Symbolic Model Checking

Lecture #14 of Advanced Model Checking

Joost-Pieter Katoen

Lehrstuhl 2: Software Modeling & Verification

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

December 14, 2006

Symbolic model checking

- CTL model checking using ROBDDs
  - determine whether $I \subseteq \mathit{Sat}(\Phi)$ for transition system *TS*
- Represent *TS* by means of ROBDDs
- Represent $\mathit{Sat}(\Psi)$ for sub-formula Ψ (of Φ) by an ROBDD
  - manipulate these ROBDDs to obtain $\mathit{Sat}(\Psi \wedge \Psi')$, $\mathit{Sat}(\bigcirc \Psi)$, and so on
  - most involved cases: until-operator and $\Box\Phi$
- Check whether $I \subseteq \mathit{Sat}(\Phi)$ using their ROBDDs

this approach is also applicable to the μ-calculus

Transition systems as boolean functions

- Assume each state is uniquely labeled
- Assume a fixed total order on propositions: $a_1 < a_2 < \ldots < a_K$
- Represent a state by a *boolean function*
  - over the 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
- Represent I and $\longrightarrow$ by their characteristic (boolean) functions
  - $f_I([\![s]\!]) = 1$ if and only if $s \in I$
  - $f_{\rightarrow}([\![s]\!], [\![\alpha]\!], [\![t]\!]) = 1$ if and only if $s \xrightarrow{\alpha} t$

Interleaved variable ordering

- Which variable ordering to use for transition relations?
- The *interleaved variable ordering*:
  - for encodings $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$ of state s and t respectively:

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

- This variable ordering yields compact ROBDDs for binary relations
  - for transition relation with $z_1 \ldots z_m$ be the encoding of action α, take:

$$\underbrace{z_1 < z_2 < \ldots < z_m}_{\text{encoding of } \alpha} < \underbrace{x_1 < y_1 < x_2 < y_2 < \ldots < x_n < y_n}_{\text{interleaved order of statea}}$$

Operations on ROBDDs

Algorithm	Inputs	Output ROBDD
Reduce	B (not reduced)	B′ (reduced) with $f_{\mathrm{B}} = f_{\mathrm{B}'}$
Not	B_f	$\mathrm{B}_{\neg f}$
Apply	B_f, B_g, binary logical operator *op*	$\mathrm{B}_{f\ op\ g}$
Restrict	B_f, variable x, boolean value b	$\mathrm{B}_{f[x:=b]}$
Rename	B_f, variables x and y	$\mathrm{B}_{f[x:=y]}$
Exists	B_f, variable x	$\mathrm{B}_{\exists x.\ f}$

Operations on ROBDDs

Algorithm	Output	Time complexity	Space complexity
REDUCE	B′ (reduced) with $f_{\mathsf{B}} = f_{\mathsf{B}'}$	$\mathcal{O}(\lvert B_f \rvert \cdot \log \lvert \mathsf{B}_f \rvert)$	$\mathcal{O}(\lvert \mathsf{B}_f \rvert)$
NOT	$\mathsf{B}_{\neg f}$	$\mathcal{O}(\lvert \mathsf{B}_f \rvert)$	$\mathcal{O}(\lvert \mathsf{B}_f \rvert)$
APPLY	$\mathsf{B}_{f \; \mathit{op} \; g}$	$\mathcal{O}(\lvert \mathsf{B}_f \rvert \cdot \lvert \mathsf{B}_g \rvert)$	$\mathcal{O}(\lvert \mathsf{B}_f \rvert \cdot \lvert \mathsf{B}_g \rvert)$
RESTRICT	$\mathsf{B}_{f[x:=b]}$	$\mathcal{O}(\lvert \mathsf{B}_f \rvert)$	$\mathcal{O}(\lvert \mathsf{B}_f \rvert)$
RENAME	$\mathsf{B}_{f[x:=y]}$	$\mathcal{O}(\lvert \mathsf{B}_f \rvert)$	$\mathcal{O}(\lvert \mathsf{B}_f \rvert)$
EXISTS	$\mathsf{B}_{\exists x.\, f}$	$\mathcal{O}(\lvert \mathsf{B}_f \rvert^2)$	$\mathcal{O}(\lvert \mathsf{B}_f \rvert^2)$

operations are only efficient if f (and g) have compact ROBDD representations

Model checking CTL using ROBDDs

The set of CTL formulas in *existential normal form* (ENF) is given by:

$$\Phi \quad ::= \quad \text{true} \;\Big|\; a \;\Big|\; \Phi_1 \wedge \Phi_2 \;\Big|\; \neg\Phi \;\Big|\; \exists\bigcirc\Phi \;\Big|\; \exists(\Phi_1 \,\mathsf{U}\, \Phi_2) \;\Big|\; \exists\Box\Phi$$

For each CTL formula, there exists an equivalent CTL formula in ENF

Model checking CTL

- Convert the formula Φ' into an equivalent Φ in ENF
- How to check whether state *TS* satisfies Φ?
  - compute *recursively* the set *Sat*(Φ) of states that satisfy Φ
  - check whether all initial states belong to *Sat*(Φ)
- Recursive bottom-up computation:
  - consider the parse-tree of Φ
  - start to compute *Sat*(a), for all leafs in the tree
  - then go one level up in the tree and check the formula of these nodes
  - then go one level up and check the formula of these nodes
  - and so on....... until the root of the tree (i.e., Φ) is checked

Computing $Sat(\Phi)$ symbolically

Input: CTL-formula Φ in ENF
Output: ROBDD $\mathsf{B}_{Sat(\Phi)}$

---

switch(Φ):

true	:	**return** $\textsc{Const}(1)$;
false	:	**return** $\textsc{Const}(0)$;
a_i	:	**return** ROBDD B_f for $f(x_1, \ldots, x_n) = x_i$;
$\neg\Psi$	:	**return** $\textsc{Not}(\mathit{bddSat}(\Psi))$
$\Phi_1 \wedge \Phi_2$	:	**return** $\textsc{Apply}(\wedge, \mathit{bddSat}(\Phi_1), \mathit{bddSat}(\Phi_2))$
$\exists\bigcirc\Psi$	:	**return** $\mathit{bddEX}(\Psi)$;
$\exists(\Phi_1 \mathsf{U} \Phi_2)$	:	**return** $\mathit{bddEU}(\Phi_1, \Phi_2)$
$\exists\Box\Psi$	:	**return** $\mathit{bddEG}(\Psi)$

end switch

The next-step operator

$$\textit{Sat}(\bigcirc\Phi) \;=\; \{\, s \in S \mid \exists s'.\, s \longrightarrow s' \text{ and } s' \in \textit{Sat}(\Phi)\,\}$$

Input: CTL-formula Φ in ENF
Output: ROBDD $\mathrm{B}_{\textit{Sat}(\bigcirc\Phi)}$

B := *bddSat*(Φ); (* *Sat*(Φ) *)
B := RENAME(B, $x_1, \ldots, x_n, x'_1, \ldots, x'_n$);
B := APPLY($\wedge$, $\mathrm{B}_{\rightarrow}$, B); (* *Pre*(*Sat*(Φ)) *)
return EXISTS(B, $x'_1, \ldots, x'_n$)

Characterization for until and globally

For all CTL formulas Φ, Ψ over *AP* it holds:

- *Sat*$(\exists(\Phi \,\mathsf{U}\, \Psi))$ is the smallest subset T of S, such that:

  (1) *Sat*$(\Psi) \subseteq T$ and

  (2) $s \in Sat(\Phi)$ and *Post*$(s) \cap T \neq \varnothing$ implies $s \in T$

- *Sat*$(\exists\Box\Phi)$ is the largest subset T of S, such that:

  (3) $T \subseteq$ *Sat*(Φ) and

  (4) $s \in T$ implies *Post*$(s) \cap T \neq \varnothing$

where $TS = (S, Act, \longrightarrow, I, AP, L)$ is a transition system without terminal states

Computation of *Sat*

```
switch(Φ):

    a              :  return { s ∈ S | a ∈ L(s) };
    ...            :  ......
    ∃(Φ1 U Φ2)     :  T := Sat(Φ2);   (* compute the smallest fixed point *)
                      while (Sat(Φ1) \ T ∩ Pre(T) ≠ ∅) do
                         let  s ∈ Sat(Φ1) \ T ∩ Pre(T);
                         T := T ∪ { s };
                      od;
                      return T;
    ∃□Ψ            :  T := Sat(Ψ);    (* compute the greatest fixed point *)
                      while ∃s ∈ T. Post(s) ∩ T = ∅ do
                         let  s ∈ { s ∈ T | Post(s) ∩ T = ∅ };
                         T := T \ { s };
                      od;
                      return T;
end switch
```

Computing *Sat*$(\exists(\Phi \, \mathsf{U} \, \Psi))$ and *Sat*$(\exists\Box\Phi)$

- Computing *Sat*$(\exists(\Phi \, \mathsf{U} \, \Psi))$ iteratively:
  - $T_0 := \mathit{Sat}(\Psi)$
  - $T_{i+1} := T_i \cup \{ s \in \mathit{Sat}(\Phi) \mid \exists s'. \, s \rightarrow s' \text{ and } s' \in T_i \}$
- Computing *Sat*$(\exists\Box\Phi)$ iteratively:
  - $T_0 := \mathit{Sat}(\Phi)$
  - $T_{i+1} := T_i \cap \{ s \in \mathit{Sat}(\Phi) \mid \exists s'. \, s \rightarrow s' \text{ and } s' \in T_i \}$

Existential until

Input: CTL-formulas Φ, Ψ in ENF
Output: ROBDD $B_{Sat(\exists(\Phi \,\mathsf{U}\, \Psi))}$

---

var N, P, B : *ROBDD*;
N $:=$ *bddSat*(Ψ);
P $:=$ CONST(0);
B $:=$ *bddSat*(Φ);
while (N $\neq$ P) **do**
  P $:=$ N; (* T_i *)
  N $:=$ RENAME(N, $x_1, \ldots, x_n, x'_1, \ldots, x'_n$);
  N $:=$ APPLY($\wedge$, B$_\rightarrow$, N); (* *Pre*(T_i) *)
  N $:=$ EXISTS(N, $x'_1, \ldots, x'_n$);
  N $:=$ APPLY($\wedge$, N, B); (* *Pre*(T_i) $\cap$ *Sat*(Φ) *)
  N $:=$ APPLY($\vee$, P, N); (* $T_{i+1} = T_i \cup \ldots\ldots$ *)
od
return N

Possibly always

Input: CTL-formula Φ in ENF
Output: ROBDD $B_{Sat(\exists\Box\Phi)}$

---

var N, P, B : *ROBDD*;
B := *bddSat*(Φ);
N := B;
P := CONST(0);
while (N $\neq$ P) **do**
  P := N; (* T_i *)
  N := RENAME(N, $x_1, \ldots, x_n, x'_1, \ldots, x'_n$);
  N := APPLY($\wedge$, $B_{\rightarrow}$, N); (* *Pre*(T_i) *)
  N := EXISTS(N, $x'_1, \ldots, x'_n$);
  N := APPLY($\wedge$, N, B); (* *Pre*(T_i) $\cap$ *Sat*(Φ) *)
  N := APPLY($\wedge$, P, N); (* $T_{i+1} = T_i \cap \ldots\ldots$ *)
od
return N

Compositional generation of ROBDDs

- Let $TS_i = (S_i, Act, \rightarrow_i, I_i, AP, L_i)$ for $i = 1, 2$ and for $H \subseteq Act$:

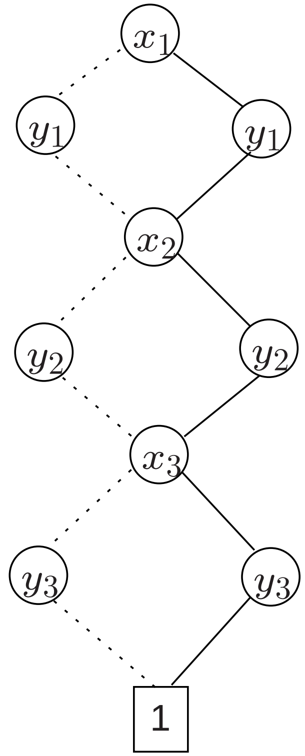
$$TS = TS_1 \, ||_H \, TS_2 \quad \text{the parallel composition of } TS_1 \text{ and } TS_2$$

- $\rightarrow_i$ is represented by B_i and H by B_H
- The ROBDD B representing the transition relation of TS is given by:

$$\underbrace{((B_1 \wedge B_H) \;\wedge\; (B_2 \wedge B_H))}_{\text{synchronization}} \;\vee\; \underbrace{(B_1 \wedge B_{\overline{H}} \wedge B_{f_{stab1}})}_{\text{own move by } TS_1} \;\vee\; \underbrace{(B_2 \wedge B_{\overline{H}} \wedge B_{f_{stab2}})}_{\text{own move by } TS_2}$$

  - where $f_{stabi} = \bigwedge_{j=1}^{n_i} (x_j^{(i)} \leftrightarrow y_j^{(i)})$ with n_i = # state variables in TS_i
  - and $x_j^{(i)}$, $y_j^{(i)}$ encode the source and target state of a transition in TS_i

The function stable with linear ROBDD



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

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

Example of compositional generation

Compositional generation of ROBDDs

- The size of *TS* is exponential in number of concurrent processes
  - $|\mathit{TS}| = |\mathit{TS}_1 \,||_H\, \mathit{TS}_2|$ is bounded from above by $|S_1|\cdot|S_2|$
- The size of B is linear in number of concurrent processes
  - $|\mathsf{B}|$ is bounded from above by:

$$|\mathit{Act}|\cdot \left(|\mathsf{B}_1| + |\mathsf{B}_2| + |\mathsf{B}_{f_{stab1}}| + |\mathsf{B}_{f_{stab2}}|\right)$$

  - by exploiting the interleaved variable ordering
- Compositional generation of ROBDDs is beneficial
  - it reduces the peak memory requirements
  - size of BDD representation is linear in number of components

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 $\forall\Box(p \rightarrow q)$ 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)