Bounded Model Checking (BMC)

Dr Olga Tveretina

Summer semester, 2009
Background: model checking

Given:
- a finite transition system $M$
- a property $P$ (in some temporal logic)

The model checking problem:
- Does $P$ holds in $M$?

$M \models P$
Temporal properties

• **Safety properties:**
  - “Always x=y”
  - $G (x=y)$

• **Liveness properties:**
  - “Reset can always be reached”
  - $GF \text{ Reset}$
  - “From some point on, always switch\_on”
  - $FG \text{ switch\_on}$
OBDDs and symbolic model checking

- OBDD is a canonical form to represent Boolean functions
- They are often more compact than 'traditional' normal forms as CNFs, DNFs and can be manipulated efficiently
- The reachable state-space is represented by a OBDD
- The property is evaluated recursively, by iterative fix point computations on the reachable state-space
BDDs are a canonical representation, but often become too large.

Variable ordering must be uniform along paths.

Selecting right variable ordering very important to build small BDDs

- time consuming or needs manual intervention
- in some cases no space efficient variable ordering exists

Alternative approaches to model checking use SAT procedures
Advantages of SAT procedures

- SAT procedures also operate on Boolean formulas but do not use canonical forms
- Do not suffer from the potential space explosion of BDDs
- Different orderings of variables are possible on different branches
- There exist very efficient implementations
SAT solver progress 1960 - 2010

(E. Clarke)
Bounded model checking

Bounded model checking

- Based on SAT
- There is a counterexample of length $k \iff$ propositional formula is satisfiable
- BMC for LTL reduced to SAT in poly time

**Example:**

- Most of the safety properties can be expressed as 'always $p$', where $p$ is a propositional variable
- Is there a state reachable within $k$ steps that satisfies $\neg p$?
Bounded model checking

- Existential model checking problem $M \models Ef$ for an LTL formula $f$ and a Knipke structure $M$

- To look for a witness to the property that can be represented within a bound of $k$ steps

- Given $k$, the problem is reduced to the satisfiability of a propositional formula $[[M,f]]_k$

- If $[[M,f]]_k$ is satisfiable then the propositional model provides a witness of $k$ steps to $f$
Bounded model checking

- The method is not complete

- If $[[M,f]]_k$ is unsatisfiable then nothing can be said about the existence of a solution for $M \models f$ models with higher bound

- The typical technique is to generate and solve $[[M,f]]_k$ for increasing values of $k$
Bounded model checking

- Effective and practical technique, especially in the process of falsification, i.e. bug funding
- Bounded model checking based on SAT procedures not BDD
- Smart DFS search of SAT potentially will get faster to a satisfying sequence (counterexample)
- No exponential space
Creation of propositional formula

- **Given:**
  - a transition system $M$
  - a temporal logic formula $f$
  - a user-supplied bound $k$

- **Construct:**
  - a propositional formula $[[M,f]]_k$ is satisfiable iff $f$ is valid along some computation path of $M$
Creation of propositional formula

- For state transition system $M$ and time bound $k$, the unrolled transition relation is

$$[[M]]_k = \bigwedge_{i=0}^{k-1} I(s_0) \land T(s_i, s_{i+1})$$

- $I(s_0)$ is the characteristic function of the set of initial states
- $T(s_i, s_{i+1})$ is the characteristic function of the transition relation

- A propositional formula $[[M,f]]_k$ is satisfiable iff $f$ is valid along some computation path of $M$
Consider the CTL formula $\text{EF } p$

Check whether $\text{EF } p$ can be verified in two time steps, i.e. $k=2$

$$[[M,f]]_2 = I(s_0) \land T(s_0,s_1) \land T(s_1,s_2) \land (p(s0) \lor p(s1) \lor p(s2))$$

Here, $(p(s0) \lor p(s1) \lor p(s2))$ is $[[\text{EF } p]]_2$
Safety property example

2-bit counter: the least significant bit represented by a Boolean variable A and the most significant by B

Transition relation:
\[(A' \leftrightarrow \neg A) \land (B' \leftrightarrow A \diamond B)\]

\(\diamond\) stands for XOR, \(\leftrightarrow\), XNOR

\[I(s0): (\neg A_0 \land \neg B_0) \land \]
\[T(s0,s1): ((A_1 \leftrightarrow \neg A_0) \land (B_1 \leftrightarrow (A_0 \diamond B_0))) \land \]
\[T(s1,s2): ((A_2 \leftrightarrow \neg A_1) \land (B_2 \leftrightarrow (A_1 \diamond B_1))) \land \]
\[p(s0): (A_0 \land B_0) \lor \]
\[p(s1): A_1 \land B_1 \lor \]
\[p(s2): A_2 \land B_2 )\]
Liveness property example

- We add a transition from state (1,0) back to itself

- Define:
  \[ \text{inc}(s, s') = (A' \iff \neg A) \land (B' \iff (A \spadesuit B)) \]
  \[ T(s, s') = \text{inc}(s, s') \lor (B \land \neg A \land B' \land \neg A') \]

- A counter must eventually reach state (1,1): \( \text{AF} (b \land a) \)

- A counterexample that demonstrates this would be a path starting at the initial state, in which the counter never reaches state (1,1): \( \text{EG} \ p, \text{where } p = \neg B \lor \neg A \)
Liveness property example

- Set the time bound $k$ for checking $\text{EG } p$ at 2
- All candidate paths will then have $k+1$, or 3 states, an initial one and two reached upon two successive transitions: $s_0$, $s_1$, $s_2$
- The transition relation must hold for $k=2$

\[
[[M]]_2 = I(s_0) \wedge T(s_0,s_1) \wedge T(s_1,s_2)
\]

- The sequence of states $s_0$, $s_1$, $s_2$ must be a part of a loop, i.e.

\[
T(s_2,s_3) \wedge (s_3=s_0 \lor s_3=s_1 \lor s_3=s_2)
\]
Liveness property example

I(s0): \((\neg A_0 \land \neg B_0)\)  
T(s0,s1): \(((A_1 \iff \neg A_0) \land (B_1 \iff (A_0 \diamond B_0))) \lor
B_1 \land \neg A_1 \land B_0 \land \neg A_0)\) \land
T(s1,s2): \(((A_2 \iff \neg A_1) \land (B_2 \iff (A_1 \diamond B_1))) \lor
B_2 \land \neg A_2 \land B_1 \land \neg A_1)\) \land
T(s2,s3): \(((A_3 \iff \neg A_2) \land (B_3 \iff (A_2 \diamond B_2))) \lor
B_2 \land \neg A_2 \land B_1 \land \neg A_1)\) \land
s3=s0: \((A_3 \iff A_0) \land (B_3 \iff B_0)\) \lor
s3=s1: \((A_3 \iff A_1) \land (B_3 \iff B_1)\) \lor
s3=s2: \((A_3 \iff A_2) \land (B_3 \iff B_2)\) \lor
p(s0): \((\neg A_0 \land \neg B_0)\) \lor
p(s1): \((\neg A_1 \land \neg B_1)\) \lor
p(s2): \((\neg A_2 \land \neg B_2)\)
Liveness property example

- The formula is **satisfiable**
- The satisfying assignment corresponds to a path from initial state (0,0) to (0,1) and then to (1,0) followed by the self-loop at state (1,0), and is a **counterexample** to $\text{AF} (B \land A)$
- Removing the self-loop would remove the lines

\[
B_i \land \neg A_i \land B_{i-1} \land \neg A_{i-1}
\]

- The formula then become **unsatisfiable**
Determining the bound $k$

- For every model $M$ and LTL property $P$ there exists $k$ such that

$$M \models_k P \implies M \models P$$

- The minimal such $k$ is the completeness threshold (CT)
Determining the bound $k$

- **Diameter $d$** = longest 'shortest path' from an initial state to any other reachable state.

- **Recurrence diameter $r_d$** = longest loop-free path.

- $r_d \geq d$

  ![Diagram](image)
Determining the bound $k$

- **Theorem:** For $Gp$ properties $CT = d$.

- **Theorem:** For $Fp$ properties $CT = rd$.

- **Open problem:** The value of $CT$ for general LTL logic is unknown.
What BMC useful for?

- A.I. planning problems:
  Can we reach a desirable state in k steps?

- Verification of safety properties:
  Can we find a bad state in k steps?

- Verification:
  Can we find a counterexample in k steps?
BMS vs. MC

- **Advantages of BMS:**
  - Counterexamples – found faster and of minimal length
  - Less space, no manual intervention (order on variables for OBDDs)
  - The modern SAT solvers are very efficient

- **Disadvantages of BMS:**
  - With the limit k, completeness cannot be always achieved
BMC

- A model checker called BMC has been implemented, based on bounded model checking.
- It’s input language is a subset of the SMV language.
- It takes in a circuit description, a property to be proven, and a user supplied time bound $k$.
- It then generates the propositional formula.