Introduction to Model Checking and its Applications

Cindy Eisner
Formal Methods Group
Overview

- What is model checking?
- Symbolic model checking
- Applications of symbolic model checking
- Conclusion
What is model checking?
Model checking?

- Given a model $M$ and a temporal logic formula $f$, decide if $M \models p$
- Formally:
  - A model is a quintuple $(S, S_0, R, P, L)$
    - $S$ is a finite set of states
    - $S_0$ is a subset of $S$
    - $R$ is the transition relation, a subset of $S \times S$
    - $P$ is a non-empty set of atomic propositions
    - $L$ is the valuation, a function $S \rightarrow 2^P$
  - We use the temporal logic PSL, a soon-to-be IEEE standard
- In the interest of time, we will continue informally
An example

Given

The following model:

The PSL formula:

- $G (\text{request} \rightarrow X[1..3] \text{ process})$
- (if a request is received, it must be processed within 3 cycles)

Determine whether the model satisfies the formula
CTL Model Checking

- Clarke, Emerson (1981)
- A branching time temporal logic
- PSL model checking can be reduced to CTL model checking (sometimes the model must be augmented with an automaton)
- Idea: unwind state graph to obtain infinite computation tree, reason about (infinite) paths in the computation tree
CTL

- Path Quantifiers
  - A – for all paths
  - E – there exists a path

- Tense Operators
  - X p - p holds at the next cycle
  - F p - p holds sometime in the future
  - G p - p holds globally in the future
  - p U q - p holds until q holds

- Temporal operators
  - AX, AF, AG, AU, EX, EF, EG, EU
CTL (continued)

- **EG p**

- **AF p**

- **AG AF p**
CTL Model Checking – Image and Pre-Image

- Forward traversal: image
- Backward traversal: pre-image
CTL Model Checking - EX

```c
ex(set_of_states p)
{
    return(PreImg(p));
}
```

- $p = \{0,1,2\}$
- $\text{PreImg}(p) = \{0,1\}$

$$
\text{EX } p = \{0,1\}
$$
CTL Model Checking - EG

eg(set_of_states p)
{
    set_of_states oldY = {};
    set_of_states Y = p;
    while (Y != oldY) {
        oldY = Y;
        Y = intersection(Y, ex(Y));
    }
    return(Y);
}

1. oldY={}; Y={0,1,2};
2. oldY={0,1,2}; ex(Y)={0,1}; Y={0,1};
3. oldY={0,1}; ex(Y)={0}; Y = {0};
4. oldY={0}; ex(Y)={0}; Y={0};

EG p = {0}
CTL Model Checking in Practice

- Bare model checking is a nice theory, but
  - In practice, models are huge

- If we limit our integers to 2 bits, then an explicit model for getmax() has 128 states
  - 2 bits for max
  - 2 bits for a
  - 3 bits for pc

- Enter symbolic model checking

```c
getmax () {
    int max, a;
    0 a = max = 0;
    1 do {
        2 if (a > max)
        3 max = a;
        4 a = input();
        5 } while (a);
    6 return(max);
    7 }
```
Symbolic Model Checking
Symbolic model checking (McMillan 1992)

- Use BDDs to:
  - Implicitly traverse the state space without building the model
  - Deal simultaneously with sets of states
BDDs (Bryant 1986)

- Binary decision diagram
- Directed acyclic graph representing a (boolean) function

\[(a \& b \& c) \lor (c \& d)\]
BDDs (continued)

- BDDs are a canonical representation if:
  - An order is imposed on the variables
  - Isomorphic sub-trees are combined into a single tree
  - Nodes whose two children are isomorphic are removed

![BDD Diagram](image-url)
Logic Operations on BDDs

By Shannon’s expansion

\[ f \cdot g = \overline{a} \cdot (f|_a \cdot g|_a) + a \cdot (f|_a \cdot g|_a) \]
BDDs in Symbolic Model Checking

- Functions
- Sets of states
- Transition relation
Symbolic model checking - example

- Design is completely deterministic, but
- Inputs behave non-deterministically
- Therefore, the model will be non-deterministic
Symbolic model checking – example (continued)

Let's see how to traverse this model without actually building it
BDDs represent functions

\[ xy' = \begin{cases} 00 & \text{if } (a \& b) \\ \{00,01,10,11\} & \text{else} \end{cases} \]
BDDs Represent Sets of States

\[
xy' = \text{if } (a \& b) \text{ then } 00 \text{ else } \{00,01,10,11\}
\]

initial states \((abxy = \{0000,0001,0010,0011\})\)
Transition relation

\[ xy' = \text{if } (a \& b) \text{ then } 00 \text{ else } \{00,01,10,11\} \]

Transition Relation (fragment)

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>x</th>
<th>y</th>
<th>a'</th>
<th>b'</th>
<th>xy'</th>
<th>TR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>00</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>01</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>11</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>01</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>11</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>00</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>01</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>11</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>00</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>01</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>00</td>
<td>0</td>
</tr>
</tbody>
</table>

Truth Table (fragment)

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>x</th>
<th>y</th>
<th>a'</th>
<th>b'</th>
<th>xy'</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>{00,01,10,11}</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>{00,01,10,11}</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>00</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>00</td>
</tr>
</tbody>
</table>
Building transition relation per bit

\[ TR = tr(a) \& tr(b) \& tr(xy) \]

\[ xy' = \text{if } (a \& b) \text{ then } 00 \text{ else } \{00,01,10,11\} \]
BDDs represent transition relation
BDDs represent transition relation (continued)
Computing the Image with the TR
Computing the Image (continued)

```
      y
     /\  
    a'  a'
   / \  /  
  b'  0 b'
  1  0 0 1 0
```

```
      a'
     / \  
    "or" a'
   / \  /  
  b'  0 b'
  1  0 1 0
```

```
      b
     /   
   b'    
  0 1 0
```

1 0 1 0

->
Sanity check

- We started these initial states:

- We took one symbolic step in this design:

- And we ended up at this set of states:

- And we did it all without building this:
Other methods of model checking

- McMillan’s BDD-based method is just one way to do symbolic model checking
- Other techniques include:
  - SAT-based methods
  - BDD-based unfolding
  - Reachability-based on-the-fly methods
  - Localization
  - And many many more
- And then there is explicit-state model checking
Applications of Symbolic Model Checking
Applications: **Hardware Design and Verification (RuleBase)**

- **Mainstream hardware design and verification tool**
  - High-level development of abstract architectural models
  - Block-level verification of RTL
- **Active area of research and development**
  - Reductions, abstractions, new symbolic algorithms
- **Recent tapeouts supported by RuleBase**
  - IBM’s high-end microprocessors and ASICs
  - STMicroelectronics – ST100® DSP Cores
  - Marvell – Discovery™ System Controllers
  - Analog Devices – TigerSHARC® Processor
Applications: **Software verification (Wolf)**

- Prototype software verification tool
- Active area of research and development
  - Adaptation of standard techniques
  - Dedicated software techniques
- Recent applications of Wolf:
  - Discovery of data race in Linux driver
  - Discovery of out-of-bound access to array in firmware from the server group due to bad return value from function

```c
getmax () {
    int max, a;
    a = max = 0;
    do {
        if (a > max)
            max = a;
        a = input();
    } while (a);
    return(max);
}
```
Applications: **Analysis of Hybrid Control Systems**

- Beta-level tool for design of nonlinear control systems
  - Discretize Matlab models to create RuleBase input
  - Temporal logic formula states that values of control parameters do not exist
  - Counter-example provides desired values of control parameters
- Active area of research and development
- Recent case study:
  - Analysis of system for tracking and intercepting a maneuvering target
  - Discovery of robust control parameters for the guidance law and extended Kalman filter
Non-traditional applications

- Railway interlockings
  - Does the interlocking prevent two trains from being on the same piece of track at the same time?

- Firefly game
  - How many initial states converge, and are there board sizes for which no converging initial states exist?

- Various amusing NP-complete problems
  - There is no free lunch: finding a Hamiltonian circuit is difficult for even small graphs
  - On a lazy Thursday afternoon, it is sometimes interesting to see how far you can push the tools
Conclusion
Conclusion

- Model checking is a fascinating area of research, with many practical applications
- Find out more by visiting our home page: