Model-Checking Exercises
Sommersemester 2009 / Sheet 2

May 6, 2009

We are going to discuss the examples together at 28.5. For questions about the exercises or examples, please send me an email `campetel@in.tum.de`.

Example 2.1: More or less behaviors?

Let $K_1 = (S, \rightarrow_1, r, \mathbf{AP}, v)$ and $K_2 = (S, \rightarrow_2, r, \mathbf{AP}, v)$ two Kripke structures with the same states S, the same start state r and the same interpretation v on the propositions $\mathbf{AP}$. We write now $K_1 \leq K_2$, where the transition relation $\rightarrow_2 \subseteq S \times S$ allows more behaviors than the transition relation $\rightarrow_1 \subseteq S \times S$, viz if holds $\rightarrow_1 \subseteq \rightarrow_2$. Show that for $K_1 \leq K_2$ holds the following relation for every **LTL** formula ϕ:

$$K_2 \models \phi \Rightarrow K_1 \models \phi$$

Example 2.2: Token Ring in Spin

A token ring consists of m independent processes, which are arranged in a circle, so that every process has exactly one left and right neighbor. The processes in the Token Ring use a token (best represented by a message in a channel) to be synchronized. After each calculation step, the token is passed to one of the neighbors. A process may only be in the critical section, if it has the token.

1. Implement a token ring in Promela with $m = 4$, where the token is passed non-deterministically to a Neighbors of the two. Simulate the token ring interactively.
2. Use SPIN to verify that at any given time no more than one process is in the critical section.
3. Use SPIN to verify that at least a process infinitely often occurs in the critical section.
4. Repeat the above steps for a deterministic model, where the token is passed always to the left.
5. Use SPIN to the deterministic and non-deterministic variant to verify whether every process infinitely often in the critical section comes.

Note: Use a Promela process type to model the Token Ring processes:

```
proctype process (chan left; chan right)
```

This process receives as a parameter one channel for the left and right neighbors. Then start four processes of this type in the init process using

```
run process (t1, t2);
run process (t2, t3);
run process (t3, t4);
run process (t4, t1)
```

whereas declare the channels using

```
mtype {
TOKEN
}

chan t1 = [1] of {mtype}
chan t2 = [1] of {mtype}
chan t3 = [1] of {mtype}
chan t4 = [1] of {mtype}
```

Example 2.3: Non-determinism in Büchi Automata

Show that non deterministic Büchi automata are strict more expressive than deterministic Büchi automata.

1. Let $L \subseteq \Sigma^\omega$ the language over the alphabet $\Sigma = \{a, b\}$, which the words contain finite many b-s:

   $$L = \{\sigma \in \Sigma^\omega \mid \#_b(\sigma) < \infty\}$$

   Provide a non-deterministic Büchi automata, which accepts L.

2. Show that there is no deterministic Büchi automata, which accepts L. *Note*: Assume, there would be a such automa $\mathbb{B}$ with k states, and consider the accepting run for the word $w_0 = a^\omega$ (the automaton must accept the word). Now add after the first accepting state of this run a b to get a word w_1 : If the automata after reading the prefix a^{l_1} for the first time passes over an accept state, compose in $w_1 = a^{l_1}ba^\omega$. Note that holds $w_1 \in L$. What property must the state satisfy after is reached the reading of the prefix $a^{l_1}b$? And how can you use this property in order to deduce a contradiction from a concatenation of words $w_0, \ldots, w_{k+1}$?