TD 1: Models

Exercise 1 (Mutual Exclusion).

1. The following program is a mutual exclusion protocol for two processes due to Pnueli. There is a shared boolean variable s, initialized to 1, and two shared boolean variables y_i, i in $\{0, 1\}$, initialized to 0. Each process P_i can read the values of s, y_0, and y_1, but only write a new value in s and y_i. Here is the code of process P_i in C-like syntax:

```
while (true)
  {
    /* 1: Noncritical section. */
    atomic { y_i = 1; s = i; };
    /* 2: Wait for turn. */
    wait until ((y_{1-i} == 0) || (s != i));
    /* 3: Critical section. */
    y_i = 0;
  }
```

   Draw the transition system of each process, and construct their parallel composition. Label the states appropriately using the atomic propositions w_i and c_i, holding when process P_i is waiting or in the critical section, respectively.

2. Does the algorithm ensure *mutual exclusion*, i.e. that the two processes can never be simultaneously inside the critical section?

3. Does the algorithm ensure *starvation freedom*, i.e. that every waiting process will eventually access the critical section, provided that the other process does not stay forever inside the critical section?

4. Recall Peterson's mutual exclusion algorithm from the lecture notes: two processes 0 and 1 execute a program to access a critical section (where they can safely modify other shared resources). In order to do so, they share three boolean variables: y_0, y_1, and s. Here is the algorithm for P_i with the first two assignments swapped:

```
while (true)
  {
    /* 1: Noncritical section. */
    s = 1 - i;
    y_i = 1;
    /* 2: Wait for turn. */
    wait until ((y_{1-i} == 0) || (s == i));
    /* 3: Critical section. */
    y_i = 0;
  }
```

Show that this modified algorithm is incorrect, i.e. that the two processes can simultaneously be in the critical section. More precisely, compute the part of the low-level transition system that leads to the error.

Exercise 2 (Rendez-vous with Data)**.** Consider the synchronization of transition systems with variables through a rendez-vous mechanism. Such a system is of form $M = (S, \Sigma, \mathcal{V}, (D_v)_{v \in \mathcal{V}}, T, I, \mathrm{AP}, l)$ where $\mathcal{V}$ the set of (typed) variables v, each with domain D_v.

We want to extend the rendez-vous mechanism between systems with variables with the ability to exchange data values. For instance, a system M_i may transmit a value m by performing

$$s_i \xrightarrow{!m} s'_i \ ,$$

only if some system M_j is ready to receive the message, i.e. to perform

$$s_j \xrightarrow{?v} s'_j \ ,$$

where v is a variable of M_j and m is in D_v. Of course the synchronization is also possible if M_j performs instead

$$s_j \xrightarrow{?m} s'_j \ .$$

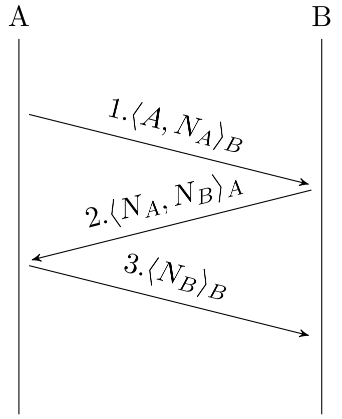
1. Propose Structural Operational Semantics for the rendez-vous with data synchronization.
2. Assume $D_v = D$ for all variables v in $\mathcal{V}$.

   Generalize these semantics to allow sending and receiving *terms* in $T(\Sigma, \mathcal{V})$ build from the variables and a finite set of symbols Σ that contains D.

Exercise 3 (Needham-Schroeder Protocol)**.** We consider the analysis of a public-key authentication protocol proposed by Needham and Schroeder in 1978. The protocol relies on

- the generation of *nounces* N_C: random numbers that should only be used in a single session, and
- on public key encryption: we denote the encryption of message M using C's public key by $\langle M \rangle_C$.

A(lice) and B(ob) try to make sure of each other's identity by the following (very simplified) exchange:



1. Alice first presents herself (the A part of the message) and challenges Bob with her nounce N_A. Assuming both cryptography and random number generation to be perfect, only Bob can decrypt $\langle A, N_A \rangle_B$ and find the correct number N_A.

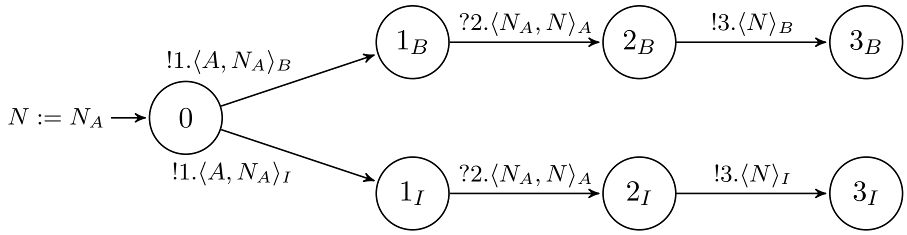
2. Bob responds by proving his identity (the N_A part) and challenges Alice with his own nounce N_B.

3. Finally, Alice proves her identity by sending N_B.

The nounces N_A and N_B are used by Alice and Bob as secret keys for their communications.

In order to account for the insecure channel, we have to add an intruder I to the model, who has his own nounce N_I, and can read and send any message it fancies, but can only decrypt $\langle M \rangle_I$ messages and cannot guess the nounces generated by Alice and Bob.

We can model the behaviour of Alice as a transition system M_A with variables and rendez-vous with data, using a single variable N ranging over $D_N = \{N_A, N_B, N_I\}$.



1. Provide a model M_B for Bob.

2. Provide a model M_I the intruder.

3. Unfold an execution path in the synchronized product of M_A, M_B, and M_I that unveils a flaw in the protocol.

Exercise 4 (Channel Systems)**.** The course notes present the semantics of FIFO channels. We consider here the case of a single finite system $M = \langle S, \Sigma, T, I, \mathrm{AP}, \ell \rangle$ along with n unbounded channels over a finite set Γ (i.e. each channel is declared as c_i`:` `channel[∞] of` Γ for each $1 \leq i \leq n$). Configurations of the full system $\hat{M}$ are thus in $S \times (\Gamma^*)^n$, i.e. of form $(s, \gamma_1, \ldots, \gamma_n)$ where s is a state of S and channel i contains γ_i. Without loss of generality, we consider the channels to be empty in the initial configurations, i.e. $\hat{I} = \{(s_i, \varepsilon, \ldots, \varepsilon) \mid s_i \in I\}$.

We are interested in the *control-state reachability problem*, i.e. given an n-channel system $\hat{M}$ and a state s, does there exist an initial state s_i in I and n strings $\gamma_1, \ldots, \gamma_n$ in Γ^* s.t. $(s_i, \varepsilon, \ldots, \varepsilon) \to^* (s, \gamma_1, \ldots, \gamma_n)$?

1. Consider the case $\Gamma = \{a\}$ and $n = 1$. Show that the control-state reachability problem is decidable in PTime.

2. Show that it becomes undecidable for $n = 1$ and $|\Gamma| \geq 2$.

3. We allow the channel systems to test the contents of a channel for emptiness:

$$\frac{\nu(c_j) = \varepsilon \wedge s_i \xrightarrow{\text{empty}(c_j)} s'_i}{(\bar{s}, \nu) \xrightarrow{\text{empty}(c_j)} (\bar{s}', \nu)}$$

   Show that the control-state reachability problem is then undecidable for $n \geq 2$ even if $|\Gamma| = 1$. Hint: reduce from the control state reachability in 2-counters Minsky machines.