Formal Methods in Software Development
Exercise 5 (July 6)

Wolfgang Schreiner
Wolfgang.Schreiner@risc.uni-linz.ac.at

May 23, 2006

The exercise is to be submitted by **July 6** (hard deadline)

1. either as a paper report (cover page with full name and Matrikelnummer, pages stapled, put into an envelope with address "Wolfgang Schreiner", "Institut für Symbolisches Rechnen") put into an university mailbox,
2. or as a single PDF file sent to me per email.

1 Dining Philosophers

A group of $n \geq 2$ philosophers sits around a pot of spaghetti with one fork between every pair of philosophers (i.e. there are n forks in total). Every philospher spends his life alternating between philosophizing and eating. When a philosopher wants to eat, she first grabs the left fork, then grabs the right fork, then uses both forks to eat some spaghetti, then returns both forks and philosophizes again. If one of the forks is already in use by a neighbor, the hungry philosopher has to wait until the fork is returned, before it can be grabbed.

1. Model this scenario as a concurrent system (state space, initial state condition, transition relation) with n processes, first for the special case $n = 2$ and then for the general case $n \geq 2$.
   - Signal by $f[i] \in \{0, 1\}$ that fork i is in use.
   - Signal by $s[i] \in \{0, 1, 2, 3\}$ the state of philosopher i (0: philosophizing, 1: waiting for left fork, 2: waiting for right fork, 3: eating).
   - Define a predicate p such that $p_i(f, s, f', s')$ describes the four possible actions of philosopher i:
     - She becomes hungry and waits for the left fork.
     - She receives the left fork and waits for the right fork.
     - She receives the right fork and starts eating.
     - She finishes eating and returns the forks.

The transition relation of the whole system can be constructed by application of this predicate.

For $n = 2$ you may use pairs of scalar variables; for $n \geq 2$, you need functions $\mathbb{N}_n \to \{\ldots\}$ to model the state.

2. We want the system to fulfill the following property: *it is never the case that all forks are in use and all philosophers are waiting.*

   Formulate this property (for $n = 2$ and for $n \geq 2$) as a LTL formula using the variables introduced in above system model.

3. Describe the scenario for $n = 3$ in Promela (using shared variables, not message passing) and use Spin to check whether it satisfies the property. If not, give a high-level interpretation of the counterexample generated by Spin. Please note that an atomic state transition with precondition p and action a must be described in Promela as

   ```
   atomic { p -> a }
   ```

4. The scenario is changed as follows: apart from the pot of spagetthi there is also a bowl with $n-1$ tokens in the middle of the n philosophers. Before grabbing his left fork, a philosopher has to grab one of the tokens from the bowl. When returning both forks, the philosopher also returns the token.

   Model this scenario as a concurrent system for $n = 2$ and $n \geq 2$ by introducing another variable t that describes the number of tokens in the bowl. The first action of the philosopher is modified such that it only takes place when $t > 0$; it then also decreases t by one. Likewise, the fourth action is modified to increase t by one.

5. Describe the modified scenario for $n = 3$ in Promela and use Spin to check whether it satisfies the property (it should).

6. Verify by manual proof for $n = 2$ that the system has the desired property.

7. **Bonus (50%):** rather than for the special case $n = 2$, perform the verification for the general case $n \geq 2$.

The result of this exercise consists of

1. the formal system models and LTL formulas,
2. the listings of the Promela models and formulas used in the verifications (including the definitions of the atomic predicates),
3. the SPIN output of each attempted verification run,
4. counterexamples produced by SPIN (if any) and their interpretation,
5. the manual verification.