2.9.2 Algorithmic Verification of Programs

Final Exam

December 3, 2015

Please write the solutions for each part independently, on a separate piece of paper. Each part is supposed to be solved in 1 hour and 30 minutes.

1 Part 1: Predicate Abstraction, Interpolants, Decision Procedures for Satisfiability

Problem 1

Consider the following program fragment (the variables are of integer type):

```
res := 0;
count := 0;
error := false;
while ( y > 0 ) {
  y := y - 1;
  count := count + 1;
  tmp := x;
  while ( tmp > 0 ) {
    tmp := tmp - 1;
    res := res + 1;
  }
  if ( x = 0 )
    if ( ! (res = 0) )
      error := true;
```

1. Prove that "`error = false`" is an invariant for this program using predicate abstraction. To this aim, choose a suitable set of predicates P (i.e., atomic formulas without boolean operators) and give the boolean program generated by the chosen set of predicates. Justify the construction of the boolean program. You are allowed to make simplifications with respect to the construction described in the lecture if they are properly justified.

   The set of predicates P should be minimal, in the sense that every strict subset of P is not sufficient to prove the invariant. Give a justification for why P is minimal.

2. Let P' be a strict subset of the chosen set of predicates P. Use the refinement method based on interpolants to synthesize the predicates in $P \setminus P'$ from spurious counter-examples found in the boolean program generated by P'.

Problem 2

Let $\mathcal{T}$ be a first-order theory that admits quantifier-elimination, i.e., for every first-order formula φ over the signature of $\mathcal{T}$ there exists a *quantifier-free* formula ϕ equivalent to φ, i.e., $\mathcal{T} \models \phi \Leftrightarrow \varphi$. Show that for every two formulas $A, B \in \mathcal{T}$ such that $A \wedge B$ is unsatisfiable there exists a *quantifier-free* interpolant I. Is it possible to compute the strongest and the weakest interpolant for A and B (the strongest, resp., weakest, interpolant of A and B is an interpolant I which implies, resp., is implied by, all the other interpolants of A and B) ?

Problem 3

Consider the class of CNF formulas in which no more than one positive literal appears in each clause. Describe a polynomial time algorithm for deciding the satisfiability of this restricted class of CNF formulas. Explain why your proposed algorithm is correct and state the complexity of your algorithm.

Problem 4

Apply the Nelson-Oppen procedure on the following formulas to decide whether they are satisfiable (the variables are interpreted over the integers):

$$f(f(x) - f(y)) \neq f(z) \wedge x \leq y \wedge y + z \leq x \wedge 0 \leq z$$

$$1 \leq x \wedge x \leq 2 \wedge f(x) \neq f(1) \wedge f(x) \neq f(2)$$

Problem 5

Suppose (V, E) is a finite directed tree ($E(v, w)$ means that w is a direct child of v). Further, let ϕ be a labelling of nodes $v \in V$ with formulae $\phi(v)$ such that $\bigwedge_{v \in V} \phi(v)$ is unsatisfiable. A *tree interpolant* I is a mapping from nodes $v \in V$ to formulae such that

- $I(v_0) = false$ for the root node $v_0 \in V$,
- for any node $v \in V$, it holds that

$$\Big(\phi(v) \wedge \bigwedge_{(v,w) \in E} I(w)\Big) \Rightarrow I(v)$$

- any non-logical symbol in $I(v)$ occurs both in some $\phi(w)$ with $E^*(v, w)$, and in some $\phi(w')$ with $\neg E^*(v, w')$ (where E^* is the reflexive and transitive closure of E).

Describe an algorithm for reducing the problem of computing tree interpolants to the problem of computing classical interpolants (i.e., interpolants for two formulas A and B with $A \wedge B$ being unsatisfiable).

Part 2: Sequential and Concurrent Programs

We recall that a pushdown system is a tuple (P, Γ, Δ) where P is a finite set of control states, Γ is finite stack alphabet, and Δ is a set of transition rules of the form $p\gamma \hookrightarrow p'w$ where $p, p' \in P$, $\gamma \in \Gamma$, and $w \in \Gamma^*$. Intuitively, if p is the current control state and γ is the current top of the stack, then by applying this rule the system will move to the control state p and its stack will be modified by replacing the top of the stack by the sequence w (which can be empty).

Configurations of a pushdown system are sequences of the form $pw \in P\Gamma^*$, where p is the current control state of the system, and w is the content of its stack. Given two configurations c and c', we write $c \Rightarrow c'$ is c' is reachable from c by applying one of the rules in Δ, and we write $c \Rightarrow^* c'$ if c' is reachable from c in 0 or more steps.

It is known that pushdown systems can model sequential programs with recursive procedure calls: control states are used to encode the global variables, while stack symbols are used to encode local variables, and the stack is used to store the contexts of the procedures at their call points.

A configuration is *stable* if it is of the form $p\gamma \in P\Gamma$, which means that the stack in such a configuration is of size precisely 1. Intuitively, they correspond to configuration of a program when the main procedure is running. (The call stack contains only the local variables of the main procedure.) Notice that the number of stable configurations is finite.

Given two configurations c and c', and a set of stable configurations s, we write $c \Rightarrow_s^* c'$ if it is possible to reach c' from c by a computation path (i.e., a sequence of $\Rightarrow$ transitions) that visits all the stable configurations of the set s.

An *interface* is a sequence $\sigma = (q_1, q_1')(q_2, q_2') \cdots (q_k, q_k')$, for some $k \geq 0$, where $q_i, q_{i'} \in P$ for every $i \in \{1, \ldots, k\}$. We write $c \overset{\sigma}{\Longrightarrow}_s c'$ if there exists a sequence of stack contents $w_1, \ldots, w_k$, and a sequence of sets of stable configurations $s_0, \ldots, s_k$, such that, $s = s_0 \cup \cdots \cup s_k$, and

- $c \Rightarrow_{s_0}^* q_1 w_1$,
- $q_i' w_i \Rightarrow_{s_i}^* q_{i+1} w_{i+1}$,
- $\forall i \in \{1, \ldots, k-1\}$, $q_k w_k \Rightarrow_{s_k}^* c'$.

We consider the following verification problems:

P1 Given two stables configurations $p\gamma$ and $p'\gamma'$, determine if $p\gamma \Rightarrow^* p'\gamma'$.

P2 Given two stables configurations $p\gamma$ and $p'\gamma'$, and given a set of stable configurations s, determine if $p\gamma \Rightarrow_s^* p'\gamma'$.

P3 Given two stables configurations $p\gamma$ and $p'\gamma'$, and given an interface σ, determine if $p\gamma \overset{\sigma}{\Longrightarrow} p'\gamma'$.

P4 Given two stables configurations $p\gamma$ and $p'\gamma'$, given a set of stable configurations s, and an interface σ, determine if $p\gamma \overset{\sigma}{\Longrightarrow}_s p'\gamma'$.

Question 1: Assuming that we have an algorithm for solving P1 (P2 respectively), give an algorithm for solving P3 (P4 respectively).

Question 2: Is P2 decidable?

We consider now 2-stack pushdown systems (2-PDS for short). They are tuples $(P, \Gamma, \Delta_1, \Delta_2)$ where P is a finite set of control states, Γ is a finite stack alphabet, and Δ_i, for $i \in \{1, 2\}$, is a set of transitions rules of the form $p\gamma \hookrightarrow_i p'w$. Intuitively, 2-PDS's are models of shared memory concurrent programs with two parallel threads.

A configuration is an element $p(w_1, w_2)$ of $P(\Gamma^* \times \Gamma^*)$. The transition relation between configurations is obtained by applying nondeterministically rules from either Δ_1 or Δ_2. The application of a rule in Δ_1 modifies the control state and the content of the first stack, whereas the application of a rule in Δ_2 modifies the control state and the content of the second stack.

Given two configurations c and c', we write $c \Rightarrow c'$ if c' is obtained from c by applying a rule in $\Delta_1 \cup \Delta_2$. As usual, $\Rightarrow^*$ is the reflexive-transitive closure of $\Rightarrow$.

A configuration of a 2-PDS is stable if it is of the form $p(\gamma_1, \gamma_2)$, where $p \in P$, and $\gamma_1, \gamma_2 \in \Gamma$. Notice again, that the number of stable state is finite in this case too.

We generalise the definition of $\Rightarrow_s$ from PDS to 2-PDS in the straightforward way.

We consider the following problems:

P5 Given two stable configurations $p(\gamma_1, \gamma_2)$ and $p'(\gamma_1', \gamma_2')$, and given a set of stable configurations s, determine if $p(\gamma_1, \gamma_2) \Rightarrow_s^* p'(\gamma_1', \gamma_2')$.

P6 Given two stable configurations $p(\gamma_1, \gamma_2)$ and $p'(\gamma_1', \gamma_2')$, determine if is possible to have an infinite computation path (i.e., and infinite sequence of $\Rightarrow$ transitions) from $p(\gamma_1, \gamma_2)$ that visits $p'(\gamma_1', \gamma_2')$ infinitely often.

Question 3: Show that P5 is decidable.

Question 4: Show that P6 is decidable.