Lecture 8: Roadmap

- Modelling Peterson’s Algorithm by IOA
- Properties of Mutex Protocols
- Invariance (Safety) Properties
- Progress (Liveness) Properties
- Verification of Peterson’s Algorithm
- Other Properties
- N-Process Mutual Exclusion: Bakery
- The Bakery Algorithm
Modelling Peterson’s Algorithm by IOA

Consider Peterson’s mutual exclusion (m.e.) algorithm from first lecture. There, both Alice and Bob, whom we’ll call $P_1$ and $P_2$ now, have a shared variable turn that indicates who has priority to let the dog out (enter the critical section.) They each also have a flag that can be written by it owner, and read by any other process.

Hence, there are three (or $n + 1$) shared variables, turn and $\text{flag}_i$ for $i = 1, 2$. The value of turn is in $\{1, 2\}$ and $\text{flag}_i$ is in boolean.

We also model a wish to enter the critical section (let the dog out) as an input action, and the entry to the critical section as an output action. This is because both should be observable. similarly, exiting the critical section is an output action, and entering the idle state is an input action. The environment should guarantee that every entry to the CS is followed by an exit from it. The system should guarantee that every request to enter the CS is followed by entry to it, and every request to exit is followed by entry to the idle state.
Modelling Peterson’s Algorithm by IOA
Peterson's Algorithm for $P_i$

\begin{verbatim}
do forever
  idle
  $y_i := 1$
  $turn := 3 - i$
  wait until ($\neg y_{3-i} \lor turn = i$)
  $cs_i$
  $y_i := 0$
\end{verbatim}
Peterson's Algorithm for $P_i$

**do forever**

$I_i$: idle

$T^1_i$: $y_i := 1$

$T^2_i$: $\text{turn} := 3 - i$

$T^3_i$: **wait until** ($-y_{3-i} \lor \text{turn} = i$)

$C_i$: $cs_i$

$E_i$: $y_i := 0$

Hence, for every $i$, we have:

\[
\begin{align*}
\text{input}_i &= \{\text{Try}_i, \text{Exit}_i\} \\
\text{output}_i &= \{\text{CS}_i, \text{Idle}_i\} \\
\text{local}_i &= \{t_{12_i}, t_{23_i}, e_i\} \\
(\text{status}_i &= \{i_i, t^1_i, t^2_i, t^3_i, c_i, e_i\})
\end{align*}
\]

where $\text{status}_i$ is a variable indicating $i$'s state.
The Transitions of $P_i$

Try:
Effect:
if $status = i$ then $status := t^1$

$t12$:
Precondition:
$status = t^1$
Effect:
$y := 1$
$y := 1$

$t23$:
Precondition:
$status = t^2$
Effect:
$turn := 3 - i$
$turn := 3 - i$

CS:
Precondition:
$status = t^3 \land (\neg y \lor \neg t12)$
Effect:
$status = c$

Exit:
Effect:
if $status = c$ then $status := t23$

Idle:
Precondition:
$status = e$
Effect:
$y := 0$
$y := 0$
$y := 0$
$y := 0$

$status = i$
Properties of Mutex Protocols

First note that the “life cycle” of a process consists of

where we denote only self loops that we allow. I.e., we assume that once in the critical section, the environment will produce an Exit signal. We need to show that once in the trying region, the process will enter the critical region, and once in the exit region, the process will enter its Idle state.
Properties of Mutex Protocols

Let $\eta : s_0, a_1, s_1, \ldots$ be an execution of any mutual exclusion protocol. For a process $i$, we define:

\begin{align*}
s_k \models T_i & \iff \exists k' \leq k : a_{k'} = \text{Try}_i \land (\forall j : k' < j \leq k \rightarrow a_j) \\
s_k \models C_i & \iff \exists k' \leq k : a_{k'} = \text{CS}_i \land (\forall j : k' < j \leq k \rightarrow a_j) \\
s_k \models E_i & \iff \exists k' \leq k : a_{k'} = \text{Exit}_i \land (\forall j : k' < j \leq k \rightarrow a_j) \\
s_k \models I_i & \iff (\exists k' \leq k : a_{k'} = \text{Idle}_i \land (\forall j : k' < j \leq k \rightarrow a_j) \\
& \lor (\forall k' \leq k : a_{k'} \neq \text{Try}_i)
\end{align*}

All of the above are state properties—their truth depends on a single state, rather then on an execution.
Invariance Properties

We say that a state property $\varphi$ is invariant if it holds in all reachable states, that is, if for every execution $\eta$ as above, for all $i \geq 0$, $s_i \models \varphi$.

Invariance properties are usually proven by a computational induction, that is, we show that they hold at the initial state (of every computation) and then we show they are preserved by every transition.

A typical example of invariance properties is the Mutual Exclusion property, which established that

$$\forall i, j : i \neq j \quad \rightarrow \quad \neg(C_i \land C_j)$$

is invariant.

Note that most invariance properties are easy to satisfy. Therefore, when specifying the requirements from a system, it is important to include liveness, or progress requirements, to avoid trivial useless solutions.
Liveness Properties

Liveness properties come in many flavors. The most common is of the form:

For every computation \( \eta : s_0, a_1, s_1, \ldots \),
for every \( i \geq 0 \)

\[
  s_i \models \varphi \iff \exists j \geq i : s_j \models \psi
\]

where both \( \varphi \) and \( \psi \) are state properties. We denote such a property by \( \varphi \rightsquigarrow \lozenge \psi \).

Liveness properties are usual proven by well-founded induction, where the prover finds some ranking function on states, with \( \psi \)-states ranked the lowest, and shows that every transition decreases the rank. Well-foundedness of the domain then establishes liveness.

A typical example of liveness property is the accessibility of mutual exclusion protocols, that for every \( i \), every \( T_i \) state is eventually followed by a \( C_i \) state, i.e., \( \forall i : T_i \rightsquigarrow \lozenge C_i \).

It is often the case that one needs to use the fairness assumptions to prove liveness.
Proving m.e. for Peterson’s Algorithm

We first some simple invariants (i.e., invariance properties), namely, that for $i = 1, 2$:

\[
\begin{align*}
y_i &\iff status_i \in \{t^2, t^3, c, e\} \\
I_i &\iff status_i \in \{i\} \\
T_i &\iff status_i \in \{t^1, t^2, t^3\} \\
C_i &\iff status_i \in \{c\} \\
E_i &\iff status_i \in \{e\}
\end{align*}
\]

as well as:

\[
I_i \lor T_i \lor C_i \lor E_i, \quad turn \in \{1, 2\}, \quad \text{and} \quad C_i \rightarrow y_i
\]
Proving m.e. for Peterson’s Algorithm

We now establish an auxiliary invariant from which we’ll derive the mutual exclusion property. For every \( i = 1, 2 \), let \( \phi_i \) be the state property:

\[
C_i \lor E_i \quad \rightarrow \quad status_{3-i} \in \{i, t^1, t^2\} \lor turn = i
\]

Next we’ll show that \( \phi_i \) is invariant by computational induction. Initially, \( \neg C_i \), hence \( \phi_i \) holds. Consider now potentially falsifying transitions:

- \( T_i \rightarrow C_i \). This is only possible when \( \neg y_{3-i} \lor turn = i \), which implies \( status_{3-i} \in \{i, t^1\} \lor turn = i \), thus \( \phi_i \) is preserved.

- \( status_{3-i} \in \{i, t^1, t^2\} \leftrightarrow status_{3-i} \not\in \{i, t^1, t^2\} \) while \( C_i \lor E_i \). But this implies that \( turn \) becomes \( i \), thus \( \phi_i \) is preserved.

- \( turn = i \leftrightarrow turn \neq i \) while \( C_i \lor E_i \). But this is only possible when \( i \) moves from one \( T_i \) state to another, thus cannot happen from a \( C_i \lor E_i \) state.
Peterson’s Mutual Exclusion Property

... follows now immediate from \( \phi_1 \) and \( \phi_2 \):

Since both \( \phi_i \) and \( \phi_2 \) are invariant, so is the their conjunction:

\[
C_1 \land C_2 \quad \longrightarrow \quad (\text{status}_2 \in \{i, t^1, t^2\} \lor \text{turn} = 1) \land (\text{status}_1 \in \{i, t^1, t^2\} \lor \text{turn} = 2)
\]

But since \( C_i \rightarrow \text{status}_i = c \) is also invariant, and we can’t have \textit{turn} equal to both 1 and 2 at the same time, we get that

\[
C_1 \land C_2 \quad \longrightarrow \quad \text{false}
\]

Note: From the invariance of \( \phi_i \) we can also derive the invariance of \( \phi_i' : T_i^3 \land \text{turn} = i \rightarrow \neg (C_{3-i} \lor E_{3-i}) \). The potentially falsifying transitions are: (1) when the l-h-s becomes true; this is only possible when \( P_{3-i} \) enters \( T^3 \), in which case the r-h-s is true, and (2) when \( P_{3-i} \) enters the critical section, in which case we have by \( \phi_{3-i} \) that the l-h-s does not hold.
Peterson's Algorithm: Liveness

We prove the liveness property “if $T_i$ then eventually $C_i$” by proving a chain of liveness properties:

1. $T_i^3 \land \neg y_{3-i} \rightarrow \Diamond C_i$
2. $T_i^3 \land y_{3-i} \land \text{turn} = i \rightarrow \Diamond C_i$
3. $T_i^3 \land y_{3-i} \land \text{turn} \neq i \land \neg (C_{3-i} \lor E_{3-i}) \rightarrow \Diamond \text{turn} = i$
4. $T_i^3 \land (C_{3-i} \lor E_{3-i}) \rightarrow \Diamond \neg y_{3-i}$
5. $T_i \rightarrow \Diamond T_i^3$

To prove (5), it suffices to note that, from the fairness properties on $P_i$, it follows that

6. $T_i^1 \rightarrow \Diamond T_i^2$
7. $T_i^2 \rightarrow \Diamond T_i^3$
Proving (1)–(2)

To prove (1), note the following:

\[ \begin{align*}
(a) \quad T_i^3 \land \neg y_{3-i} & \leftrightarrow T_i^3 \land (I_{3-i} \lor T_{3-i}^1) \\
(b) \quad T_i^3 \land I_{3-i} & \rightarrow \diamond (C_i \lor T_{3-i}^1) \\
(c) \quad T_i^3 \land T_{3-i}^1 & \rightarrow \diamond (C_i \lor T_{3-i}^2) \\
(d) \quad T_i^3 \land T_{3-i}^2 & \rightarrow \diamond (C_i \lor (\text{turn} = i \land T_{3-i}^3)) \\
(e) \quad T_i^3 \land T_{3-i}^3 \land \text{turn} = i & \rightarrow \diamond C_i \\
(f) \quad T_i^3 \land \neg y_{3-i} & \rightarrow \diamond C_i
\end{align*} \]

To prove (2), note:

\[ \begin{align*}
(g) \quad T_i^3 \land y_{3-i} \land \text{turn} = i & \leftrightarrow \\
& T_i^3 \land (T_{3-i}^2 \lor T_{3-i}^3) \land \text{turn} = i \quad \text{(invariants included)} \\
(h) \quad T_i^3 \land (T_{3-i}^2 \lor T_{3-i}^3) \land \text{turn} = i & \rightarrow \diamond C_i \quad ((d) \text{ and } (e) \text{ ab}) \\
(i) \quad T_i^3 \land \text{turn} = i \land y_{3-i} & \rightarrow \diamond C_i \quad ((g) \text{ and } (h))
\end{align*} \]
Other Properties of Mutex

In addition to safety and liveness there are other properties one may require from a solution to the mutual exclusion problem.

One such property is **livelock freedom** or **communal liveness**, that requires that if some process is trying then eventually some process will access the critical section. I.e., $\exists i : T_i \leadsto \Diamond \exists j : C_j$. While this may not seem to be sufficient, it is sometimes the only liveness one can attain, and a system that satisfied it can be used as a building block in a system that satisfied the "full" individual liveness property.

Another property is **bounded overtaking**, that bounds the number of times one process can enter the critical section while another is trying to enter it. Bounded overtaking is a safety property. Peterson's 2-process mutex algorithm has a bounded overtaking of 1: once the system reached a $T_i$ state, $P_j$ can enter the critical section at most once before $P_i$ does.