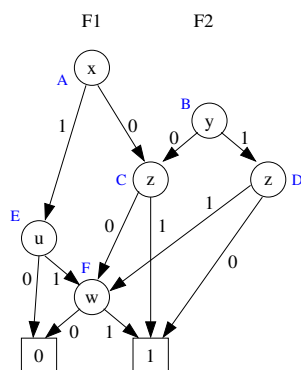


# Examen – Initiation à la vérification – Solutions

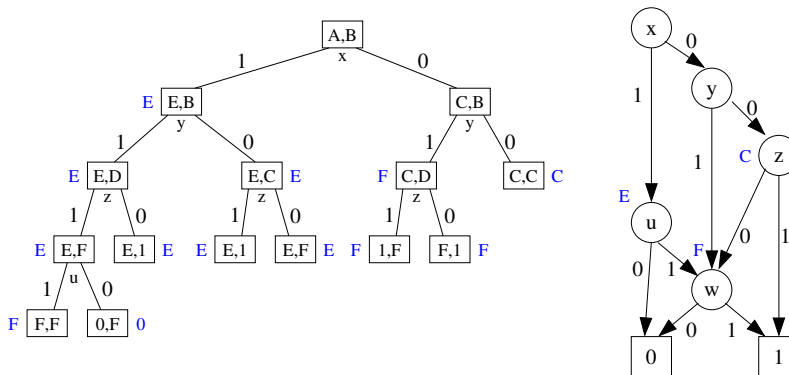
8 février 2010

- Generally, one can obtain those BDDs by multiple methods, e.g. recursive substitution or simply ad hoc (no single algorithm was proscribed).

(a)  $F_1$  becomes easy once one realizes that  $F_1[x/1] = u \wedge w$  and  $F_1[x/0] = w \vee z$ . In the drawing below, common subgraphs of the two BDDs are already shared.

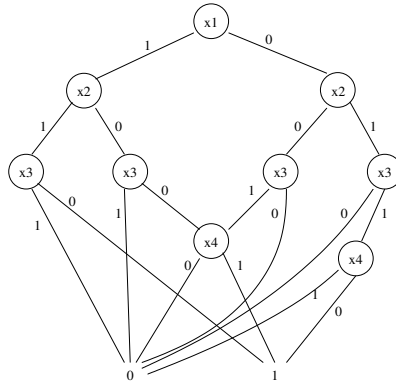


(b) Here's how the recursive algorithm would work in this case: Each box indicates one recursive call, the names are the nodes from (a). Next to each box the result is denoted (in those cases where it corresponds to a node that's already known). Note that the result for  $(E, F)$  is  $E$  because  $E$  has children  $F$  and  $0$ ; this result is used twice.



- Reminder: In the following, we count only those nodes that are labelled by variables.

(a) With four variables, one can have at most 9 nodes, with 1, 2, and 4 nodes at levels  $x_1$  to  $x_3$ . At the last level, one can have only two nodes; the 1-labelled arc must go to one of the leaves, the 0-labelled arc to the other.

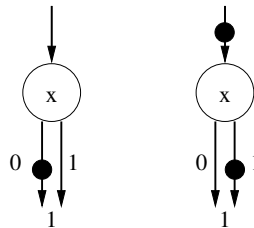


(b) With five or six variables, the number of nodes at the last level remains at most 2, let us call these nodes  $a$  and  $b$ . At the penultimate level, one has at most  $4 \cdot 3 = 12$  nodes; four choices for the 1-labelled edge  $(a, b, 1, 0)$  and the remaining three for the 0-labelled edge. Thus, for five variables, one has  $1 + 2 + 4 + 8 + 2 = 17$  nodes, and for six variables,  $1 + 2 + 4 + 8 + 12 + 2 = 29$  nodes.

(c) Generally, the number of nodes at level  $i$  is bounded by two factors: from “above”, i.e. the branching degree 2 of a BDD means that one can have at most  $2^{i-1}$  nodes; and from “below”, meaning that the number of possible targets for the outgoing edges is limited. Both the 1-edge and the 0-edge point to some other BDD node, which represents a Boolean function of  $n - i$  variables. Since there are  $S_i^n := 2^{2^{n-i}}$  such functions, there are at most  $S_i^n$  targets for the 1-edge, and the 0-edge must point to one of the other  $S_i^n - 1$  targets. Thus, we have  $N_i^n = \min\{2^{i-1}, S_i^n \cdot (S_i^n - 1)\}$ .

3. General remark: Every positive negation bit inverts the meaning of the CBDD starting at its target. Thus, to decide whether a given valuation is accepted by a CBDD, one can count the number of negations on the corresponding path from the initial edges to the unique leaf. If it is even, the answer is yes, otherwise no.

(a) To show non-uniqueness, it suffices to provide an alternative for  $x$ , the function represented by CBDD  $A$ :

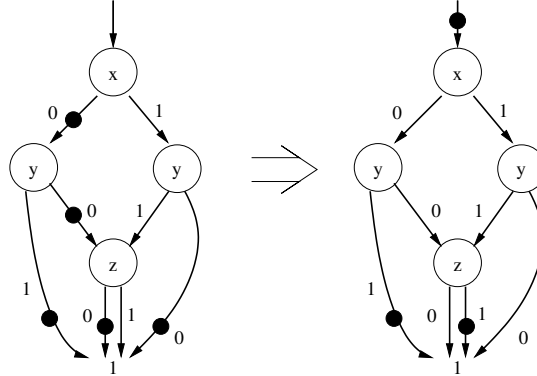


(b) Suppose that we have a node  $n$  whose outgoing 0-edge has a positive negation bit. We mentioned above that only the parity of positive negation bits matters along each path. Now, if we invert the negation bit on all adjacent edges of  $n$  (the 0-edge, the 1-edge, and all incoming edges), the parity on all paths remains the same.

On a logical level, if  $n$  is labelled by  $x$ , then the CBDD starting at  $n$  represents a formula  $ite(x, F, G)$  (for some suitable  $F, G$ ). This is equivalent to  $\neg ite(x, \neg F, \neg G)$ , which in turn is represented by the result of the transformation.

Naturally, this procedure may create negated 0-edges that did not exist before. However, these appear only on edges that are “higher up”, i.e. closer to the root. Therefore, one starts by successively eliminating those negated 0-edges that are closest to the leaves. One can easily see that this terminates. (E.g., if one wants to be formal, then one can consider the tuples  $(n, m)$  with lexicographic ordering, where  $m$  is the highest variable index of a node with an outgoing 0-edge, and  $n$  is the number of such nodes; this tuple strictly decreases during each operation.)

- (c) For  $A$ , see above. For  $C$ , see below, it is the result of the procedure from (b).



4. (a) The only pair of bisimilar structures 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  $\mathbf{EX EX} \alpha$  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. Finally,  $S_1$  and  $S_3$  are distinguished by the formula  $\mathbf{EX AX} \beta$ .

- (b) There is no simulation from either  $S_1$  or  $S_3$  to either  $S_2$  or  $S_4$ : The latter satisfy the LTL formula  $(\mathbf{X X} \beta) \rightarrow (\mathbf{G} \neg \gamma)$  but the former do not. There is also no simulation in the reverse direction;  $S_1$  and  $S_3$  satisfy  $\mathbf{X G} \neg \alpha$  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

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

and from  $S_3$  to  $S_1$ , one can take

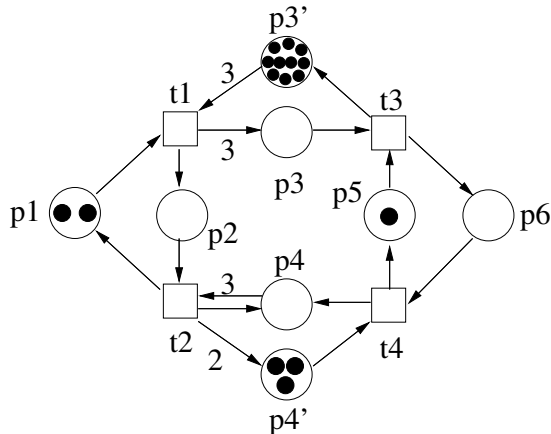
$$\{(m, a), (n, c), (o, d), (p, d)\} \cup (\{q, r\} \times \{e, f\}).$$

5. In a P/T net, let us call two places  $p, p'$  *complementary* if there exists a bound  $k$  such that in every reachable marking  $M$  we have  $M(p) + M(p') = k$ . For this to hold, every transition must remove from  $p'$  as many tokens as it adds to  $p$  and vice versa.

If we interpret the given net as a P/T net (i.e., by removing the capacities), then  $p_1$  and  $p_2$  are complementary, and so are  $p_5$  and  $p_6$ .

Note that if  $p, p'$  are complementary with bound  $k$ , then both  $p$  and  $p'$  always have between 0 and  $k$  tokens. This means that  $p_1, p_2, p_5, p_6$  already

respect the given capacities. To achieve the same for  $p_3$  and  $p_4$ , we add new places  $p'_3$  and  $p'_4$  and modify all transitions according to the condition mentioned above to make them complements of  $p_3$  and  $p_4$ . Initially, they get 10 and 3 tokens, respectively. The result is shown below.



There is a little catch here w.r.t.  $p'_4$  and  $t_2$ : it is not correct to draw back-and-forth arrows with weights 1 and 3. This would require 4 tokens from  $p_4$  and  $p'_4$  together, meaning that  $t_2$  could never fire.

6. (a) The incidence matrix of the net is given below:

$$\begin{pmatrix} & t_1 & t_2 & t_3 & t_4 & t_5 \\ p_1 & 0 & 1 & -1 & 0 & 0 \\ p_2 & -1 & 0 & -1 & 1 & 0 \\ p_3 & 1 & 0 & 0 & 0 & -1 \\ p_4 & 0 & -1 & 1 & 0 & 0 \\ p_5 & 0 & 0 & 1 & -1 & 1 \end{pmatrix}$$

Farkas' algorithm yields two P-invariants  $\langle 1, 0, 0, 1, 0 \rangle^T$  and  $\langle 0, 1, 1, 0, 1 \rangle^T$ . Since initially there are two tokens in the net, we get for all reachable markings  $M$ :

$$M(p_1) + M(p_2) + M(p_3) + M(p_4) + M(p_5) = 2$$

Now,  $S := \{p_1, p_2, p_5\}$  is a trap since in fact every transition puts at least one token into  $S$ . Since  $S$  is marked initially, we get for every reachable marking  $M$ :

$$M(p_1) + M(p_2) + M(p_5) \geq 1$$

The desired property  $M(p_3) + M(p_4) \leq 1$  follows from subtracting these two (in-)equalities.

- (b) For the unfolding, there are two different legal outcomes, depending on the order in which one handles  $t_2$  and  $t_5$ . One possible outcome is shown below, where cut-offs are indicated by dashed lines.

