6.0/4.0 VU Formale Methoden der Informatik

185.291 June, 28 2013

Kennzahl (study id)	Matrikelnummer (student id)	Familienname (family name)	Vorname (first name)	Gruppe (version)
				A

1.) Consider the following problem:

> **3-PROGRAM-EQUIVALENCE (3PE)**
>
> INSTANCE: A triple (Π_1, Π_2, Π_3) of programs that take a single string as input.
>
> QUESTION: Are Π_1, Π_2 and Π_3 equivalent? That is, is it true that for all input strings I, the programs Π_1, Π_2 and Π_3 produce the same output value, or they all do not terminate?

By providing a reduction from **PROGRAM-EQUIVALENCE** to **3PE**, prove that **3PE** is undecidable. Argue formally that your reduction is correct.

We remind that **PROGRAM-EQUIVALENCE** is defined as follows:

> **PROGRAM-EQUIVALENCE (PE)**
>
> INSTANCE: A pair (Π_1, Π_2) of programs that take a single string as input.
>
> QUESTION: Are Π_1 and Π_2 equivalent? That is, is it true that for all input strings I, the programs Π_1 and Π_2 produce the same output value, or they both do not terminate?

(15 points)

2.) (a) Consider the following clauses:

$$c_1 : (\neg A \vee B) \qquad c_2 : (\neg A \vee \neg C) \qquad c_3 : (D \vee E) \qquad c_4 : (\neg B \vee C \vee \neg E)$$

Draw an implication graph starting with the decisions $D = 0@1$ and $A = 1@2$ until you reach a conflict. Mark/Underline all UIPs in the implication graph and state which UIP is the first one. **(4 points)**

(b) Prove: *During the run of a SAT solver, the implication graph G_k at step k is acyclic.*
Hints:

1) Perform a proof by induction over k.
2) Consider the following events that can occur:
   (i) making a decision,
   (ii) unit propagation (one step of BCP),
   (iii) a clause is unsatisfiable,
   (iv) backtracking.

(11 points)

3.) (a) Some programming languages offer loops of the form repeat p until e. The program p is executed repeatedly until the condition e becomes true. The condition is tested for the first time after having executed p once.

Define the syntax and the (structural operational or natural) semantics of a programming language like the one in the course, but with repeat- instead of while-loops. Extend the Hoare calculus accordingly and define the weakest precondition of repeat-loops.

You may use your knowledge of while-loops, but the final definitions should be self-contained and should not refer to while-statements. You don't have to copy the syntax and semantics of the other statements, but indicate clearly which parts of the old definition occur in which places of your new definition.

Remember the following definitions of wp and the Hoare calculus.

wp(while e do p od, G) $= \exists i\,(i \geq 0 \wedge F_i)$
where $F_0 = \neg e \wedge G$ and $F_{i+1} = e \wedge \mathrm{wp}(p, F_i)$

$$\frac{\{\, Inv \wedge e \wedge t = t_0 \,\}\, p \,\{\, Inv \wedge 0 \leq t < t_0 \,\}}{\{\, Inv \,\}\ \text{while } e \text{ do } p \text{ od}\ \{\, Inv \wedge \neg e \,\}}$$

(7 points)

(b) Compute a formula that describes all states for which the following program terminates.

$$y := x;\ \text{while } 3x \neq 2y \text{ do } x := x - 1; y := y + 2 \text{ od}$$

List three states for which the program terminates. **(8 points)**

4.) Simulation

Let $M_1 = (S_1, I_1, R_1, L_1)$ and $M_2 = (S_2, I_2, R_2, L_2)$ be two Kripke structures.

Simulation

Remember, a relation $H \subseteq S_1 \times S_2$ is a simulation relation if for each $(s, s') \in H$ holds:

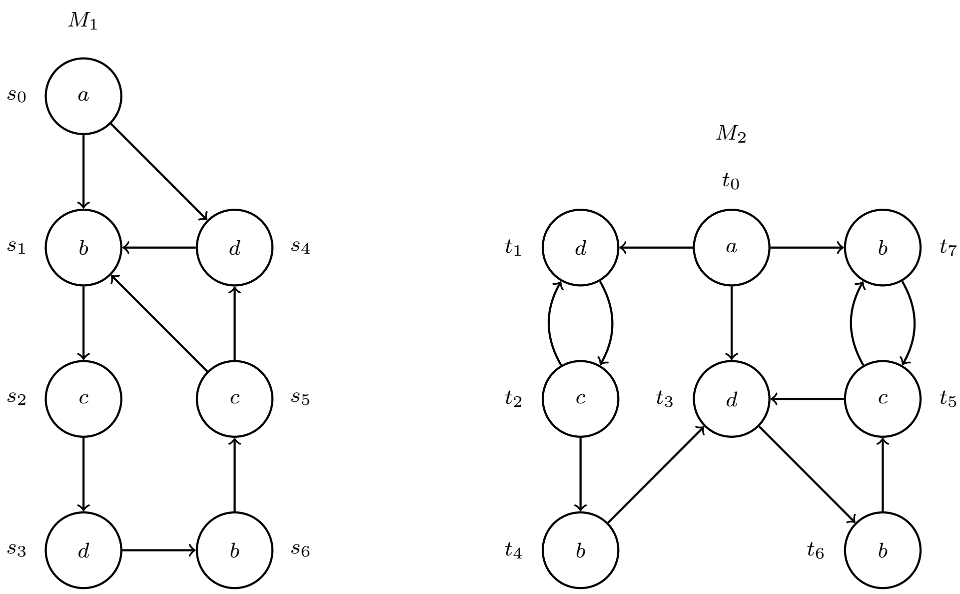
- $L_1(s) = L_2(s')$, and
- for each $(s, t) \in R_1$ there is a $(s', t') \in R_2$ such that $(t, t') \in H$.

Further remember, M_2 *simulates* M_1, in signs $M_1 \leq M_2$, if there is a simulation relation $H \subseteq S_1 \times S_2$ *such that*

- for each initial state $s \in I_1$ there is an initial state $s' \in I_2$ with $(s, s') \in H$.

In the following, we say that H *witnesses the similarity of* M_1 *and* M_2 in case H is a simulation relation from M_1 to M_2 that satisfies the condition stated above.

(a) Give a simulation relation showing $M_1 \leq M_2$.



(3 points)

(b) Algorithm 1 computes the biggest simulation relation between two given Kripke structures $M_1 = (S_1, I_1, R_1, L_1)$ and $M_2 = (S_2, I_2, R_2, L_2)$. Extend the algorithm such that the algorithm outputs a winning strategy for the spoiler for the tuples $(s, s') \in S_1 \times S_2$, for which such a strategy exists. A *winning strategy* for the spoiler is a mapping $\sigma : S_1 \times S_2 \to S_1$ such that $\sigma(s, s')$ is the next position in S_1 for the spoiler in structure S_1.

(5 points)

Data: Two Kripke structures $M_1 = (S_1, I_1, R_1, L_1)$ and $M_2 = (S_2, I_2, R_2, L_2)$.
Result: A simulation relation H between M_1 and M_2.
$H = \{(s, s') \in S_1 \times S_2 \mid L_1(s) = L_2(s')\}$;
$H' = \emptyset$;
while $H \neq H'$ **do**
    $H' = H$;
    $H = H \setminus \{(s, s') \in H \mid \exists (s, t) \in R_1 . \forall (s', t') \in R_2 . (t, t') \notin H\}$
end
return H

Algorithm 1: Simulation Algorithm

(c) We define a simplified subset of ACTL:
A formula $\varphi \in$ ACTLSimp is either

- an atomic proposition p,
- $\varphi_1 \vee \varphi_2$, where $\varphi_1, \varphi_2 \in$ ACTLSimp,
- $\varphi_1 \wedge \varphi_2$, where $\varphi_1, \varphi_2 \in$ ACTLSimp, or
- $\mathbf{AX}\varphi_1$, where $\varphi_1 \in$ ACTLSimp.

Let $K_1 = (S_1, R_1, L_1)$ and $M_2 = (S_2, I_2, R_2, L_2)$ be two Kripke structures and let $H \subseteq S_1 \times S_2$ be a simulation relation. Show that for each $(s, s') \in H$ and for each formula $\varphi \in$ ACTLSimp it holds that $K_2, s \models \varphi$ implies $K_1, s' \models \varphi$. **(7 points)**