TD 1: Modeling

1 Parallel Processes

1.1 Concurrent Access

We are given three processes P_1, P_2, and P_3 with shared integer variable x. The program of process P_i increments x ten times as follows:

```
int k, t;
for (k = 0; k < 10; k++)
  {
    t = x;    /* load x in t  */
    t++;      /* increment t  */
    x = t;    /* store t in x */
  }
```

Supposing that x is initially 0 and that the three P_i instances are run in parallel, is there an execution that halts with value 2 for x? What happens if we replace the three instructions by a single `x++`?

1.2 Mutual Exclusion

The following program is a mutual exclusion protocol for two processes du to Pnueli. There is a single shared variable s which is either 0 or 1, and initially 1. Besides, each process has a public local Boolean variable y, that initially equals 0. Here is the code of process P_i:

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

Draw the transition system of each process, construct their parallel composition, and check whether the algorithm ensures mutual exclusion. Does it ensure starvation freedom?

2 Modeling

The purpose of the two last exercises is to model the following two systems. Try to identify the consequences of your choices: which simplification, which abstractions did you make? Which granularity did you choose?

2.1 Lunch Protocol

Three researchers agree on a protocol for the choice of the lunch restaurant. The first one who starts feeling hungry chooses one of the two restaurants on the campus and calls one of his two coworkers, tells him where to go and leaves. This second researcher calls the remaining one and leaves, and the last one also leaves after this second call.

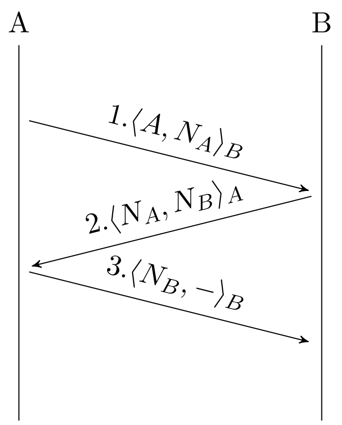
Do the three researchers end up in the same restaurant?

2.2 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 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.

Can an intruder pass as Alice or Bob, and gain knowledge of N_A and N_B?