Model Checking WS 2015: Assignment 1

Institute for Formal Models and Verification, JKU Linz

Due 22.10.2015

Exercise 1

Draw an automaton $P$ which accepts all words $\omega \in \{x, y, z\}^*$ which satisfy the following conditions:

- In $\omega$, every $x$ is followed by a $y$.
- In $\omega$, every $y$ is followed by two $z$.
- $\omega$ contains an even number of $y$.
- $\omega$ starts with at most two $z$.

Exercise 2

Let $A$ and $B$ be two finite automata and $P := A \times B$ the product automaton of $A$ and $B$. Do the following two statements hold? If so then give a proof sketch for the claim. Otherwise, provide a concrete counterexample, i.e. concrete $A, B$ and $P$ refuting the claim.

a) If both $A$ and $B$ are deterministic then $P$ is deterministic.

b) If $P$ is deterministic then both $A$ and $B$ are deterministic.
Exercise 3

a) Given two FA $A_I$ and $A_S$ describing an implementation $I$ and specification $S$, respectively. Explain in detail how to check whether $I$ conforms to $S$, given $A_I$ and $A_S$. Illustrate your explanations using set diagrams.

b) Check conformance of implementation $I$ and specification $S$ given as FA on the right.

Exercise 4

Check conformance of implementation $I$ and specification $S$ given as FA on the right.

Exercise 5

Given the predicates $A := \langle x < y \rangle$ and $B := \langle (x \mod 2) \neq 0 \rangle$ and the action $\alpha := \langle y := y + x \rangle$.

For variables $x$, $y$ and action $\alpha$, assume 4-bit signed integer modular arithmetic and two’s complement representation, that is values can overflow at the borders of the value range. This corresponds e.g. to Java semantics of integer arithmetic.

Given the abstract transition system with abstract states $S = \{AB, \overline{AB}, A\overline{B}, \overline{A}\overline{B}\}$ shown below. Notation $\overline{A}$ ($\overline{B}$) means that predicate $A$ ($B$) does not hold. Edges represent transitions between states by action $\alpha$.

For each transition from a state $s$ to state $s'$ given by an edge, add concrete values for $x$ and $y$ in $s$, if possible. If a transition cannot be executed, then delete the corresponding edge. You do not have to introduce new edges.
Exercise 6

a) Given variables $i, n \in \mathbb{Z}$ (integers), the predicate $a \leftrightarrow (i = 0)$ and the action $\alpha := i++$. Predicate $a$ defines two abstract states $a$ and $\neg a$, i.e. $a$ can hold or not. Draw an abstract transition system by adding all possible transitions between states $a$ and $\neg a$ when action $\alpha$ is executed: how does executing $\alpha$ influence the value of predicate $a$?

b) As above, but $a \leftrightarrow (i > n)$. What is the difference when interpreting $i, n$ and $\alpha$ over 32-bit Java integers with overflow semantics?

The abstract transition system from part b) is used to abstract the code fragment shown below (left). It is assumed that $i, n \in \mathbb{Z}$ (integers), i.e. values can not overflow. In the abstraction (right), value * denotes nondeterministic choice. Relational expression $i > n$ is replaced by predicate $a$.

```plaintext
assert (i <= n);
lock ();
do {
  i++;
  if (i > n) unlock ();
} while (i <= n);

Bool a = false;
assert (!a);
lock ();
do {
  if (!a) a = *;
  if (a) unlock ();
} while (!a);
```

Bonus Exercise

Read sections I and III “Software Model Checking” in the survey on software verification\(^1\) and describe the approach of counterexample-guided abstraction refinement (CEGAR).

\(^1\)V. D’Silva, D. Kroening, G. Weissenbacher: A Survey of Automated Techniques for Formal Software Verification. IEEE TCAD 27(7), 2008. The article can be found in KUSSS.