6.0/4.0 VU Formale Methoden der Informatik

185.291 WS 2010, SS 2011 July, 1 2011

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

1.) Consider the following problem:

SOME-TERMINATING-INPUT (STI)

INSTANCE: A program (i.e. a source code) Π such that Π takes one string as input and outputs either *true* or *false*.

QUESTION: Does there exist an input string I for Π such that Π terminates on I?

By providing a reduction from **HALTING** to **STI**, prove that **STI** is undecidable. Argue formally that your reduction is correct.

(15 points)

2.) (a) Consider the following clause set $\hat{\delta}(\varphi)$ which has been derived from an (unknown) formula φ by Tseitin translation (atoms have not been labeled).

$$\begin{array}{rlrlrl}
C_1: & \ell_1 \lor \neg x \lor \neg y & C_2: & \neg\ell_1 \lor x & C_3: & \neg\ell_1 \lor y \\
C_4: & \neg\ell_2 \lor \neg y \lor z & C_5: & \ell_2 \lor y & C_6: & \ell_2 \lor \neg z \\
C_7: & \neg\ell_3 \lor \neg\ell_1 \lor z & C_8: & \ell_3 \lor \ell_1 & C_9: & \ell_3 \lor \neg z \\
C_{10}: & \neg\ell_4 \lor \neg x \lor \ell_2 & C_{11}: & \ell_4 \lor x & C_{12}: & \ell_4 \lor \neg\ell_2 \\
C_{13}: & \neg\ell_5 \lor \neg\ell_3 \lor \ell_4 & C_{14}: & \ell_5 \lor \ell_3 & C_{15}: & \ell_5 \lor \neg\ell_4
\end{array}$$

(i) Reconstruct φ from $\hat{\delta}(\varphi)$.

(ii) Start from $\hat{\delta}(\varphi)$ and extend it by a single nonempty clause C in such a way that φ is valid iff $\hat{\delta}(\varphi) \land C$ is unsatisfiable.

(iii) Prove the validity of φ by resolution (no additional translation to normal form is allowed!).

(iv) Can you use the shorter Plaisted-Greenbaum translation here instead of the Tseitin translation? Justify your answer.

(5 points)

(b) Model the following *pigeonhole principle*. If n pigeons are put into $m < n$ holes, then at least one hole must contain more than one pigeon (m, n are natural numbers). Let $p_{i,j}$ denote the fact that pigeon i is in hole j. Give clauses for the following facts.

(i) Every pigeon must be in at least one hole.

(ii) No two pigeons can be in the same hole.

(4 points)

(c) Let R be $\forall x\, p(x,x)$, and let φ be $\exists x \exists y \forall z \big[p(x,y) \land p(y,z)\big]$, where p is a binary predicate symbol. Check whether $R \models \varphi$ holds. If yes, then give a proof; otherwise give a counter-example and prove that the entailment does not hold. **(6 points)**

3.) (a) Some programming languages allow 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 semantics of a programming language like the one in the course, but containing `repeat`- instead of `while`-loops; extend the Hoare calculus accordingly. 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. **(6 points)**

(b) Prove the total correctness of the assertion below using the invariant $3x+2y = 3x_0$. Note that you have to strengthen the precondition and the invariant to show termination.

```
{ 1: $x = x_0$ }
$y \leftarrow 0$;
while $x \neq 0$ do
    $x \leftarrow x - 2$;
    $y \leftarrow y + 3$
od
{ 2: $2y = 3x_0$ }
```

(9 points)

4.) (a) Consider the two formulæ $\mathsf{AGAF}p$ and $\mathsf{AGF}p$. Prove that the two formulæ are equivalent or provide a counterexample. **(5 points)**

(b) Consider the following program

```
do {
    x = y;
    if(w == 1)
        x = x + 1;
    z = w - 1;
} until (x != y);
assert(z == 0);
```

i. Provide a labeled transition system for the given program.

ii. Provide an abstraction for the labeled transition system that uses the predicates $x = y$ and $z = 0$. As a shorthand, use p in case predicate $x = y$ holds and $\bar{p}$ in case it does not hold. Use q in case predicate $z = 0$ holds and $\bar{q}$ otherwise.

iii. Give an error trace in the abstraction.

iv. State a new predicate which can be used to refine the abstraction in order to get rid of the error state. Note, you just have to give the predicate, you don't have to draw the new abstraction.

(5 points)

(c) Given n cities $0, \ldots, n-1$ and a nonnegative integer distance d_{ij} between any two cities i and j (such that $d_{ij} = d_{ji}$), and a "budget" B, write a C program such that CBMC can determine whether there is a tour of length at most B. Augment the following given code corresponding to the following subtasks.

```
#define N 4 // number of cities
#define B 12 // budget

int nondet();

int distances[N][N] =
  {{0, 1, 2, 5},
   {1, 0, 3, 7},
   {2, 3, 0, 4},
   {5, 7, 4, 0}};
int tour[N];
```

i. Write a loop that nondeterministically guesses a tour. A tour is a sequence of visited cities (use the array `tour`, to represent a tour).

ii. Write a loop that checks whether every city is visited once in the tour.

iii. Write a loop that checks whether the tour is within the budget. Note, you have to consider the distance between the last and the first city of the tour as well, since a tour represents a circle through all cities.

iv. Ensure, that CBMC reports a tour within the given budget in case there exists one.

(5 points)