Symbolic Model Checking with ROBDDs

Lecture #14 of Advanced Model Checking

Joost-Pieter Katoen

Lehrstuhl 2: Software Modeling & Verification

E-mail: `katoen@cs.rwth-aachen.de`

January 10, 2011

Symbolic representation of transition systems

- let $\mathit{TS} = (S, \rightarrow, I, \mathit{AP}, L)$ be a "large" finite transition system
  - the set of actions is irrelevant here and has been omitted, i.e., $\rightarrow \subseteq S \times S$
- For $n \geqslant \lceil \log |S| \rceil$, let injective function $\mathit{enc} : S \rightarrow \{ 0, 1 \}^n$
  - note: $\mathit{enc}(S) = \{0, 1\}^n$ is no restriction, as all elements $\{ 0, 1 \}^n \setminus \mathit{enc}(S)$ can be treated as the encoding of pseudo states that are unreachable
- Identify the states $s \in S = \mathit{enc}^{-1}(\{ 0, 1 \}^n)$ with $\mathit{enc}(s) \in \{0, 1\}^n$
- And $T \subseteq S$ by its characteristic function $\chi_T : \{ 0, 1 \}^n \rightarrow \{ 0, 1 \}$
  - that is $\chi_T(\mathit{enc}(s)) = 1$ if and only if $s \in T$
- And $\rightarrow \subseteq S \times S$ by the Boolean function $\Delta : \{ 0, 1 \}^{2n} \rightarrow \{ 0, 1 \}$
  - such that $\Delta\left(\mathit{enc}(s), \mathit{enc}(s')\right) = 1$ if and only if $s \rightarrow s'$

Switching functions

- Let *Var* $= \{z_1, \ldots, z_m\}$ be a finite set of Boolean variables
- An evaluation is a function η : *Var* $\rightarrow \{ 0, 1 \}$
  - let *Eval*$(z_1, \ldots, z_m)$ denote the set of evaluations for $z_1, \ldots, z_m$
  - shorthand $[z_1 = b_1, \ldots, z_m = b_m]$ for $\eta(z_1) = b_1, \ldots, \eta(z_m) = b_m$
- f : *Eval*(*Var*) $\rightarrow \{ 0, 1 \}$ is a *switching function* for *Var* $= \{z_1, \ldots, z_m\}$
- Logical operations and quantification are defined by:

$$
\begin{aligned}
f_1(\cdot) \wedge f_2(\cdot) &= \min\{ f_1(\cdot), f_2(\cdot) \} \\
f_1(\cdot) \vee f_2(\cdot) &= \max\{ f_1(\cdot), f_2(\cdot) \} \\
\exists z.\, f(\cdot) &= f(\cdot)|_{z=0} \vee f(\cdot)|_{z=1}, \text{ and} \\
\forall z.\, f(\cdot) &= f(\cdot)|_{z=0} \wedge f(\cdot)|_{z=1}
\end{aligned}
$$

Symbolic model checking

- Take a symbolic representation of a transition system (Δ and χ_B)
- Backward reachability $\mathit{Pre}^*(B) = \{ s \in S \mid s \models \exists \Diamond B \}$
- Initially: $f_0 = \chi_B$ characterizes the set $T_0 = B$
- Then, successively compute the functions $f_{j+1} = \chi_{T_{j+1}}$ for:

$$T_{j+1} = T_j \cup \{ s \in S \mid \exists s' \in S.\, s' \in \mathit{Post}(s) \wedge s' \in T_j \}$$

- Second set is given by: $\exists \overline{x}'.\, (\underbrace{\Delta(\overline{x}, \overline{x}')}_{s' \in \mathit{Post}(s)} \;\wedge\; \underbrace{f_j(\overline{x}')}_{s' \in T_j})$
  - $f_j(\overline{x}')$ arises from f_j by renaming the variables x_i into their primed copies x_i'

Symbolic computation of $Sat(\exists(C \mathsf{U} B))$

$f_0(\overline{x}) := \chi_B(\overline{x})$;
$j := 0$;
repeat
  $f_{j+1}(\overline{x}) := f_j(\overline{x}) \vee \big(\chi_C(\overline{x}) \wedge \exists \overline{x}'. (\Delta(\overline{x}, \overline{x}') \wedge f_j(\overline{x}')) \big)$;
  $j := j + 1$
until $f_j(\overline{x}) = f_{j-1}(\overline{x})$;
return $f_j(\overline{x})$.

Symbolic computation of $Sat(\exists\Box\, B)$

Compute the largest set $T \subseteq B$ with*Post*$(t) \cap T \neq \varnothing$ for all $t \in T$

Take $T_0 = B$ and $T_{j+1} = T_j \cap \{ s \in S \mid \exists s' \in S.\, s' \in$ *Post*$(s) \wedge s' \in T_j \}$

Symbolically this amounts to:

$f_0(\overline{x}) := \chi_B(\overline{x})$;
$j := 0$;
repeat
  $f_{j+1}(\overline{x}) := f_j(\overline{x}) \wedge \exists \overline{x}'.\, (\, \Delta(\overline{x}, \overline{x}') \wedge f_j(\overline{x}')\,)$;
  $j := j + 1$
until $f_j(\overline{x}) = f_{j-1}(\overline{x})$;
return $f_j(\overline{x})$.

Symbolic model checkers mostly use ROBDDs to represent switching functions

Ordered Binary Decision Diagram

Let $\wp$ be a variable ordering for *Var* where $z_1 <_\wp \ldots <_\wp z_m$

An $\wp$-OBDD is a tuple $\mathfrak{B} = (V, V_I, V_T, \mathit{succ}_0, \mathit{succ}_1, \mathit{var}, \mathit{val}, v_0)$ with

- a finite set V of nodes, partitioned into V_I (inner) and V_T (terminals)
  - and a distinguished root $v_0 \in V$
- successor functions $\mathit{succ}_0, \mathit{succ}_1 : V_I \rightarrow V$
  - such that each node $v \in V \setminus \{v_0\}$ has at least one predecessor
- labeling functions $\mathit{var} : V_I \rightarrow \mathit{Var}$ and $\mathit{val} : V_T \rightarrow \{0, 1\}$ satisfying

$$v \in V_I \ \wedge \ w \in \{\mathit{succ}_0(v), \mathit{succ}_1(v)\} \cap V_I \ \Rightarrow \ \mathit{var}(v) <_\wp \mathit{var}(w)$$

Reduced OBDDs

A $\wp$-OBDD $\mathfrak{B}$ is *reduced* if for every pair (v, w) of nodes in $\mathfrak{B}$:

$$v \neq w \text{ implies } f_v \neq f_w$$

$\Rightarrow$ $\wp$-ROBDDs any $\wp$-consistent cofactor is represented by exactly one node

Universality and canonicity theorem

[Fortune, Hopcroft & Schmidt, 1978]

Let *Var* be a finite set of Boolean variables and $\wp$ a variable ordering for *Var*. Then:

(a) For each switching function f for *Var* there exists a $\wp$-ROBDD $\mathfrak{B}$ with $f_{\mathfrak{B}} = f$

(b) Any $\wp$-ROBDDs $\mathfrak{B}$ and $\mathfrak{C}$ with $f_{\mathfrak{B}} = f_{\mathfrak{C}}$ are isomorphic

Any $\wp$-OBDD $\mathfrak{B}$ for f is reduced iff *size*$(\mathfrak{B}) \leqslant$ *size*$(\mathfrak{C})$ for each $\wp$-OBDD $\mathfrak{C}$ for f

Synthesis of ROBDDs

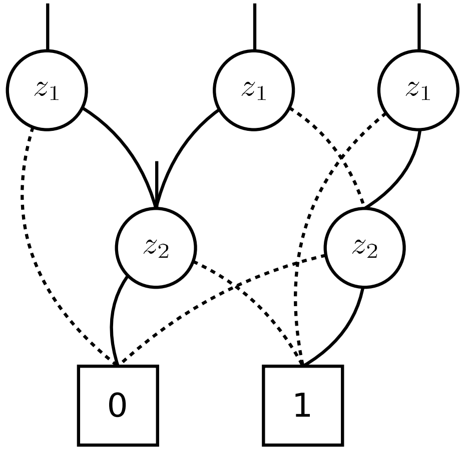
- Construct a $\wp$-ROBDD for f_1 *op* f_2 given $\wp$-ROBDDs for f_1 and f_2
  - where *op* is a Boolean connective such as disjunction, implication, etc.
- Idea: use a single ROBDD with (global) variable ordering $\wp$ to represent several switching functions
- This yields a shared OBDD, which is:

  a combination of several ROBDDs with variable ordering $\wp$
  by sharing nodes for common $\wp$-consistent cofactors

- The size of $\wp$-SOBDD $\overline{\mathfrak{B}}$ for functions $f_1, \ldots, f_k$ is at most $N_{f_1} + \ldots + N_{f_k}$ where N_f denotes the size of the $\wp$-ROBDD for f

Shared OBDDs

A shared $\wp$-OBDD is an OBDD with multiple roots



Shared OBDD representing $\underbrace{z_1 \wedge \neg z_2}_{f_1}$, $\underbrace{\neg z_2}_{f_2}$, $\underbrace{z_1 \oplus z_2}_{f_3}$ and $\underbrace{\neg z_1 \vee z_2}_{f_4}$

Main underlying idea: combine several OBDDs with same variable ordering such that common $\wp$-consistent co-factors are shared

Using shared OBDDs for model checking Φ

Use a single SOBDD for:

- $\Delta(\overline{x}, \overline{x}')$ for the transition relation
- $f_a(\overline{x})$, $a \in AP$, for the satisfaction sets of the atomic propositions
- The satisfaction sets *Sat*(Ψ) for the state subformulae Ψ of Φ

In practice, often the interleaved variable order for Δ is used.

Synthesizing shared ROBDDs

Relies on the use of two tables

- The unique table
  - keeps track of ROBDD nodes that already have been created
  - table entry $\langle \mathit{var}(v), \mathit{succ}_1(v), \mathit{succ}_0(v) \rangle$ for each inner node v
  - main operation: *find_or_add*(z, v_1, v_0) with $v_1 \neq v_0$
    - return v if there exists a node $v = \langle z, v_1, v_0 \rangle$ in the ROBDD
    - if not, create a new z-node v with $\mathit{succ}_0(v) = v_0$ and $\mathit{succ}_1(v) = v_1$
  - implemented using hash functions (expected access time is $\mathcal{O}(1)$)

- The computed table
  - keeps track of tuples for which ITE has been executed (memoization)

  $\Rightarrow$ realizes a kind of dynamic programming

ITE normal form

The ITE (if-then-else) operator: $\mathit{ITE}(g, f_1, f_2) \;=\; (g \wedge f_1) \;\vee\; (\neg\, g \wedge f_2)$

The ITE operator and the representation of the SOBDD nodes in the unique table:

$$f_v \;=\; \mathit{ITE}\Big(\, z,\; f_{\mathit{succ}_1(v)},\; f_{\mathit{succ}_0(v)} \,\Big)$$

Then:

$$\begin{aligned}
\neg f \;&=\; \mathit{ITE}(f, 0, 1) \\
f_1 \vee f_2 \;&=\; \mathit{ITE}(f_1, 1, f_2) \\
f_1 \wedge f_2 \;&=\; \mathit{ITE}(f_1, f_2, 0) \\
f_1 \oplus f_2 \;&=\; \mathit{ITE}(f_1, \neg f_2, f_2) \;=\; \mathit{ITE}(f_1, \mathit{ITE}(f_2, 0, 1), f_2)
\end{aligned}$$

If g, f_1, f_2 are switching functions for *Var*, $z \in \mathit{Var}$ and $b \in \{0, 1\}$, then

$$\mathit{ITE}(g, f_1, f_2)|_{z=b} \;=\; \mathit{ITE}(g|_{z=b}, f_1|_{z=b}, f_2|_{z=b})$$

ITE-operator on shared OBDDs

- A node in a $\wp$-SOBDD for representing *ITE*(g, f_1, f_2) is a node w with *info*$\langle z, w_1, w_0 \rangle$ where:
  - z is the minimal (wrt. $\wp$) essential variable of *ITE*(g, f_1, f_2)
  - w_b is an SOBDD-node with $f_{w_b} =$ *ITE*$(g|_{z=b}, f_1|_{z=b}, f_2|_{z=b})$
- This suggests a recursive algorithm:
  - determine z
  - recursively compute the nodes for ITE for the cofactors of g, f_1 and f_2

$ITE(u, v_1, v_2)$ on shared OBDDs (initial version)

if u is terminal **then**
  if *val*$(u) = 1$ **then**
    $w := v_1$ (* $ITE(1, f_{v_1}, f_{v_2}) = f_{v_1}$ *)
  else
    $w := v_2$ (* $ITE(0, f_{v_1}, f_{v_2}) = f_{v_2}$ *)
  fi
else
  $z := \min\{var(u), var(v_1), var(v_2)\}$; (* minimal essential variable *)
  $w_1 := ITE(u|_{z=1}, v_1|_{z=1}, v_2|_{z=1})$;
  $w_0 := ITE(u|_{z=0}, v_1|_{z=0}, v_2|_{z=0})$;
  if $w_0 = w_1$ **then**
    $w := w_1$; (* elimination rule *)
  else
    $w :=$ *find_or_add*(z, w_1, w_0); (* isomorphism rule *)
  fi
fi
return w

ROBDD size under ITE

The size of the $\wp$-ROBDD for *ITE*(g, f_1, f_2) is bounded by $N_g \cdot N_{f_1} \cdot N_{f_2}$
where N_f denotes the size of the $\wp$-ROBDD for f

for some ITE-functions optimisations are possible, e.g., $f \oplus g$

ROBDD size under ITE

The size of the $\wp$-ROBDD for *ITE*(g, f_1, f_2) is bounded by $N_g \cdot N_{f_1} \cdot N_{f_2}$

where N_f denotes the size of the $\wp$-ROBDD for f

But how to avoid multiple invocations to ITE?

$\Rightarrow$ Store triples (u, v_1, v_2) for which ITE already has been computed

Efficiency improvement by memoization

if there is an entry for (u, v_1, v_2, w) in the computed table **then**
  return node w
else
  if u is terminal **then**
    if *val*$(u) = 1$ **then** $w := v_1$ **else** $w := v_2$ **fi**
  else
    $z := \min\{var(u), var(v_1), var(v_2)\}$;
    $w_1 := ITE(u|_{z=1}, v_1|_{z=1}, v_2|_{z=1})$;
    $w_0 := ITE(u|_{z=0}, v_1|_{z=0}, v_2|_{z=0})$;
    if $w_0 = w_1$ **then** $w := w_1$ **else** $w :=$ *find_or_add*(z, w_1, w_0) **fi**;
    insert (u, v_1, v_2, w) in the computed table;
    return node w
  fi
fi

The number of recursive calls for the nodes u, v_1, v_2 equals the $\wp$-ROBDD size

of $ITE(f_u, f_{v_1}, f_{v_2})$, which is bounded by $N_u \cdot N_{v_1} \cdot N_{v_2}$

Some experimental results

- Traffic alert and collision avoidance system (TCAS) (1998)
  - 277 boolean variables, reachable state space is about $9.6 10^{56}$ states
  - $|\mathrm{B}| = 124,618$ vertices (about 7.1 MB), construction time 46.6 *sec*
  - checking $\forall\Box\,(p \longrightarrow q)$ takes 290 sec and 717,000 BDD vertices
- Synchronous pipeline circuit (1992)
  - pipeline with 12 bits: reachable state space of $1.5 10^{29}$ states
  - checking safety property takes about $10^4 - 10^5$ *sec*
  - $|\mathrm{B}_{\rightarrow}|$ is linear in data path width
  - verification of 32 bits (about 10^{120} states): 1h 25m
  - using partitioned transition relations

Some other types of BDDs

- Zero-suppressed BDDs
  - like ROBDDs, but non-terminals whose 1-child is leaf 0 are omitted
- Parity BDDs
  - like ROBDDs, but non-terminals may be labeled with $\oplus$; no canonical form
- Edge-valued BDDs
- Multi-terminal BDDs (or: algebraic BDDs)
  - like ROBDDs, but terminals have values in $\mathbb{R}$, or $\mathbb{N}$, etc.
- Binary moment diagrams (BMD)
  - generalization of ROBDD to linear functions over bool, int and real
  - uses edge weights

Further reading

- R. Bryant: Graph-based algorithms for Boolean function manipulation, 1986
- R. Bryant: Symbolic boolean manipulation with OBDDs, Computing Surveys, 1992
- M. Huth and M. Ryan: Binary decision diagrams, Ch 6 of book on Logics, 1999
- H.R. Andersen: Introduction to BDDs, Tech Rep, 1994
- K. McMillan: Symbolic model checking, 1992
- Rudell: Dynamic variable reordering for OBDDs, 1993

Advanced reading: Ch. Meinel & Th. Theobald (Springer 1998)