Graph-Based Algorithms for Boolean Function Manipulation

Sofia Cassel

March 9, 2012
Boolean Algebra

- Building blocks:
  0, 1 (true, false)
  \( x \land y \)
  \( x \lor y \)
  \( x \rightarrow y \)
  \( x \leftrightarrow y \)

- Any Boolean expression can be written using these (and parentheses)

- Truth table: represents assignment of truth values to variables

- Tautology: always true regardless of truth assignments
  Satisfiable: there is a truth assignment that renders the formula true

- Normal forms: \textit{CNF, DNF}

- Satisfiability: NP-complete
"if $x$ then $t_1$ else $t_0$" denotes $(x \rightarrow t_1) \land (\neg x \rightarrow t_0)$

$t = x \rightarrow t_1, t_0$

Boolean expression built from an if-then-else operator and $\{0, 1\}$: all tests performed on variables

Every Boolean formula has an INF

Example (INF)

$\neg p : \text{if } p \text{ then } \bot \text{ else } \top$
Shannon expansion

- Represent a Boolean function as the sum of two subfunctions:

\[ f = x_i \cdot f\big|_{x_i=1} + \neg x_i \cdot f\big|_{x_i=0} \]

- \( f \) is expanded around variable \( x_i \)
- \( f\big|_{x_i=b} \) = the restriction of \( f \) to the case where \( x_i = b \)

- Use Shannon expansion to generate an INF from any Boolean expression:
  - Expression contains no variables \( \rightarrow 0, 1 \) (true, false)
  - Expression contains variables \( \rightarrow \) Do Shannon expansion

- Result of Shannon expansion: binary decision tree
- A binary decision tree can be transformed into a BDD!
A BDD is a rooted DAG with:

- one or two terminal nodes, outdegree 0, labeled 0 or 1
- a set of nonterminal nodes $u$ of outdegree 2. The edge are $\text{high}(u)$; $\text{low}(u)$; the associated variable is $\text{var}(u)$

Introduced by Lee & Akers
Ordered and Reduced BDDs

- Introduced by Bryant [this paper]
- **OBDD**: a BDD where variables are ordered
  - Minimality depends on ordering of variables
- **ROBDD**: a reduced OBDD
  - All identical nodes are shared
  - All redundant tests are eliminated
- Example [on blackboard]
Operations on ROBDDs

- **Apply**: Takes graphs representing $f_1$ and $f_2$ and an operator $op$, produces graph representing $f_1 \ op \ f_2$
  
  Start at the root of both graphs ($v_1$, $v_2$)
  
  Reduce if necessary.

- **Restriction**: restricts a Boolean function with respect to truth value of a variable $x_i$
  
  Replace each node with variable $x_i$ by the corresponding branch
  
  Transforms $f$ into $f|_{x_i=b}$ where $b$ is a constant

- **Composition, Satisfy**
BDDs in Verification

- Used in hardware verification (equivalence of circuits)
- Used in model checking to determine whether model $M$ satisfies set of properties $P$
- Every Boolean expression has a unique canonical BDD representation
References

- Randal E. Bryant (1986): *Graph-Based Algorithms for Boolean Function Manipulation* [the main paper]