Algorithmic Verification

Comp4151
Lecture 9-B
Ansgar Fehnker

Overview

Model checking Approaches

- Explicit State Model Checking
  - Combat the state explosion problem by reduction techniques such as partial order or symmetry reduction
  - Common tool: Spin

- Symbolic Model Checking
  - Represents set of states and transition relation as BDD
  - Use fast and efficient BDD algorithms
  - Common tool: NuSMV

Overview (cont)

- Bounded Model Checking
  - Relies on fast and efficient SAT-solvers
  - Transforms the bounded model checking problem to a satisfiability problem
  - BMC allows for tailored optimizations of SAT procedure

- Counterexample Guided Abstraction Refinement
  - Model check a small abstraction rather than full model
  - Refine the abstraction, if necessary, automatically

Abstraction

Existential Abstraction

Definition

Given a transition system $M = (S, S_0, R, L)$, and a surjective (many-to-one) mapping $h: S \rightarrow \hat{S}$.

The (minimal) existential abstraction is a transition system $M'$ with

- state space $\hat{S}$
- initial states $\hat{S}_0 = \{ h(s) \mid s \in S_0 \}$
- transition relation $\hat{R} = \{(h(s_0), h(s_1)) \mid (s_0, s_1) \in R\}$
- and labeling $\hat{L}(\hat{s}) = \bigcup_{h(s) = \hat{s}} L(s)$
Abstraction

Existential Abstraction

Definition
Given a transition system \( M = (S, S_0, R, L) \), and a surjective (many-to-one) mapping \( h: S \to \hat{S} \), maps many states to a few states

The (minimal) existential abstraction is a transition system \( M' \) with
- state space \( \hat{S} \)
- initial states \( \hat{S}_0 = \{ h(s) \mid s \in S_0 \} \)
- transition relation \( \hat{R} = \{ (h(s_0), h(s_1)) \mid (s_0, s_1) \in R \} \)
- and labeling \( \hat{L}(s) = \bigcup_{s \in \text{corresponding states}} L(s) \)

union of all labels in corresponding states

Types of abstraction

Predicate abstraction
Given a set of predicates \( p_0, \ldots, p_n \), the abstract state is a boolean vector \( (b_0, \ldots, b_n) \), such that state \( s \) is mapped to an abstract state iff
\[ b_i \Leftrightarrow s| = p_i \]

Example
variables
- bool x, y, z;

transitions
- pre: \( x = 0 \);
- post: \( x' = 1, y' = \neg z \)
- pre: \( y = z \);
- post: \( x' = y, z' = \neg x \)
- pre: \( x = 1, y = \neg z \);
- post: \( x' = \neg y, z' = \neg x \)

assert \((x = 0 \lor y = 0)\)
Given predicates $p_1$ and $p_2$, the abstract state is a tuple $(b_0, b_1)$.

**Abstraction**

Example

<table>
<thead>
<tr>
<th>$p_1: \neg z \land z = 1$</th>
<th>$p_2: y = 0 \land z = y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0, 0$</td>
<td>$0, 1$</td>
</tr>
<tr>
<td>$1, 1$</td>
<td>$1, 0$</td>
</tr>
</tbody>
</table>

**Abstraction**

Other types of abstraction

Abstraction by hiding variables

**Abstraction**

Consistent labelling

Definition

Labeling $\hat{L}$ is consistent with $h$ if $L(s) = \hat{L}(h(s))$ for all $s$. 

**Abstraction**

Example by Daniel Kroening

C program

```c
int main() {
    int i;
    i = 0;
    while (even(i))
        i++;
}
```

Predicates

```c
p_1 \land \neg p_2
p_2 \land \neg p_1
```

Boolean program

```c
void main() {
    bool p1, p2;
    p1 = TRUE;
    p2 = TRUE;
    while (p2)
        { p1 = !p1; FALSE: nondet(); p2 = !p2; }
}
```
### Abstraction

**Example**

This abstraction is not consistent with assertion \( (x=0 \lor y=0) \)

### Abstraction

**Example**

This abstraction is consistent with assertion \( (x=0 \land y=0) \)

### Abstraction

**Preservation Theorem**

Theorem

Let \( M' \) be a existential abstraction of \( M \), for abstraction function \( h \). Assume that the labeling of \( M' \) is consistent with \( h \). Given an ACTL property \( \phi \) we have

\[
M' = f \implies M = \emptyset
\]

If we can show by model checking that \( M' \) is correct, we don't need to check \( M \).

We hope that \( M' \) is much smaller than \( M \), thus easier to model check.

### Abstraction

**Example**

Show \( AG \neg (x=1 \land y=1) \)
Abstraction Refinement

Definition
Given a transition system $M$ and an abstraction function $h$ from $S$ to $\hat{S}$. An abstraction function $h'$ from $S$ to $\hat{S}$ is a refinement if
\[ h'(s_0) = h'(s_1) \implies h(s_0) = h(s_1) \]

What does this mean?

Abstraction Refinement

Series of Abstractions
Given and transition system $M$, abstraction function $h$, and a refinement $h'$ of $h$
- $M'$ is an abstraction of $M$ w.r.t $h$
- $M''$ is also an abstraction of $M$, but w.r.t $h'$
- And also: $M'$ is an abstraction of $M''$

It is possible to show that there exists a surjective mapping from $\hat{S}$ to $S'$

Example
Refinement by introduction of a new predicate

Example
Refinement by introduction of a new predicate
Abstraction Refinement

Example

Update the abstraction function

Abstraction Refinement

Example

Update the transition relation

Abstraction Refinement

Series of Abstraction

1. Construct automatically a series of abstractions $M', M'', M'''$, ..., such that for some $M^n |= \phi$
2. If $M^n |= \phi$ use the abstract counterexample to obtain information about how to refine $M'$.
3. Check if the abstract counterexample of $M'$ corresponds to a real one in $M$. Then $M|\not= \phi$

Spurious Counterexample

Definition

An abstract counterexample $(s_0, ..., s_n)$ of $M'$ is spurious, if there exists no path $(s_0, ..., s_n)$ in concrete model $M$, such that $h(s_i) = s'_i$ for all $i$. 

6
**Spurious Counterexample**

- **Concrete**
  - Down arrows indicate transitions.
  - States and transitions are represented visually.
- **Abstract**
  - States and transitions are simplified.

**Automatic Abstraction Refinement**

- **Refinement Loop**
  - Generate initial abstraction.
  - Model check.
  - Check if CE exists.
  - Refinement.

- **Issues**
  - We know how to model check (symbolically or explicitly).
  - But how to check a counterexample and how to refine?

---

**Definitions**

- **Bad States**
- **Deadend States**
- **Failure State**

**CE**

- **Spurious**
- **Not Spurious**
- **Check if CE is Spurious**
- **Stop**

**Model**

- **\( M \)**
- **\( \psi \)**
- **\( M_i \)**
- **\( M_i+1 \)**
Automatic Abstraction Refinement

Checking Counterexamples

Automatic Theorem Proving
- Given an abstract counterexample \((s_0, \ldots, s_n)\) of \(M’\)
- Use automatic theorem prover to show that there exists no series of states \((s_0, \ldots, s_n)\) such that
  - \(s_0 \in S_0\)
  - \((s_i, s_{i+1}) \in R\) for all \(i\)
  - \(h(s_i) = s_i\) for all \(i\)

Automatic Abstraction Refinement

Checking Counterexamples

SAT solving
- Given an abstract counterexample \((s_0, \ldots, s_n)\) of \(M’\)
- There exists no corresponding concrete path by if the following is unsatisfiable
  \[
  \Omega = l(s_0) \land \bigwedge_{i=0}^{n-1} R(s_i, s_{i+1}) \land \bigwedge_{i=0}^{n-1} h(s_i) = s_i
  \]
- A satisfying assignment gives a real counterexample.
- If we find a satisfying assignment, then \(M|\neq \emptyset\)

Abstraction Refinement

Refining the abstraction

Automatic Theorem Prover
- Use predicates found by the theorem prover
- A lot of effort in finding the right predicates (small and useful)
- Details exceed scope of this lecture

Abstraction Refinement

Refining the abstraction

SAT-solving
- The conflict clauses show why there exist no counterexample in \(M\)
- Use predicates found in conflict clauses, or
- Make variable visible that appear (a lot) in conflict clauses
- Reducing the set of relevant clauses by analysis of the conflict dependency graph.
Abstraction Refinement

Refining the abstraction

Conflict Dependency?
- A conflict clause $A$ may appear in the implication graph of a second conflict $B$
- We then say that $B$ depends directly on $A$

Example

Driver verification with SLAM
- Microsoft blames most Windows crashes on third party device drivers
- SLAM: Tool to automatically check device drivers for certain errors
- Specification in SLIC: Finite state language for stating rules
  - monitors behavior of C code
  - temporal safety properties – similar to what SPIN does
  - familiar C syntax

Example

Locking Rule

```
state {
  enum {Locked, Unlocked}
  s = Unlocked;
}

AcquireSpinLock.entry {
  if (s==Locked)
    abort;
  else s = Locked;
}

ReleaseSpinLock.entry {
  if (s==Unlocked)
    abort;
  else s = Unlocked;
}
```

Locking Rule in SLIC
Example

\[
\begin{align*}
d & \rightarrow (\text{KeAcquireSpinLock}(); \\
& nPacketsOld = nPackets; \\
if & (request) \{ \\
& \text{request} = \text{request}\rightarrow\text{Next}; \\
& \text{KeReleaseSpinLock}(); \\
& nPackets++; \\
\} \text{ while } (nPackets \neq nPacketsOld); \\
\text{KeReleaseSpinLock}(); \\
\end{align*}
\]

Example

\[\text{KeReleaseSpinLock}();
\]

Example

\[\text{KeReleaseSpinLock}();
\]

Example

\[\text{KeReleaseSpinLock}();
\]

Example

\[\text{KeReleaseSpinLock}();
\]
Example

do {
    KeAcquireSpinLock();
    b = true;
    if(!*){
        KeReleaseSpinLock();
        b = b ? false : *;
    }
} while (!b);

KeReleaseSpinLock();

Example

do {
    KeAcquireSpinLock();
    b = true;
    if(!*){
        KeReleaseSpinLock();
        b = b ? false : *;
    }
} while (!b);

KeReleaseSpinLock();

Summary

Counterexample guided abstraction refinement

Initial Abstraction
Refinement
Refinement
Correct!

Original Model

Summary

Counterexample guided abstraction refinement

Initial Abstraction
Refinement
Refinement
Real CE!

Original Model
Conclusions

Predicate abstraction and abstraction refinement have become a standard technique in software verification.

(C programs) SLAM '00
- Microsoft Research
- Abstract C programs to Boolean programs

(C programs) BLAST
- Berkeley and Los Angeles
- On-the-fly Predicate Abstraction and proof-based CE analysis

(C programs) MAGIC
- CMU
- SAT-based CE analysis

(Java programs) ESC/Java, Bandera, ...

Conclusion

Next week

Static Analysis
What else can you do to check the correctness of software?