Algorithms for Model Checking (2IW55)

Lecture 5

Bounded Model Checking

Handout: A. Biere, A. Cimatti, E.M. Clarke, O. Strichman, Y. Zhu: Bounded model checking. Advances in Computers 58: 118-149 (2003)

Tim Willemse
(timw@win.tue.nl)
http://www.win.tue.nl/∼timw
HG 6.81

Outline

LTL Model Checking

LTL-based model checking:

- checks temporal operators along single paths
- LTL is claimed to be more intuitive than CTL (see e.g. [1]):
  - in LTL: X F $p \equiv$ F X p (p holds sometimes in the strict future)
  - in CTL: A X A F $p \stackrel{?}{\equiv}$ A F A X p; does at least one of these express *"p holds sometimes in the strict future"*?
- counter examples are easy: "lasso"
- typical tool: SPIN

[1]. Moshe Vardi, *Branching vs. Linear Time: Final Showdown*, Proc. of TACAS'01, 2001.

LTL Model Checking

Let $M = \langle S, R, L \rangle$ be a Kripke Structure. Recall the syntax and semantics of LTL:

$\mathcal{P}$ ::= true | false | *AP* | $\neg\mathcal{P}$ | $\mathcal{P} \wedge \mathcal{P}$ | $\mathcal{P} \vee \mathcal{P}$ | X $\mathcal{P}$ | F $\mathcal{P}$ | G $\mathcal{P}$ | [$\mathcal{P}$ U $\mathcal{P}$] | [$\mathcal{P}$ R $\mathcal{P}$]

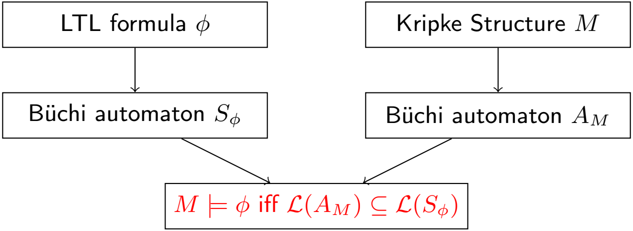
For a path π, we have:

$\pi \models$ true		
$\pi \not\models$ false		
$\pi \models p$	iff	$p \in L(\pi(0))$
$\pi \models \neg f$	iff	$\pi \not\models f$
$\pi \models f \wedge g$	iff	$\pi \models f$ and $\pi \models g$
$\pi \models f \vee g$	iff	$\pi \models f$ or $\pi \models g$
$\pi \models \mathsf{X}\ f$	iff	$\pi^1 \models f$
$\pi \models \mathsf{F}\ f$	iff	for some $i \geq 0, \pi^i \models f$
$\pi \models \mathsf{G}\ f$	iff	for all $i \geq 0, \pi^i \models f$
$\pi \models [f\ \mathsf{U}\ g]$	iff	$\exists i \geq 0.\ \pi^i \models g \wedge \forall j < i.\ \pi^j \models f$
$\pi \models [f\ \mathsf{R}\ g]$	iff	$\forall j \geq 0.\ ((\forall i < j.\ \pi^i \not\models f) \Rightarrow \pi^j \models g)$

Checking $M \models f$ requires checking that $\pi \models f$ holds for all initialised paths

LTL Model Checking

LTL has a nice automata-theoretic algorithm (see Chapter 9.2–9.4):



- Complexity of LTL model checking is **PSPACE**-complete.
- for a state space of size n and a formula of size m, the problem has complexity $n2^{\mathcal{O}(m)}$.
- Hence, checking for $M \models \phi$ is not always feasible.

Alternative: Bounded Model Checking

Outline

Bounded Model Checking

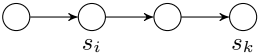
- Observation: LTL model checking requires checking all initialised paths.
- On the other hand: a counterexample to an LTL formula f corresponds to the question whether there exists a witness for $\neg f$
  - A counterexample for G f is a finite prefix of a path in which F $\neg f$ holds.
  - A counterexample for F f is a finite prefix of a path that is a lasso in which G $\neg f$ holds.

Idea behind BMC:

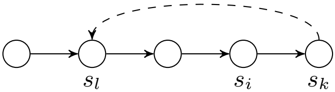
- BMC is performed only on the basis of finite, bounded prefixes of paths $|[M]|^k$ of the system M
- BMC searches for a witness to an existentially quantified LTL formula f, interpreted over bounded prefixes of paths: $|[f]|^k$.
- BMC can efficiently be solved using SAT-solvers:
  - If the formula $|[M]|^k \wedge |[f]|^k$ is satisfiable, a counterexample has been found
  - If the formula $|[M]|^k \wedge |[f]|^k$ is unsatisfiable, no counterexample of length k exists

Bounded Model Checking

Let $M = \langle S, R, L \rangle$ be a Kripke Structure.



(a) no loop



(b) (k, l)-loop

Consider a k-bounded path π. Such a bounded path can represent

- all its infinite extensions (case a)
- a (k, l)-loop (case b), i.e. if $\pi(k)\ R\ \pi(l)$ then π represents an infinite path $\rho = u\ v^\omega$, with $u = \pi(0)\ \ldots\ \pi(l-1)$ and $v = \pi(l)\ \ldots\ \pi(k)$ for some $l \leq k$.

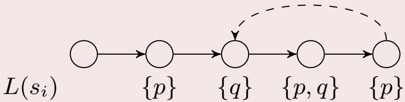
Definition (k-loops)

If there is an $l \leq k$, such that π is a (k, l)-loop, π is called a k-loop.

Bounded Model Checking

Example (k-loops)

Consider the following 4-bounded path π:

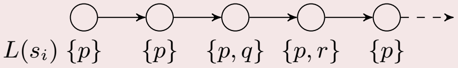


- π is actually a $(4, 2)$-loop.
- We can check whether $\pi \models \phi$ for all formulae ϕ
- For instance: $\phi = \mathsf{F}\ [p\ \mathsf{U}\ q]$ or $\phi = \mathsf{F}\ \mathsf{G}\ \neg(p \wedge q)$

Bounded Model Checking

Example (no loop)

Consider the following 4-bounded path π:



- π is not a 4-loop.
- Observe that we have $\rho \models \mathsf{F}\ q$ for all infinite extensions ρ of π
- We do not know $\rho \models \mathsf{G}\ p$ for any infinite extension ρ of π.

Bounded Model Checking

- From hereon, restrict to LTL formulae in Normal Form (NF)
- formulae in NF only have negation in front of atomic propositions
- NF is not a restriction: every LTL formula can be translated to an equivalent NF formula.

Formulae in NF are given a Bounded Semantics.

- Bounded Semantics approximates the unbounded (i.e. ordinary) semantics
- Bounded Semantics is based on k-bounded paths.

Bounded Model Checking

Definition

Let $\pi = s_0 \ s_1 \ \ldots$ be a bounded path, and let $k \geq 0$ be a bound. Then an LTL formula f is valid along the path π with bound k (denoted $\pi \models_k f$) iff:

- π is a k-loop and $\pi \models f$
- π is not a k-loop and $\pi \models_k^0 f$, where for non-temporal operators:

$\pi \models_k^i$ true		always holds
$\pi \models_k^i$ false		is always false
$\pi \models_k^i p$	iff	$p \in L(\pi(i))$
$\pi \models_k^i \neg p$	iff	$p \notin L(\pi(i))$
$\pi \models_k^i f \wedge g$	iff	$\pi \models_k^i f$ and $\pi \models_k^i g$
$\pi \models_k^i f \vee g$	iff	$\pi \models_k^i f$ or $\pi \models_k^i g$

Bounded Model Checking

Definition

Let $\pi = s_0\ s_1\ \ldots$ be a bounded path, and let $k \geq 0$ be a bound. Then an LTL formula f is valid along the path π with bound k (denoted $\pi \models_k f$) iff:

- π is a k-loop and $\pi \models f$
- π is not a k-loop and $\pi \models_k^0 f$, where for temporal operators:

$\pi \models_k^i \mathsf{G}\ f$		is always false
$\pi \models_k^i \mathsf{F}\ f$	iff	$\exists j. i \leq j \leq k \wedge \pi \models_k^j f$
$\pi \models_k^i \mathsf{X}\ f$	iff	$i < k$ and $\pi \models_k^{i+1} f$
$\pi \models_k^i [f\ \mathsf{U}\ g]$	iff	$\exists j. i \leq j \leq k \wedge \pi \models_k^j g$ and $\forall n. i \leq n < j \Rightarrow \pi \models_k^n f$
$\pi \models_k^i [f\ \mathsf{R}\ g]$	iff	$\exists j. i \leq j \leq k \wedge \pi \models_k^j f$ and $\forall n. i \leq n < j \Rightarrow \pi \models_k^n g$

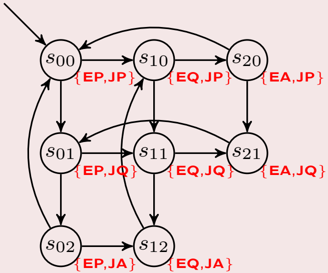
Bounded Model Checking

Some properties of $\models_k$:

- $\models_k$ under-approximates $\models$:
  - if f holds for a k-bounded path, it also holds a longer path: if $\pi \models_k f$ then $\pi \models_{k+1} f$.
  - for all paths π and all k: $\pi \models_k f$ then $\pi \models f$.
- For each ultimately periodic path π there is a k such that π is a k-loop and thus $\pi \models f$ iff $\pi \models_k f$ for some k.
- From this, it follows that the existential model checking question $M \models \mathsf{E}\ f$ can be solved by computing $M \models_k \mathsf{E}\ f$ for a sufficiently large k.

Bounded Model Checking

Example



Let $\pi = s_{00}\ s_{10}\ s_{11}\ s_{12}$ be a bounded path

- π is a $(3,1)$-loop
- $\pi \models_3 \mathsf{G}\ (EP \vee EQ)$
- $\pi \not\models_3 \mathsf{G}\ EP \vee \mathsf{G}\ EQ$

Consider the bounded path $\rho = s_{00}\ s_{10}\ s_{11}\ s_{21}$

- ρ is not a looping path
- $\rho \models_3 \mathsf{F}\ EA$
- $\rho \not\models_3 \mathsf{G}\ (\neg JA)$

Outline

Reduction of BMC to SAT

SAT-problem: given a propositional formula ϕ, find a valuation for the variables of ϕ that make ϕ true.

- Boolean satisfiability is NP-complete.
- a SAT-solver computes a valuation (if it exists) or it returns *unsatisfiable*.
- SAT-solvers accept formulae in Conjunctive Normal Form (CNF), i.e. a conjunction of clauses (disjunctions of literals and negated literals).
- turning a formula ϕ into CNF can be done either:
  - naively (yields formulae exponential in the size of ϕ, think of an example), or
  - cleverly, by introducing $\mathcal{O}(|\phi|)$ auxiliary variables, where $|\phi|$ is the number of sub expressions in ϕ.
- Typical tools: minisat and zchaff

Reduction of BMC to SAT

Given a Kripke Structure $M = \langle S, R, L \rangle$, a formula f and a bound k.

$[M, f]_k$ encodes the problem $M \models_k f$ as a propositional formula.

The encoding $[\_]_k$ proceeds in three steps:

- Compute $[M]_k$, encoding all initialised paths of length k.
- Compute L_k, encoding the loop condition as a proposition.
- Constrain the encoded paths to paths that satisfy f

Note: the size of $[M, f]_k$ is $\mathcal{O}(|f| \times k \times |M|)$

Reduction of BMC to SAT

Given a Kripke Structure $M = \langle S, R, L\rangle$ and a bound k.

- Represent all states in S uniquely by a state vector s of n Boolean state variables $\langle s[0], s[1], \ldots, s[n-1]\rangle$
- Take $k+1$ copies of the system state vector, denoted by $s_0, s_1, \ldots, s_k$
- Let $S_0(s)$ be the initial state(s) of the system, and $R(s, s')$ be the transition relation, both expressed as propositional formulae.

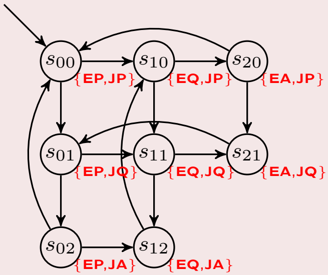
Definition

The k-unfolding $[\![M]\!]_k$ of a Kripke Structure is given by the following propositional formula

$$[\![M]\!]_k := S_0(s_0) \ \wedge \ \bigwedge_{i=1}^{k} R(s_{i-1}, s_i)$$

Reduction of BMC to SAT

Example



Symbolic representation of M:

- $\mathcal{S}_0(s) := \; s[E] = p \wedge s[J] = p$
- $\mathcal{R}(s, s') := \; R_1 \vee R_2 \vee R_3 \vee R_4 \vee R_5 \vee R_6$, where:
  - $R_1 := \; s[E] = p \wedge s'[E] = q \wedge s[J] = s'[J]$
  - $R_2 := \; s[E] = q \wedge s'[E] = a \wedge s'[J] = s[J] \wedge s[J] \neq a$
  - $R_3 := \; s[E] = a \wedge s'[E] = p \wedge s'[J] = s[J]$
  - $R_4 := \; s[J] = p \wedge s'[J] = q \wedge s'[E] = s[E]$
  - $R_5 := \; s[J] = q \wedge s'[J] = a \wedge s'[E] = s[E] \wedge s[E] \neq a$
  - $R_6 := \; s[J] = a \wedge s'[J] = p \wedge s'[E] = s[E]$

Use vectors s_0, s_1 and s_2 to represent the states of the system; use propositional variables to represent $s_0[E] = p$, etc.
The 2-unfolding of M is given by the following propositional formula :

$$(s_0[E] = p \wedge s_0[J] = p) \wedge \mathcal{R}(s_0, s_1) \wedge \mathcal{R}(s_1, s_2)$$

Reduction of BMC to SAT

Recall that the Bounded Semantics for LTL depends on the structure of the path:

- for loops, the Bounded Semantics coincides with the ordinary semantics
- for loop-free paths, the Bounded Semantics differs.

The propositional formula ${}_lL_k$ is true iff there is a transition from state s_k to state s_l:

$${}_lL_k := R(s_k, s_l)$$

Definition

The loop-condition L_k is given by the following proposition:

$$L_k := \bigvee_{l=0}^{k} {}_lL_k$$

Reduction of BMC to SAT

Given a Kripke Structure $M = \langle S, R, L \rangle$, a bound k and an LTL formula f

The encoding of f in case f is interpreted over a path that is a (k, l)-loop:

$$
\begin{array}{ll}
{}_l[p]_k^i & := p(s_i) \\
{}_l[\neg p]_k^i & := \neg p(s_i) \\
{}_l[f \vee g]_k^i & :=_l [f]_k^i \vee_l [g]_k^i \\
{}_l[f \wedge g]_k^i & :=_l [f]_k^i \wedge_l [g]_k^i \\
{}_l[\mathsf{X}\ f]_k^i & :=_l [f]_k^{\mathsf{succ}(i)} \\
{}_l[\mathsf{G}\ f]_k^i & :=_l [f]_k^i \wedge_l [\mathsf{G}\ f]_k^{\mathsf{succ}(i)} \\
{}_l[\mathsf{F}\ f]_k^i & :=_l [f]_k^i \vee_l [\mathsf{F}\ f]_k^{\mathsf{succ}(i)} \\
{}_l[[f\ \mathsf{U}\ g]]_k^i & :=_l [g]_k^i \vee ({}_l[f]_k^i \wedge_l [[f\ \mathsf{U}\ g]]_k^{\mathsf{succ}(i)} \\
{}_l[[f\ \mathsf{R}\ g]]_k^i & :=_l [g]_k^i \wedge ({}_l[f]_k^i \vee_l [[f\ \mathsf{R}\ g]]_k^{\mathsf{succ}(i)}
\end{array}
$$

$\mathsf{succ}(i)$ is defined as:

$$
\begin{cases} i+1 & \text{if } i < k \\ l & \text{if } i = k \end{cases}
$$

Note: i, $(i \leq k)$ indicates the depth of "unfolding"

Reduction of BMC to SAT

Given a Kripke Structure $M = \langle S, R, L \rangle$, a bound k and an LTL formula f

The encoding of f in case f is interpreted over a path that is *not* a loop:

$$
\begin{array}{ll}
[p]_k^i & := p(s_i) \\
[\neg p]_k^i & := \neg p(s_i) \\
[f \vee g]_k^i & := [f]_k^i \vee [g]_k^i \\
[f \wedge g]_k^i & := [f]_k^i \wedge [g]_k^i \\
[\mathsf{X}\ f]_k^i & := [f]_k^{i+1} \\
[\mathsf{G}\ f]_k^i & := [f]_k^i \wedge [\mathsf{G}\ f]_k^{i+1} \\
[\mathsf{F}\ f]_k^i & := [f]_k^i \vee [\mathsf{F}\ f]_k^{i+1} \\
[[f\ \mathsf{U}\ g]]_k^i & := [g]_k^i \vee ([f]_k^i \wedge [[f\ \mathsf{U}\ g]]_k^{i+1} \\
[[f\ \mathsf{R}\ g]]_k^i & := [g]_k^i \wedge ([f]_k^i \vee [[f\ \mathsf{R}\ g]]_k^{i+1}
\end{array}
$$

Formulae beyond depth k never hold:

$[f]_k^j :=$ false for $j > k$

Note: i, $(i \leq k)$ indicates the depth of “unfolding”

Reduction of BMC to SAT

Given a Kripke Structure $M = \langle S, R, L \rangle$, an LTL formula f and a bound $k \geq 0$.

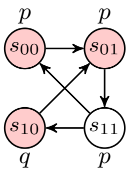
The propositional formula corresponding to the Existential Bounded Model Checking problem is given by $[M, f]_k$:

$$[M, f]_k := [M]_k \wedge \left(\left(\neg L_k \wedge [f]_k^0\right) \vee \bigvee_{l=0}^{k} \left({}_l L_k \wedge {}_l [f]_k^0\right) \right)$$

- The left side of the disjunction represents the case when there is no back-loop in a path of length k (L_k does *not* hold)
- The right side of the disjunction represents the case when there is a back-loop at some point between 0 and k (${}_l L_k$ holds for some l)
- $[M, f]_k$ is satisfiable iff $M \models_k$ E f.

Outline

Example



- Kripke Structure M, represented by:
- Initial state proposition: $\mathcal{S}_0(s) = \neg s[0] \wedge \neg s[1]$.
- Transition relation: $\mathcal{R}(s, s') =$

$$\begin{array}{ll} & (s[0] \leftrightarrow s[1] \wedge (s'[0] \leftrightarrow \neg s[0]) \wedge (s'[1] \leftrightarrow s[1])) \\ \vee & (\neg s[0] \wedge s[1] \wedge s'[0] \wedge s'[1]) \\ \vee & (s[0] \wedge (s'[0] \leftrightarrow \neg s[0]) \wedge (s'[1] \leftrightarrow \neg s[1])) \end{array}$$

- To check: G p

- paths starting in s_{00} have (a.o.) a $(2, 0)$-loop and a $(3, 1)$-loop.
- $[\![M, \mathsf{F}\ \neg p]\!]_2$ is not satisfiable.
- $[\![M, \mathsf{F}\ \neg p]\!]_3$ is satisfiable:

$$\left\{ \begin{array}{ll} (s_0[0], s_0[1]) & = (\text{false}, \text{false}) \\ (s_1[0], s_1[1]) & = (\text{false}, \text{true}) \\ (s_2[0], s_2[1]) & = (\text{true}, \text{true}) \\ (s_3[0], s_3[1]) & = (\text{true}, \text{false}) \end{array} \right.$$