Modelling Software-based Systems
Lecture 3 Checking contracts with Event-B

Telecom Nancy IL

Dominique Méry
Telecom Nancy,Université de Lorraine

10 avril 2025(12:10 A.M.)
dominique.mery@loria.fr

General Summary

1 Programming by contract

2 Verification

3 Floyd to Hoare

Current Summary

1. **Programming by contract**

2. Verification

3. 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 : $v_0 \xrightarrow{\text{P}} v_f$
- v_0 satisfies pre : $\text{pre}(v_0)$ and v_f satisfies post : $\text{post}(v_0, v_f)$
- $\text{pre}(v_0) \wedge v_0 \xrightarrow{\text{P}} v_f \Rightarrow \text{post}(v_0, v_f)$
- $\mathbb{D}$ est le domaine RTE de V

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 : $v_0 \xrightarrow{\text{P}} v_f$
- v_0 satisfies pre : $\text{pre}(v_0)$ and v_f satisfies post : $\text{post}(v_0, v_f)$
- $\text{pre}(v_0) \wedge v_0 \xrightarrow{\text{P}} v_f \Rightarrow \text{post}(v_0, v_f)$
- $\mathbb{D}$ est le domaine RTE de V

```
requires pre(v_0)
ensures post(v_0, v_f)
variables X
        begin
        0 : P_0(v_0, v)
        instruction_0
        ...
        i : P_i(v_0, v)
        ...
        instruction_{f-1}
        f : P_f(v_0, v)
        end
```

- $pre(v_0) \wedge v = v_0 \Rightarrow P_0(v_0, v)$
- $pre(v_0) \wedge P_f(v_0, v) \Rightarrow post(v_0, v)$
- For any pair of labels ℓ, ℓ' such that $\ell \longrightarrow \ell'$, one verifies that, pour any values $v, v' \in \text{MEMORY}$

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

```
contract P
variables v
requires pre(v₀)
ensures post(v₀, vf)
        ┌  begin
        │    0 : P₀(v₀, v)
        │    S₀
        │    . . .
        │    i : Pᵢ(v₀, v)
        │    . . .
        │    Sf−1
        │    f : Pf(v₀, v)
        └  end
```

contract P
variables v
requires $pre(v_0)$
ensures $post(v_0, v_f)$
$\quad$ begin
$\quad\quad 0 : P_0(v_0, v)$
$\quad\quad$ S$_0$
$\quad\quad \ldots$
$\quad\quad i : P_i(v_0, v)$
$\quad\quad \ldots$
$\quad\quad$ S$_{f-1}$
$\quad\quad f : P_f(v_0, v)$
$\quad$ end

Contracts - Verification Conditions

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 ℓ, ℓ'
  such that $\ell \longrightarrow \ell'$, one checks that,
  for any value $v, v' \in \mathrm{MEMORY}$
  $$\left(\begin{array}{l} (pre(v_0) \wedge P_\ell(v_0, v)) \\ \wedge cond_{\ell,\ell'}(v) \wedge v' = f_{\ell,\ell'}(v) \\ \Rightarrow P_{\ell'}(v_0, v') \end{array} \right),$$

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

```
contract P
variables v
requires pre(v_0)
ensures post(v_0, v_f)
    begin
    0 : P_0(v_0, v)
    S_0
    . . .
    i : P_i(v_0, v)
    . . .
    S_{f-1}
    f : P_f(v_0, v)
    end
```

```
MACHINE M
SEES   C0
VARIABLES
   v, pc
INVARIANTS
   typing : v ∈ D
   control : pc ∈ L
   . . .
   atℓ : pc = ℓ ⇒ Pℓ(v0, v)
   . . .
th1 : pre(v₀) ∧ v = v₀ ⇒ P₀(v₀, v)
th2 : pre(v₀) ∧ Pf(v₀, v)
                ⇒ post(v₀, v)
. . .
END
. . .
END
```

$$\text{typing} : v \in D$$
$$\text{control} : pc \in L$$
$$\text{at}\ell : pc = \ell \Rightarrow P_\ell(v0, v)$$
$$th1 : pre(v_0) \land v = v_0 \Rightarrow P_0(v_0, v)$$
$$th2 : pre(v_0) \land P_f(v_0, v) \Rightarrow post(v_0, v)$$

From PAP to Rodin . . .

```
MACHINE M
SEES   C0
VARIABLES
   v, pc
INVARIANTS
   typing : v ∈ D
   control : pc ∈ L
   . . .
   atℓ : pc = ℓ ⇒ Pℓ(v0, v)
   . . .
th1 : pre(v₀) ∧ v = v₀ ⇒ P₀(v₀, v)
th2 : pre(v₀) ∧ P_f(v₀, v)
                ⇒ post(v₀, v)
. . .
END
. . .
END
```

```
MACHINE M
EVENTS
INITIALISATION
BEGIN
(pc, v) : | ( pc′ = l0 ∧ v′ = v0
             ∧pre(v0) )
END
. . .
e(ℓ, ℓ′)
   WHEN
      pc = ℓ
      cond_{ℓ,ℓ′}(v)
   THEN
      pc := ℓ′
      v := f_{ℓ,ℓ′}(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. $\forall z0, z \in \mathsf{L} \times \mathsf{D}.init(z0) \wedge z = z0 \Rightarrow I(z0, z)$
2. $\forall z0, z, z' \in \mathsf{L} \times \mathsf{D}.init(z0) \wedge I(z0, z) \wedge (z \underset{P}{\rightarrow} z') \Rightarrow I(z0, z')$
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 $S(\ell0, x0, \ell, x)$ is a safety property for an annotated program P if, and only if, there exists a property $I(\ell0, x0, \ell, x)$ satisfying :

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

Technical problems for students

(Induction Principle (II))

A property $S(\ell0, x0, \ell, x)$ is a safety property for an annotated program P if, and only if, there exists a property $I(\ell0, x0, \ell, x)$ satisfying :

1. $\forall \ell0, \in \mathsf{L}, x0 \in \mathsf{D}.\ell0 \in \mathsf{L0} \wedge pre(x0) \wedge x = x0 \wedge pc = \ell0 \Rightarrow J(\ell0, x0, \ell, x)$

2. $\forall \ell, \ell' \in \mathsf{L}, x, x0 \in \mathsf{D}.\ell0 \in \mathsf{L0} \wedge pre(x0) \wedge J(\ell0, x0, \ell, x) \wedge$
   $BA(e(\ell, \ell'),)(\ell, x, \ell', x') \Rightarrow J(\ell0, x0, \ell', x')$

3. $\forall \ell0, \ell \in \mathsf{L}, x0, x \in \mathsf{D}.pre(x0) \wedge \ell0 \in$
   $\mathsf{L0} \wedge J(\ell0, x0, \ell, x) \Rightarrow S(\ell0, x0, \ell, x)$

(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 :

1. $\forall x0 \in \mathsf{D}.pre(x0) \wedge x = x0 \wedge \ell = \ell0 \Rightarrow J(x0, \ell, x)$

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

3. $\forall \ell \in \mathsf{L}, x0, x \in \mathsf{D}.pre(x0) \wedge J(x0, \ell, x) \Rightarrow S(x0, \ell, x)$

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

A short example

s

contract SIMPLE
variables 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 $0 < x$ **do**
$\quad \ell_1 : \{0 < x \wedge x \leq x_0 \wedge x_0 \in \mathbb{N}\}$
$\quad x := x - 1;$
od
$\ell_2 : \{x = 0\}$ end

INVARIANTS
$inv1 : x \in \mathbb{N}$
$inv2 : l \in L$
$inv3 : l = l0 \Rightarrow$
$0 \leq x \wedge x \leq x0 \wedge x0 \in \mathbb{N}$
$inv4 : l = l1 \Rightarrow$
$0 < x \wedge x \leq x0 \wedge x0 \in \mathbb{N}$
$inv5 : l = l2 \Rightarrow x = 0$
$requires : x0 \in \mathbb{N} \wedge x = x0$
$\Rightarrow x = x0 \wedge x0 \in \mathbb{N}$
$ensures : x = 0 \wedge x = x0$
$\Rightarrow x = 0$

Event $el0l2$
WHEN
$\quad grd1 : l = l0$
$\quad grd2 : \neg(0 < x)$
THEN
$\quad act1 : l := l2$

Event $Init$
THEN
$\quad act1 : x := x0$
$\quad act2 : l := l0$

Event $el0l1$
WHEN
$\quad grd1 : l = l0$
$\quad grd2 : 0 < x$
THEN
$\quad act1 : l := l1$

Event $el1l0$
WHEN
$\quad grd1 : l = l1$
THEN
$\quad act1 : l := l0$
$\quad act2 : x := x - 1$

Current Summary

❶ Programming by contract

❷ Verification

❸ Floyd to Hoare

Summary

Annotation of programs

$$\ell : \{P_\ell(v)\}$$
$$cond_{\ell,\ell'}(v) \longrightarrow v := f_{\ell,\ell'}(v)$$
$$\ell' : \{P_{\ell'}(v)\}$$

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

```
e(ℓ, ℓ')
   WHEN
      c = ℓ
      cond_{ℓ,ℓ'}(v)
   THEN
      c := ℓ'
      v := f_{ℓ,ℓ'}(v)
   END
```

- 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 ℓ and moving to ℓ'; v is updated.

INVARIANTS

$inv_i : c \in \text{LOCATIONS}$

$inv_j : v \in Type$

$\ldots$

$inv_k : c = \ell \Rightarrow P_\ell(v)$

$inv_m : c = \ell' \Rightarrow P_{\ell'}(v)$

$\ldots$

$th_n : A(c, v)$

- $Type$ 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 ℓ, (resp. in ℓ').

- $A(c, v)$ 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 ℓ, ℓ', 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 **pre**$(AA)(v)$ and postcondition **post**$(AA)(v_0,v)$. Let the context C and the machine M generated from AA using the construction given previously. We assume that ℓ_0 is the first label and ℓ_e is the last label. We add the following safety properties in the machine M :

- $c = \ell_0 \wedge$ **pre**$(AA)(v) \Rightarrow P_{\ell_0}(v)$
- $c = \ell_e \Rightarrow (P_{\ell_e}(v) \Rightarrow$ **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.

Current Summary

1. Programming by contract

2. Verification

3. Floyd to Hoare

- $\forall x_f, x_0.\mathsf{pre}(x_0) \wedge x_0 \xrightarrow{\ \mathsf{P}\ } x_f \Rightarrow \mathsf{post}(x_0, x_f)$

From Floyd to Hoare

- $\forall x_f, x_0.\mathsf{pre}(x_0) \wedge x_0 \xrightarrow{\mathsf{P}} x_f \Rightarrow \mathsf{post}(x_0, x_f)$
- $\forall x_f, x_0.\mathsf{pre}(x_0) \Rightarrow x_0 \xrightarrow{\mathsf{P}} x_f \Rightarrow \mathsf{post}(x_0, x_f)$

From Floyd to Hoare

- $\forall x_f, x_0.\mathsf{pre}(x_0) \wedge x_0 \xrightarrow{\mathsf{P}} x_f \Rightarrow \mathsf{post}(x_0, x_f)$
- $\forall x_f, x_0.\mathsf{pre}(x_0) \Rightarrow x_0 \xrightarrow{\mathsf{P}} x_f \Rightarrow \mathsf{post}(x_0, x_f)$
- $\forall x_f, x_0.\mathsf{pre}(x_0) \Rightarrow x_0 \xrightarrow{\mathsf{P}} x_f \Rightarrow \mathsf{post}(x_0, x_f)$

- $\forall x_f, x_0.\mathsf{pre}(x_0) \wedge x_0 \xrightarrow{\mathsf{P}} x_f \Rightarrow \mathsf{post}(x_0, x_f)$

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

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

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

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

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

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

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

- $\forall x_0.\mathsf{pre}(x_0) \Rightarrow [P]\mathsf{post}(x_0, x_f)$

- wlp calculus is introduced

From Floyd to Hoare

- $\forall x_f, x_0.\mathsf{pre}(x_0) \wedge x_0 \xrightarrow{\mathsf{P}} x_f \Rightarrow \mathsf{post}(x_0, x_f)$

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

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

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

- $\forall x_0.\mathsf{pre}(x_0) \Rightarrow [P]\mathsf{post}(x_0, x_f)$

- wlp calculus is introduced

- $[x := e]P(x) = P[x \mapsto e]$

From Floyd to Hoare

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

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

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

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

- $\forall x_0.\mathsf{pre}(x_0) \Rightarrow [P]\mathsf{post}(x_0, x_f)$

- wlp calculus is introduced

- $[x := e]P(x) = P[x \mapsto e]$

- $[\text{if } b(x) \text{ then } S1 \text{ else } S2\,]P(x) = b(x) \land [S1]P(x) \lor \text{ not } b(x)\ [S2]P(x)$

From Floyd to Hoare

- $\forall x_f, x_0.\mathsf{pre}(x_0) \wedge x_0 \xrightarrow{\mathsf{P}} x_f \Rightarrow \mathsf{post}(x_0, x_f)$

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

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

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

- $\forall x_0.\mathsf{pre}(x_0) \Rightarrow [P]\mathsf{post}(x_0, x_f)$

- 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.

From Floyd to Hoare

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

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

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

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

- $\forall x_0.\mathsf{pre}(x_0) \Rightarrow [P]\mathsf{post}(x_0, x_f)$

- wlp calculus is introduced

- $[x := e]P(x) = P[x \mapsto e]$

- [if $b(x)$ then $S1$ else $S2$]$P(x) = b(x) \land [S1]P(x) \lor$ 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 ...