



# Modelling Software-based Systems Lecture 5 Checking contracts with Event-B Master Informatique

Dominique Méry Telecom Nancy, Université de Lorraine

21 novembre 2024 dominique.mery@loria.fr

# General Summary

Programming by contract

Verification

3 Floyd to Hoare

# **Current Summary**

- 1 Programming by contract
- 2 Verification
- S Floyd to Hoare

# Verifying program correctness

## A program P satisfies a (pre,post) contract :

- P transforms a variable v from initial values  $v_0$  and produces a final value  $v_f: v_0 \xrightarrow{P} v_f$
- $\mathsf{v}_0$  satisfies  $\mathsf{pre} : \mathsf{pre}(v_0)$  and  $\mathsf{v}_f$  satisfies  $\mathsf{post} : \mathsf{post}(v_0, v_f)$
- $\bullet \ \operatorname{pre}(v_0) \wedge v_0 \stackrel{\mathsf{P}}{\longrightarrow} v_f \Rightarrow \operatorname{post}(v_0, v_f)$
- D est le domaine RTE de V

# Verifying program correctness

## A program P satisfies a (pre,post) contract :

- P transforms a variable v from initial values  $v_0$  and produces a final value  $v_f: v_0 \xrightarrow{P} v_f$
- $\mathsf{v}_0$  satisfies pre :  $\mathsf{pre}(v_0)$  and  $\mathsf{v}_f$  satisfies post :  $\mathsf{post}(v_0, v_f)$
- $\operatorname{pre}(v_0) \wedge v_0 \xrightarrow{\mathsf{P}} v_f \Rightarrow \operatorname{post}(v_0, v_f)$
- D est le domaine RTE de V

```
requires pre(v_0) ensures post(v_0, v_f) variables X
\begin{array}{c} \text{begin} \\ 0: P_0(v_0, v) \\ \text{instruction}_0 \\ \dots \\ i: P_i(v_0, v) \\ \dots \\ \text{instruction}_{f-1} \\ f: P_f(v_0, v) \\ \text{end} \end{array}
```

- $pre(v_0) \wedge v = v_0 \Rightarrow P_0(v_0, v)$
- $pre(v_0) \wedge P_f(v_0, v) \Rightarrow post(v_0, v)$
- For any pair of labels  $\ell, \ell'$  such that  $\ell \longrightarrow \ell'$ , one verifies that, pour any values  $v, v' \in \operatorname{MEMORY}$   $\left( \begin{array}{c} pre(v_0) \wedge P_\ell(v_0, v) \\ \wedge cond_{\ell, \ell'}(v) \wedge v' = f_{\ell, \ell'}(v) \end{array} \right),$   $\Rightarrow P_{\ell'}(v_0, v')$

## **Contracts - Verification Conditions**

```
 \begin{array}{|c|c|} \hline \text{contract P} \\ \text{variables v} \\ \text{requires } pre(v_0) \\ \text{ensures } post(v_0, v_f) \\ \hline \\ begin \\ 0: P_0(v_0, v) \\ S_0 \\ \dots \\ i: P_i(v_0, v) \\ \dots \\ S_{f-1} \\ f: P_f(v_0, v) \\ \text{end} \\ \hline \end{array}
```

## Contracts - Verification Conditions

contract P variables v requires  $pre(v_0)$ ensures  $post(v_0, v_f)$  Verification conditions are listed as follows:

- (initialisation)  $pre(v_0) \wedge v = v_0 \Rightarrow P_0(v_0, v)$
- (finalisation)  $pre(v_0) \wedge P_f(v_0, v) \Rightarrow post(v_0, v)$
- (induction) For each labels pair  $\ell, \ell'$ such that  $\ell \longrightarrow \ell'$ , one checks that, for any value  $v, v' \in MEMORY$

$$\left(\begin{array}{c} pre(v_0) \wedge P_{\ell}(v_0, v)) \\ \wedge cond_{\ell, \ell'}(v) \wedge v' = f_{\ell, \ell'}(v) \end{array}\right),$$

$$\Rightarrow P_{\ell'}(v_0, v')$$

Three kinds of verification conditions should be checked and we justify the method in the full version..

## From PAP to Rodin ...

## From PAP to Rodin ...

```
MACHINE M
SEES CO
VARIABLES
  v, pc
INVARIANTS
   \mathsf{typing}: v \in D
  control : pc \in L
  \mathsf{at}\ell: pc = \ell \Rightarrow P_\ell(v0, v)
th1: pre(v_0) \wedge v = v_0 \Rightarrow P_0(v_0, v)
th2: pre(v_0) \wedge P_f(v_0, v)
                  \Rightarrow post(v_0, v)
END
END
```

## From PAP to Rodin ...

```
MACHINE M
SEES CO
VARIABLES
  v, pc
INVARIANTS
   \mathsf{typing}: v \in D
   control : pc \in L
  \mathsf{at}\ell: pc = \ell \Rightarrow P_\ell(v0, v)
th1: pre(v_0) \wedge v = v_0 \Rightarrow P_0(v_0, v)
th2: pre(v_0) \wedge P_f(v_0, v)
                  \Rightarrow post(v_0, v)
END
END
```

```
MACHINE M
EVENTS
INITIALISATION
BEGIN
(pc, v) : | \begin{pmatrix} pc' = l0 \land v' = v0 \\ \land pre(v0) \end{pmatrix}
END
e(\ell, \ell')
  WHEN
     pc = \ell
     cond_{\ell,\ell'}(v)
  THEN
     pc := \ell'
     v := f_{\ell,\ell'}(v)
  END
END
```

# Technical problems for students

## (Induction Principle (I))

A property S(z0,z) is a safety for an annotated program P if, and only if, there exists a property I(z0,z) satisfying :

- $1 \forall z0, z \in \mathsf{L} \times \mathsf{D}.init(z0) \land z = z0 \Rightarrow I(z0, z)$
- $2 \ \forall z0,z,z' \in \mathsf{L} \times \mathsf{D}.init(z0) \wedge I(z0,z) \wedge (z \underset{P}{\longrightarrow} z') \Rightarrow I(z0,z')$
- $3 \ \forall z0,z \in \mathsf{L} \times \mathsf{D}.init(z0) \wedge I(z0,z) \Rightarrow S(z0,z)$

## (Induction Principle (II))

A property  $S(\ell 0,x0,\ell,x)$  is a safety property for an annotated program P if, and only if, there exists a property  $I(\ell 0,x0,\ell,x)$  satisfying :

- $\bullet \forall \ell 0, \in \mathsf{L}, x 0 \in \mathsf{D}.\ell 0 \in \mathsf{L}0 \land pre(x 0) \land x = x 0 \land pc = \ell 0 \Rightarrow J(\ell 0, x 0, \ell, x)$
- 2  $\forall \ell, \ell' \in \mathsf{L}, x, x0 \in \mathsf{D}.\ell0 \in \mathsf{L0} \land pre(x0) \land J(\ell0, x0, \ell, x) \land BA(e(\ell, \ell'), )(\ell, x, \ell', x') \Rightarrow J(\ell0, x0, \ell', x')$
- 3  $\forall \ell 0, \ell \in \mathsf{L}, x0, x \in \mathsf{D}.pre(x0) \land \ell 0 \in \mathsf{L}0 \land J(\ell 0, x0, \ell, x) \Rightarrow S(\ell 0, x0, \ell, x)$

## Technical problems for students

## (Induction Principle (II))

A property  $S(\ell 0,x0,\ell,x)$  is a safety property for an annotated program P if, and only if, there exists a property  $I(\ell 0,x0,\ell,x)$  satisfying :

- $\bullet \forall \ell 0, \in \mathsf{L}, x 0 \in \mathsf{D}.\ell 0 \in \mathsf{L}0 \land pre(x 0) \land x = x 0 \land pc = \ell 0 \Rightarrow J(\ell 0, x 0, \ell, x)$
- 2  $\forall \ell, \ell' \in \mathsf{L}, x, x0 \in \mathsf{D}.\ell0 \in \mathsf{L}0 \land pre(x0) \land J(\ell0, x0, \ell, x) \land BA(e(\ell, \ell'), )(\ell, x, \ell', x') \Rightarrow J(\ell0, x0, \ell', x')$

## (Induction Principle (III))

A property  $S(x0, \ell, x)$  is a safety for an annotated program P with one entry point if, and only if, there exists a property  $I(x0, \ell, x)$  satisfying :

- $\forall x 0 \in \mathsf{D}.pre(x0) \land x = x0 \land \ell = \ell 0 \Rightarrow J(x0, \ell, x)$
- ②  $\forall \ell, \ell' \in \mathsf{L}, x, x0 \in \mathsf{D}.pre(x0) \land J(x0, \ell, x) \land BA(e(\ell, \ell'), )(\ell, x, \ell', x') \Rightarrow J(x0, \ell', x')$

## Soundness of the translation

## (Soundness of the method)

If the initialisation init, the generalisation gen and the step induction are proved to be correct by the Rodin platform, the property  $S(x0,\ell,x)$  is a correct safety property for the program P. In particular, one can handle the partial correctness and the run time error safety properties.

## Soundness of the translation

## (Soundness of the method)

If the initialisation init, the generalisation gen and the step induction are proved to be correct by the Rodin platform, the property  $S(x0,\ell,x)$  is a correct safety property for the program P. In particular, one can handle the partial correctness and the run time error safety properties.

- Contract and verification conditions are translated into Event-B and are discharged by Rodin and its provers.
- Verification conditions are derived from Floyd's method.
- Annotation as assertion

S

```
contract SIMPLE variables \mathbf{x} requires x_0 \in \mathbb{N} ensures x_f = 0 begin \ell_0 : \{0 \leq x \leq x_0 \wedge x_0 \in \mathbb{N}\} while 0 < \mathbf{x} \ \mathbf{do} \ell_1 : \{0 < x \wedge x \leq x_0 \wedge x_0 \in \mathbb{N}\} \mathbf{x} := \mathbf{x} - 1; od \ell_2 : \{x = 0\} \mathrm{end}
```

#### INVARIANTS

```
inv1: x \in \mathbb{N}
inv2: l \in L
inv3: l = l0 \Rightarrow
0 \le x \land x \le x0 \land x0 \in \mathbb{N}
inv4: l = l1 \Rightarrow
0 < x \land x \le x0 \land x0 \in \mathbb{N}
inv5: l = l2 \Rightarrow x = 0
requires: x0 \in \mathbb{N} \land x = x0
\Rightarrow x = x0 \land x0 \in \mathbb{N}
ensures: x = 0 \land x = x0
\Rightarrow x = 0
```

```
\begin{aligned} & \text{Event } el0l2 \\ & \text{WHEN} \\ & grd1: l = l0 \\ & grd2: \neg (0 < x) \\ & \text{THEN} \\ & act1: l := l2 \end{aligned}
```

```
\label{eq:controller} \begin{split} & \text{Event } el1l0 \\ & \text{WHEN} \\ & grd1: l = l1 \\ & \text{THEN} \\ & act1: l := l0 \\ & act2: x := x-1 \end{split}
```

# **Current Summary**

- Programming by contract
- 2 Verification
- S Floyd to Hoare

# Summary

- Programming by contract
- 2 Verification
- S Floyd to Hoare

## Annotation of programs

```
\begin{array}{c} \ell: \{P_{\ell}(v)\} \\ cond_{\ell,\ell'}(v) \longrightarrow v := f_{\ell,\ell'}(v) \\ \ell': \{P_{\ell'}(v)\} \end{array}
```

```
\begin{aligned} e(\ell,\ell') & \text{WHEN} \\ c &= \ell \\ cond_{\ell,\ell'}(v) & \text{THEN} \\ c &= \ell' \\ v &:= f_{\ell,\ell'}(v) \\ \text{END} & \end{aligned}
```

$$\ell_0^1 : \{x = 0\} \\ x := x + 1; \\ \ell_0^1 : \{x = 1\}$$

- v is the state meory variable or list of memory variables; v includes the local variables and the results variables.
- c is a new variable which is modelling the control flow and its type is LOCATIONS.
- $e(\ell,\ell')$  is simulating the computation flow starting from  $\ell$  and moving to  $\ell'$ ; v is updated.

## From annotations to invariants

#### INVARIANTS

```
inv_i : c \in \text{LOCATIONS}
inv_j : v \in Type
\vdots
inv_k : c = \ell \Rightarrow P_{\ell}(v)
inv_m : c = \ell' \Rightarrow P_{\ell'}(v)
\vdots
th_n : A(c, v)
```

- Type is the type of the variables v and is a set of possible values defined in the context C.
- The annotation is giving us for free the conditions satisfied by v when the control is in \(\ell\), (resp. in \(\ell'\)).
- A(c, v) is a safety property that we are supposed to check and the case of Event-B, it is a theorem.

## Partial correctness using Event-B models

For each pair of successive labels  $\ell,\ell'$ , the three statements are equivalent :

- $P_{\ell}(v) \wedge cond_{\ell,\ell'}(v) \wedge v' = f_{\ell,\ell'}(v) \Rightarrow P_{\ell'}(v')$
- $I(c,v) \wedge c = \ell \wedge cond_{\ell,\ell'}(v) \wedge c' = \ell' \wedge v' = f_{\ell,\ell'}(v) \Rightarrow (c' = \ell' \Rightarrow P_{\ell'}(v'))$
- $I(c,v) \wedge BA(e(\ell,\ell'))(c,v,c',v') \Rightarrow (c'=\ell' \Rightarrow P_{\ell'}(v'))$

#### L

et AA an annotated algorithm with precondition  $\operatorname{pre}(AA)(v)$  and postcondition  $\operatorname{post}(AA)(v_0,v)$ . Let the context C and the machine M generated from AA using the construction given previously. We assume that  $\ell_0$  is the first label and  $\ell_e$  is the last label. We add the following safety properties in the machine M:

- $c = \ell_0 \land \operatorname{pre}(AA)(v) \Rightarrow P_{\ell_0}(v)$
- $c = \ell_e \Rightarrow (P_{\ell_e}(v) \Rightarrow \mathsf{post}(AA)(v_0, v)$

If proof obligations are discharged, then the annotated algorithm AA is partially correct with respect to ist pre/post specification.

Master Informatique 2024-2025 (Dominique Méry) 15/17.

# **Current Summary**

- Programming by contract
- 2 Verification
- 3 Floyd to Hoare

 $\bullet \ \forall x_f, x_0.\mathsf{pre}(x_0) \land x_0 \overset{\mathsf{P}}{\longrightarrow} x_f \Rightarrow \mathsf{post}(x_0, x_f)$ 

- $\bullet \ \forall x_f, x_0.\mathsf{pre}(x_0) \land x_0 \overset{\mathsf{P}}{\longrightarrow} x_f \Rightarrow \mathsf{post}(x_0, x_f)$
- $\forall x_f, x_0.\mathsf{pre}(x_0) \Rightarrow x_0 \xrightarrow{\mathsf{P}} x_f \Rightarrow \mathsf{post}(x_0, x_f)$

- $\bullet \ \forall x_f, x_0.\mathsf{pre}(x_0) \land x_0 \overset{\mathsf{P}}{\longrightarrow} x_f \Rightarrow \mathsf{post}(x_0, x_f)$
- $\forall x_f, x_0.\mathsf{pre}(x_0) \Rightarrow x_0 \xrightarrow{\mathsf{P}} x_f \Rightarrow \mathsf{post}(x_0, x_f)$
- $\bullet \ \forall x_f, x_0.\mathsf{pre}(x_0) \Rightarrow x_0 \overset{\mathsf{P}}{\longrightarrow} x_f \Rightarrow \mathsf{post}(x_0, x_f)$

- $\bullet \ \forall x_f, x_0.\mathsf{pre}(x_0) \land x_0 \overset{\mathsf{P}}{\longrightarrow} x_f \Rightarrow \mathsf{post}(x_0, x_f)$
- $\forall x_f, x_0.\mathsf{pre}(x_0) \Rightarrow x_0 \xrightarrow{\mathsf{P}} x_f \Rightarrow \mathsf{post}(x_0, x_f)$
- $\bullet \ \forall x_f, x_0.\mathsf{pre}(x_0) \Rightarrow x_0 \overset{\mathsf{P}}{\longrightarrow} x_f \Rightarrow \mathsf{post}(x_0, x_f)$
- $\forall x_0.\mathsf{pre}(x_0) \Rightarrow \forall x_f.x_0 \xrightarrow{\mathsf{P}} x_f \Rightarrow \mathsf{post}(x_0,x_f)$

- $\forall x_f, x_0.\mathsf{pre}(x_0) \land x_0 \xrightarrow{\mathsf{P}} x_f \Rightarrow \mathsf{post}(x_0, x_f)$
- $\forall x_f, x_0.\mathsf{pre}(x_0) \Rightarrow x_0 \xrightarrow{\mathsf{P}} x_f \Rightarrow \mathsf{post}(x_0, x_f)$
- $\forall x_f, x_0.\mathsf{pre}(x_0) \Rightarrow x_0 \overset{\mathsf{P}}{\longrightarrow} x_f \Rightarrow \mathsf{post}(x_0, x_f)$
- $\forall x_0.\mathsf{pre}(x_0) \Rightarrow \forall x_f.x_0 \overset{\mathsf{P}}{\longrightarrow} x_f \Rightarrow \mathsf{post}(x_0,x_f)$
- $\forall x_0.\mathsf{pre}(x_0) \Rightarrow [P]\mathsf{post}(x_0,x_f)$
- wlp calculus is introduced

- $\forall x_f, x_0.\operatorname{pre}(x_0) \land x_0 \xrightarrow{\mathsf{P}} x_f \Rightarrow \operatorname{post}(x_0, x_f)$
- $\forall x_f, x_0.\mathsf{pre}(x_0) \Rightarrow x_0 \xrightarrow{\mathsf{P}} x_f \Rightarrow \mathsf{post}(x_0, x_f)$
- $\forall x_f, x_0.\mathsf{pre}(x_0) \Rightarrow x_0 \xrightarrow{\mathsf{P}} x_f \Rightarrow \mathsf{post}(x_0, x_f)$
- $\forall x_0.\mathsf{pre}(x_0) \Rightarrow \forall x_f.x_0 \overset{\mathsf{P}}{\longrightarrow} x_f \Rightarrow \mathsf{post}(x_0,x_f)$
- $\forall x_0.\mathsf{pre}(x_0) \Rightarrow [P]\mathsf{post}(x_0,x_f)$
- wlp calculus is introduced
- $[x := e]P(x) = P[x \mapsto e]$

- $\forall x_f, x_0.\mathsf{pre}(x_0) \land x_0 \overset{\mathsf{P}}{\longrightarrow} x_f \Rightarrow \mathsf{post}(x_0, x_f)$
- $\forall x_f, x_0.\mathsf{pre}(x_0) \Rightarrow x_0 \xrightarrow{\mathsf{P}} x_f \Rightarrow \mathsf{post}(x_0, x_f)$
- $\forall x_f, x_0.\mathsf{pre}(x_0) \Rightarrow x_0 \overset{\mathsf{P}}{\longrightarrow} x_f \Rightarrow \mathsf{post}(x_0, x_f)$
- $\forall x_0.\mathsf{pre}(x_0) \Rightarrow \forall x_f.x_0 \overset{\mathsf{P}}{\longrightarrow} x_f \Rightarrow \mathsf{post}(x_0,x_f)$
- $\forall x_0.\mathsf{pre}(x_0) \Rightarrow [P]\mathsf{post}(x_0,x_f)$
- wlp calculus is introduced
- $[x := e]P(x) = P[x \mapsto e]$
- [if b(x) then S1 else S2 ]  $P(x) = b(x) \wedge [S1]P(x) \vee \text{ not } b(x)$  [S2] P(x)

- $\forall x_f, x_0.\mathsf{pre}(x_0) \land x_0 \overset{\mathsf{P}}{\longrightarrow} x_f \Rightarrow \mathsf{post}(x_0, x_f)$
- $\forall x_f, x_0.\mathsf{pre}(x_0) \Rightarrow x_0 \xrightarrow{\mathsf{P}} x_f \Rightarrow \mathsf{post}(x_0, x_f)$
- $\forall x_f, x_0.\mathsf{pre}(x_0) \Rightarrow x_0 \xrightarrow{\mathsf{P}} x_f \Rightarrow \mathsf{post}(x_0, x_f)$
- $\forall x_0.\mathsf{pre}(x_0) \Rightarrow \forall x_f.x_0 \overset{\mathsf{P}}{\longrightarrow} x_f \Rightarrow \mathsf{post}(x_0,x_f)$
- $\forall x_0.\mathsf{pre}(x_0) \Rightarrow [P]\mathsf{post}(x_0,x_f)$
- wlp calculus is introduced
- $[x := e]P(x) = P[x \mapsto e]$
- [if b(x) then S1 else S2 ]  $P(x) = b(x) \wedge [S1]P(x) \vee$  not b(x) [S2]P(x)
- Frama-c uses the HOARE logic for defining the verification conditions as R. Leino in DAFNY.

- $\forall x_f, x_0.\mathsf{pre}(x_0) \land x_0 \overset{\mathsf{P}}{\longrightarrow} x_f \Rightarrow \mathsf{post}(x_0, x_f)$
- $\forall x_f, x_0.\mathsf{pre}(x_0) \Rightarrow x_0 \xrightarrow{\mathsf{P}} x_f \Rightarrow \mathsf{post}(x_0, x_f)$
- $\forall x_f, x_0.\mathsf{pre}(x_0) \Rightarrow x_0 \xrightarrow{\mathsf{P}} x_f \Rightarrow \mathsf{post}(x_0, x_f)$
- $\forall x_0.\mathsf{pre}(x_0) \Rightarrow \forall x_f.x_0 \overset{\mathsf{P}}{\longrightarrow} x_f \Rightarrow \mathsf{post}(x_0,x_f)$
- $\forall x_0.\mathsf{pre}(x_0) \Rightarrow [P]\mathsf{post}(x_0,x_f)$
- wlp calculus is introduced
- $[x := e]P(x) = P[x \mapsto e]$
- [if b(x) then S1 else S2 ] $P(x) = b(x) \land [S1]P(x) \lor \text{ not } b(x)$  [S2]P(x)
- Frama-c uses the HOARE logic for defining the verification conditions as R. Leino in DAFNY.
- Questions of termination require the wp calculus . . .