Symbolic model checking

Boolean functions

- Boolean functions $f : \mathbb{B}^n \to \mathbb{B}$ for $n \geq 0$ where $\mathbb{B} = \{ 0, 1 \}$
  - examples: $f(x_1, x_2) = x_1 \wedge (x_2 \vee \neg x_1)$, and $f(x_1, x_2) = x_1 \leftrightarrow x_2$
- Finite sets are boolean functions
  - let $|S| = N$ and $2^{n-1} < N \leq 2^n$
  - encode each element $s \in S$ as boolean vector of length n: $[\![\,]\!] : S \to \mathbb{B}^n$
  - $T \subseteq S$ is represented by f_T such that:

$$f_T([\![s]\!]) = 1 \quad \text{iff} \quad s \in T$$

  - this is the **characteristic function** of T
- Relations are boolean functions
  - $\mathcal{R} \subseteq S \times S$ is represented by $f_{\mathcal{R}}$ such that:

$$f_{\mathcal{R}}([\![s]\!], [\![t]\!]) = 1 \quad \text{iff} \quad (s, t) \in \mathcal{R}$$

Transition systems as boolean functions

- Assume each state is uniquely labeled
  - $L(s) = L(s')$ implies $s = s'$
  - no restriction: if needed extend AP and label states uniquely
- Assume a fixed total order on propositions: $a_1 < a_2 < \ldots < a_K$
- Represent a state by a **boolean function**
  - over boolean variables x_1 through x_K such that

$$[\![s]\!] = x_1^* \wedge x_2^* \wedge \ldots \wedge x_K^*$$

  - where the literal x_i^* equals x_i if $a_i \in L(s)$, and $\neg x_i$ otherwise

  $\Rightarrow$ no need to explicitly represent function L
- Represent I and $\to$ by their characteristic (boolean) functions
  - e.g., $f_{\to}([\![s]\!], [\![\alpha]\!], [\![t]\!]) = 1$ if and only if $s \xrightarrow{\alpha} t$

Representing boolean functions

representation	compact?	sat	$\wedge$	$\vee$	$\neg$
propositional formula	often	hard	easy	easy	easy
DNF	sometimes	easy	hard	easy	hard
CNF	sometimes	hard	easy	hard	hard
(ordered) truth table	never	hard	hard	hard	hard
reduced ordered binary decision diagram	often	easy	medium	medium	easy

Binary decision trees

- Let X be a set of boolean variables.
- Let $<$ be a total order on X
- **Binary decision tree** (BDT) is a complete binary tree over $\langle X, < \rangle$
  - each leaf v is labeled with a boolean value $val(v) \in \mathbb{B}$
  - non-leaf v is labeled by a boolean variable $Var(v) \in X$
  - such that for each non-leaf v and vertex w:

    $$w \in \{ left(v), right(v) \} \Rightarrow (Var(v) < Var(w) \vee w \text{ is a leaf})$$

$\Rightarrow$ On each path from root to leaf, variables occur in the same order

Shannon expansion

- Each boolean function $f : \mathbb{B}^n \longrightarrow \mathbb{B}$ can be written as:

  $$f(x_1, \ldots, x_n) = (x_i \wedge f[x_i := 1]) \vee (\neg x_i \wedge f[x_i := 0])$$

  - where $f[x_i := 1]$ stands for $f(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n)$
  - and $f[x_i := 0]$ for $f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n)$
- The boolean function $f_B(v)$ represented by vertex v in BDT B is:
  - for v a leaf: $f_B(v) = val(v)$
  - otherwise:

    $$f_B(v) = (Var(v) \wedge f_B(right(v))) \vee (\neg Var(v) \wedge f_B(left(v)))$$
- $f_B = f_B(v)$ where v is the root of B

Considerations on BDTs

- BDTs are not compact
  - a BDT for boolean function $f : \mathbb{B}^b \to \mathbb{B}$ has 2^n leafs
  - $\Rightarrow$ they are as space inefficient as truth tables!

$\Rightarrow$ BDTs contain quite some redundancy
- all leafs with value one (zero) could be collapsed into a single leaf
- a similar scheme could be adopted for isomorphic subtrees
- The size of a BDT does not change if the variable order changes

Ordered Binary Decision Diagram

share equivalent expressions [Akers 76, Lee 59]

- A **ordered binary decision diagram** (OBDD) is a directed graph over $\langle X, < \rangle$ with:
  - each leaf v is labeled with a boolean value $val(v) \in \{ 0, 1 \}$
  - non-leaf v is labeled by a boolean variable $Var(v) \in X$
  - such that for each non-leaf v and vertex w:

    $$w \in \{ left(v), right(v) \} \Rightarrow (Var(v) < Var(w) \vee w \text{ is a leaf})$$

$\Rightarrow$ An OBDD is acyclic
- f_B for OBDD B is obtained as for BDTs

Reduced OBDDs

OBDD B over $\langle X, < \rangle$ is called **reduced** iff:

1. for each leaf v, w: $(val(v) = val(w)) \Rightarrow v = w$
   $\Rightarrow$ identical terminal vertices are forbidden
2. for each non-leaf v: $left(v) \neq right(v)$
   $\Rightarrow$ non-leafs may not have identical children
3. for each non-leaf v, w:
   $(Var(v) = Var(w) \wedge right(v) \cong right(w) \wedge left(v) \cong left(w)) \Rightarrow v = w$
   $\Rightarrow$ vertices may not have isomorphic sub-dags

this is what is mostly called BDD; in fact it is an ROBDD!

Dynamic generation of ROBDDs

Main idea:

- Construct directly an ROBDD from a boolean expression
- Create vertices in depth-first search order
- On-the-fly reduction by applying hashing
  - on encountering a new vertex v, check whether:
  - an equivalent vertex w has been created (same label and children)
  - $left(v) = right(v)$, i.e., vertex v is a "don't care" vertex

ROBDDs are canonical

[Fortune, Hopcroft & Schmidt, 1978]

For ROBDDs B and B′ over $\langle X, < \rangle$ we have:
$(f_B = f_{B'})$ implies B and B′ are isomorphic

$\Rightarrow$ for a fixed variable ordering, any boolean function
can be uniquely represented by an ROBDD (up to isomorphism)

The importance of canonicity

- Absence of redundant vertices
  - if f_B does not depend on x_i, ROBDD B does not contain an x_i vertex
- Test for equivalence: $f(x_1, \ldots, x_n) \equiv g(x_1, \ldots, x_n)$?
  - generate ROBDDs B_f and B_g, and check isomorphism
- Test for validity: $f(x_1, \ldots, x_n) = 1$?
  - generate ROBDD B_f and check whether it is the 1-leaf
- Test for implication: $f(x_1, \ldots, x_n) \rightarrow g(x_1, \ldots, x_n)$?
  - generate ROBDD $B_f \wedge \neg B_g$ and check if it is the 0-leaf
- Test for satisfiability
  - f is satisfiable if and only if B_f is not the 0-leaf

Variable ordering

- Different ROBDDs are obtained for different variable orderings
- The size of the ROBDD depends on the variable ordering
- Some boolean functions have linear and exponential ROBDDs
- Some boolean functions only have polynomial ROBDDs
- Some boolean functions only have exponential ROBDDs

The even parity function

$f(x_1, \ldots, x_n) = 1$ iff the number of variables x_i with value 1 is even

truth table or propositional formula for f has exponential size

but an ROBDD of linear size is possible

Symmetric functions

$$f[x_1 := b_1, \ldots, x_n := b_n] = f[x_1 := b_{i_1}, \ldots, x_{i_n} := b_{i_n}]$$

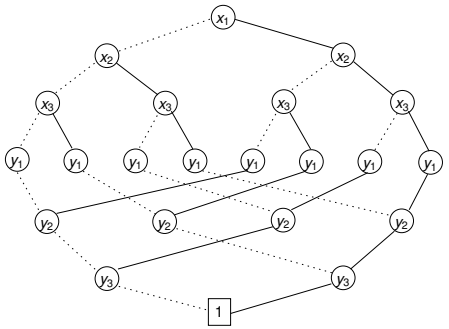
for each permutation $(i_1, \ldots, i_n)$ of $(1, \ldots, n)$

$\Rightarrow$ The value of f depends only on the number of ones!

Examples: $f(\ldots) = x_1 \oplus \ldots \oplus x_n$,
$f(\ldots) = 1$ iff $\geq k$ variables x_i are true

symmetric boolean functions have ROBDDs of size in $\mathcal{O}(n^2)$

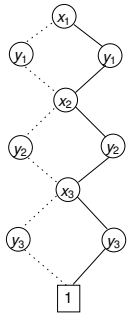
The function stable with exponential ROBDD



The ROBDD of $f(\overline{x}, \overline{y}) = (x_1 \leftrightarrow y_1) \wedge \ldots \wedge (x_n \leftrightarrow y_n)$

has $3 \cdot 2^n - 1$ vertices under ordering $x_1 < \ldots < x_n < y_1 < \ldots < y_n$

The function stable with linear ROBDD



The ROBDD of $f(\overline{x}, \overline{y}) = (x_1 \leftrightarrow y_1) \wedge \ldots \wedge (x_n \leftrightarrow y_n)$

has $3 \cdot n + 2$ vertices under ordering $x_1 < y_1 < \ldots < x_n < y_n$

Optimal variable ordering

- The size of ROBDDs is dependent on the variable ordering
- Is it possible to determine $<$ such that the ROBDD has minimal size?
  - the optimal variable ordering problem for ROBDDs is NP-complete (Bollig & Wegener, 1996)
- There are many boolean functions with large ROBDDs
- How to deal with this problem in practice?
  - guess a variable ordering in advance
  - rearrange the variable ordering during the manipulations of ROBDDs