SAT-BASED BOUNDED MODEL CHECKING

Formal Models SS18

Martina Seidl
Institute for Formal Models and Verification
Model Checking

- Requirements
  - Formal Specification
    - (Temporal Formula)
- Model Checker
- Implementation
  - Model
    - (Kripke Structure)

VERIFIED
Model Checking

Requirements

Formal Specification
(Temporal Formula)

Model Checker

ERROR +
Error Trace

Implementation

Model
(Kripke Structure)

detail
Types of Model Checking

**General question**: Given a system $K$ and a property $p$, does $p$ hold for $K$ (i.e., for all initial states of $K$)?

- **Explicit state model checking**
  - enumeration of the state space
  - state explosion problem

- **Symbolic model checking**
  - representation of model checking problem as logical formula (e.g., in propositional logic (SAT) or QBF)
Some Properties

- **Reachability**: property $p$ holds in one reachable state
- **Invariant**: property $p$ holds in all reachable states
- **Safety**: some bad property $p$ never holds
  “something bad will never happen”
- **Liveness**: something good will eventually happen
- **Fairness**: under certain conditions, some property holds repeatedly
Example: Mutual Exclusion

Given two processes P and Q which share a resource R.

- If R is accessed by P, then property \( p \) is true.
- If R is accessed by Q, then property \( q \) is true.

The behavior of P and Q is modeled by this Kripke structure:

![Kripke structure diagram]

**Question:** Does \( F(p \land q) \) hold?
Limboole

- SAT-solver for formulas in non-CNF
- available at http://fmv.jku.at/limboole/
- input format in BNF:

\[
\langle expr \rangle ::= \langle iff \rangle \\
\langle iff \rangle ::= \langle implies \rangle \mid \langle implies \rangle "<->" \langle implies \rangle \\
\langle implies \rangle ::= \langle or \rangle \mid \langle or \rangle "->" \langle or \rangle \mid \langle or \rangle "<-" \langle or \rangle \\
\langle or \rangle ::= \langle and \rangle \mid \langle and \rangle "|" \langle and \rangle \\
\langle and \rangle ::= \langle not \rangle \mid \langle not \rangle "&" \langle not \rangle \\
\langle not \rangle ::= \langle basic \rangle \mid "!" \langle not \rangle \\
\langle basic \rangle ::= \langle var \rangle \mid "(" \langle expr \rangle ")"
\]

where 'var' is a string over letters, digits, and
- _ . [ ] $ @
Symbolic Encoding of Kripke Structures

Given Kripke structure $K = (S, I, T, L)$ over $A = \{a_1, \ldots, a_n\}$.

1. Introduce sets $A' = \{a'_1, \ldots, a'_n\}$ and $A'' = \{a''_1, \ldots, a''_n\}$ for the definition of one transition step $T$ over $A'$ and $A''$.

2. Associate each state $s \in S$ with two conjunctions of literals $\text{current}(s)$ and $\text{next}(s)$:

   $\square \text{current}(s) := (l_1 \land \ldots \land l_n)$ 
   such that $l_i = a'_i$ if $a_i \in L(s)$ else $l_i = \overline{a'}_i$;

   $\square \text{next}(s) := (k_1 \land \ldots \land k_n)$ 
   such that $k_i = a''_i$ if $a_i \in L(s)$ else $k_i = \overline{a''}_i$.

3. Define prop. formula $T$ over $A', A''$ such that $\forall s_i, s_j \in S$

   $(T \land \text{current}(s_i) \land \text{next}(s_j))$ is satisfiable iff $(s_i, s_j) \in T$.

\[1\] note: the mapping “state to conjunction” has to be bijective
Naive Encoding of Kripke Structures in SAT

Let $K = (S, I, T, L)$ be a Kripke structure over $\mathcal{A}$.

$\mathcal{T} := \top$

while $S \neq \emptyset$ do

select $s \in S$

$S := S \setminus \{s\}$

$N := \bot$

for all $(s, t) \in T$ do

$N := N \lor \text{next}(t)$

end for

$\mathcal{T} := \mathcal{T} \land (\text{current}(s) \rightarrow N)$

end while

return $\mathcal{T}$
Naive Encoding of Kripke Structures in SAT

\[ \mathcal{T} := \top \land (\lnot p \land \lnot q) \rightarrow (\bot \lor (\lnot p' \land q') \lor (p' \land \lnot q')) \land \\
(\lnot p \land q) \rightarrow (\bot \lor (\lnot p' \land \lnot q') \lor (p' \land q')) \land \\
(p \land \lnot q) \rightarrow (\bot \lor (\lnot p' \land q') \lor (p' \land \lnot q')) \land \\
(\lnot p \land q) \rightarrow (\bot \lor (\lnot p' \land q')) \land \\
(p \land q) \rightarrow (\bot \lor (\lnot p' \land \lnot q')) \]
Naive Encoding of Kripke Structures in SAT

Encoding in Limboole syntax:

\[
\begin{align*}
((\neg p \land \neg q) & \rightarrow (\neg p\text{-next} \land q\text{-next}) \lor (p\text{-next} \land \neg q\text{-next})) \land \\
((p \land \neg q) & \rightarrow (\neg p\text{-next} \land \neg q\text{-next}) \lor (p\text{-next} \land q\text{-next})) \land \\
((\neg p \land q) & \rightarrow (\neg p\text{-next} \land \neg q\text{-next})) \land \\
((p \land q) & \rightarrow (\neg p\text{-next} \land \neg q\text{-next}))
\end{align*}
\]

> limboole limboole/mutual.boole -s
% SATISFIABLE formula (satisfying assignment follows)

\[
\begin{align*}
p &= 0 \\
q &= 0 \\
p\text{-next} &= 0 \\
q\text{-next} &= 1
\end{align*}
\]
Symbolic Encoding of Kripke Structures

Alternative encoding of transition function:

**Successor states** $p'$, $q'$:

$$(p' \leftrightarrow (\bar{p} \land \bar{q}) \land q' \leftrightarrow 0) \lor (p' \leftrightarrow (p \land \bar{q}) \land q' \leftrightarrow \bar{q})$$

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>$p'$</th>
<th>$q'$</th>
<th>or</th>
<th>$p'$</th>
<th>$q'$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Diagram:
Example: One Step

Encoding in Limboole syntax:

$$(((p\text{-next} \leftrightarrow (\neg p \& q)) \& (\neg q\text{-next})) \mid
(((p\text{-next} \leftrightarrow (p \& \neg q)) \& (q\text{-next} \leftrightarrow \neg q)))$$

> limboole -s mutual2.boole
% SATISFIABLE formula (satisfying assignment follows)

p = 0
q = 0
p\text{-next} = 1
q\text{-next} = 0
Multiple Transition Steps

- $\mathcal{T}$ over $\mathcal{A}'$ and $\mathcal{A}''$ defines one transition step
  - we also write $\mathcal{T}(s_0, s_1)$ indicating that we can go from state $s_0$ to a state $s_1$
- $\mathcal{T}$ over $\mathcal{A}''$ and $\mathcal{A}'''$ defines one transition step
  - we also write $\mathcal{T}(s_1, s_2)$ indicating that we can go from state $s_1$ to a state $s_2$
- $\mathcal{T}(s_0, s_1) \land \mathcal{T}(s_1, s_2)$ defines two transition steps from a state $s_0$ to a state $s_1$
- Example (previous slides):
  
  $$( ((p' \leftrightarrow (\neg p \land \neg q)) \land (q' \leftrightarrow 0)) \lor ((p' \leftrightarrow (p \land \neg q)) \land (q' \leftrightarrow \neg q)) \land ((p'' \leftrightarrow (\neg p' \land \neg q')) \land (q'' \leftrightarrow 0)) \lor ((p'' \leftrightarrow (p' \land \neg q')) \land (q'' \leftrightarrow \neg q')))$$
Example: Two Steps

Encoding in Limboole syntax:

\[
((((p\text{-next} \leftrightarrow (!p \& !q)) \& (!q\text{-next})) \mid
((p\text{-next} \leftrightarrow (p \& !q)) \& (q\text{-next} \leftrightarrow !q))) \&
(((p\text{-next2} \leftrightarrow (!p\text{-next} \& !q\text{-next2})) \& (!q\text{-next2})) \mid
((p\text{-next2} \leftrightarrow (p\text{-next} \& !q\text{-next2})) \& (q\text{-next2} \leftrightarrow !q\text{-next2})))
\]

> limboole -s mutual2-twoSteps.boole

% SATISFIABLE formula (satisfying assignment follows)

p = 0
q = 1
p\text{-next} = 0
q\text{-next} = 0
p\text{-next2} = 1
q\text{-next2} = 0
Example: Three Steps

Encoding in Limboole syntax:

\[
(((p\text{-next} <-> (!p & !q)) & (!q\text{-next})) | \\
((p\text{-next} <-> (p & !q)) & (q\text{-next} <-> !q))) & \\
(((p\text{-next2} <-> (!p\text{-next} & !q\text{-next})) & (!q\text{-next2})) | \\
((p\text{-next2} <-> (p\text{-next} & !q\text{-next})) & (q\text{-next2} <-> !q\text{-next}))) & \\
(((p\text{-next3} <-> (!p\text{-next2} & !q\text{-next2})) & (!q\text{-next3})) | \\
((p\text{-next3} <-> (p\text{-next2} & !q\text{-next2})) & (q\text{-next3} <-> !q\text{-next2})))
\]

limboole -s mutual2-threeSteps.boole
% SATISFIABLE formula (satisfying assignment follows)
p = 0
q = 1
p\text{-next} = 0
q\text{-next} = 0
p\text{-next2} = 1
q\text{-next2} = 0
p\text{-next3} = 1
q\text{-next3} = 1
Bounded Model Checking (Safety)

- Given a Kripke structure $K$. Is there a path of length $k$ to a bad state $s$, i.e., a certain property $p$ is violated in $s$?

- In other words: there is a path where $Gp$ does not hold in $K$.

- Observation: if $Gp$ does not hold in $K$, there is a finite counter-example.

- Idea: consider paths of fixed length $k$.
  - encode problem to propositional formula $\phi$
  - pass problem to SAT solver
  - $\phi$ is true $\iff$ model of $\phi$ is counter-example
  - if $\phi$ is false, then increase $k$.
Bounded Model Checking (Safety)

A bounded model checking (BMC) problem for Kripke structure $K$ and safety property $G_p$ is encoded by

$$I(s_0) \land T(s_0, s_1) \land T(s_1, s_2) \land \ldots \land T(s_{k-1}, s_k) \land B(s_k)$$

where

- $I(s_0)$ is true $\iff$ $s_0$ is an initial state
- $T$ is the transition function of $K$
- $B(s_k)$ is true $\iff$ $s_k$ is a bad state, i.e., $\neg p$ holds in $s_k$
We want to know if $G(\bar{p} \vee \bar{q})$ holds for Kripke structure $K$:
BMC Example

We want to know if $G(\overline{p} \lor \overline{q})$ holds for Kripke structure $K$:

$k = 1$ (one step):

$$(\overline{p} \land \overline{q}) \land ((p' \leftrightarrow (\overline{p} \land \overline{q})) \land (q' \leftrightarrow 0)) \lor ((p' \leftrightarrow (p \land \overline{q})) \land (q' \leftrightarrow \overline{q})) \land (p' \land q')$$
We want to know if $G(\bar{p} \lor \bar{q})$ holds for Kripke structure $K$:

$k = 2$ (two steps):

\[
(\bar{p} \land \bar{q}) \land \\
(((p' \leftrightarrow (\bar{p} \land \bar{q})) \land (q' \leftrightarrow 0)) \lor ((p' \leftrightarrow (p \land \bar{q})) \land (q' \leftrightarrow \bar{q}))) \land \\
(((p'' \leftrightarrow (\bar{p}' \land \bar{q}')) \land (q'' \leftrightarrow 0)) \lor \\
((p'' \leftrightarrow (p' \land \bar{q}')) \land (q'' \leftrightarrow \bar{q}'))) \land \\
(p'' \land q'')
\]
Bounded Model Checking (Fairness)

- Given a Kripke structure $K$. Is there a path such that a property $\neg p$ holds forever?

- In other words: there is a path such that $Fp$ does not hold in $K$

- Observation 1: if $Fp$ does not hold in $K$, there is an infinite counter-example.

- Observation 2: if the counter-example is infinite, then it has to be because of a cycle.
Bounded Model Checking (Fairness)

A bounded model checking (BMC) problem for Kripke structure $K$ and fairness property $F_p$ is encoded by

$$ I(s_0) \land \bigwedge_{l=0}^{k-1} \mathcal{T}(s_l, s_{l+1}) \land \bigvee_{i=0}^{k} \mathcal{T}(s_k, s_i) \land \bigwedge_{j=0}^{k} \neg \mathcal{F}(s_j) $$

where

- $I(s_0)$ is true $\iff s_0$ is an initial state
- $\mathcal{T}$ is the transition function of $K$
- $\mathcal{F}(s_k)$ is true $\iff \neg p$ holds in $s_k$
BMC Fairness

We want to know if $F_q$ holds for Kripke structure $K$:

Initial State:
$$(\overline{p} \land \overline{q})$$

One Step:
$$(((p' \leftrightarrow (\overline{p} \land \overline{q})) \land (q' \leftrightarrow 0)) \lor ((p' \leftrightarrow (p \land \overline{q})) \land (q' \leftrightarrow \overline{q})))$$

Cycle Check:
$$(((p \leftrightarrow (\overline{p}' \land \overline{q}'))) \land (q \leftrightarrow 0)) \lor ((p \leftrightarrow (p' \land \overline{q}'))) \land (q \leftrightarrow \overline{q}'))$$

$$(((p' \leftrightarrow (\overline{p}' \land \overline{q}'))) \land (q' \leftrightarrow 0)) \lor ((p' \leftrightarrow (p' \land \overline{q}'))) \land (q' \leftrightarrow \overline{q}')))$$

Property Check:
$$\overline{q} \land \overline{q}'$$
BMC Summary

- BMC is incomplete ...
  - if all checked formulas are unsat, no insight
  - how to choose $k$? when to stop increasing $k$?

- ... very efficient (e.g., debugging)

- many tuning techniques
  - exploit similarities between two transition steps
    (structure sharing
    - simplification of formula by rewritings )
How to choose $k$ for Safety?

Given Kripke structure $K$, the **diameter** is the smallest number $d$ such that for every path $s_0, \ldots, s_{d+1}$ there exists a path $t_0, \ldots, t_l$ such that $l \leq d$ and $t_0 = s_0$ and $t_l = s_{d+1}$.

- If a state $s$ is reachable from state $t$, then there is a path of length $d$ or less where $d$ is the diameter.
- The diameter is the longest shortest path.
- Computing the diameter is difficult (solve a QBF).