Introduction to IC3 & IC3CFA

Tim Lange

Software Modeling and Verification Group, RWTH Aachen

MOVES Seminar, September 23, 2016
Outline

Inductive Invariants

Finite State Inductive Strengthening

IC3

IC3 on Control Flow Automata
### Inductive Invariants

#### Inductivity

A property $P$ is inductive if it satisfies initiation and consecution:

- **Initiation:** $I \Rightarrow P$
- **Consecution:** $P \land T \Rightarrow P'$

#### Inductive invariant

Given a property $P$ of a system, if $P$ is inductive on $S$ it is an invariant on $S$.

---

Inductive Invariants

Inductivity

A property $P$ is inductive if it satisfies initiation and consecution:

Initiation: $I \Rightarrow P$

Consecution: $P \land T \Rightarrow P'$

Inductive invariant

Given a property $P$ of a system, if $P$ is inductive on $S$ it is an invariant on $S$.

Not vice versa

Even if $P$ is an invariant on $S$, it may not be inductive.

Finite State Inductive Strengthening

Proving a property $P$ on a system $S$

Proving that $P$ holds on $S$ in general is hard. But if $P$ would be inductive, it would be trivial. If we only had a way to make $P$ inductive ...

Inductive Strengthening

Try to find a formula $F$ that is an inductive strengthening of $P$, i.e.

Initiation: $I \Rightarrow P \land F$

Consecution: $P \land F \land T \Rightarrow P' \land F'$

Finite State Inductive Strengthening

The bad guys

An $F$-state that has a transition to a $\neg F$-state is called *Counterexample to Induction* (CTI) and is a direct witness for why $F$ is not inductive.

Finding CTIs

A CTI can be found using a simple satisfiability query

\[ \text{sat}(F \land T \land \neg F). \]

If the query is sat, there exists a CTI and we can extract the state from the model of the solver (satisfying variable assignment).
## Handling CTIs

After finding a CTI $s$ we check whether there exists a *minimal inductive subclause* for $\neg s$. If not, we update $P$ that we have to prove that $s$ is not reachable. Otherwise we can add $\neg s$ to the strengthening $F$.

---

**Finite State Inductive Strengthening**
Handling CTIs

After finding a CTI $s$ we check whether there exists a *minimal inductive subclause* for $\neg s$. If not, we update $P$ that we have to prove that $s$ is not reachable. Otherwise we can add $\neg s$ to the strengthening $F$.

Happy End?

FSIS terminates if either

- $P \land F$ becomes inductive, or
- $\not I \Rightarrow P$ anymore.
Finding $F$ is hard

In many cases it can be pretty hard to come up with such a strengthening. Especially finding a minimal inductive subclause $e$ that is inductive relative to $P \land F$ is hard.

Solution

Instead of computing $F$ directly, compute a sequence $F_0, \ldots, F_k$, called frames, to find an inductive strengthening within these $F_i$. 

---

Frames

For a frame sequence $F_0, \ldots, F_k$ to be an inductive strengthening, the following must hold:

1. $I \Rightarrow F_0$  
2. $F_i \Rightarrow F_{i+1}$  
3. $F_i \Rightarrow P$  
4. $F_i \land T \Rightarrow F'_{i+1}$
Algorithm 1 Outer loop

```plaintext
function bool prove
    Check 0- and 1-step counterexamples
    Initialise frames $F_0 = I$, $F_1 = P$
    for $k = 1$ to ...
        Blocking phase
        Propagation phase
        Check termination
```

IC3

Handling CTIs

Given a CTI $c$ in the last frame $F_k$, we check whether $c$ is reachable from $F_{k-1}$ in one step.

Thinking inductive:

From an $F_{k-1}$ state that is not $c$, do we stay in not $c$ after one step? In other words: Is $\neg c$ inductive relative to $F_k$:

$$F_k \land \neg c \land T \Rightarrow \neg c$$

How to check validity

Validity of the implication can be solved as $\text{unsat}(F_k \land \neg c \land T \land c)$.
Termination

If for any frame $F_i$, $0 \leq i < k$ it holds that $F_i = F_{i+1}$ then

$$F_i \land T \Rightarrow F_{i+1}$$

$$\Leftrightarrow F_i \land T \Rightarrow F_i$$
Consider the transition system $\mathcal{M} = (X, I, T)$
Consider the transition system $\mathcal{M} = (X, I, T)$ and the property $P(X)$. 
Consider the transition system $\mathcal{M} = (X, I, T)$ and the property $P(X)$. 
Consider the transition system $\mathcal{M} = (X, I, T)$ and the property $P(X)$. 
Consider the transition system \( \mathcal{M} = (X, I, T) \) and the property \( P(X) \).
Consider the transition system \( \mathcal{M} = (X, I, T) \) and the property \( P(X) \).
Consider the transition system $\mathcal{M} = (X, I, T)$ and the property $P(X)$. 

![Diagram of a transition system with nodes a, b, c, d, e, v, and w, and edges connecting them with labels $F_0$, $F_1$, and $F_2 = P$.]
Consider the transition system $\mathcal{M} = (X, I, T)$ and the property $P(X)$. 
Consider the transition system $\mathcal{M} = (X, I, T)$ and the property $P(X)$. 
Consider the transition system $\mathcal{M} = (X, I, T)$ and the property $P(X)$.  

\[ F_0 \leq F_1 \leq F_2 \leq F_3 = P \]
Consider the transition system $\mathcal{M} = (X, I, T)$ and the property $P(X)$. 

![Diagram showing states and transitions related to the transition system $\mathcal{M}$.]
Consider the transition system $\mathcal{M} = (X, I, T)$ and the property $P(X)$. 

\[ F_3 = P \] 

\[ F_0 \rightarrow F_1 \rightarrow F_2 \rightarrow F_3 \]
Consider the transition system $\mathcal{M} = (X, I, T)$ and the property $P(X)$. 
Consider the transition system $\mathcal{M} = (X, I, T)$ and the property $P(X)$. 
IC3 on Control Flow Automata

Motivation

Lifting to software model checking

- IC3 had a deep impact in hardware model checking
- Showed much better performance than known techniques
- Nowadays employed in most major hardware model checking tools

Challenges

- Domain in hardware model checking finite (bit-level)
- How to handle infinite state spaces?
- How to encode finite control flow?
Control Flow Automaton (CFA)

A CFA $A = (L, G, l_0, l_E)$ consists of a set of locations $L = \{0, \ldots, n\}$ and edges in $G \subseteq L \times QFFO \times L$ labeled with quantifier-free first-order formulas, an initial location $l_0$, and an error location $l_E$. 

---

1. CG12.
IC3 on Control Flow Automata

Control Flow Automaton (CFA)

A CFA $\mathcal{A} = (L, G, l_0, l_E)$ consists of a set of locations $L = \{0, \ldots, n\}$ and edges in $G \subseteq L \times QFFO \times L$ labeled with quantifier-free first-order formulas, an initial location $l_0$, and an error location $l_E$.

Idea

- Encoding of control flow using special $pc$ variable not efficient\(^1\)
- Extraction of control flow advantageous
- Instead of unrolling into ART [CG12] apply IC3 directly on CFA
- For every location in the CFA construct frames $F_0, \ldots, F_k$
- Frames represent overapproximations of $i$-step reachability in location
- Explicit control flow locations allow to take only single transitions into account

---

\(^1\)CG12.
IC3 on Control Flow Automata

Example

Initial location: $l_0$
Error location: $l_E$
Terminating location: 2
IC3 on Control Flow Automata

Example

Frames $F_{(i,l)}$

<table>
<thead>
<tr>
<th>$i:$</th>
<th>$l_0$</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>1</td>
<td>true</td>
<td>true</td>
</tr>
</tbody>
</table>
IC3 on Control Flow Automata

Example

Frames $F_{(i,l)}$

<table>
<thead>
<tr>
<th>$i:$</th>
<th>$l:$</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$l_0$</td>
<td>true</td>
<td>false</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>true</td>
<td>true</td>
<td></td>
</tr>
</tbody>
</table>

CTI $(1,x \neq y)$, level 1
IC3 on Control Flow Automata

Example

Frames $F_{(i,l)}$

<table>
<thead>
<tr>
<th>$i$</th>
<th>$l$</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>true</td>
<td>false</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>true</td>
<td>true</td>
<td></td>
</tr>
</tbody>
</table>

CTI $(1,x \neq y)$, level 1

$SAT(F_{(0,1)} \land \neg (x \neq y) \land T_{1 \rightarrow 1} \land x' \neq y')$
IC3 on Control Flow Automata

Example

Frames $F_{(i,l)}$

<table>
<thead>
<tr>
<th>$i$</th>
<th>$l_0$</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>1</td>
<td>true</td>
<td>true</td>
</tr>
</tbody>
</table>

CTI $(1, x \neq y)$, level 1

$\text{SAT}(F_{(0,1)} \land \neg(x \neq y) \land T_{1 \rightarrow 1} \land x' \neq y') \times$
IC3 on Control Flow Automata

Example

Frames $F_{(i,l)}$

<table>
<thead>
<tr>
<th>$i$</th>
<th>$l$</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$l_0$</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>1</td>
<td>true</td>
<td>true</td>
<td></td>
</tr>
</tbody>
</table>

CTI $(1, x \neq y)$, level 1

\[ SAT(F_{(0,1)} \land \neg(x \neq y) \land T_{1 \rightarrow 1} \land x' \neq y') \times \]
\[ SAT(F_{(0,l_0)} \land T_{l_0 \rightarrow 1} \land x' \neq y') \]
IC3 on Control Flow Automata

Example

Frames $F_{(i,l)}$

<table>
<thead>
<tr>
<th>$i$</th>
<th>$l$</th>
<th>$l_0$</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>true</td>
<td>false</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>true</td>
<td>true</td>
<td></td>
</tr>
</tbody>
</table>

CTI $(1, x \neq y)$, level 1

$SAT(F_{(0,1)} \land \neg(x \neq y) \land T_{1 \rightarrow 1} \land x' \neq y')$  

$SAT(F_{(0, l_0)} \land T_{l_0 \rightarrow 1} \land x' \neq y')$  

IC3 on Control Flow Automata

Example

Frames $F_{(i,l)}$

<table>
<thead>
<tr>
<th>$l$:</th>
<th>$l_0$</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$i$:</td>
<td>0</td>
<td>true</td>
</tr>
<tr>
<td>1</td>
<td>true</td>
<td>$x = y$</td>
</tr>
</tbody>
</table>

CTI $(1, x \neq y)$, level 1

$SAT(F_{(0,1)} \land \neg(x \neq y) \land T_{1\to1} \land x' \neq y') \times$

$SAT(F_{(0,l_0)} \land T_{l_0\to1} \land x' \neq y') \times$
IC3 on Control Flow Automata

Example

I: $l_0$

\[ x++; \]
\[ y++; \]

Frames $F_{(i,l)}$

<table>
<thead>
<tr>
<th>$i$</th>
<th>$l_0$</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>1</td>
<td>true</td>
<td>$x = y$</td>
</tr>
</tbody>
</table>
### Example

A control flow automaton with transitions:
- $x++$ and $y++$ transitions to state 1 from $l_0$.
- $x \neq y$ transitions to $l_E$ from 1.
- $x = y$ transitions from 1 to 2.

### Frames $F(i,l)$

<table>
<thead>
<tr>
<th>$i:$</th>
<th>$l:$</th>
<th>$l_0$</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>true</td>
<td>false</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>true</td>
<td></td>
<td>$x = y$</td>
</tr>
<tr>
<td>2</td>
<td>true</td>
<td></td>
<td>$x = y$</td>
</tr>
</tbody>
</table>
References


Aaron R. Bradley. “SAT-Based Model Checking without Unrolling”. In: VMCAI. 2011, pp. 70–87.

Alessandro Cimatti and Alberto Griggio. “Software Model Checking via IC3”. In: CAV. 2012, pp. 277–293.