Second Exam – Introduction to Verification

January 19, 2024

Answers can be given in either French or English. Justify all your answers. All documents are allowed, in non-electronic form.

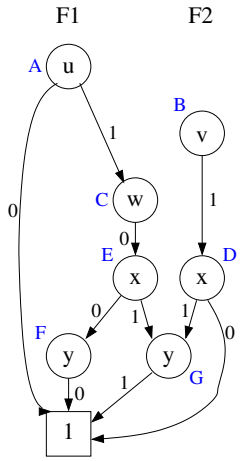
1 Binary decision diagrams

Let us consider variables with the order $u < v < w < x < y$. Let F_1, F_2 be propositional formulae with

$$F_1 := u \to (\neg w \wedge (x \leftrightarrow y)) \qquad F_2 := v \wedge (x \to y)$$

(a) Draw BDDs for F_1 and F_2, using the given order. No justification for the construction is necessary. You may omit the 0-node and edges leading to it.

Solution:



Let F, G be any two formulae and x a variable. Using the Shannon partitioning $F = ite(x, F[x/1], F[x/0])$ we derived the following equation for the *conjunction* of two formulae:

$$F \wedge G \equiv \begin{cases} F & \text{if } F \equiv G \\ 0 & \text{if } F = 0 \text{ or } G = 0 \\ G & \text{if } F = 1 \\ F & \text{if } G = 1 \\ ite(x, F[x/1] \wedge G[x/1], F[x/0] \wedge G[x/0]) & \text{otherwise} \end{cases}$$

(b) What is the corresponding equation for *implication* $F \to G$? (Hint: For $G = 0$, the result is $\neg F$, which can be obtained in linear time from F. There are four base cases where the result can be obtained in constant time.)

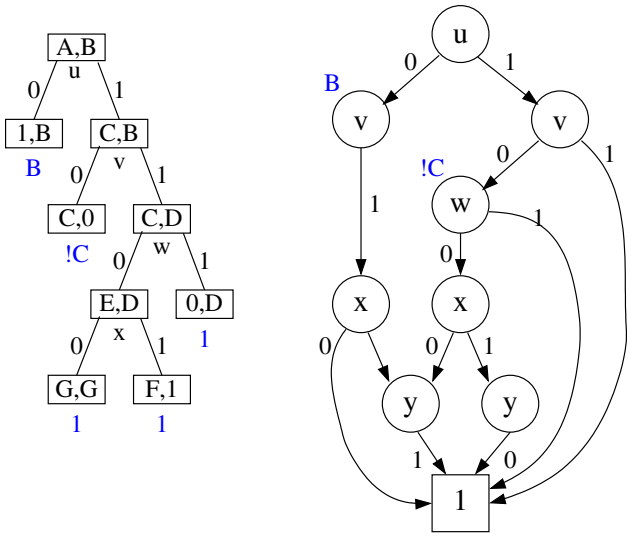
Solution:

1. The expected equation is as follows:

$$F \to G \equiv \begin{cases} 1 & \text{if } F = 0 \\ G & \text{if } F = 1 \\ \neg F & \text{if } G = 0 \\ 1 & \text{if } G = 1 \\ 1 & \text{if } F = G \\ ite(x, F[x/1] \to G[x/1], F[x/0] \to G[x/0]) & \text{otherwise} \end{cases}$$

(c) Using the equation from (b), construct a BDD for $F_1 \to F_2$.

Solution: Here is how the recursive algorithm would work: Each box indicates one recursive call, the names are the nodes from (a). Next to each box the result is denoted ('!' stands for negation). The five terminal cases correspond to those of the equation in (b).



The resulting BDD is shown on the right-hand side. Notice that the right-hand side of the box (C, B) collapses to 1. The negation of node C has to be made "manually".

2 Reachability in Petri nets.

Let $N = \langle P, T, F, W, m_0 \rangle$ be a Petri net. We say that N is *acyclic* if the directed graph $\langle P \cup T, F \rangle$ does not contain any cycles. Let $\mathcal{A}$ denote the class of Petri nets that are (i) 1-safe and (ii) acyclic.

(a) Show that in no firing sequence of a net $N \in \mathcal{A}$ can contain the same transition twice.

Solution: During the exam, I realized that there was a slight mistake in this question: If a transition has empty preset *and* empty postset, then it could fire repeatedly while leaving the marking unchanged. So let us assume for now that every transition is connected to at least one arc.

Then 1-safeness implies that no transition has an empty preset, otherwise the transition could produce infinitely many tokens. Therefore, if a transition t could fire twice in some execution, then some place $p \in {}^\bullet t$ must receive two tokens during the execution.

Since the net is acyclic, such an execution must have the form $\sigma t \tau$, where $\sigma, \tau \in T^*$ are firing sequences, such that after σ there is a first token on p (which t can comsume), and

such that after τ there is another token on p. Let τ' denote the subsequence of τ that contains the predecessors of t, and let τ'' denote the subsequence composed of all others. Now, τ' must contain a transition that puts a token onto p.

Again, since the net is acyclic, t is not predecessor of any element of τ' (nor, by transitivity, is any element of τ''). Therefore, $\sigma\tau'$ is also a valid firing sequence that places two tokens on p; but then N would not be 1-safe.

(b) Show that the reachability problem for the class $\mathcal{A}$ is in NP.

Solution: To show that the reachability problem for $\mathcal{A}$ is in NP, we must show that a non-deterministic machine can answer 'yes' for a positive instance in polynomial time.

If a net has a transition with empty preset and postset, that transition is irrelevant for reachability and can be ignored. Other than this, because of (a), any execution of N consists of at most $|T|$ transition firings.

A nondeterministic Turing machine can simulate some firing sequence with space $|P|$ (to remember the marking), answer 'yes' if the desired marking is reached, or 'no' if it is not reached after $|T|$ steps.

(c) Show that the reachability problem for the class $\mathcal{A}$ is NP-hard, by reduction from the SAT problem. (Without loss of generality, one can assume that a SAT formula is given in conjunctive normal form.)

Solution: Recall that SAT is a restriction of the QBF problem to existential quantification. In class, we discussed how to reduce QBF to the reachability problem for 1-safe nets; this reduction produced an almost acyclic net when only existential quantification was involved. Thus, it was sufficient to slightly modify that construction.

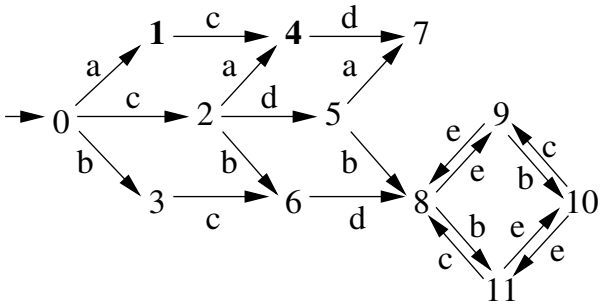
Let X be a set of Boolean variables and $F = \bigwedge_{i=1}^{k} \bigvee_{j=1}^{m_i} \ell_{i,j}$ be a formula in CNF, where the literals $\ell_{i,j}$ are positive or negative instances from X; denote by $\mathcal{L}$ the set of those literals. We construct the following net N_F:

- The places of N_F are $P_X \uplus P_L \uplus P_C \cup P'_C \uplus \{p\}$, where
  - $P_X := \{\, p_x \mid x \in X \,\}$ (one place per variable);
  - $P_L := \{\, p_{i,j} \mid \ell_{i,j} \in \mathcal{L} \,\}$ (one place per literal);
  - $P_C := \{\, c_i \mid 1 \le i \le k \,\}$ and $P'_C := \{\, c'_i \mid 1 \le i \le k \,\}$ (two places per clause).
- The initially marked places are $P_X \cup P_C$.
- The transitions are $T_\top \uplus T_\bot \uplus T_L \uplus \{t\} \uplus T'_L$, where
  - $T_\top := \{\, t^x_\top \mid x \in X \,\}$, ${}^\bullet t^x_\top = \{p_x\}$, ${t^x_\top}^\bullet = \{\, p_{i,j} \mid \ell_{i,j} = x \,\}$
    (choose to generate tokens for all positive x-literals);
  - $T_\bot := \{\, t^x_\bot \mid x \in X \,\}$, ${}^\bullet t^x_\bot = \{p_x\}$, ${t^x_\bot}^\bullet = \{\, p_{i,j} \mid \ell_{i,j} = \neg x \,\}$
    (choose to generate tokens for all negative x-literals);
  - $T_L := \{\, t_{i,j} \mid \ell_{i,j} \in \mathcal{L} \,\}$, ${}^\bullet t_{i,j} = \{c_i, p_{i,j}\}$, ${t_{i,j}}^\bullet = \{c'_i\}$
    (mark clause c_i as satisfied if any literal is marked);
  - ${}^\bullet t := \{\, c'_i \mid 1 \le i \le n \,\}$, $t^\bullet := \{p\}$
    (put a token on p if all clauses are satisfied);
  - $T'_L := \{\, t'_{i,j} \mid \ell_{i,j} \in \mathcal{L} \,\}$, ${}^\bullet t_{i,j} = \{p_{i,j}\}$, ${t_{i,j}}^\bullet = \emptyset$
    (allow the token for any literal to disappear).

It is easy to see that the size of N_F is polynomial (linear) w.r.t. the size of F and that the marking $\{p\}$ can be reached if and only if F is satisfiable. Moreover, the net N_F is acyclic and 1-safe.

3 Partial-order reduction

(a) Consider the Kripke structure shown below with actions a, b, c, d, e, where 0 is the initial state, and $AP = \{p\}$ such that p holds in states 1 and 4 only.



Determine all pairs of independent actions I and indicate which actions are visible and which are not.

Solution: (a, b) are not independent (see states 0,2,5), all other pairs of different actions are. Actions a and d are visible, the others invisible.

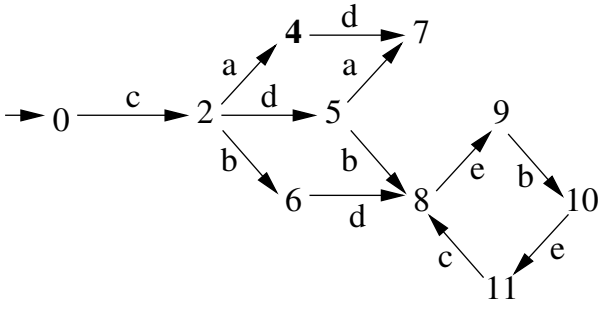
(b) Compute a reduction function red that satisfies the conditions C0–C3 explained in the course. Wherever possible, $red(s)$ should be a strict subset of $en(s)$, for each state s of $\mathcal{K}$. Draw the reduced structure $red(\mathcal{K})$.

Solution:

1. State by state: (without discussing the obvious case C0)

   - In state 0 we can reduce to c, which is invisible and independent of any other action. All other proper subsets would not work: $\{a, b\}$ because of C2, the others because of C1.
   - In state 2 we need to include all actions; again $\{a, b\}$ is prevented by C2, so is $\{d\}$; all others by C1.
   - In state 5 we need $\{a, b\}$ due to C1.
   - In states 4, 7, and 6, we take the only possible action.
   - In 8, 9, 10, 11, we can go either clockwise or counter-clockwise to obtain a loop containing all four states. (C1 and C2 are irrelevant here, and C3 prevents us from closing the loop too early.)

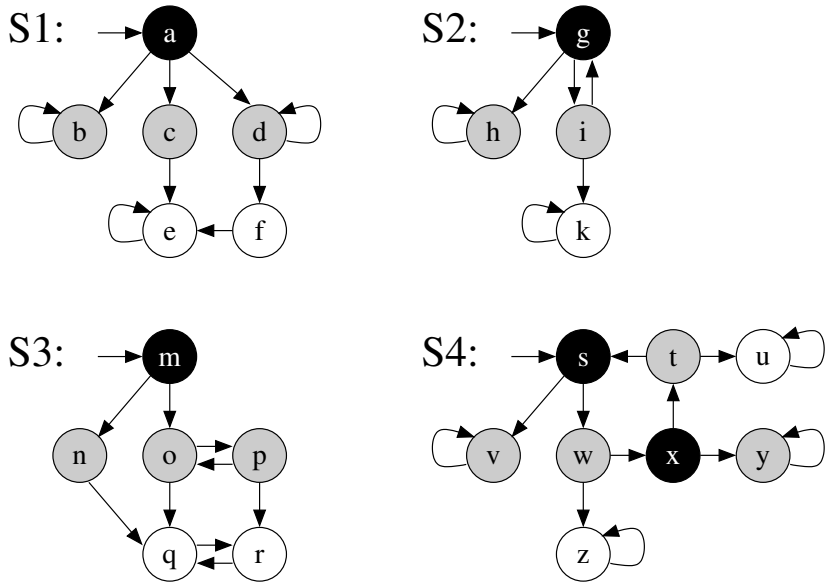
   A possible result:



4 Free the animals

The elephant and the marmot have been captured by the wizard Kripke. The wizard asks them to choose two of the structures shown below. The elephant will be placed on the initial state of one and the marmot on the initial state of the other. In each turn, the wizard will then instruct

either the elephant or the marmot to move to a neighbouring state of the wizard's choosing. The other animal must then respond by moving to a state of the same colour. (E.g., if at some point the elephant sits on o and the marmot on w, the wizard could instruct the elephant to move to q and the marmot could respond by moving to z.) If the wizard gets the two into a situation where they cannot respond with a correct move, the poor animals remain in captivity, but if they can play the game for at least ten moves, they will be let free.



(a) The animals need your help – which pair of structures should they choose, and how should they play? They have attended the ENS and know about bisimulation, so they need you to provide a bisimulation relation between the two structures.

For each other pair of structures, explain how the wizard would trap them, by giving a CTL formula distinguishing them. The CTL formula would consist of only the modalities EX and AX, and the atomic propositions *white*, *grey*, *black*, possibly negated.

Solution: The only pair of bisimilar structures, which will guarantee our heros' freedom, are S_2 and S_4, as demonstrated by the following relation:

$$\{(g,s),(g,x),(h,v),(h,y),(i,w),(i,t),(k,u),(k,z)\}$$

The CTL formula **EX EX** *black* is satisfied by S_2 and S_4 but not by S_1 or S_3, proving that none of these four pairs are bisimilar. (If our friends made such a choice, the wizard could tell one of them to go to a black state in the second move, which the other could not respond to.)

Finally, S_1 and S_3 are distinguished by the formula **EX AX** *grey*. Thus, if our friends chose S_1 and S_3, the wizard could first tell the elephant to go to b, and whatever the marmot responds, he would tell it to move to q.

(b) Do the elephant and the marmot have another choice if they know in advance that the wizard will always instruct the same animal to move?

For each pair of structures that are not bisimilar, check whether there exists a simulation one way or the other. If a simulation relation exists, provide it, otherwise exhibit an LTL formula satisfied by one structure but not the other.

Solution: There is no simulation from either S_1 or S_3 to either S_2 or S_4: All runs in the latter satisfy the LTL formula (**X X** *grey*) $\rightarrow$ (**G** $\neg$*white*) but the former do not. There is

also no simulation in the reverse direction; all runs in S_1 and S_3 satisfy $\mathbf{X}\,\mathbf{G}\,\neg \mathit{black}$ but S_2 and S_4 do not.

Finally, while S_1 and S_3 are not bisimilar, there are simulations both ways. From S_1 to S_3, one can take

$$\{(a,m)\} \cup \big(\{b,c,d\} \times \{o,p\}\big) \cup \big(\{e,f\} \times \{q,r\}\big),$$

and from S_3 to S_1, one can take

$$\{(m,a),(n,c),(o,d),(p,d)\} \cup \big(\{q,r\} \times \{e,f\}\big).$$