Assessed Coursework: Systems Verification

Ioannis Kassinopoulos

February 12, 2013

Question 1

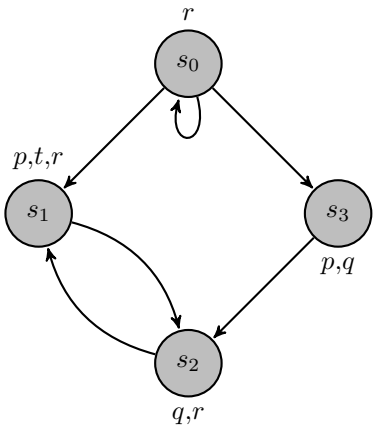

Figure 1: The transition system $\mathcal{M}_1$.

Algebraic Form

A transition system $\mathcal{M} = (S, \rightarrow, \pi)$ is a set of states S endowed with a transition relation $\rightarrow$ (a binary relation on S), such that every $s \in S$ has some $s' \in S$ with $s \rightarrow s'$, and an inverse labeling function $\pi : \mathcal{P} \rightarrow S$.

Our system $\mathcal{M}_1$ (figure: 1) can be described as following:
$\mathcal{P} = \{p, q, r, t\}$
$\mathcal{M}_1 = \{\{s_0, s_1, s_2, s_3\}, \{(s_0, s_0), (s_0, s_1), (s_0, s_3), (s_1, s_2), (s_2, s_1), (s_3, s_1)\}, \pi\}$
$\pi(p) = \{s_1, s_3\}$
$\pi(q) = \{s_2, s_3\}$
$\pi(r) = \{s_0, s_1, s_2\}$
$\pi(t) = \{s_1\}$

Infinite Tree

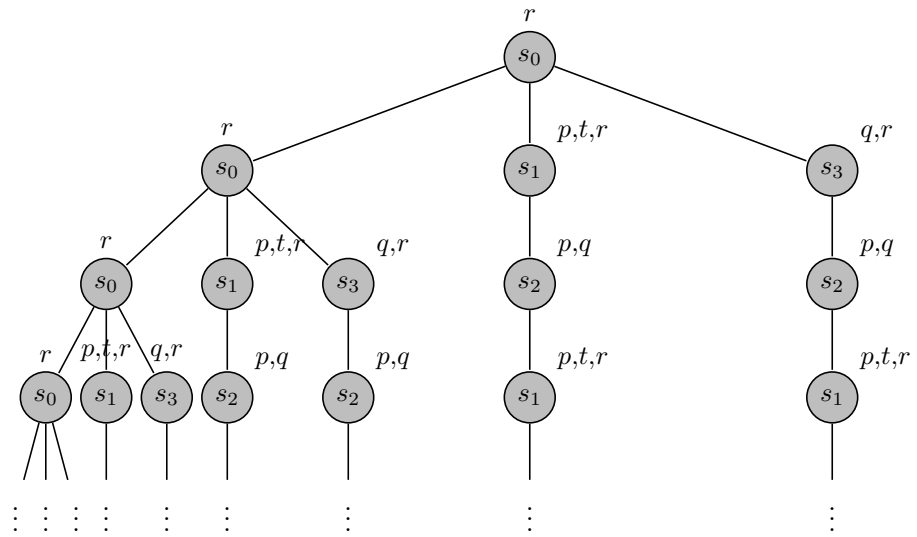

Figure 2: Unwinding the system described by $\mathcal{M}_1$ as an infinite tree of all computation paths beginning in s_0 (first layer).

Satisfiability

Question 2

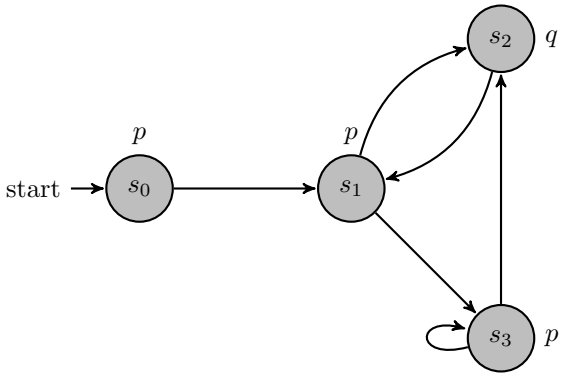

Figure 3: The transition system $\mathcal{M}_2$.

Calculating

$\phi = p$

$SAT(p) = \{s \in S \mid p \in L(s)\} = \{s_0, s_1, s_2\}$

$\Rightarrow [\![\phi]\!] = \{s_0, s_1, s_2\}$

$\phi = AGEFp$

$SAT(EFp) = SAT(E[trueUp]) = SAT_{eu}(true, p)$

$SAT_{eu}(true, p)$

$W = \{s_0, s_1, s_2, s_3\}$

$Y_0 = \{s_0, s_1, s_3\}$

$Y_1 = \{s_0, s_1, s_2, s_3\}$

$Y_2 = Y_1$

$SAT(AGEFp) = SAT(\neg EF \neg EFp)$

$SAT(EF\neg EFp) = SAT(E[trueU\neg EFp]) = SAT_{eu}(true, \neg EFp)$

$SAT_{eu}(true, \neg EFp)$

$W = \{s_0, s_1, s_2, s_3\}$

$Y_0 = S - \{s_0, s_1, s_2, s_3\} = \{\}$

$Y_1 = Y_0$

$SAT(AGEFp) = SAT(\neg EF \neg EFp) = S - SAT(EF\neg EFp) = \{s_0, s_1, s_2, s_3\}$

$\Rightarrow [\![\phi]\!] = \{s_0, s_1, s_2, s_3\}$

$\phi = AFq$

$\phi = AGp \vee Afq$

$\phi = E(pU(AFq))$

Question 3

Question 4

Let $\phi = (x_1 \wedge x_2) \vee (y_1 \wedge y_2)$, the following truth table is derived to help us with our calculations

x_1	x_2	y_1	y_2	$x_1 \wedge x_2$	$y_1 \wedge y_2$	$(x_1 \wedge x_2) \vee (y_1 \wedge y_2)$
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	0	0	0
0	0	1	1	0	1	1
0	1	0	0	0	0	0
0	1	0	1	0	0	0
0	1	1	0	0	0	0
0	1	1	1	0	1	1
1	0	0	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	0	0	0
1	0	1	1	0	1	1
1	1	0	0	1	0	1
1	1	0	1	1	0	1
1	1	1	0	1	0	1
1	1	1	1	1	1	1

Binary Decision Tree

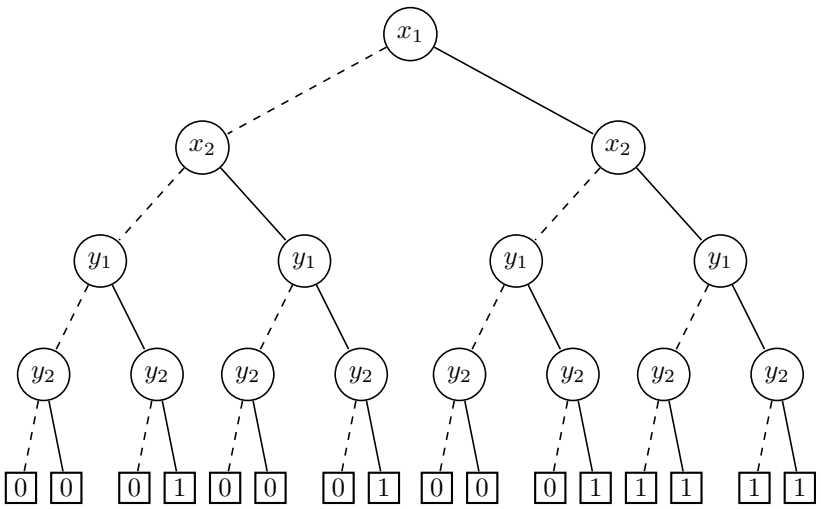

Figure 4: A BDT is easily derived from the truth table. Every non-terminal node is labelled with a variable and every terminal node is labelled with either 0 or 1.

Reduced Ordered Binary Decision Diagrams

In order to reduce the size of the BDT we can produce a Binary Decision Diagram which is a reduced form of the BDT. Making this diagram ordered

over a list of variables, results in getting an Ordered Binary Decision Diagram (OBDD) which is then unique when it is reduced until no more reduction can occur. This reduced form is called canonical form and it can be used to extract equivalences since two different but equivalent Boolean functions always have identically structured Reduced Ordered Binary Decision Diagrams if they have compatible variable orderings.

Reduction Algorithm

In order to reduce BDTs we use iteratively the rules C1-C3 until no more reductions can occur.

- **C1:** Removal of duplicate terminals.
- **C2:** Removal of redundant tests.
- **C3:** Removal of duplicate non-terminals.

ROBDD under the $[x_1, x_2, y_1, y_2]$ ordering.

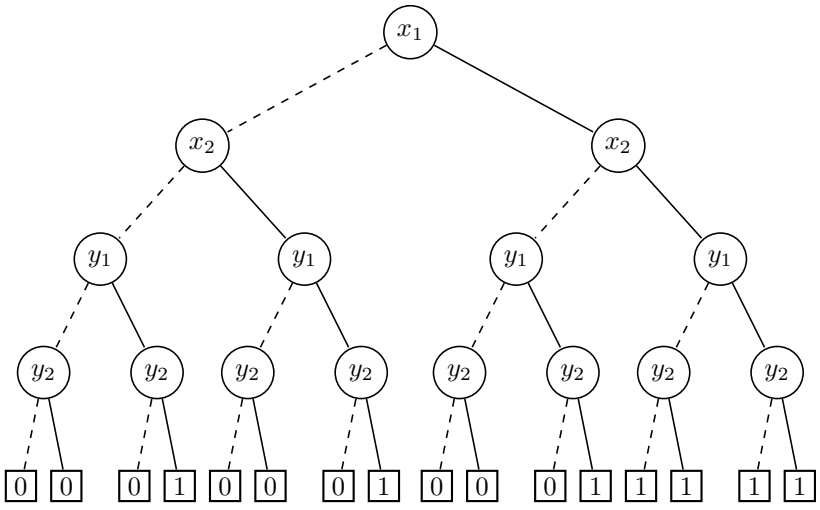

Figure 5: We start with the BDT over our ordering.

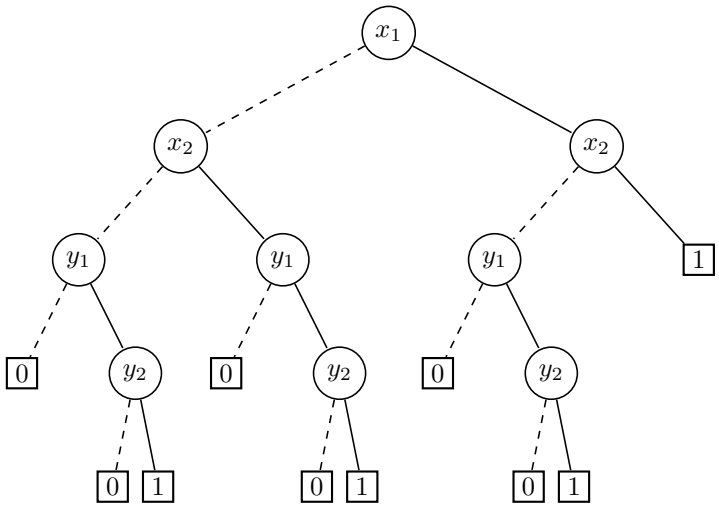

Figure 6: Using C2 we remove the redundant tests and eliminate the nodes leading to them. We derive the above reduced diagram.

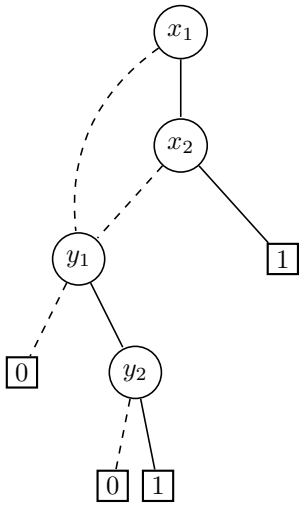

Figure 7: Using C3 we remove the duplicate non-terminals, redirect the incoming edges and derive the above reduced diagram.

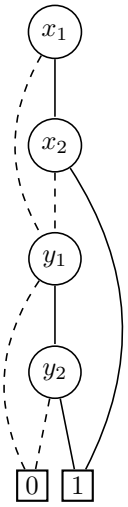

Figure 8: Using C1 we remove all the duplicate terminals. This OBDD cannot be reduced any further so we can now call it the canonical of the previous diagrams.

ROBDD under the $[x_1, y_1, y_2, x_2]$ ordering.

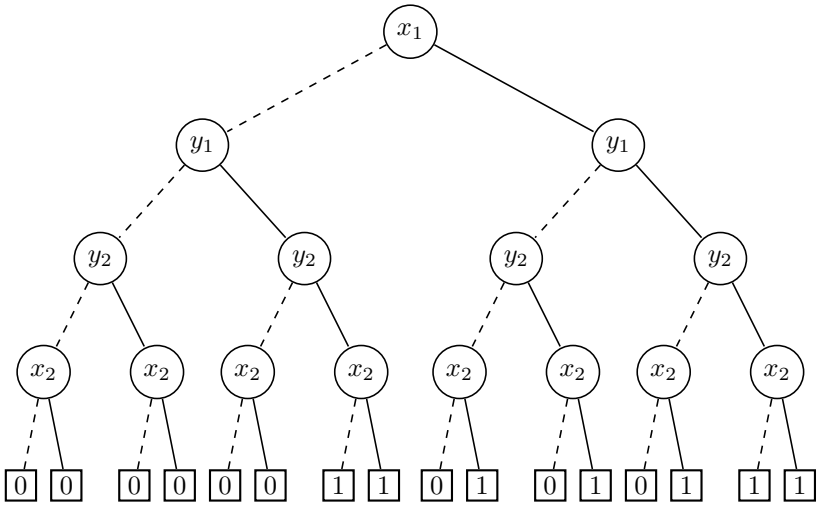

Figure 9: We start with the BDT over our ordering.

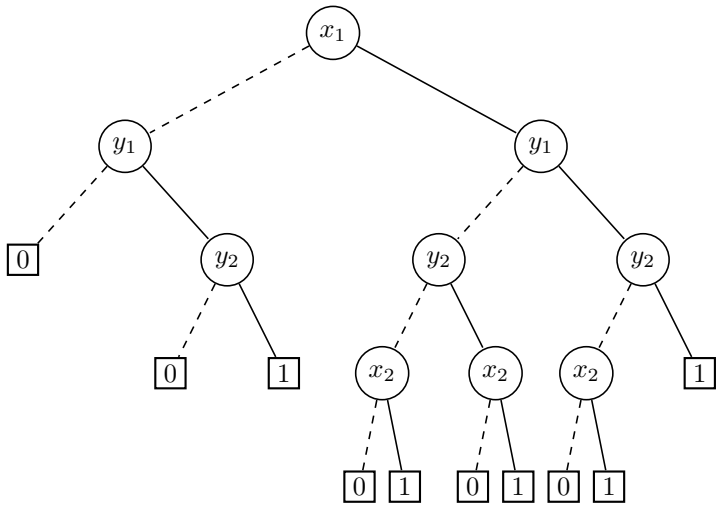

Figure 10: Using C2 we remove the redundant tests and eliminate the nodes leading to them. We derive the above reduced diagram.

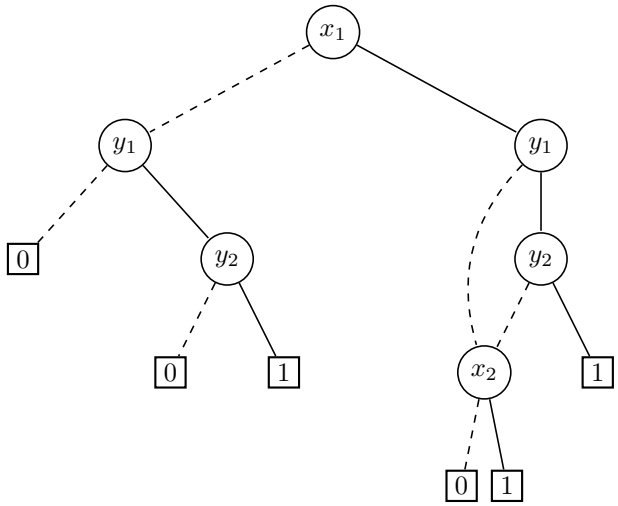

Figure 11: Using C3 we remove the duplicate non-terminals, redirect the incoming edges and derive the above reduced diagram.

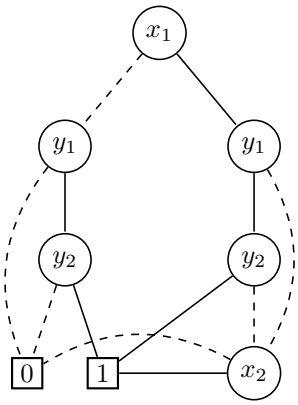

Figure 12: Using C1 we remove all the duplicate terminals. This OBDD cannot be reduced any further so we can now call it the canonical of the previous diagrams.

Discuss how ordering impacts ROBDD

Suggest an algorithm for choosing ordering