarrow{\text{axiom}}$	$p(U), \forall Y p(Y)$
		\downarrow	impl-right
5	$p(V)$	$\xrightarrow{\quad}$	$p(U), p(U) \rightarrow \forall Y p(Y)$
		\downarrow	ex-right
4	$p(V)$	$\xrightarrow{\quad}$	$p(U), \exists X(p(X) \rightarrow \forall Y p(Y))$
		\downarrow	all-right
3	$p(V)$	$\xrightarrow{\quad}$	$\forall Y p(Y), \exists X(p(X) \rightarrow \forall Y p(Y))$
		\downarrow	impl-right
2		$\xrightarrow{\quad}$	$p(V) \rightarrow \forall Y p(Y), \exists X(p(X) \rightarrow \forall Y p(Y))$
		\downarrow	ex-right
1		$\xrightarrow{\quad}$	$\exists X(p(X) \rightarrow \forall Y p(Y))$

Admissibility of Substitutions

Motivation

We want to have that

$$\forall XF \rightarrow F\sigma$$

$$F\sigma \rightarrow \exists XF$$

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 occurrence of X in the scope of that quantification

Sequent Calculus for Dynamic Logic

Cut rule

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

Sequent Calculus for Dynamic Logic

Cut rule

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

Equality rules

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

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

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:

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

A Sequent Calculus for Dynamic Logic

Conditional rule

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

Reasoning about Loops

To prove

$[\underline{\text{while}}\ B\ \underline{\text{do}}\ \textit{body}\ \underline{\text{od}}]\ F$

find an (arbitrary) formula \textit{Inv} such that

1. \textit{Inv} is true before execution of the loop
2. $\textit{Inv} \wedge B \rightarrow [\textit{body}]\textit{Inv}$ is true
3. $\textit{Inv} \wedge \neg B \rightarrow F$ is true

Note

\textit{Inv} is a loop invariant

Sequent Calculus for Dynamic Logic

Loop rule

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

Example

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

$$B: I < X$$

$$\alpha: Y := Y + 2 * I + 1; I := I + 1$$

Example

→ $[I := 0; Y := 0; \text{while } B \text{ do } \alpha \text{ od}] Y \doteq X * X$

→ $[\alpha_{\text{square}}] Y \doteq X * X$

$B: I < X$

$\alpha: Y := Y + 2 * I + 1; I := I + 1$

Example

- $[I := 0] [Y := 0] [\text{while } B \text{ do } \alpha \text{ od}] Y \doteq X * X$
- $[I := 0; Y := 0; \text{while } B \text{ do } \alpha \text{ od}] Y \doteq X * X$
- $[\alpha_{\text{square}}] Y \doteq X * X$

$B:$ $I < X$

$\alpha:$ $Y := Y + 2 * I + 1; I := I + 1$

Example

$I \doteq 0 \quad \rightarrow \quad [Y := 0] [\text{while } B \text{ do } \alpha \text{ od}] Y \doteq X * X$
 $\rightarrow \quad [I := 0] [Y := 0] [\text{while } B \text{ do } \alpha \text{ od}] Y \doteq X * X$
 $\rightarrow \quad [I := 0; Y := 0; \text{while } B \text{ do } \alpha \text{ od}] Y \doteq X * X$
 $\rightarrow \quad [\alpha_{\text{square}}] Y \doteq X * X$

$B: \quad I < X$

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

Example

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

$B: \ I < X$

$\alpha: \ Y := Y + 2 * I + 1; \ I := I + 1$

Example

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

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

$I \doteq 0, Y \doteq 0 \rightarrow$ **[while B do α od]** $Y \doteq X * X$

$I \doteq 0 \rightarrow$ $[Y := 0]$ **[while B do α od]** $Y \doteq X * X$

\rightarrow $[I := 0]$ $[Y := 0]$ **[while B do α od]** $Y \doteq X * X$

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

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

$B: I < X$

$\alpha: Y := Y + 2 * I + 1; I := I + 1$

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)

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

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

Example

Left branch (pre-condition implies invariant)

$$I \doteq 0, Y \doteq 0 \rightarrow 0 \leq X$$

$$I \doteq 0, Y \doteq 0 \rightarrow Y \doteq 0 * 0$$

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

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

Example

Middle branch (invariant is indeed invariant)

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

Example

Middle branch (invariant is indeed invariant)

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

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

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)

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

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

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

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

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

Example

Middle branch (invariant is indeed invariant)

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

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

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

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

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

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

$$\text{Inv}, B \rightarrow [\alpha] \text{Inv}$$

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)

$$\begin{array}{l} I \leq X, Y \doteq I * I, \neg (I < X) \quad \rightarrow \quad Y \doteq X * X \\ \text{Inv} \wedge \neg B \quad \rightarrow \quad Q \end{array}$$

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)

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

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

$$\begin{array}{l} X \doteq A, Y \doteq B, Z \doteq 0 \quad \rightarrow \quad \textit{Inv} \\ \textit{Inv}, \neg B \doteq 0 \quad \rightarrow \quad [\alpha_{body}] \textit{Inv} \\ \textit{Inv}, B \doteq 0 \quad \rightarrow \quad Z \doteq X \end{array}$$

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

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

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

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**

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

$$\Gamma \rightarrow \Delta$$

N not occurring in Γ, Δ

N not occurring in any program in F

Induction Rule: Example

$\rightarrow \text{even}(2 * 0), \text{even}(2 * 3)$

$\text{even}(2 * N) \rightarrow \text{even}(2 * (N + 1)), \text{even}(2 * 3)$

$\forall N (\text{even}(2 * N)) \rightarrow \text{even}(2 * 3)$

$\rightarrow \text{even}(2 * 3)$

Loop Unwind Rule

Rule

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

Loop Unwind Rule

Rule

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

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

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

Induction hypothesis

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

Admissibility of Substitutions Revisited

Problem

Previous definition of admissibility
is not sufficient if formulas contain programs

Example

$$F \equiv J \dot{=} K \rightarrow [I := 0] (J \dot{=} K) \quad \text{valid}$$

Admissibility of Substitutions Revisited

Problem

Previous definition of admissibility
is not sufficient if formulas contain programs

Example

$$F \equiv J \dot{=} K \rightarrow [I := 0] (J \dot{=} K) \quad \text{valid}$$

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

Admissibility of Substitutions Revisited

Problem

Previous definition of admissibility
is not sufficient if formulas contain programs

Example

$$\begin{array}{lll} F & \equiv & J \dot{=} K \rightarrow [I := 0] (J \dot{=} K) & \text{valid} \\ F\{I \leftarrow J\} & \equiv & J \dot{=} K \rightarrow [J := 0] (J \dot{=} K) & \text{not valid} \\ F\{J \leftarrow I\} & \equiv & I \dot{=} K \rightarrow [I := 0] (I \dot{=} K) & \text{not valid} \end{array}$$

Admissibility of Substitutions Revisited

Problem

Previous definition of admissibility
is not sufficient if formulas contain programs

Example

F	\equiv	$J \dot{=} K \rightarrow [I := 0](J \dot{=} K)$	valid
$F\{I \leftarrow J\}$	\equiv	$J \dot{=} K \rightarrow [J := 0](J \dot{=} K)$	not valid
$F\{J \leftarrow I\}$	\equiv	$I \dot{=} K \rightarrow [I := 0](I \dot{=} K)$	not valid
$F\{I \leftarrow 1\}$	\equiv	$J \dot{=} K \rightarrow [1 := 0](J \dot{=} K)$	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 $Y := s$