Exam – Initiation à la vérification

March 6, 2013

Duration: 2.5 hours. All course materials can be used. Answers can be given in either French or English. Justify all your answers. The numbers given in brackets indicate the estimated difficulty or length of the exercise.

1. Consider an \( \omega \)-automaton \( A = \langle S, I, \delta, \mathcal{F} \rangle \), where \( S \) the finite set of states, \( I \subseteq S \) the set of initial states, \( \delta \subseteq S \times S \) the transition relation and \( \mathcal{F} \) an acceptance condition. (We omit the alphabet, which will be unimportant.) A path from state \( s \in S \) is a sequence \( \rho \in S^\omega \) if \( \rho(0) = s \) and \( (\rho(i), \rho(i + 1)) \in \delta \) for all \( i \geq 0 \). For a path \( \rho \), we denote by \( \inf(\rho) \) the set of states occurring infinitely often in \( \rho \). A run is a path from any \( s \in I \). We write \( s \rightarrow s' \) if \( (s, s') \in \delta \) and \( 
rightarrow^+, \rightarrow^\ast \) for the transitive and transitive/reflexive closure of \( \rightarrow \).

\( A \) is a Büchi automaton if \( \mathcal{F} = F \), where \( F \subseteq S \); in that case \( A \) is non-empty if there exists a run \( \rho \) where \( \inf(\rho) \cap F \neq \emptyset \). \( A \) is a generalized Büchi automaton if \( \mathcal{F} = \{ F_1, \ldots, F_k \} \), for some \( k \geq 1 \), and \( A \) is non-empty if there exists a run \( \rho \) where for all \( 1 \leq i \leq k \) we have \( F_i \cap \inf(\rho) \neq \emptyset \).

We study “symbolic algorithms” for \( A \). A symbolic algorithm consists of a collection of functions that can have local variables (no global variables will be needed). All variables are integers or subsets of \( S \) (representable, e.g., by BDDs). The following operations on sets are allowed: comparison for subset inclusion (\( \subseteq \)) or (in-)equality (\( =, \neq \)), as well as union, intersection, complement (w.r.t. \( S \)), and predecessors w.r.t. \( \delta \), i.e. \( \text{pre}(M) \), where \( M \) is a set, is defined as \( \{ s \mid s' \in M : (s, s') \in \delta \} \). Additionally, \( \text{pick}(M) \), where \( M \) is non-empty, returns a singleton set containing some random \( s \in M \). Variables/symbolic constants for certain sets, such as \( \emptyset, S, I, F, F_i \), can be supposed as given.

As an example, an algorithm for computing the set of transitive predecessors for a set \( M \) could be as follows:

```plaintext
function prestar(set M):
    var set P := M, Q := \emptyset;
    while P \neq Q do
        Q := P;
        P := P \cup \text{pre}(P);
    od
    return P
```

We can similarly assume the existence of functions \( \text{post} \) and \( \text{poststar} \) for the successors. Your task is to write some algorithms for solving problems on \( A \). If you cannot solve one task, you may still assume the existence of a solution for the next task.

(a) State \( s \) is said to touch set \( G \) twice if there is a path from \( s \) containing at least two occurrences of \( G \), i.e. there are \( s', s'' \in G \) such that \( s \rightarrow^* s' \rightarrow^* s'' \). Write a function \( \text{tworeach}(G) \) that computes all those states that touch \( G \) twice.
(b) Given a singleton set \( M = \{s\} \), write a function \( \text{scc}(M) \) that computes the strongly connected component containing \( s \).

(c) Assuming \( A \) is a Büchi automaton, write a function \( \text{buchi} \) for checking whether \( A \) is empty.

(d) Assuming \( A \) is a generalized Büchi automaton, write a function \( \text{genbuchi} \) for checking whether \( A \) is empty.

Hint: (c) and (d) can be solved with or without taking recourse to SCCs. Extra points if you can provide solutions for both.

2. We consider Petri nets \( \langle P, T, F, W, M_0 \rangle \) with places \( P = \{p_1, \ldots, p_n\} \) and transitions \( T = \{t_1, \ldots, t_k\} \) for some \( n, k > 0 \). The incidence matrix \( C \) of such a net is an \( n \times k \) integer matrix with \( C_{i,j} = W(t_j, p_i) - W(p_i, t_j) \), i.e. the difference that firing \( t_j \) makes for the number of tokens on \( p_i \). A marking (such as \( M_0 \)) can be seen as an \( n \)-column vector. For a firing sequence of transitions, its Parikh vector is a \( k \)-column vector denoting how often each transition was fired.

As an example, consider the net \( N \) shown below. Its initial marking is \( \langle 1, 1, 0, 0, 0 \rangle \), and the Parikh vector of the firing sequence \( t_3 t_2 t_4 t_3 \) is \( \langle 0, 1, 2, 1, 0 \rangle \) (note that in this particular case \( n = k \)).

An invariant is an \( n \)-column vector \( x \) such that all entries of \( x \) are non-negative integers and \( C^\top x = 0 \). A trap is a set of places \( S \) such that \( S^\bullet \subseteq ^\bullet S \). \( S \) is initially marked if \( S \cap M = \emptyset \).

(a) Prove that, for any invariant \( x \) and reachable marking \( M \), the equation \( Mx = M_0x \) holds.

(b) Prove that if \( S \) is an initially marked trap, then \( S \cap M_0 = \emptyset \) for any reachable marking \( M \).

(c) In the net \( N \) shown above, give examples of an initially marked trap and two (linearly independent and non-null) invariants.

(d) Using invariants and traps, prove that \( p_3 \) and \( p_4 \) cannot be marked concurrently in any reachable marking.

(e) Unfold \( N \) using some adequate order to obtain a finite complete prefix. When drawing your unfolding, please indicate the order in which events were added and which markings are associated with each event.
3. Let \( \langle P, \Gamma, \Delta \rangle \) be a pushdown system where any rule \( pA \rightarrow qw \) satisfies \( |w| \leq 2 \), i.e. the stack height can change by at most one in each rule.

Let \( m \) be the number of different triples \( p, A, C \) such that there exists a rule \( pA \rightarrow qBC \), and let \( n \) be the number of different control states \( q \) such that there exists a rule \( pA \rightarrow q \).

(a) Let \( p, A, q \) be a triple such that \( pA \Rightarrow q \) holds, i.e. from the control state \( p \) with only \( A \) on the stack we can go to control state \( q \) with the empty stack. Show that there exists a sequence of rules from \( pA \) to \( q \) where the stack height never exceeds \( O(nm) \).

(b) Find an example PDS with a triple \( p, A, q \) such that any execution leading from \( pA \) to \( q \) has to pass through a configuration of stack height at least \( m \cdot n \). Hint: it works with a two-letter alphabet.

4. Consider the labeled Kripke structure \( K \) shown below with actions \( \{ A, B, C, D, E \} \) and one atomic proposition \( q \), where \( q \) holds only on state 4.

(a) Determine a maximal independence relation \( I \).

(b) Indicate which actions are visible and which are not.

(c) Compute a reduction function \( \text{red} \) that satisfies the conditions C0–C3 explained in the course. Wherever possible, \( \text{red}(s) \) should be a strict subset of \( \text{en}(s) \), for each state \( s \) of \( K \). Draw the reduced structure \( \text{red}(K) \).