Technische Universität Wien
Fakultät für Informatik
Univ. Prof. Helmut Veith
Dr. Igor Konnov
Dr. Florian Zuleger
Andreas Holzer, M.Sc.
Moritz Sinn, M.Sc.

SS 2012
Übungsblatt 4
Wednesday, 30 May 2012

Exercises on Formal Methods in Computer Science

Version: 3 / 05062012

Name:	Mustermann
Vorname:	Max
Matr.-nr.:	0123456
Gruppe:	Mustermann (0123456), Musterfrau (0113456), Doe (0346121)

The deadline for handing in the solution is Sunday, 17th of June, before midnight.

Exercise 1 ACTL & LTL **(2 Points)**

Given an LTL formula φ in Negation Normal Form, the following function $\mathsf{trans} : \mathbf{LTL} \rightarrow \mathbf{ACTL}$ translates φ into an ACTL formula $\mathsf{trans}(\varphi)$ as follows:

φ	$\mathsf{trans}(\varphi)$
true	**true**
false	**false**
a	a
$\neg a$	$\neg a$
$\varphi_1 \vee \varphi_2$	$\mathsf{trans}(\varphi_1) \vee \mathsf{trans}(\varphi_2)$
$\varphi_1 \wedge \varphi_2$	$\mathsf{trans}(\varphi_1) \wedge \mathsf{trans}(\varphi_1)$
$\mathbf{X}\varphi_1$	$\mathbf{AX}\ \mathsf{trans}(\varphi_1)$
$\mathbf{F}\varphi_1$	$\mathbf{AF}\ \mathsf{trans}(\varphi_1)$
$\mathbf{G}\varphi_1$	$\mathbf{AG}\ \mathsf{trans}(\varphi_1)$
$\varphi_1\mathbf{U}\varphi_2$	$\mathbf{A}\left[\mathsf{trans}(\varphi_1)\ \mathbf{U}\ \mathsf{trans}(\varphi_2)\right]$
$\varphi_1\mathbf{R}\varphi_2$	$\mathbf{A}\left[\mathsf{trans}(\varphi_1)\ \mathbf{R}\ \mathsf{trans}(\varphi_2)\right]$

The semantics of the "release" operator $\mathbf{R}$ is defined as follows:

$$M, \pi \models \varphi_1\mathbf{R}\varphi_2 \overset{def}{\Leftrightarrow} \forall j \geq 0.\pi^j \models \varphi_2 \text{ or } \exists i \geq 0.(\pi^i \models \varphi_1) \wedge (\forall k \leq i.\pi^k \models \varphi_2)$$

a) Show that, for all **LTL** formulas φ in negation normal form, the **CTL*** formula $\mathsf{trans}(\varphi) \Rightarrow \mathbf{A}\varphi$ is a tautology. *Hint: Show this by showing that $M, s \models \mathsf{trans}(\varphi)$ implies $M, s \models \mathbf{A}\varphi$ for all* **LTL** *formulas φ in negation normal form, all Kripke structures M and all states s in M. Use induction over the structure of φ.* **(1 Point)**

b) Show that, in general, $M, s \models \varphi$ does not imply $M, s \models \mathsf{trans}(\varphi)$. *Hint: Give a Kripke structure M and an* **LTL** *formula φ such that $M, s \models \varphi$ and $M, s \not\models \mathsf{trans}(\varphi)$. Discuss why $M, s \models \varphi$ and $M, s \not\models \mathsf{trans}(\varphi)$ holds on M and the given state s in M.* **(1 Point)**

Exercise 2 LTL - Monotonicity and Negation Normal Form **(2 Point)**

(a) Let $K_1 = (S, R, L_1)$ and $K_2 = (S, R, L_2)$ be two Kripke structures with the same set of states S and the same transition relation R such that $L_1(s) \subseteq L_2(s)$ for all states $s \in S$. Prove that $K_1, s \models \phi$ implies $K_2, s \models \phi$ for all LTL formulae ϕ that do not contain negation. *(Hint: prove this statement by structural induction).*

(b) Exercise 1 defined the *release operator* **R**. Prove that the release operator enjoys the following equivalence using the semantics of LTL *(Hint: use the semantics of LTL formulae)*:

$$\phi \mathbf{R} \psi \equiv \neg(\neg\psi \mathbf{U} \neg\phi)$$

(c) An LTL formula in *negation normal form*, if

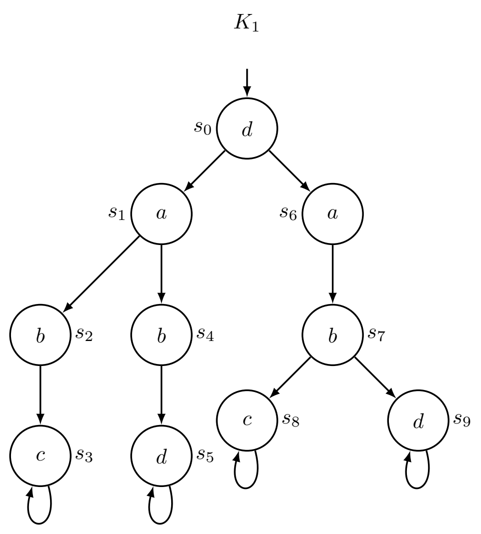
- all negations appear only in front of the atomic propositions,
- only the logical operators *true*, *false*, $\vee$, and $\wedge$ are used, and
- only the temporal operators **X**, **U**, and **R** are used.

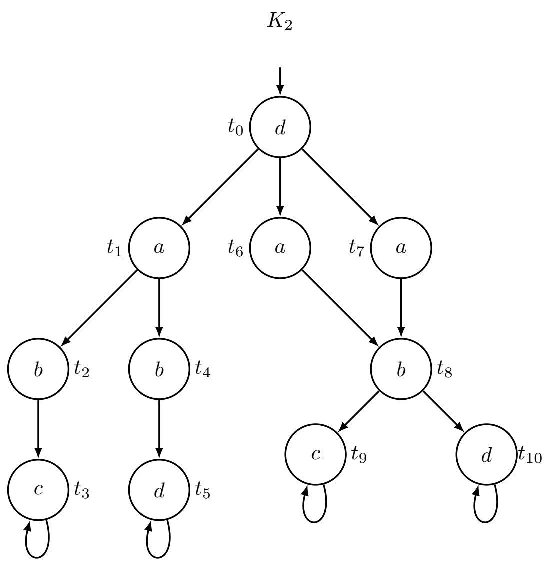
Show that every LTL formula ϕ can be transformed into an equivalent formula ψ that is in negation normal form. *(Hint: prove this statement by structural induction).*

Exercise 3 Bisimulation & Simulation Relations **(1 Point)**

Consider the following Kripke structures K_1 and K_2 and the bisimulation relation

$$H = \{(s_0,t_0),(s_1,t_1),(s_2,t_2),(s_3,t_3),(s_4,t_4),(s_5,t_5),$$
$$(s_6,t_6),(s_6,t_7),(s_7,t_8),(s_8,t_9),(s_9,t_{10})\}.$$





a) Give a simulation relation H' such that $K_1 \leq K_2$ and $|H'| < |H|$. **(0.5 Points)**

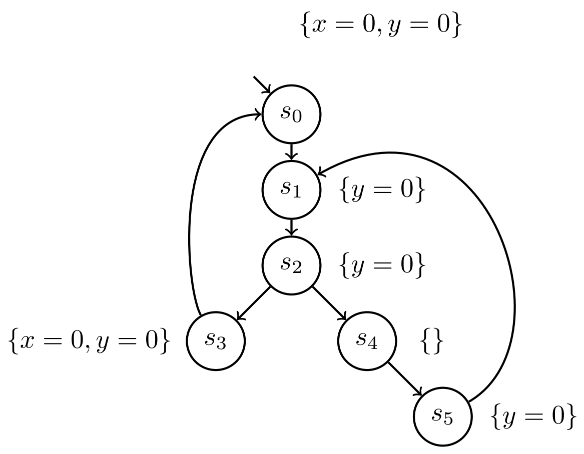
b) Give a simulation relation H'' such that $K_1 \leq K_2$ and $|H''| > |H|$. **(0.5 Points)**

Exercise 4 Simulation as refinement **(2 Points)**

Given a C program and a set AP of atomic propositions, one can construct a Kripke structure over AP which models the behaviour of the program with respect to the atomic propositions. For example, given the following program:

```
int x = 0, y = 0;
l0:
    y = 0;
    for (int i = 0; i < 2; i++)
l1:     x = 1;
    y = *;
    if (y != 0)
l2:     x = 1;
    else
l3:     x = 0;
    goto l0;
```

One may construct the following Kripke structure over $AP = \{x = 0, y = 0\}$ as follows:



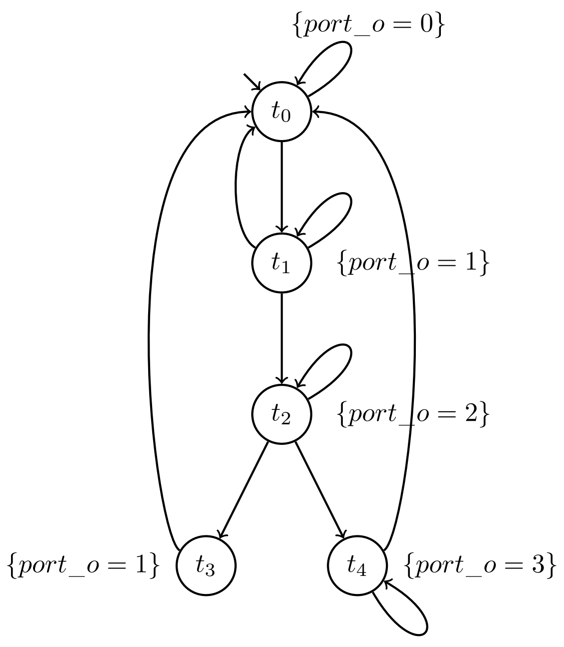
In the example above, the special form of assignment `y = *` denotes a non-deterministic assignment to `y` of any value from the domain of `y`, e.g., it can be assigned concurrently by another thread. The effect of a statement sequence residing under the same label ℓ_j may be merged into one state, whereas the effect of statements marked with distinct labels ℓ_j and ℓ_k, $j \neq k$, must not be merged into one state.

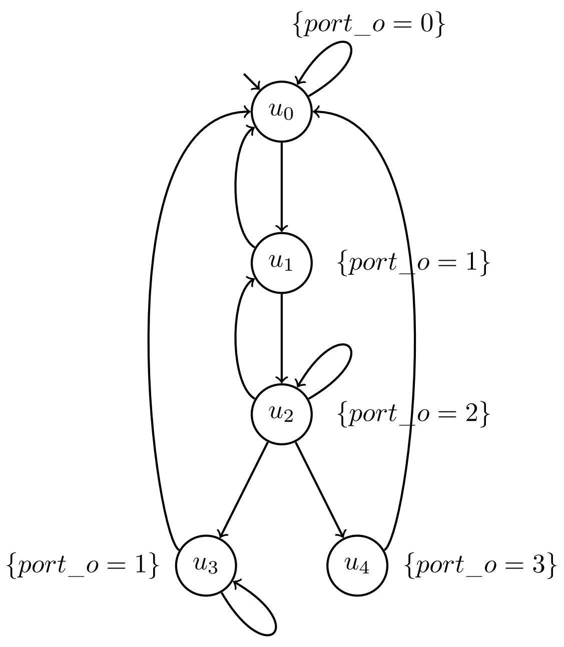
a) Construct a Kripke structure K over $AP = \{port\_o = 0, port\_o = 1, port\_o = 2, port\_o = 3\}$ for the following program:

```
int locked_dev = 0, port_o = 0, port_i = 0;
l0:
    for (int i = 0; i < 3; i++)
l1:     port_o = 1;

    port_i = *;
    if (port_i < 3) {
        goto l0;
    }
    do {
l2:     port_o = 2; locked_dev = *;
    } while (locked_dev);
l3: locked_dev = 1; port_i = *;
    if (port_i == 1)
l4:     port_o = 1;
    else
l5:     port_o = 3;
l6: port_o = 0; locked_dev = 0;
    goto l0;
```

b) Give simulation relations between K and the following Kripke structures S' (left) and S'' (right):





Exercise 5 Predicate Abstraction **(1.5 Point)**

Consider the following program:

```
void foo(int j, int z) {
  assume(z != 0);
  int i := j;
  while(z != 0) {
      i := i + z;
      if(z > 0)
          z--
      else
          z ++;
  };
  assert(i != j)
}
```

The *assume statement* at the beginning of the function forces the parameter z not to be 0 when the function is called.

(a) Argue in your own words why the assertion at the end of the program allways holds, i.e., why the error state can never be reached.

(b) Provide a labeled transition system for the given program.

(c) Provide an abstraction for the labeled transition system that uses the predicates $i = j$, $i < j$, $i > j$.

(d) Check whether the error state can be reached in the abstraction, if so state a trace to the error state and refine the abstraction with suitable predicates such that the error state is not reachable anymore.

Exercise 6 Bounded Model Checking **(1.5 Point)**

Consider the following Program:

```
#define N 5
#define K 0

void main() {

  bool G[N][N] = {
     {0, 1, 0, 0, 1},
     {0, 0, 1, 0, 0},
     {0, 1, 0, 1, 0},
     {1, 0, 0, 0, 0},
     {0, 0, 0, 0, 1},
  };

  bool result = 1;
  int node = 0;
  for(i = 0; i < K, i++) {
       int next = nondet() % N;
       result = result && G[node][next];
       node = next;
  }

  result = result && (node == 0) && (K > 0);

  assert(!result);

}
```

The Matrix G models a graph with N nodes, i.e, $G[i][j]$ is true iff there is a directed edge from node i to node j. The method *nondet* chooses an integer non-deterministically.

(a) Use CBMC to find the lowest value for K such that the assertion at the end of the program can be violated. What does it mean that the error state is or is not reachable for a given graph G and a given value K?

(b) Perform an unwinding of the loop for $K = 3$.

(c) Transform the unwinded program into the SSA form.

(d) Build an SMT formula from the SSA form which is satisfiable if and only if the assertion at the end of the program may be violated. *Note:* A call to *nondet* can be modeled by introducing a new integer variable.

(e) Can you draw a conclusion about the satisfiability of the formula by comparing the value for K that you determined under (a) with the number of times the loop was unwinded for building the formula? Check whether the formula can be satisfied (You may use an SMT solver such as *Yices* or *Z3*) to be sure that the result of the satisfiability check is consistent with your expectation.

Exercise 7 Computing the (Greatest) Bisimulation Relation **(1 Point)**

Let $K_1 = (S_1, R_1, L_1)$ and $K_2 = (S_2, R_2, L_2)$ be two Kripke structures over a set of atomic predicates AP. The relations $H_n \subseteq S_1 \times S_2$ are inductively defined by:

- $(s_1, s_2) \in H_0$ iff $L_1(s_1) = L_2(s_2)$.
- $(s_1, s_2) \in H_{n+1}$ iff
  - (i) $(s_1, s_2) \in H_n$,
  - (ii) for all $(s_1, t_1) \in R_1$ there exists a $(s_2, t_2) \in R_2$ with $(t_1, t_2) \in H_n$, and
  - (iii) for all $(s_2, t_2) \in R_2$ there exists a $(s_1, t_1) \in R_1$ with $(t_1, t_2) \in H_n$.

(a) Compute the sequence $H_0, H_1, H_2, \ldots$ for the Kripke structures K_1 and K_2 from Exercise 2.

(b) Show that the sequence $H_0, H_1, H_2, \ldots$ *stabilizes* for all finite Kripke structures K_1 and K_2, i.e., there is a $n \geq 0$ such that $H_n = H_{n+1}$.

(c) Construct Kripke structures K_1^n and K_2^n such that the sequence $H_0, H_1, H_2, \ldots$ stabilizes after exactly n steps.