# 7 Verification by Rules

Next we look at an alternative approach to verification. We will use a rule-based reasoning to establish properties of computations.

We define *invariance property* as a property that can be specified by a formula $\mathsf{G}\varphi$ where $\varphi$ is a propositional formula. An invariance property $p$ can be established by showing that

1. $p$ holds initially and that
2. $p$ is preserved during each step of all executions of the program.

Let the specifying formula for property $p$ be $\varphi_p$. Thus we have to show that, for all computations $\sigma$,

1. $(\mathcal{M}, \sigma) \models \varphi_p$ and
2. $(\mathcal{M}, \sigma) \models \mathsf{G}(\varphi_p \rightarrow \mathsf{X}\varphi_p)$.

Then, by an inductive argument, we have $(\mathcal{M}, \sigma) \models \mathsf{G}\varphi_p$.

In general, we will consider *verification conditions* defined as follows. Let $\mathcal{T}$ be a labelled transition system, $\varphi$ and $\psi$ be propositional formulas, and $\tau$ be a transition. The verification condition of $\varphi$ and $\psi$ relative to $\tau$ is the triple $\{\varphi\}\tau\{\psi\}$. We say that $\{\varphi\}\tau\{\psi\}$ holds if the following holds:

> for all states $s$ and $t$ such that $s\tau t$, if $\varphi$ is true at $s$, then $\psi$ is true at $t$.

Note that if we can establish the verification condition $\{\varphi\}\tau\{\varphi\}$ for all transitions $\tau$, then $\varphi$ is preserved during all computations. Thus to prove that $p$ is an invariance property, it suffices to show that

1. $\varphi_p$ is true at all initial states and
2. the verification condition $\{\varphi_p\}\tau\{\varphi_p\}$ holds for all transitions $\tau$.

## 7.1 Proving Invariance Properties

Let us fix a program $P$ and an invariance property $p$ with specifying formula $\varphi_p$. Let the labelled transition system associated with $P$ be $\mathcal{T} = (S, L, R_a : a \in A)$ and the corresponding linear temporal model be $\mathcal{M} = (S, R, V)$. Assume that there are finitely many initial states in $\mathcal{T}$ and that they can be defined by a propositional formula *init*, i.e., for every state $s$, $s$ is an initial state iff *init* is true at $s$ under the valuation $V$.

Consider the following rule.

$$\begin{array}{ll} B1 & (\mathcal{M}, \sigma) \models \mathit{init} \rightarrow \varphi_p \\ B2 & \{\varphi_p\}R_a\{\varphi_p\} \quad \text{for all } a \in A \\ \hline B3 & (\mathcal{M}, \sigma) \models \mathsf{G}\varphi_p \end{array}$$

Rule INV-B

where $\sigma$ is a computation. The basic invariance rule INV-B says, that the invariance formula $\varphi_p$ is true in $\mathcal{M}$ during computation $\sigma$ if $\varphi_p$ is true at every initial state and the verification condition $\{\varphi_p\}R_a\{\varphi_p\}$ holds for every transition $R_a$. Note that checking the verification condition involves checking all possible transitions between states, and it is not restricted to transitions between states *occurring in computations*.[2]

[2] In general, it is difficult to check which transitions between states occur in computations.

We can formulate Rule INV-B using temporal logic as follows.

$$\begin{array}{ll} B1 & (\mathcal{M},\sigma) \models \mathit{init} \rightarrow \varphi_p \\ B2 & (\mathcal{M},\pi) \models \mathsf{G}(\varphi_p \rightarrow \mathsf{X}\varphi_p) \quad \text{for every run } \pi \\ \hline B3 & (\mathcal{M},\sigma) \models \mathsf{G}\varphi_p \end{array}$$

Rule INV-B

An invariance formula $\varphi$ is called *inductive* if both B1 and B2 in INV-B hold for $\varphi$.

**Example 7.1** Consider the program MUX-SEM and the invariance formula $r_0 \vee r_1$. Define *init* as $at\_l_0 \wedge at\_m_0 \wedge r_1$. Thus *init* is true precisely at the initial state $(l_0, m_0, r = 1)$. Obviously, $(\mathcal{M},\sigma) \models \mathit{init} \rightarrow r_0 \vee r_1$ for every run (computation) $\sigma$. Let $\mathit{move}(l_i, l_j)$ denote that process 1 makes a move from location $l_i$ to $l_j$ by executing some instruction (and similarly for process 2). Using the definition of a semaphore, it is routine to check that the invariance conditions

$$\{r_0 \vee r_1\}\mathit{move}(l_i, l_j)\{r_0 \vee r_1\} \quad \text{and} \quad \{r_0 \vee r_1\}\mathit{move}(m_i, m_j)\{r_0 \vee r_1\}$$

hold for any $i, j \in \{0, 1, 2, 3, 4\}$ (recall that a process cannot perform **request**$(r)$ if $r = 0$). Thus we can apply INV-B to establish that the semaphore $r$ remains non-negative during all computations.

Invariance properties are monotone, i.e., the following rule is valid: for (propositional) formulas $\varphi$ and $\psi$,

$$\begin{array}{l} (\mathcal{M},\sigma) \models \mathsf{G}\varphi \\ \hline (\mathcal{M},\sigma) \models \mathsf{G}(\varphi \rightarrow \psi) \\ \hline (\mathcal{M},\sigma) \models \mathsf{G}\psi \end{array}$$

Rule MON-I, monotonicity of invariance

If we have two invariant formulas $\varphi$ and $\psi$, then their conjunction is invariant as well. That is, the following rule is valid:

$$\begin{array}{l} (\mathcal{M},\sigma) \models \mathsf{G}\varphi \\ \hline (\mathcal{M},\sigma) \models \mathsf{G}\psi \\ \hline (\mathcal{M},\sigma) \models \mathsf{G}(\varphi \wedge \psi) \end{array}$$

Rule CON-I, conjunction of invariances

A basic limitation to rule INV-B is when $\varphi_p$ is invariant but not inductive. This happens when $(\mathcal{M},\sigma) \models \mathsf{G}(\varphi \rightarrow \mathsf{X}\varphi)$ holds for all *computations* $\sigma$, but there is a *run* that falsifies the formula; see the following example.

**Example 7.2** We want to use rule INV-B to establish that the program MUX-SEM has the *mutual exclusion* property, i.e., for every computation $\sigma$, we have $(\mathcal{M},\sigma) \models \mathsf{G}\neg(at\_l_3 \wedge at\_m_3)$. But $B2$ requires that $\{\neg(at\_l_3 \wedge at\_m_3)\}\tau\{\neg(at\_l_3 \wedge at\_m_3)\}$ should hold for all transitions $\tau$. This fails, however, for the transition $\mathit{move}(l_2, l_3)$ from state $(l_2, m_3, r = 1)$ to state $(l_3, m_3, r = 0)$.[3]

## 7.2 Assertion Strengthening

Consider the following rule. For all computations $\sigma$,

$$\begin{array}{ll} I1 & (\mathcal{M},\pi) \models \mathsf{G}(\varphi \rightarrow \psi) \quad \text{for every run } \pi \\ I2 & (\mathcal{M},\sigma) \models \mathit{init} \rightarrow \varphi \\ I3 & \{\varphi\}R_a\{\varphi\} \quad \text{for all } a \in A \\ \hline I4 & (\mathcal{M},\sigma) \models \mathsf{G}\psi \end{array}$$

Rule INV, general invariance

[3] Of course, the state $(l_2, m_3, r = 1)$, hence the transition, cannot occur in any computation, but this is what we would like to establish.

Rule INV states that if $\varphi$ is inductive and $\varphi$ implies $\psi$ at each state in all runs, then $\psi$ is true at each state in all computations. Again, we can replace $I3$ by:

$$\text{for every run } \pi,\ (\mathcal{M}, \pi) \models \mathsf{G}(\varphi \rightarrow \mathsf{X}\varphi).$$

**Example 7.3** Assume we want to prove mutual exclusion $\psi : \neg(at\_l_3 \wedge at\_m_3)$ for MUX-SEM. Let $\rho$ be the propositional formula expressing that precisely one of $at\_m_3 \vee at\_m_4$, $at\_l_3 \vee at\_l_4$, or $r_1$ holds. **DEFINE** $\rho$**.** We define $\varphi$ as

$$(r_0 \vee r_1) \wedge \rho$$

We have to establish the following for every computation $\sigma$:

- $I1$: $(\mathcal{M}, \pi) \models \mathsf{G}(\varphi \rightarrow \psi)$ for every run $\pi$,
- $I2$: $(\mathcal{M}, \sigma) \models \mathit{init} \rightarrow \varphi$,
- $I3$: $(\mathcal{M}, \pi) \models \mathsf{G}(\varphi \rightarrow \mathsf{X}\varphi)$ for every run $\pi$,

so that we have

- $(\mathcal{M}, \sigma) \models \mathsf{G}\psi$

by rule INV.

Premise $I1$: The implication

$$[(r_0 \vee r_1) \wedge \rho] \rightarrow \neg(at\_l_3 \wedge at\_m_3)$$

is true at every state (since $(at\_l_3 \wedge at\_m_3)$ would contradict to $\rho$).

Premise $I2$: Obviously, the implication

$$(at\_m_0 \wedge at\_l_0 \wedge r_1) \rightarrow [(r_0 \vee r_1) \wedge \rho]$$

is true (since $\rho$ is true in the initial state).

Premise $I3$: We have to show that

$$(\mathcal{M}, \pi_i) \models (r_0 \vee r_1) \wedge \rho \text{ implies } (\mathcal{M}, \pi_{i+1}) \models (r_0 \vee r_1) \wedge \rho$$

for every run $\pi$ and $i \in \omega$. Consider the transitions of process 1. Transitions $move(l_0, l_1)$, $move(l_1, l_2)$ and $move(l_3, l_4)$ do not change the truth of $\rho$ (since these transitions do not change the value of $r$, $(at\_l_3 \vee at\_l_4)$ and $(at\_m_3 \vee at\_m_4)$). Transition $move(l_2, l_3)$ can be made only if $r = 1$. But $\rho$ implies that process 2 cannot be in its critical section in this case. Also, the value of $r$ after the transition is 0, by the definition of **request**$(r)$. It follows that $\rho$ is true after the transition. Finally, transition $move(l_4, l_0)$ changes $at\_l_3 \vee at\_l_4$ from true to false and $r_1$ from false to true (by the definition of **release**$(r)$), while the truth value of $at\_m_3 \vee at\_m_4$ is unchanged. Hence $\rho$ is "preserved" under all transitions of process 1. A similar argument shows that transitions of process 2 cannot change the truth value of $\rho$ either.

Thus $\varphi$ is true along all runs, and $\psi$ is true during all computations.

**Exercise 7.4** Show that MUX-SEM avoids deadlock, i.e., that at every moment at least one transition is enabled.

**Exercise 7.5** The **exchange**$(x, y)$ instruction is defined in Figure 8. Consider the program in Figure 9.

1. Show that MEX achieves mutual exclusion (i.e., only one of the $n$ processes can be in the critical section at any time).

   Hint: Let $\varphi \oplus \psi$ denote exclusive disjunction $(\neg\varphi \wedge \psi) \vee (\varphi \wedge \neg\psi)$ and $\bigwedge_i \gamma_i$ denote the conjunction $\gamma_0 \wedge \ldots \wedge \gamma_i \wedge \ldots \wedge \gamma_n$. Let $crit_i$ be an atomic proposition expressing that 'process $i$ is in its critical section'.

```
int temp

void exchange(int x, int y)
{
 temp = x;
 x = y;
 y = temp;
}
```

Figure 8: The **exchange** instruction

```
int n    /* number of processes */
int bolt = 0
int key[n] = 1    /* a variable for each process */

void P(int i)
{
 while(true){
  while(key[i]! = 0) exchange(key[i],bolt);
  critical;
  exchange(key[i],bolt);
  noncritical;
 }
}
```

Figure 9: MEX — mutual exclusion with **exchange**

- Show that bolt and all key[$i$] are either 0 or 1: that is,

$$(\text{bolt} = 0 \oplus \text{bolt} = 1) \wedge \bigwedge_i (\text{key}[i] = 0 \oplus \text{key}[i] = 1)$$

  is inductive.
- Show that

$$\text{bolt} + \sum_i \text{key}[i] = n$$

  is inductive.
- Show that 'if process $i$ is in its critical section, then key[$i$] = 0': that is,

$$crit_i \rightarrow \text{key}[i] = 0$$

  is inductive for every $i$.
- Using the above inductive formulas show that

$$crit_i \rightarrow (\text{bolt} = 1 \wedge \bigwedge_{j \neq i} \text{key}[j] = 1)$$

  is invariant for every $i$.
- Using the above invariant formulas show that

$$crit_i \rightarrow \bigwedge_{j \neq i} \neg crit_j$$

  is invariant for every $i$.

2. Show "progress", i.e., 'if there is no process in the critical section, then one process can enter the critical section'.

   Hint: Define the formula $enabled(crit_i)$ as $\text{key}_i = 0 \vee \text{bolt} = 0$ and show that

   $$(\bigwedge_i \neg crit_i) \rightarrow (\bigvee_i enabled(crit_i))$$

   is invariant.

**Exercise 7.6** Recall Peterson's solution, see Figure 10. There are two processes, identified by 0 and 1.

```
int t
int y_0 = 0
int y_1 = 0

void P(int i)
{
L1 : while(true){
L2 :  y_i = 1;
L3 :  t = i;
L4 :  while{t = i & y_{1-i} = 1}      /* loop/;
L5 :  critical;
L6 :  y_i = 0;
L7 :  noncritical;
 }
}
```

Figure 10: PET

1. Show that mutual exclusion holds for PET.

   Hint: Let $Lj_i$ be an atomic proposition for each $1 \leq j \leq 7$ and $0 \leq i \leq 1$ expressing that process $i$ is at location $Lj$.

   - Show that
     $$(y_0 = 0 \oplus y_0 = 1) \wedge (y_1 = 0 \oplus y_1 = 1) \wedge (t = 0 \oplus t = 1)$$
     is inductive.
   - Show that
     $$(L3_i \vee L4_i \vee L5_i \vee L6_i) \rightarrow y_i = 1$$
     is inductive for both $i = 0$ and $i = 1$.
   - Using the above inductive formulas show that
     $$((L5_i \vee L6_i) \wedge L4_{1-i}) \rightarrow (t = 1 - i \wedge y_i = 1)$$
     is invariant for $i = 0, 1$.
   - Using the above invariant formulas show that
     $$(L5_i \vee L6_i) \rightarrow \neg(L5_{1-i} \vee L6_{i-1})$$
     is invariant for both $i = 0$ and $i = 1$.

2. Show that PET avoids "livelock", i.e., at least one process can enter the critical section.

   Hint: Define the formula $enabled(L5_i)$ as $L4_i \wedge (t \neq i \vee y_{1-i} \neq 1)$ and show that

   $$(L4_0 \wedge L4_1) \rightarrow (enabled(L5_0) \vee enabled(L5_1))$$

   is invariant.