CS 512, Spring 2017, Handout 14

Binary Decision Diagrams (BDD's)

Assaf Kfoury

February 26, 2017

background and reading material

- The last chapter, Chapter 6, in the book [LCS] is entirely devoted to BDD's. You should read at least Sections 6.1 and 6.2.

  Sections 6.3 and 6.4 go into topics that will not be covered this semester (***symbolic model-checking*** and ***mu-calculus***), but still cover material that will deepen your knowledge of BDD's, if you can handle them.

  My presentation is somewhat different from that in [LCS], especially in regard to explaining connections between propositional WFF's and BDD's.

- Although there is rather little on BDD's, especially from a persepctive stressing formal methods and formal modeling, in textooks (of which I am aware),[1] there is a lot on BDD's that you can find by searching the Web.

  For a good expository account of BDD's and their history, click **here**.

[1] There is a book by Rolf Drechsler and Bernd Becker, *Binary Decision Diagrams, Theory and Practice*, 1998, written from the perspective of people working on VLSI (Very Large Scale Integration) and the design of electronic circuits. From an algorithmic perspective, there is a very nice section (Section 7.1.4) in Donald Knuth, *The Art of Computer Programming, Vol. 4*, 2008.

canonical representations of WFF's of propositional logic?

- given a WFF φ of propositional logic, is there a **canonical representation** of φ, call it $\varphi^\star$, satisfying the following condition:

  for every WFF ψ of propositional logic,
  φ and ψ are equivalent iff $\varphi^\star = \psi^\star$??

  ("$\varphi^\star = \psi^\star$" means $\varphi^\star$ and $\psi^\star$ are syntactically the same.)

canonical representations of WFF's of propositional logic?

- given a WFF φ of propositional logic, is there a **canonical representation** of φ, call it $\varphi^\star$, satisfying the following condition:

  for every WFF ψ of propositional logic,
  φ and ψ are equivalent iff $\varphi^\star = \psi^\star$??

  ("$\varphi^\star = \psi^\star$" means $\varphi^\star$ and $\psi^\star$ are syntactically the same.)

- if yes, hopefully $\varphi^\star$ and $\psi^\star$ are obtained by "easy" syntactic transformation, allowing for a "quick" syntactic test $\varphi^\star = \psi^\star$

- perhaps the CNF's of propositional WFF's can be the desired canonical representations???

- *or* perhaps the DNF's of propositional WFF's can be the desired canonical representations???

bad news: CNF's and DNF's are not canonical representations

- Two WFF's of propositional logic:

$$\varphi \triangleq (x \wedge (y \vee z))$$
$$\psi \triangleq (x \wedge (x \vee y) \wedge (y \vee z))$$

- φ and ψ are both in CNF

- φ and ψ are equivalent

- yet, φ and ψ are syntactically different

- **Conclusion:**
  CNF's are **not** canonical representations of propositional WFF's.
  Same conclusion for DNF's.

truth-table representation of propositional WFF's is canonical

- **Canonicity of Truth Tables**: For arbitrary propositional WFF's φ_1 and φ_2, φ_1 and φ_2 are equivalent iff $\textbf{table}(\varphi_1) = \textbf{table}(\varphi_2)$.[2]

  The equivalence of φ_1 and φ_2 is therefore reduced

  to a syntactic test of equality between $\textbf{table}(\varphi_1)$ and $\textbf{table}(\varphi_2)$.

[2] We limit $\textbf{table}(\varphi)$ to the columns corresponding to the variables in φ together with the last column in the truth-table of φ.

truth-table representation of propositional WFF's is canonical

- **Canonicity of Truth Tables**: For arbitrary propositional WFF's φ_1 and φ_2, φ_1 and φ_2 are equivalent iff **table**$(\varphi_1) =$ **table**(φ_2).[2]
  The equivalence of φ_1 and φ_2 is therefore reduced to a syntactic test of equality between **table**(φ_1) and **table**(φ_2).
- **Example**: for the WFF's $\varphi = (x \wedge (y \vee z))$ and $\psi = (x \wedge (x \vee y) \wedge (y \vee z))$ on slide 5, **table**$(\varphi) =$ **table**(ψ) is the following truth-table:

x	y	z	φ	ψ
F	**F**	**F**	**F**	**F**
F	**F**	**T**	**F**	**F**
F	**T**	**F**	**F**	**F**
F	**T**	**T**	**F**	**F**
T	**F**	**F**	**F**	**F**
T	**F**	**T**	**T**	**T**
T	**T**	**F**	**T**	**T**
T	**T**	**T**	**T**	**T**

- **But** canonicity of truth tables comes with a heavy price, which is . . .

[2] We limit **table**(φ) to the columns corresponding to the variables in φ together with the last column in the truth-table of φ.

in search of a canonical representation of propositional WFF's

In the next few slides, we show:

- how to transform an arbitrary propositional WFF φ to a binary decision tree (BDT) representing φ,
- how to translate a binary decision tree (BDT) T back to a propositional WFF that T represents,
- how to transform a binary decision tree (BDT) T to an equivalent binary decision diagram (BDD) D.
- how to transform a binary decision diagram (BDD) D to an equivalent reduced ordered binary decision diagram (OBDD) D'.

from a propositional WFF to a binary decision tree (BDT)

for propositional WFF φ with atoms in $X = \{x_1, \ldots, x_n\}$, two basic approaches:

(A) substitute $\perp$ (left branch) and $\top$ (right branch) for the atoms in X in some order, delaying simplification until all atoms are replaced.

(B) substitute $\perp$ (left branch) and $\top$ (right branch) for the atoms in X in some order, without delaying simplification until all atoms are replaced.

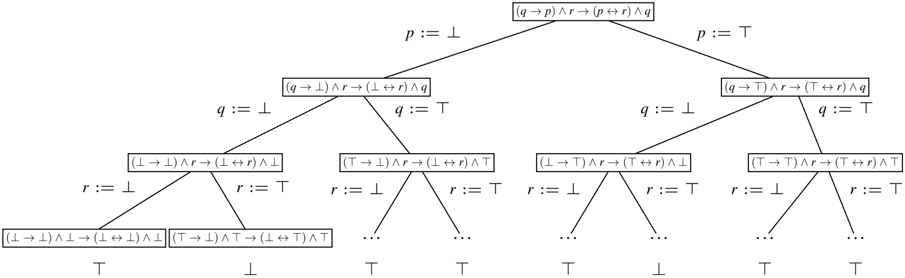
- method (A) produces a full binary tree with exactly $(2^n - 1)$ internal nodes and 2^n leaf nodes.
- method (B) produces a binary tree with at most $(2^n - 1)$ internal nodes and 2^n leaf nodes.
- simplification in both methods based on, for arbitrary WFF ψ:

$$
\begin{array}{lll}
\neg\neg\psi \equiv \psi & \psi \vee \neg\psi \equiv \top & \psi \wedge \neg\psi \equiv \perp \\
\top \vee \psi \equiv \top & \perp \vee \psi \equiv \psi & \\
\top \wedge \psi \equiv \psi & \perp \wedge \psi \equiv \perp &
\end{array}
$$

as well as $(\psi \to \psi') \equiv (\neg\psi \vee \psi')$, commutativity of "$\vee$" and "$\wedge$", etc.

from a propositional WFF to a binary decision tree (BDT)

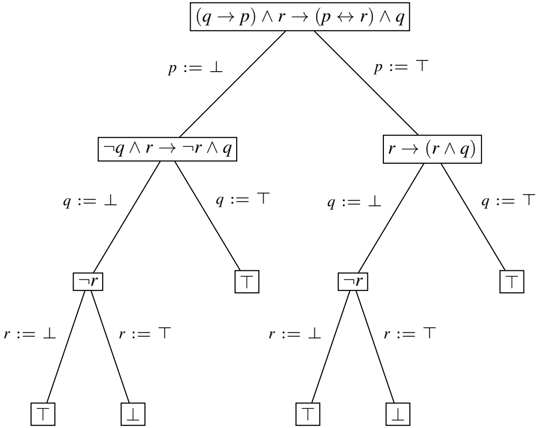
- **Example:** applying method (A) to WFF $\varphi \triangleq (q \to p) \wedge r \to (p \leftrightarrow r) \wedge q$:



The preceding is a binary tree, labelled in a particular way, but NOT yet a BDT!

from a propositional WFF to a binary decision tree (BDT)

- **Example:** applying method (B) to WFF $\varphi \triangleq (q \to p) \wedge r \to (p \leftrightarrow r) \wedge q$:



The preceding is a binary tree, labelled in a particular way, but NOT yet a BDT!

from a propositional WFF to a binary decision tree (BDT)

Remarks:

- for the same WFF $\varphi \triangleq (q \to p) \wedge r \to (p \leftrightarrow r) \wedge q$ in slide 11, method (B) produces different trees for different orderings of the atoms $\{p, q, r\}$.

  Exercise: apply method (B) to φ using the ordering: (1) r, (2) q, and (3) p.

- the trees returned by methods (A) and (B) give the same complete semantic information about the input WFF φ.

  for the input $\varphi \triangleq (q \to p) \wedge r \to (p \leftrightarrow r) \wedge q$ in slides 10 and 11:

  φ is **not** a tautology/valid WFF – some leaf nodes are $\perp$

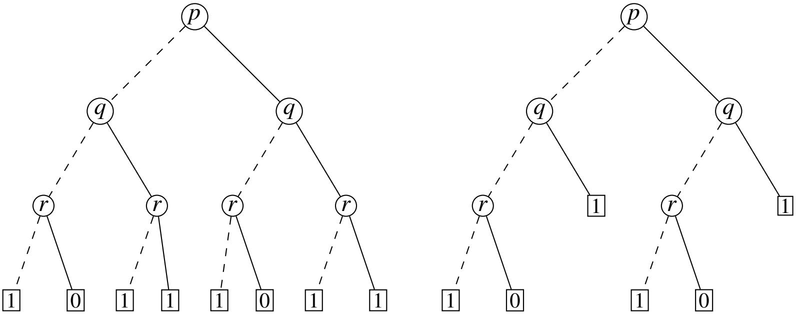
  φ is **not** unsatisfiable/contradictory WFF – some leaf nodes are $\top$

  φ is contingent WFF :

  - φ is satisfied by any valuation of $\{p, q, r\}$ induced by a path from the root to a leaf node $\top$
  - φ is falsified by any valuation of $\{p, q, r\}$ induced by a path from the root to a leaf node $\perp$

from a propositional WFF to a binary decision tree (BDT)

- one more step to transform the trees in slides 10 and 11 returned by methods (A) and (B) into what are called binary decision trees (**BDT's**) :



Note that, starting from the same WFF, we obtained two different BDT's!

And the shape of the BDT on the right changes with the orderings of $\{p, q, r\}$!!

from a binary decision tree (BDT) to a propositional WFF

- one approach is to write a DNF (disjunction of conjuncts) where each conjunct represents the truth assignment along a path from the root of the BDT to a leaf node labelled "1".

Example: We can write the DNF's φ_A and φ_B, below, for the BDT's on the left and on the right in slide 13, respectively:

$$\varphi_A \triangleq (\neg p \wedge \neg q \wedge \neg r) \vee (\neg p \wedge q \wedge \neg r) \vee (\neg p \wedge q \wedge r) \vee (p \wedge \neg q \wedge \neg r) \vee (p \wedge q \wedge \neg r) \vee (p \wedge q \wedge r)$$

$$\varphi_B \triangleq (\neg p \wedge \neg q \wedge \neg r) \vee (\neg p \wedge q) \vee (p \wedge \neg q \wedge \neg r) \vee (p \wedge q)$$

there are 6 conjuncts in φ_A and 4 conjuncts in φ_B, corresponding to the number of paths in each of the two BDT's leading to a leaf node "1".

from a binary decision tree (BDT) to a propositional WFF

- another approach is to write a WFF using the logical connective **if-then-else**.

Example: For the BDT on the right in slide 13 (leaving the BDT on the left in slide 13 to you), we can write:

$$
\begin{array}{llllllllll}
\psi_B \triangleq & \textbf{if} & p & \textbf{then} & \textbf{if} & q & \textbf{then} & \top & & \\
 & & & & \textbf{else} & \textbf{if} & r & \textbf{then} & \bot \\
 & & & & & & & \textbf{else} & \top \\
 & & & \textbf{else} & \textbf{if} & q & \textbf{then} & \top & & \\
 & & & & \textbf{else} & \textbf{if} & r & \textbf{then} & \bot \\
 & & & & & & & \textbf{else} & \top
\end{array}
$$

Exercise: the logical connective **if-then-else** is not directly available in the syntax of propositional logic. Show how to define **if-then-else** using the standard connectives in $\{\to, \wedge, \vee, \neg\}$.

binary decision trees (BDT), binary decision diagrams (BDD)

- definition of BDT is in first paragraph of Sect 6.1.2 [LCS, page 361]
- definition of BDD in Definition 6.5 [LCS, page 364]
- BDT's are a special case of BDD's
- BDD's allow three optimizations $\{\mathbf{C1}, \mathbf{C2}, \mathbf{C3}\}$ [LCS, page 363], which are not allowed in BDT's

binary decision trees (BDT's) vs. reduced ordered binary decision diagrams (ROBDD's)

- consider the propositional WFF φ (written as a Boolean function of 6 variables):

$$\varphi \triangleq (x_1 + x_2) \cdot (x_3 + x_4) \cdot (x_5 + x_6)$$

(φ as a function, we follow the convention: "$+$" instead of "$\vee$" and "$\cdot$" instead of "$\wedge$")

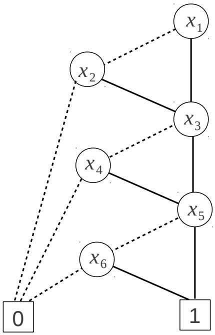
- if we include all propositional variables along all paths from the root, then the corresponding **BDT**(φ) has $2^6 = 64$ leaf nodes and $2^6 - 1 = 63$ internal nodes (just too large to draw on this slide!!)
- if **BDT**(φ) is produced using method (A) in slide 9, then its size is not affected by the ordering of the variables $\{x_1,\ x_2,\ x_3,\ x_4,\ x_5,\ x_6\}$, it is the same regardless of the ordering
- relative to a fixed ordering of the variables, *e.g.*, $x_1 < x_2 < x_3 < x_4 < x_5 < x_6$, starting from the root, **BDT**(φ) is unique (as an unordered binary tree)

binary decision trees (BDT's) vs. reduced ordered binary decision diagrams (ROBDD's)

- applying repeatedly reduction rules $\{$**C1**, **C2**, **C3**$\}$ to **BDT**(φ) on slide 17:
  C1: merge leaf nodes into two nodes "0" and "1"
  C2: remove redundant nodes
  C3: merge isomorphic sub-dags
  we obtain a ROBDD w.r.t. to the ordering $x_1 < x_2 < x_3 < x_4 < x_5 < x_6$:

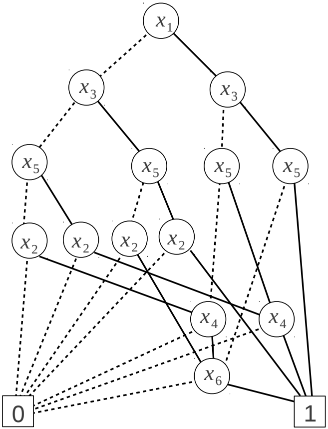
binary decision trees (BDT's) vs. reduced ordered binary decision diagrams (ROBDD's)

- applying repeatedly reduction rules $\{$**C1**, **C2**, **C3**$\}$ to **BDT**(φ) on slide 17:
  C1: merge leaf nodes into two nodes "0" and "1"
  C2: remove redundant nodes
  C3: merge isomorphic sub-dags
  we obtain a ROBDD w.r.t. to the ordering $x_1 < x_2 < x_3 < x_4 < x_5 < x_6$:



binary decision trees (BDT's) vs. reduced ordered binary decision diagrams (ROBDD's)

- **however,** w.r.t. the (different) ordering $x_1 < x_3 < x_5 < x_2 < x_4 < x_6$, applying the 3 reduction rules repeatedly produces a much larger ROBDD:

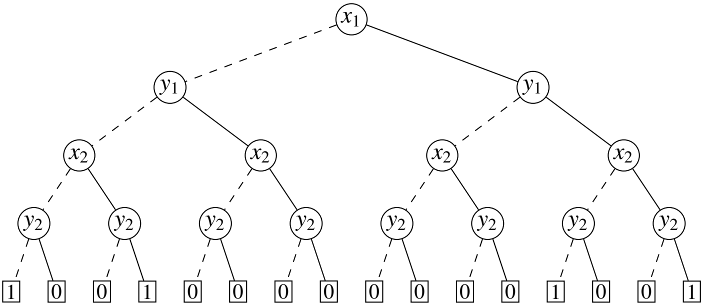


another example: binary decision trees (BDT's) vs. reduced ordered binary decision diagrams (ROBDD's)

- consider the so-called **two-bit comparator**:

$$\psi \triangleq (x_1 \leftrightarrow y_1) \wedge (x_2 \leftrightarrow y_2)$$

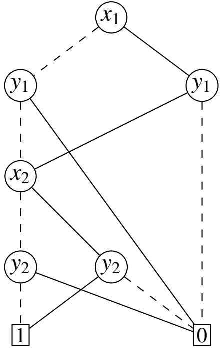
and the corresponding **BDT**(ψ), which has 15 internal nodes/decision points and 16 leaf nodes:



(I used method (A) in slide 9 to obtain **BDT**(ψ) from ψ.)

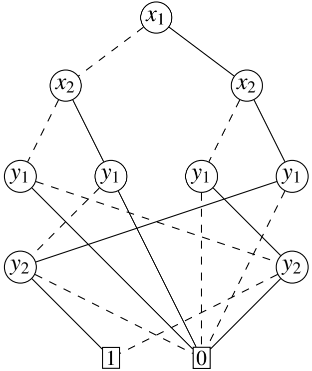
another example: binary decision trees (BDT's) vs. reduced ordered binary decision diagrams (ROBDD's)

- applying repeatedly reduction rules $\{\mathbf{C1}, \mathbf{C2}, \mathbf{C3}\}$ to $\mathbf{BDT}(\psi)$ on slide 21, we obtain a ROBDD w.r.t. to the ordering $x_1 < y_1 < x_2 < y_2$, with 6 internal nodes and 2 leaf nodes:



another example: binary decision trees (BDT's) vs. reduced ordered binary decision diagrams (ROBDD's)

- **however,** if we use the ordering $x_1 < x_2 < y_1 < y_2$ for the BDT of the two-bit comparator ψ, and apply the 3 reduction rules repeatedly, we obtain a larger ROBDD, with 9 internal nodes and 2 leaf nodes:



facts about ROBDD's – some **bad** news!

- The n-bit comparator is the following WFF:

  $$\psi_n \triangleq (x_1 \leftrightarrow y_1) \wedge (x_2 \leftrightarrow y_2) \wedge \cdots \wedge (x_n \leftrightarrow y_n)$$

  - **Fact**: If we use the ordering $x_1 < y_1 < \cdots < x_n < y_n$, the number of nodes in **ROBDD**(ψ_n) is $3 \cdot n + 2$ (linear in n).
  - **Fact**: If we use the ordering $x_1 < \cdots < x_n < y_1 < \cdots < y_n$, the number of nodes in **ROBDD**(ψ_n) is $3 \cdot 2^n - 1$ (exponential in n).

  Exercise: Prove two preceding facts (easy!).

- **Fact**: There are propositional WFF's φ whose ROBDD's have sizes exponential in $|\varphi|$ for all orderings of variables (bad news!).

  Exercise: Prove this fact (not easy!).

- **Fact**: Finding an ordering of the variables in an arbitrary φ so that the size of **ROBDD**(φ) is minimized is an NP-hard problem (more bad news!).

  Exercise: Search the Web for a paper proving this fact.

facts about ROBDD's – some **good** news!

- **Fact**: **ROBDD's are canonical**.

  Specifically, relative to a fixed ordering of the variables in WFF φ (imposing the same ordering on var in all paths from root to terminals), **ROBDD**(φ) is a uniquely defined dag.

- **Fact**: Relative to the same ordering of variables along all paths from the root to a terminal, the transformation from **BDT**(φ) to **ROBDD**(φ) can be carried out in **linear time**.

facts about ROBDD's – still some **good** news!

Exploiting canonicity of ROBDD's.

- **Fact**: checking **equivalence** of φ and ψ is the same as checking if **ROBDD**(φ) and **ROBDD**(ψ) are **equal**, w.r.t. same ordering of variables.
- **Fact**: **tautological validity** of φ can be determined by checking if **ROBDD**(φ) is **equal** to the ROBDD with a single terminal label "1"
- **Fact**: **unsatisfiability** of φ can be determined by checking if **ROBDD**(φ) is **equal** to the ROBDD with a single terminal label "0"

facts about ROBDD's – more **good** news!

Exploiting canonicity of ROBDD's.

- **Fact**: **satisfiability** of φ can be determined by first checking if **ROBDD**(φ) is **equal** to the ROBDD with a single terminal label "0", in which case φ is unsatisfiable, otherwise . . ..

  Exercise: Fill in the missing part in preceding statement (easy!) .

  Exercise: determine if φ is satisfiable **and** construct a satisfying assignment (more interesting!) .

  Exercise: determine if φ is satisfiable **and** count the number of satisfying assignments (still more interesting!) .

- **Fact**: **implication**, *i.e.*, φ **implies** ψ, can be determined by checking if **ROBDD**$(\varphi \wedge \neg\psi)$ is **equal** to the ROBDD with a single terminal label "0"

  Exercise: Prove this fact (easy!) .