17. Rigorous reviews

The outline of this part:

- The principle of rigorous review
- Properties
- Review Task Tree
- Property reviews
- Important points
17.1 The principle of rigorous review

Definition 17.1 A property $P$ of a specification $S$ is called a **critical property** if it satisfies the following condition:

$S$ is not correct if $P$ is not satisfied by $S$.

Definition 17.2. A specification $S$ is not correct means that $S$ is either inconsistent or invalid.
Definition 17.3 A specification is said to be internally consistent if and only if there is no contradiction with the semantics of the SOFL language in the specification.

Definition 17.4 A specification is valid if and only if it satisfies the user's requirements.
The principle of the rigorous review technique:

Let $P_1, P_2, \ldots, P_n$ be critical properties of specification $S$. Then, the rigorous review aims to detect faults that prevents $S$ from satisfying one of those properties.
The procedure for rigorous review:

Step 1: Derive all the critical properties from the specification to be reviewed.

Step 2: Construct a Review Task Tree (RTT) for each property.

Step 3: Review each RTT to detect potential faults.

Step 4: Analyze the review results.
17.2 Properties

- Internal consistency of process
- Satisfiability of process
- Invariant-conformance consistency
- Internal consistency of CDFD
module M;
type
  UsableInt = int;
inv          I;
behav CDFD_1;
process P(a: T_1) b: T_2
  ext wr x: T_3
  rd y: T_4
  wr z: T_5
  pre Q_1
  post Q_2
end_process;
...
end_module;
Notation:

- Input(P): the set of all input data flow variables of process P
- Output(P): the set of all output data flow variables of process P
- WR(P): the set of all writable (wr) external variables (including both decorated and undecorated variables) of process P
- RD(P): the set of all readable (rd) external variables of process P
- Variables(C): the set of all free variables occurring in condition C
Definition 17.5 A process $P$ is internally consistent if the following conditions hold:

1. \(\forall v: \text{Output}(P) \ | \ v \notin \text{Variables}(\text{pre}_P)\)

2. \(v \in \text{union}(\text{Variables}(\text{pre}_P), \text{Variables}(\text{post}_P)) \Rightarrow v \in \text{union}(	ext{Input}(P), \text{Output}(P), \text{WR}(P), \text{RD}(P))\)
For example, let process P be defined as follows:

```
process P(a: UsableInt) b: UsableInt
  ext wr x: UsableInt
    rd y: UsableInt
    wr z: UsableInt
  pre a > 0 and y > 0
  post x = a + y and b > x - a and z = ~z + a
end_process
```
Then,

Input(P) = \{a\}
Output(P) = \{b\}
WR(P) = \{x, z, \sim x, \sim z\}
RD(P) = \{y\}
Variables(pre_P) = \{a, y\}
Variables(post_P) = \{x, a, y, b, z, \sim z\}

Obviously, process P satisfies the two conditions necessary for being an internally consistent process.
Definition 17.6 Let a type invariant I be defined as
\[ \exists [x_1: T_1, x_2: T_2, \ldots, x_n: T_n] | Q(x_1, x_2, \ldots, x_n). \]
Then, a process P and invariant I are consistent if and only if the following two conditions hold.
(1) (pre\_P(y\_1, y\_2, ..., y\_m) and
(exists[x\_2: T\_2, ..., x\_n: T\_n] |
  Q(x\_1, x\_2, ..., x\_n)[y\_1/x\_1]) and
(exists[x\_1: T\_1, x\_3: T\_3, ..., x\_n: T\_n] |
  Q(x\_1, x\_2, x\_3, ..., x\_n)[y\_2/x\_2]) and
... and
(exists[x\_1: T\_1, x\_2: T\_3, ..., x\_n-1: T\_n-1] |
  Q(x\_1, x\_2, x\_3, ..., x\_n-1, x\_n)[y\_n/x\_n])) <> false

(2) (pre\_P(y\_1, y\_2, ..., y\_m) and
post\_P(z\_1, z\_2, ..., z\_w) and
(exists[x\_2: T\_2, ..., x\_n: T\_n] |
  Q(x\_1, x\_2, ..., x\_n)[y\_1/x\_1]) and
(exists[x\_1: T\_1, x\_3: T\_3, ..., x\_n: T\_n] |
  Q(x\_1, x\_2, x\_3, ..., x\_n)[y\_2/x\_2]) and
... and
(exists[x\_1: T\_1, x\_2: T\_3, ..., x\_n-1: T\_n-1] |
  Q(x\_1, x\_2, x\_3, ..., x\_n-1, x\_n)[y\_n/x\_n])) <> false
where \( n \geq 1 \); \( y_1, y_2, \ldots, y_m \) are the variables of types \( T_1, T_2, \ldots, T_m \) \( (m \leq n) \), respectively; and likewise \( z_1, z_2, \ldots, z_w \) are the variables of \( T_1, T_2, \ldots, T_m \) \( (w \leq n) \), respectively (assuming that \( x_i \) \( (i = 1 \ldots n) \) are all different from \( y_j \) \( (j=1..m) \) and \( z_k \) \( (k=1..w) \)).
For example, let an invariant on type UsableInt be

\[ \forall [i: \text{UsableInt}] \mid i \leq 10000 \]

Then the conditions for process P given previously to be consistent with this invariant is

1. \((a > 0 \text{ and } y > 0) \text{ and } a \leq 10000 \text{ and } y \leq 10000\) \(\not\equiv\) false

2. \((a > 0 \text{ and } y > 0) \text{ and } (x = a + y \text{ and } b > x - a \text{ and } z = \sim z + a) \) \text{ and } \ a \leq 10000 \text{ and } y \leq 10000 \text{ and } x \leq 10000 \text{ and } b \leq 10000 \text{ and } z \leq 10000 \text{ and } \sim z \leq 10000\) \(\not\equiv\) false
Definition 17.7 The satisfiability of process P is defined as:

\[
\text{forall}\{a, y, \sim x, \sim z: \text{UsableInt}\} \mid
(p_{\text{pre}}(a, y)[\sim x/x, \sim z/z] \Rightarrow
\text{exists}\{b, x, z: \text{UsableInt}\} \mid
\text{post}_{\text{P}}(a, b, x, \sim x, y, \sim z, z))
\]
For example, the condition for process $P$ given previously to be satisfiable is:

$$\forall [a, y, \sim x, \sim z: \text{UsableInt}] \mid a > 0 \text{ and } y > 0 \Rightarrow \exists [b, x, z: \text{UsableInt}] \mid x = a + y \text{ and } b > x - a \text{ and } z = \sim z + a$$
Definition 17.8 The internal consistency of a CDFD is a property that the output data flows of the CDFD can be generated based on its input data flows under the condition that the pre and postconditions of all the processes involved in the execution of the CDFD evaluate to true.

See the example next.
The conditions for the CDFD above to be internally consistent:

1. bound(a) and pre_P => bound(t) and post_P3
2. post_P => pre_P1
3. post_P1 => pre_P2
4. post_P1 and post_P2 => pre_P3
17.3 Review Task Tree

A review task tree is a graphical notation for systematically, logically, and hierarchically presenting the review task as its top node and all the necessary subtasks as the sub-nodes of the top node.
Example:

A is the property to be reviewed. This RTT indicates that the correctness of A is ensured by B, C, and D. The correctness of B is ensured by G and F. The correctness of C is ensured by E. The correctness of D is ensured by H and W in the order of left-right.
**Basic components of RTT**

<table>
<thead>
<tr>
<th>Diagram</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Diagram" /></td>
<td>Property A (or B) can hold (or holds) if all of its child properties hold.</td>
</tr>
<tr>
<td><img src="image2" alt="Diagram" /></td>
<td>Property A (or B) can hold (or holds) if one of its child properties hold.</td>
</tr>
<tr>
<td><img src="image3" alt="Diagram" /></td>
<td>Property A (or B) can hold (or holds) if all of its child properties hold in the order from left to right.</td>
</tr>
<tr>
<td><img src="image4" alt="Diagram" /></td>
<td>Property A (or B) can hold (or holds) if one of its child properties holds in the order from left to right.</td>
</tr>
<tr>
<td><img src="image5" alt="Diagram" /></td>
<td>Property A (or B) can hold (or holds) if its only child property holds.</td>
</tr>
<tr>
<td><img src="image6" alt="Diagram" /></td>
<td>Property B holds if its right child property holds under the assumption that its left child property holds.</td>
</tr>
<tr>
<td><img src="image7" alt="Diagram" /></td>
<td>Property A (or B) can hold (or holds). It is an atomic property that has no decomposition.</td>
</tr>
</tbody>
</table>

**Input connecting node,** meaning the connection will continue to reach the output connecting node with the same number.

**Output connecting node,** meaning it takes the connection from the input connecting node with the same number.
Definition 17.9 Let T be an RTT. A minimal cut set of T is a smallest combination of the leaf tasks that implies the top-level task of T.

For example,

The minimal cut set of this RTT is:
\[ A = \{G, E, H, W\} + \{F, E, H, W\} \]
Rules for constructing RTTs from properties

(1) can_hold(A and B)
(2) can_hold(A or B)

(3) can_hold(Atomic(x₁,x₂,...,xₙ))
(4) hold(A and B)

(5) hold(A or B)
(6) \text{hold}(\text{Atomic}(x_1, x_2, \ldots, x_n))

(7) \text{can\_hold}(\text{forall}[x: T] \mid Q(x))
(8) hold(forall\([x: T] \mid Q(x)\))

(9) can_hold(exists\([x: T] \mid Q(x)\))
(10) hold(exists[x: T] | Q(x))
17.4 Property Review

To review a property, we take the following steps:

Step 1: Construct a review task tree for the property to show the overall review task and its decomposition.

Step 2: Identify the minimal cut sets of the review task tree.

Step 3: Review all the leaf tasks to determine their truth (remember that a task in a review task tree is also a property).

Step 4: Determine whether the top-level task holds based on the review results of the minimal cut sets.
Review of Consistency Between Process and Invariant

inv
  forall[i: UsableInt] | i <= 10000

process P(a: UsableInt) b: UsableInt
  ext wr x: UsableInt
  rd y: UsableInt
  wr z: UsableInt
  pre  a > 0 and y > 0
  post x = a + y and b > x - a and z = ~z + a
end_process
An RTT for the property of precondition and invariant consistency

\[ a > 0 \text{ and } y > 0 \text{ and } a \leq 10000 \text{ and } y \leq 10000 \]

\[ \text{inter} \{ \{ a \mid a: \text{UsableInt} \& a > 0 \}, \{ y \mid y: \text{UsableInt} \& y > 0 \}, \{ a \mid a: \text{UsableInt} \& a \leq 10000 \}, \{ y \mid y: \text{UsableInt} \& y \leq 10000 \} \} \]
\[ <> \{ \} \]
Review of the internal consistency of process

P is consistent

for all \( v: \text{Output}(P) \) \( \mid \) \( v \) not in \( \text{Variables} \text{(pre}_P \text{)} \)

\( b \) not in \( \{a, y\} \)

\( v \) inset union(\( \text{Variables} \text{(pre}_P \text{)}, \text{Variables} \text{(post}_P \text{)} \) \( \Rightarrow \)
\( v \) inset union(\( \text{Input}(P), \text{Output}(P), \text{WR}(P), \text{RD}(P) \) \)

\( v \) inset \( \{a, b, x, y, z, \sim z\} \Rightarrow \)
\( v \) inset \( \{a, b, x, \sim x, y, z, \sim z\} \)
Review of process satisfiability

The proof obligation for the satisfiability of process P:

\[
\forall a, y, z: \text{UsableInt} \mid \\
\neg \text{pre}_P(a, y, z)[\sim x/x, \sim z/z] \text{ or } \\
\exists b, x, z: \text{UsableInt} \mid \text{post}_P(a, b, x, \sim x, y, \sim z, z)
\]
An RTT for reviewing the satisfiability of process P:

P is satisfiable

\[ \text{not pre}_P(a, y, z)[\sim x/x, \sim z/z] \]

\[ \{(a, \sim x, \sim z) | a, \sim x, \sim z: \text{UsableInt} \land \text{not } a > 0 \land y > 0\} \nsucc\{\} \]

Existentially quantified expression

\[ \text{post}_P(a, b, x, \sim x, y, \sim z, z) \]

\[ \{(b, x, z) | b, x, z: \text{UsableInt} \land x = a + y \land b > x - a \land z = \sim z + a\} \nsucc\{\} \]
Review of the internal consistency of CDFD

For example, to ensure the internal consistency of CDFD 1, we must ensure the following conditions:

(1) $d_1$ and $d_3$ are generated consistently.
(2) $P_3$ is satisfiable.
(3) $t$ occurs in the postcondition of $P_3$ (i.e., $t$ inset Variables(post$_P_3$)).
An RTT for reviewing the internal consistency of the CDFD

The CDFD of module M is internally consistent

Date flow t is generated consistently

- d1 and d2 are generated consistently
- P3 is satisfiable
  - t inset Variables(post_P3)
Definition 17.10 A constructive review of a property aims to establish the property, while a critical review aims to detect faults that prevent the establishment of the property.

For example, an RTT for a critical review of the invariant-conformance consistency discussed previously is given next.
\(\neg (a > 0 \text{ and } y > 0 \text{ and } a \leq 10000 \text{ and } y \leq 10000)\)

\[
\begin{align*}
\text{not } a > 0 & \quad \text{not } y > 0 & \quad \text{not } a \leq 10000 & \quad \text{not } y \leq 10000 \\
\{a \mid a: \text{UsableInt} \& \neg a > 0\} \nsubseteq \{\} & \quad \{y \mid y: \text{UsableInt} \& \neg y > 0\} \nsubseteq \{\} & \quad \{a \mid a: \text{UsableInt} \& \neg a \leq 10000\} \nsubseteq \{\} & \quad \{y \mid y: \text{UsableInt} \& \neg y \leq 10000\} \nsubseteq \{\}
\end{align*}
\]

\[\text{inter}(\{a \mid a: \text{UsableInt} \& \neg a > 0\}, \{y \mid y: \text{UsableInt} \& \neg y > 0\}, \{a \mid a: \text{UsableInt} \& \neg a \leq 10000\}, \{y \mid y: \text{UsableInt} \& \neg y \leq 10000\}) \nsubseteq \{\}\]
17.5 Important points

(1) The RTT approach offers several potential advantages over traditional review techniques:

- the review task tree can be constructed automatically based on a property derived from the specification.
- it allows the reviewer to focus on a manageable review task at a time.
- the review results of manageable tasks can be automatically utilized to determine the result of the overall review.
- the review task tree notation is comprehensible in conveying the ideas of a review, which will be useful and helpful when a review is explained to other people, such as the teammates or the managers involved in the same project.
(2) The review of a specific task in an RTT is usually done by human being. It is difficult to be automated. The reason for this is that only human being can make correct judgments in validation of the specification.

(3) The review can be done by generating test cases.

(4) The RTT approach can be effective if it is supported by a powerful software tool.

Tool Demonstration!
17.6 Exercises

1. Suppose the process P is defined as follows:
   process P(a: int) b: set of int
   ext wr x: set of int
   rd y: int
   pre card(x) <> 0
   post inter(x, b) = union({a, y}, ~x)
   end_process

   Build a review task tree for reviewing the internal consistency of process P, and determine whether the process is internally consistent.

2. Build a review task tree for both constructive review and critical review of the satisfiability of process P given above, and determine if process P is satisfiable.