Intelligent Systems Programming

Lecture 5: **Boolean Expression Representations & Binary Decision Diagrams (BDDs)**

Today’s Program

- **[10:00-10:55] Computation process representations**
  - Boolean expressions and Boolean functions
  - Desirable properties of representations of Boolean expressions
  - Classical representations of Boolean expressions
    - Truth tables
    - Two-level normal forms: CNF, DNF
    - Multi-level representations: circuits and formulas
- **[11:05-12:00] Evaluation process representations**
  - If-then-else normal form (INF)
  - Decision trees
  - Ordered Binary Decision Diagrams (OBDDs)
  - Reduced Ordered Binary Decision Diagrams (ROBDDs / BDDs)

Boolean Expressions

- Boolean Expressions

  $t \ ::= x \mid 0 \mid 1 \mid \neg t \mid t \wedge t \mid t \vee t \mid t \Rightarrow t \mid t \Leftrightarrow t$

- Literals

  $l \ ::= x \mid \neg x$

- Precedence

  $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$ (obs. two last swapped in HRA notes)

- Terminology
  - **Boolean expression** = Boolean formula/Propositional formula/Sentence in propositional logic
  - **Boolean variable** = Propositional symbol/letter/variable

Boolean Functions

- **Definition**

  An n-ary function $f : \boldsymbol{B}^n \rightarrow \boldsymbol{B}$

  $$f(x_1,x_2,\ldots,x_n) = E(x_1,x_2,\ldots,x_n),$$

  where E is a Boolean expression

- **Example**

  $$f(x_1,x_2,x_3) = x_1 \Leftrightarrow \neg x_2$$

Properties of Boolean Functions

- Equality

  $f = g \quad \text{iff} \quad \forall \boldsymbol{x} \,.\, f(\boldsymbol{x}) = g(\boldsymbol{x})$

- Several expressions may represent a function

  $f(x,y) = x \Rightarrow y = \neg x \vee y = (\neg x \vee y) \wedge (\neg x \vee x) = \ldots$

- Order of arguments matter

  $f(x,y) = x \Rightarrow y \quad \neq \quad g(y,x) = x \Rightarrow y$

- Number of Boolean functions $f : \boldsymbol{B}^n \rightarrow \boldsymbol{B}$

$$2^{\left(2^n\right)}$$

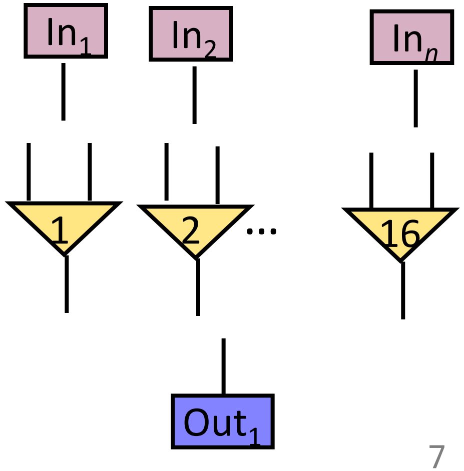
Desirable properties of a representation

1. Compact
2. Equality check easy
3. Easy to evaluate the truth-value of an assignment
4. Boolean operations efficient
5. SAT check efficient
6. Tautology check efficient
7. Canonicity: exactly one representation of each Boolean function
   – Solves 2, 5, and 6, why?

Compact representations are rare

- $2^{\left(2^{n}\right)}$ Boolean functions in *n* variables…
  - How do we find a single compact representation for them all?

    Ex. Boolean circuits : how many Boolean circuits of size g with n inputs and 1 output?
  - Choice of gates: $(16)^{g}$
  - Choice of connections: $(2g+1)^{(n+g)}$
  - Total: $(16)^{g}\,(2g+1)^{(n+g)}$



Compact representations are rare (cont.)

- Assume $g = O(n^k)$
- Thus the fraction of Boolean functions of n variables with a polynomial circuit size in n is

$$\frac{16^{O(n^k)}(2O(n^k)+1)^{(O(n^k)+n)}}{2^{2^n}} \to 0 \;\text{ for }\; n \to \infty$$

Curse of Boolean function representations:

This problem exists for all representations we know!

Classical Representations of Boolean Expressions

Truth tables

- Compact ☹ table size 2^n
- Equality check easy ☺ canonical
- Easy to evaluate the truth-value of an assignment ☺ log n or constant
- Boolean operations efficient ☺ linear
- SAT check efficient ☺ linear
- Tautology check efficient ☺ linear

x	y	z	$x \wedge y \vee z$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Two-level normal forms: DNF CNF

- Is there a DNF and CNF of every expression?
- Given a truth table representation of a Boolean formula, can we easily define a DNF and CNF of the formula?

x	y	z	e
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Two-level normal forms: DNF CNF

- Example DNF of e

x	y	z	e	
0	0	0	0	
0	0	1	1	$\neg x \wedge \neg y \wedge z \ \vee$
0	1	0	0	
0	1	1	1	$\neg x \wedge y \wedge z \ \vee$
1	0	0	0	
1	0	1	1	$x \wedge \neg y \wedge z \ \vee$
1	1	0	1	$x \wedge y \wedge \neg z \ \vee$
1	1	1	1	$x \wedge y \wedge z$

Two-level normal forms: DNF CNF

- Example CNF of *e*

x	y	z	e	
0	0	0	0	$\neg(\neg x \wedge \neg y \wedge \neg z) \;\wedge$
0	0	1	1	
0	1	0	0	$\neg(\neg x \wedge y \wedge \neg z) \;\wedge$
0	1	1	1	
1	0	0	0	$\neg(x \wedge \neg y \wedge \neg z) \;\wedge$
1	0	1	1	
1	1	0	1	
1	1	1	1	

Two-level normal forms: DNF CNF

- Example CNF of e

x	y	z	e	
0	0	0	0	$(x \vee y \vee z) \;\wedge$
0	0	1	1	
0	1	0	0	$(x \vee \neg y \vee z) \;\wedge$
0	1	1	1	
1	0	0	0	$(\neg x \vee y \vee z)$
1	0	1	1	
1	1	0	1	
1	1	1	1	

Two-level normal forms: DNF CNF

Every Boolean formula has a DNF and CNF representation

- The special version DNF and CNF representations produced from *on* and *off*-tuples are canonical and called cDNF and cCNF
- Are cDNF and cCNF minimum size DNF and CNF representations?

Two-level normal forms: DNF CNF

- Symmetry properties of DNF and CNF

	SAT	Tautology
CNF	NP complete	Polynomial (exercise)
DNF	Polynomial (exercise)	Co-NP complete

- Idea: Solve CNF-SAT by conversion to DNF-SAT
  – Problem: conversion between CNF and DNF may be exponential

Two-level normal forms: DNF CNF

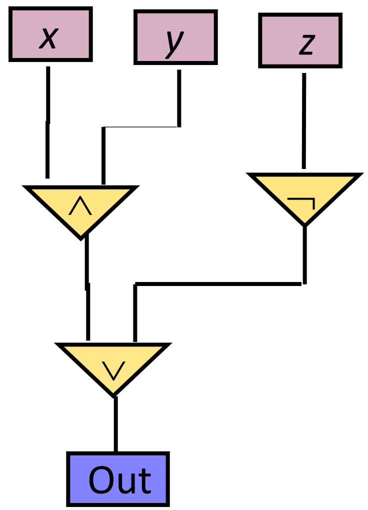
- Example
  - CNF $\left(x_0^1 \vee x_1^1\right) \wedge \left(x_0^2 \vee x_1^2\right) \wedge \cdots \wedge \left(x_0^n \vee x_1^n\right)$
  - Corresponding DNF blows up

$$
\begin{array}{l}
\left(x_0^1 \wedge x_0^2 \wedge \cdots \wedge x_0^{n-1} \wedge x_0^n\right) \vee \\
\left(x_0^1 \wedge x_0^2 \wedge \cdots \wedge x_0^{n-1} \wedge x_1^n\right) \vee \\
\qquad\qquad \vdots \\
\left(x_1^1 \wedge x_1^2 \wedge \cdots \wedge x_1^{n-1} \wedge x_0^n\right) \vee \\
\left(x_1^1 \wedge x_1^2 \wedge \cdots \wedge x_1^{n-1} \wedge x_1^n\right)
\end{array}
$$

Two-level normal forms: DNF CNF

- Compact Translating a formula to either CNF or DNF may cause an exponential blow-up in expression size
- Equality check easy ”compact” CNF and DNF formulas are not canonical
- Easy to evaluate the truth-value of an assignment linear
- Boolean operations efficient disjunction e.g. efficient for DNF
- SAT check efficient Not for CNF
- Tautology check efficient Not for DNF

Circuits



- Circuits are not canonical

Circuits

- Compact
- Equality check easy Not canonical
- Easy to evaluate the truth-value of an assignment linear
- Boolean operations efficient constant
- SAT check efficient (NP-Complete)
- Tautology check efficient Not canonical

Formulas

$$((x \vee y) \Rightarrow (z \vee y)) \vee ((x \vee y) \Rightarrow (z \vee x))$$

- Formulas are circuits where each node has out-degree 1
- Multiple use of intermediate functions is not allowed

Formulas

- Compact 😐 At least as large as a circuit
- Equality check easy ☹ Not canonical
- Easy to evaluate the truth-value of an assignment ☺ linear
- Boolean operations efficient ☺ constant
- SAT check efficient ☹ (Circuit-SAT $\leq_p$ Formula-SAT)
- Tautology check efficient ☹ Not canonical

Classical Representations

Observations

- There is a trade off between the size and accessibility of a representation
- **Truthtables**: very large, but all checks and Boolean operations can be carried out efficiently
- **Circuits**: quite compact, but SAT and equality checks are very hard

Next

Binary decision diagrams, with a good balance between size and accessibility!

Binary Decision Diagrams

Computation and Evaluation Based Representation

- **Computation Process Representation**
  - Representation gives a way to compute the expression value
  - Truth-tables, DNF, CNF, circuits and formulas
- **Evaluation Process Representation**
  - Representation gives a way to classify the expression value by means of assignments tests
  - Decision trees

If-then-else operator

- The *if-then-else* Boolean operator is defined by

$$x \to y_1, y_0 \;\equiv\; (x \wedge y_1) \vee (\neg x \wedge y_0)$$

- We have

$$(x \to y_1, y_0)[x/1] \;\equiv\; (1 \wedge y_1) \vee (0 \wedge y_0) \equiv y_1$$

$$(x \to y_1, y_0)[x/0] \;\equiv\; (0 \wedge y_1) \vee (1 \wedge y_0) \equiv y_0$$

If-then-else operator

- All operators in propositional logic can be expressed using only $\rightarrow$ operators with:
  - $\rightarrow$ expressions and 0 and 1 for y_1 and y_0
  - tests on un-negated variables

- What are *if-then-else* expressions for
  - $x, \neg x$
  - $x \wedge y$
  - $x \vee y$
  - $x \Rightarrow y$

If-then-else Normal Form (INF)

An *if-then-else* Normal Form (INF) is a Boolean expression build entirely from the if-then-else operator and the constants 0 and 1 such that all test are performed only on un-negated variables

- **Proposition**: any Boolean expression t is equivalent to an expression in INF

Proof:

$t \equiv x \rightarrow t[1/x], t[0/x]$ (Shannon expansion of t)

Apply the Shannon expansion recursively on t. The recursion must terminate in 0 or 1, since the number of variables is finite

Example

- Example: $t = (x_1 \wedge y_1) \vee (x_2 \wedge y_2)$
- Shannon expansion of t in order x_1, y_1, x_2, y_2

$$t \quad = x_1 \rightarrow t_1, t_0$$

Example

- Example: $t = (x_1 \wedge y_1) \vee (x_2 \wedge y_2)$
- Shannon expansion of t in order x_1, y_1, x_2, y_2

$$t = x_1 \rightarrow t_1, t_0$$

$$t_0 = y_1 \rightarrow t_{01}, t_{00}$$

$$t_1 = y_1 \rightarrow 1, t_{10}$$

Example

- Example: $t = (x_1 \wedge y_1) \vee (x_2 \wedge y_2)$
- Shannon expansion of t in order x_1, y_1, x_2, y_2

$$
\begin{aligned}
t &= x_1 \rightarrow t_1, t_0 \\
t_0 &= y_1 \rightarrow t_{01}, t_{00} \\
t_1 &= y_1 \rightarrow 1, t_{10} \\
t_{01} &= x_2 \rightarrow t_{011}, 0 \\
t_{00} &= x_2 \rightarrow t_{001}, 0 \\
t_{10} &= x_2 \rightarrow t_{101}, 0
\end{aligned}
$$

Example

- Example: $t = (x_1 \wedge y_1) \vee (x_2 \wedge y_2)$
- Shannon expansion of t in order x_1, y_1, x_2, y_2

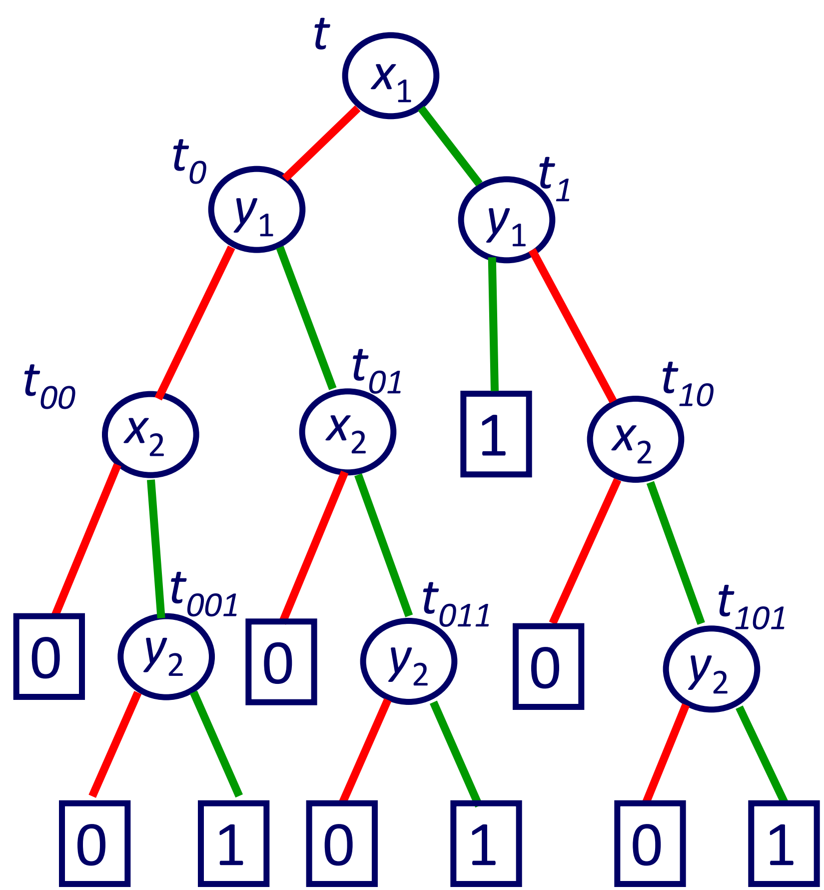
$$
\begin{aligned}
t &= x_1 \rightarrow t_1, t_0 \\
t_0 &= y_1 \rightarrow t_{01}, t_{00} \\
t_1 &= y_1 \rightarrow 1, t_{10} \\
t_{01} &= x_2 \rightarrow t_{011}, 0 \\
t_{00} &= x_2 \rightarrow t_{001}, 0 \\
t_{10} &= x_2 \rightarrow t_{101}, 0 \\
t_{011} &= y_2 \rightarrow 1,0 \\
t_{001} &= y_2 \rightarrow 1,0 \\
t_{101} &= y_2 \rightarrow 1,0
\end{aligned}
$$

$$
t = x_1 \rightarrow (y_1 \rightarrow 1,(x_2 \rightarrow (y_2 \rightarrow 1,0), 0)), (y_1 \rightarrow (x_2 \rightarrow (y_2 \rightarrow 1,0), 0), (x_2 \rightarrow (y_2 \rightarrow 1,0), 0))
$$

Decision tree

- Example: $t = (x_1 \wedge y_1) \vee (x_2 \wedge y_2)$
- Shannon expansion of t in order x_1, y_1, x_2, y_2

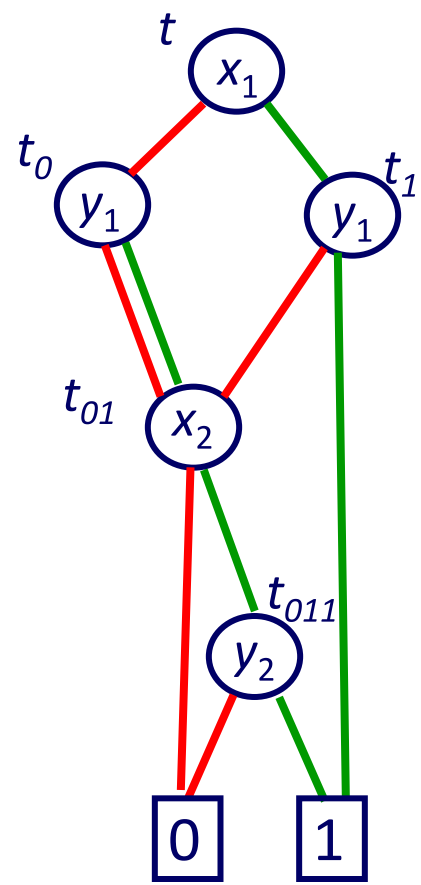
$$
\begin{aligned}
t &= x_1 \to t_1, t_0 \\
t_0 &= y_1 \to t_{01}, t_{00} \\
t_1 &= y_1 \to 1, t_{10} \\
t_{01} &= x_2 \to t_{011}, 0 \\
t_{00} &= x_2 \to t_{001}, 0 \\
t_{10} &= x_2 \to t_{101}, 0 \\
t_{011} &= y_2 \to 1,0 \\
t_{001} &= y_2 \to 1,0 \\
t_{101} &= y_2 \to 1,0
\end{aligned}
$$



Reduction I: substitute identical subtrees

- Example: $t = (x_1 \wedge y_1) \vee (x_2 \wedge y_2)$
- Shannon expansion of t in order x_1, y_1, x_2, y_2

$$
\begin{aligned}
t &= x_1 \rightarrow t_1, t_0 \\
t_0 &= y_1 \rightarrow t_{01}, t_{01} \\
t_1 &= y_1 \rightarrow 1, t_{01} \\
t_{01} &= x_2 \rightarrow t_{011}, 0 \\
t_{011} &= y_2 \rightarrow 1,0
\end{aligned}
$$



Result: an Ordered Binary Decision Diagram (OBDD)

Reduction II: remove redundant tests

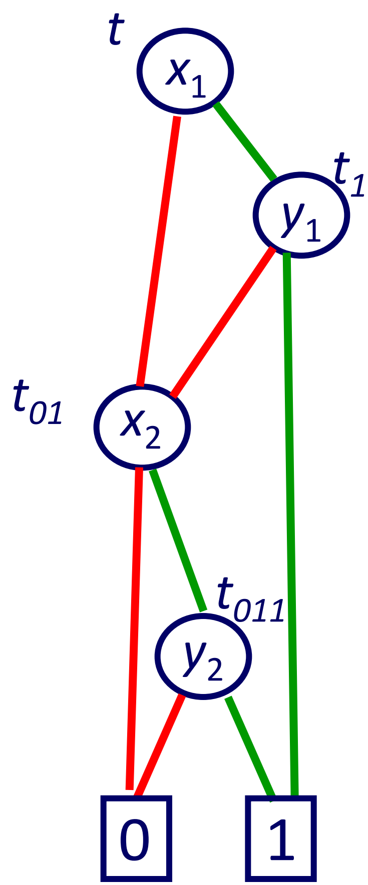
- Example: $t = (x_1 \wedge y_1) \vee (x_2 \wedge y_2)$
- Shannon expansion of t in order x_1, y_1, x_2, y_2

$$t = x_1 \rightarrow t_1, t_{01}$$

$$t_1 = y_1 \rightarrow 1, t_{01}$$

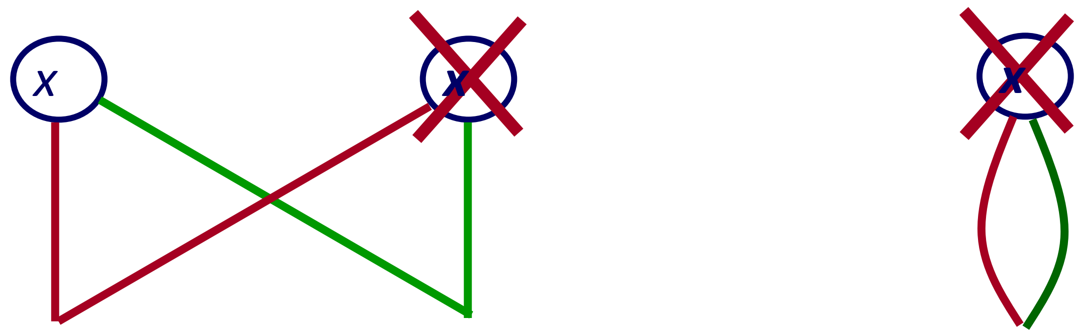
$$t_{01} = x_2 \rightarrow t_{011}, 0$$

$$t_{011} = y_2 \rightarrow 1,0$$



Result: a Reduced Ordered Binary Decision Diagram (ROBDD)
[often called a BDD]

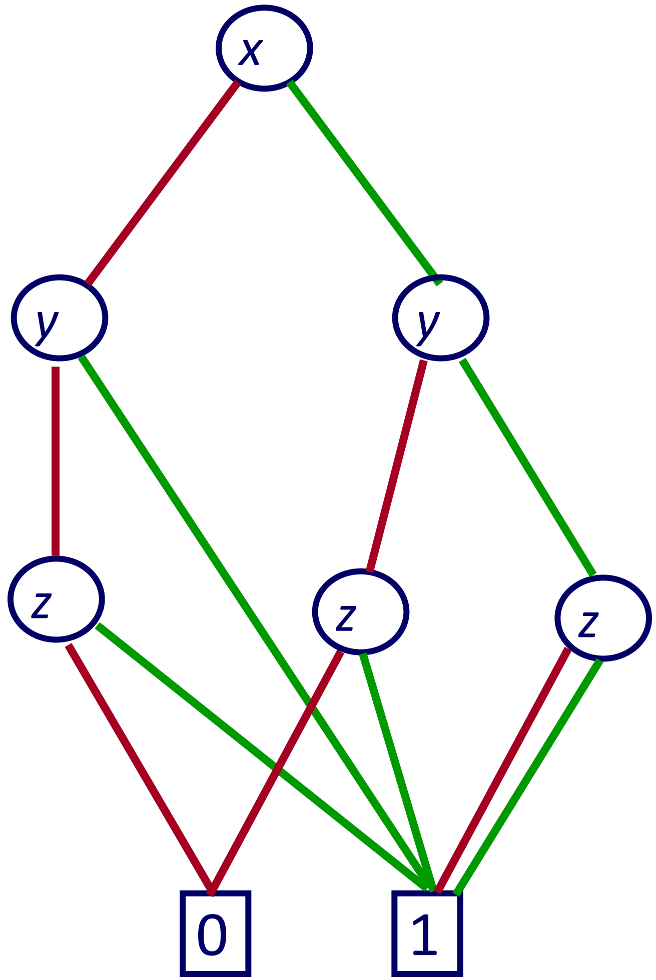
Reductions



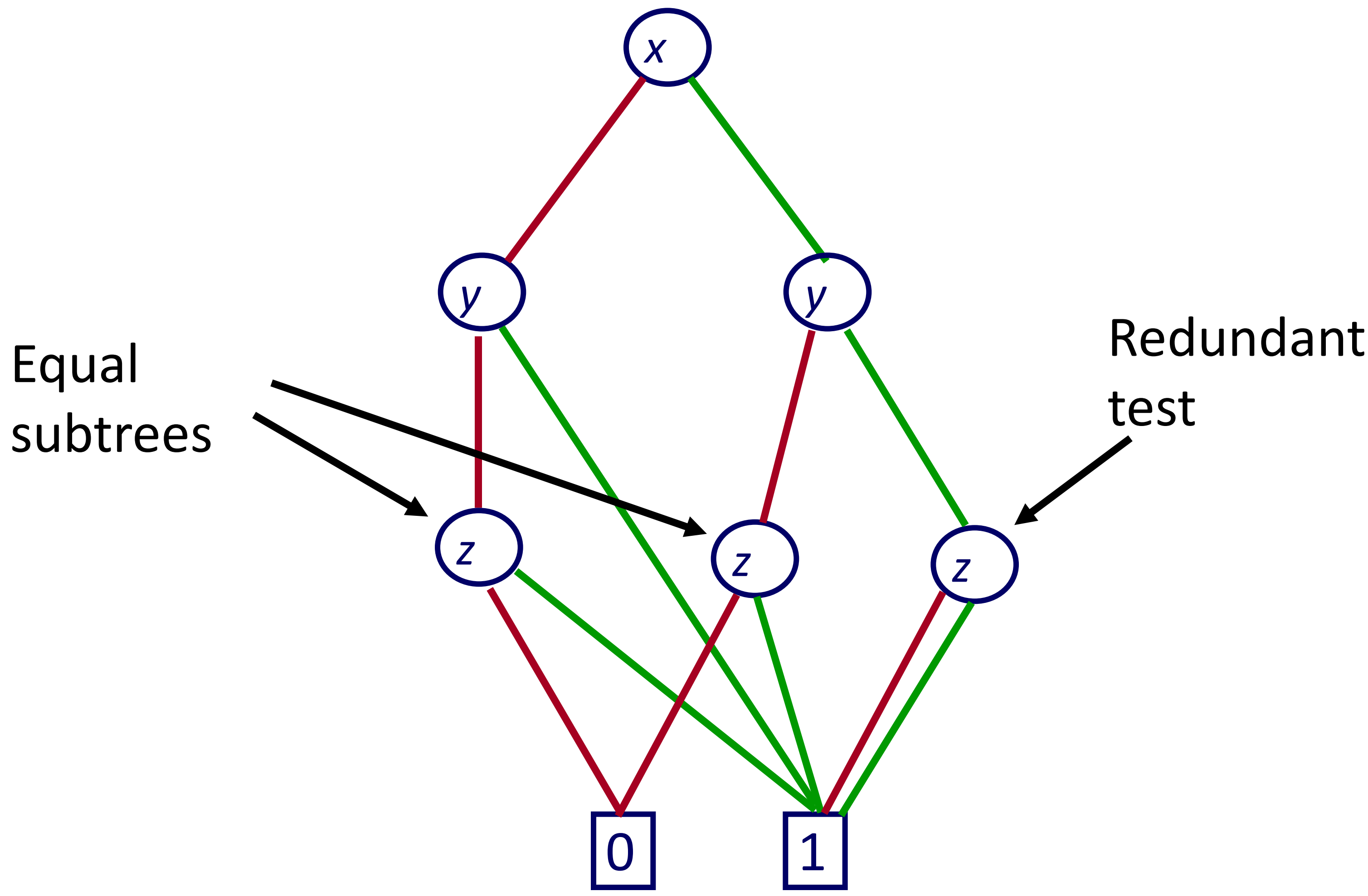
Uniqueness requirement

Non-redundant tests requirement

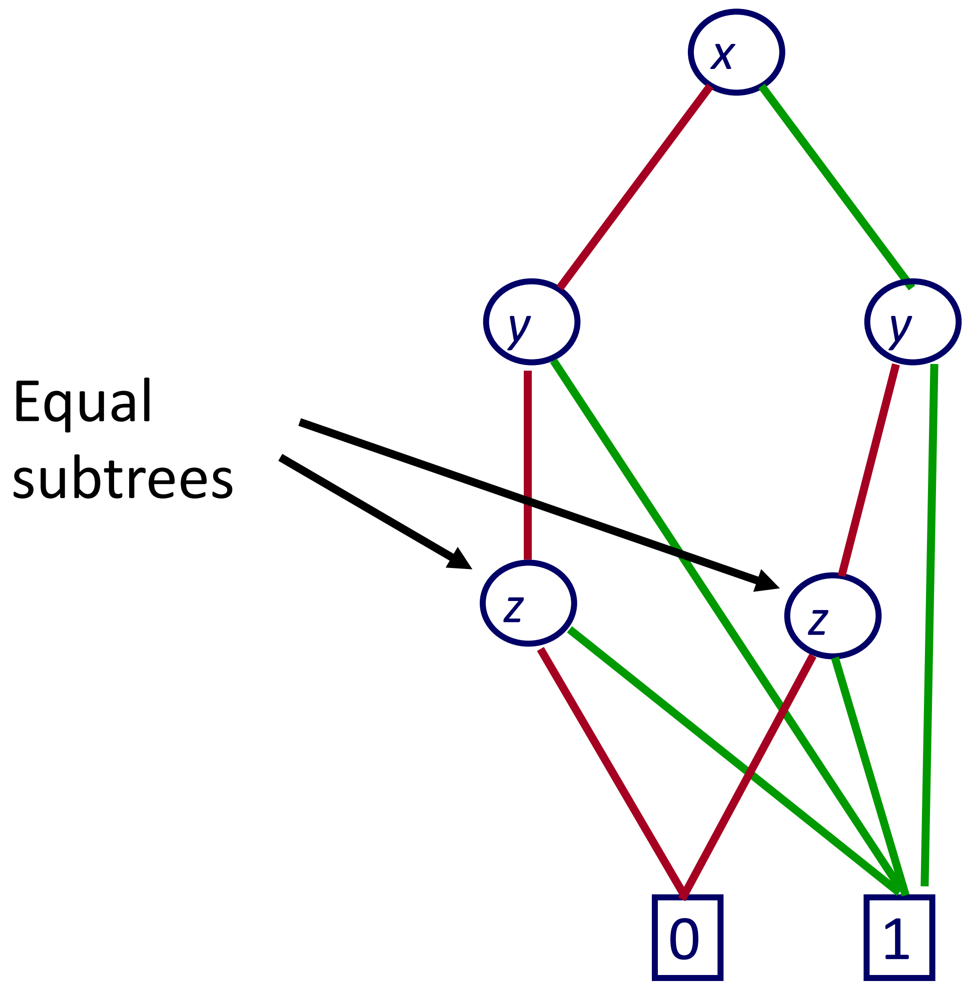
Another reduction example



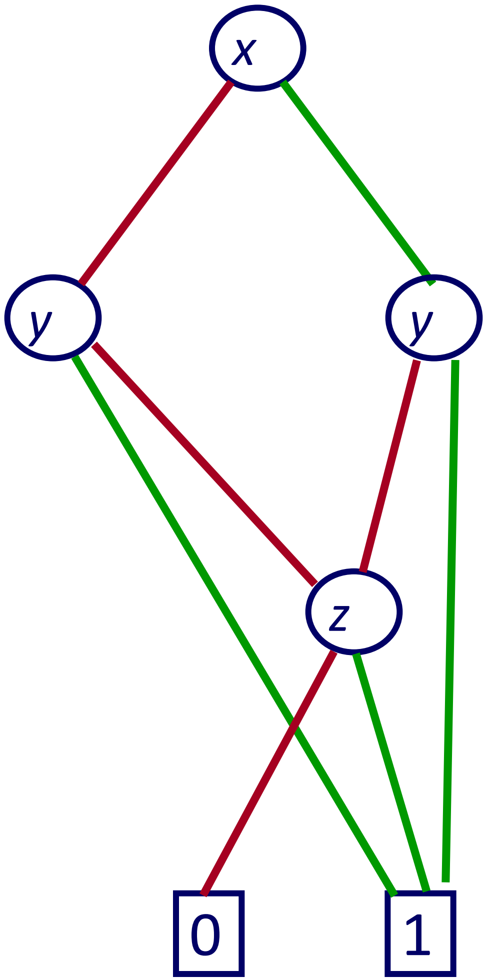
Another reduction example



Another reduction example

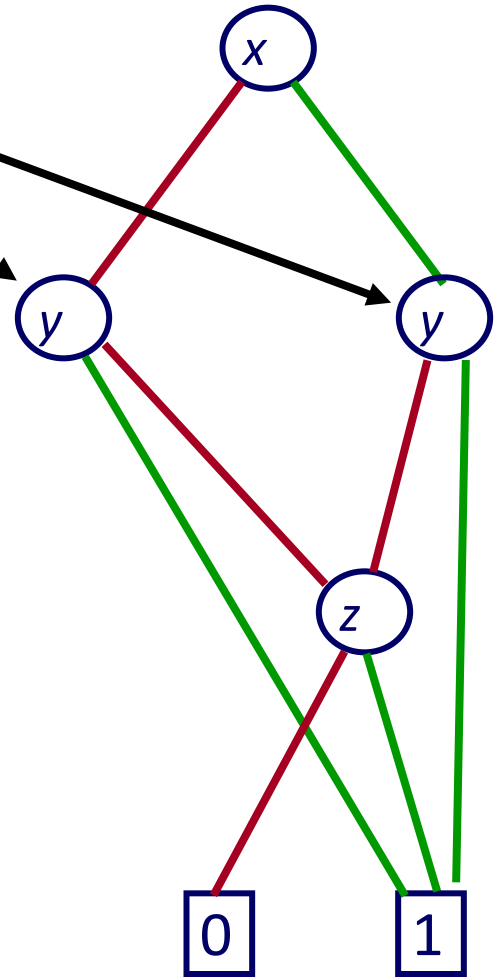


Another reduction example

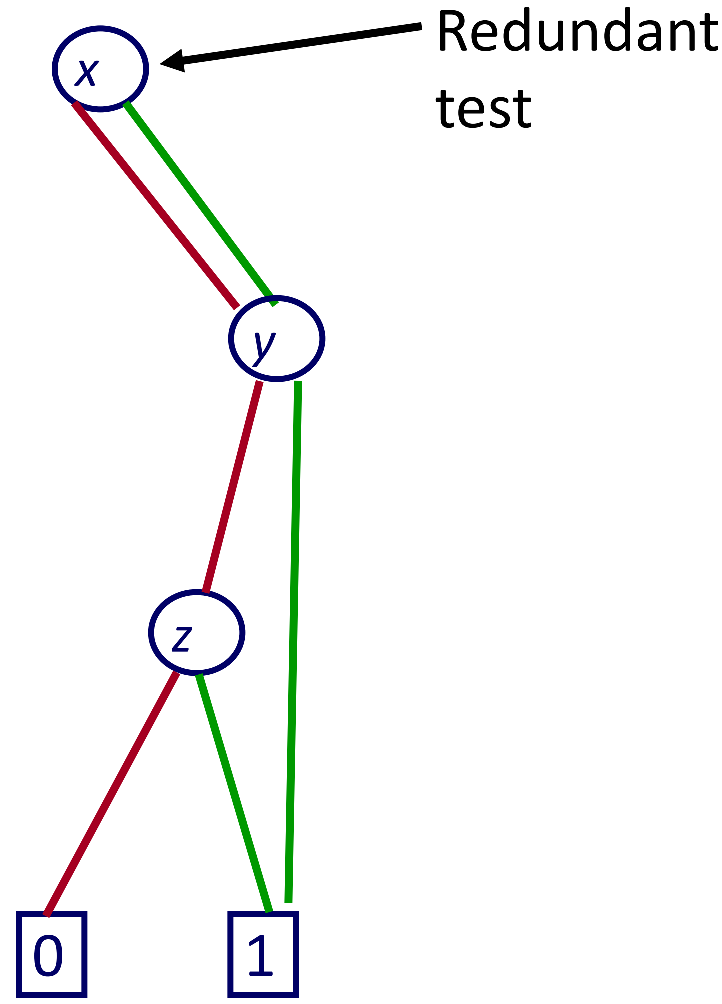


Another reduction example

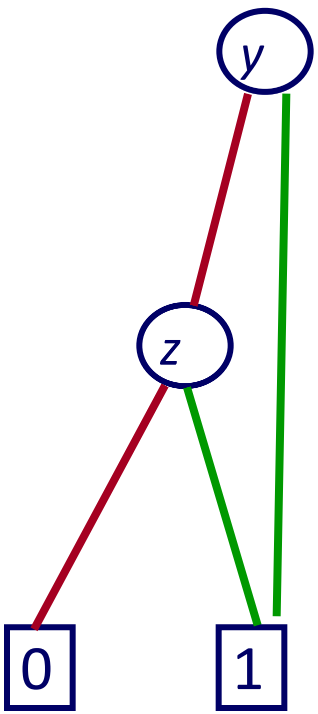
Equal subtrees



Another reduction example

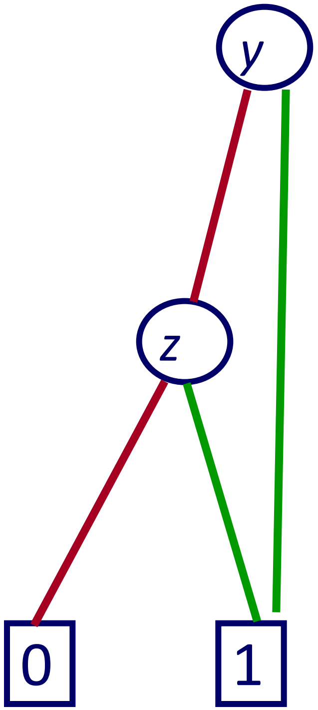


Another reduction example



Another reduction example

$y \vee z$



Canonicity of ROBDDs

- **Canonicity Lemma**: for any function $f : \boldsymbol{B}^n \rightarrow \boldsymbol{B}$ there is exactly one ROBDD u with a variable ordering $x_1 < x_2 < \ldots < x_n$ such that $f_u = f(x_1,\ldots,x_n)$

Proof (by induction on n)

Read on your own!

Canonicity of ROBDDs

- The canonicity of ROBDDs simplifies
  - Equality check
  - Tautology check
  - Satisfiability check

  Why? (hint: same answer as previously)

Practice

- What are the ROBDDs of
  - x
  - 1
  - 0
  - $x \wedge y$ order x, y
  - $(x \Rightarrow y) \wedge z$ order x, y, z

Size of ROBDDs

- ROBDDs of many practically important Boolean functions are small
  - E.g., Adders have polynomial sized ROBDDs
- Have all functions polynomial ROBDD size? NO
  - ROBDDs do not escape the curse of Boolean function representation
  - E.g., Multipliers have exponential ROBDD size independent of ordering

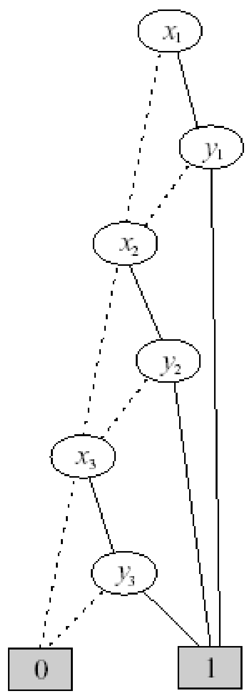
Size of ROBDDs

- The size of an ROBDD depends heavily on the variable ordering
- Example: $t = (x_1 \wedge y_1) \vee (x_2 \wedge y_2)$
- Build ROBDD of t in order x_1, x_2, y_1, y_2

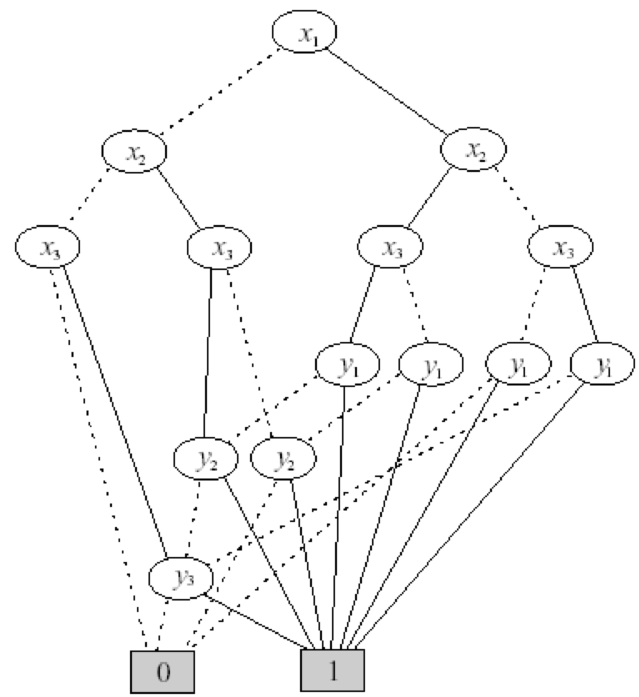
Size of ROBDDs

- The size of an ROBDD depends heavily on the variable ordering

$(x_1 \wedge y_1) \vee (x_2 \wedge y_2) \vee \ldots \vee (x_n \wedge y_n)$



$x_1 < y_1 < x_2 < y_2 < \ldots < x_n < y_n$



$x_1 < x_2 < \ldots < x_n < y_1 < x_2 < \ldots < y_n$