Outline

1. Proving Invariants by Induction
   - Induction for Transition Systems
   - Strengthening
   - Relative Induction

2. Proving Safety Properties with IC3
   - Basic Algorithm
   - Examples
   - Efficiency

3. Proving Progress Properties
   - FAIR
   - Examples

4. Proving Branching Time Properties
   - ICTL
   - Example
Outline

1. Proving Invariants by Induction
   Induction for Transition Systems
   Strengthening
   Relative Induction

2. Proving Safety Properties with IC3
   Basic Algorithm
   Examples
   Efficiency

3. Proving Progress Properties
   FAIR
   Examples

4. Proving Branching Time Properties
   IICTL
   Example
Finite-State Transition Systems

IIV algorithms work on a symbolic representation of a system:

\[ S : (\bar{i}, \bar{x}, I(\bar{x}), T(\bar{i}, \bar{x}, \bar{x}')) \]

- \( \bar{i} \): primary inputs
- \( \bar{x} \): state variables
- \( \bar{x}' \): next state variables
- \( I(\bar{x}) \): initial states
- \( T(\bar{i}, \bar{x}, \bar{x}') \): transition relation
IC3 proves (or refutes) invariants

- Prove that every reachable state satisfies $P(\overline{x})$
  - $P$ is a propositional formula
- Checking safety properties is reduced to checking invariance properties
  - Compose system with (safety) automaton that accepts the counterexamples to the property
  - Check that no reachable state is accepting
Mutual Exclusion for a Simple Arbiter

\[ I(\overline{g}) = \neg g_1 \land \neg g_2 \]
\[ \exists r_1, r_2 \cdot T(\overline{r}, \overline{g}, \overline{g}') = \neg g'_1 \lor \neg g'_2 \]
\[ P(\overline{g}) = \neg g_1 \lor \neg g_2 \]
Inductive Proofs for Transition Systems

- **Prove initiation** (base case)
  - \(I(\overline{x}) \Rightarrow P(\overline{x})\)
  - All initial states satisfy \(P\)
  - \((\neg g_1 \land \neg g_2) \Rightarrow (\neg g_1 \lor \neg g_2)\)

- **Prove consecution** (inductive step)
  - \(P(\overline{x}) \land T(i, \overline{x}, \overline{x}') \Rightarrow P(\overline{x}')\)
  - All successors of states satisfying \(P\) satisfy \(P\)
  - \((\neg g_1 \lor \neg g_2) \land (\neg g'_1 \lor \neg g'_2) \Rightarrow (\neg g'_1 \lor \neg g'_2)\)

- If both pass, all reachable states satisfy the property
  - \(S \models P\)
Visualizing Inductive Proofs

The inductive assertion (yellow) contains all initial (blue) states and no arrow leaves it (it is closed under the transition relation)
Counterexamples to Induction: The Troublemakers
Counterexamples to Induction: The Troublemakers

\[ 00 \rightarrow 01 \rightarrow 11 \rightarrow 10 \]

CTI
Invariant Strengthening
Invariant Strengthening
Invariant Strengthening
Invariant Strengthening
Strong and Weak Invariants

Induction is not restricted to:

- the strongest inductive invariant (forward-reachable states)
- ... or the weakest inductive invariant (complement of the backward-reachable states)
- \( \neg x_1 \) is simpler than \( \neg x_1 \land (\neg x_2 \lor \neg x_3) \) (strongest) and \( (\neg x_1 \lor \neg x_3) \) (weakest)
Strong and Weak Invariants

Induction is not restricted to:

- the strongest inductive invariant (forward-reachable states)
- ... or the weakest inductive invariant (complement of the backward-reachable states)
- \( \neg x_1 \) is simpler than \( \neg x_1 \land (\neg x_2 \lor \neg x_3) \) (strongest) and \( (\neg x_1 \lor \neg x_3) \) (weakest)
Completeness for Finite-State Systems

- CTIs are effectively bad states
  - If a CTI is reachable so is at least one bad state
- Remove CTI from $P$ and try again
- Eventually either:
  - An inductive strengthening of $P$ results
  - An initial state is removed from $P$
- In the latter case, a counterexample is obtained
Examples of Strengthening Strategies

- Removing one CTI at a time is very inefficient!
  - Several strategies in use to avoid that
- Fixpoint-based invariant checking: if $\nu Z . p \land AX Z$ converges in $n > 0$ iterations, then $\land_{0 \leq i < n} AX^i p$ is an inductive invariant
  - In fact, the weakest inductive invariant
- $k$-induction: if all states on length-$k$ paths from the initial states satisfy $p$, and $k$ distinct consecutive states satisfying $p$ are always followed by a state satisfying $p$, then all states reachable from the initial states satisfy $p$.
- fsis algorithm: try to extract an inductive clause from CTI to exclude multiple CTIs
Relative Induction

\[ \varphi = \neg x_1 \land (x_1 \lor \neg x_2) \]
Relative Induction

$\neg x_1$ is not inductive
Relative Induction

$x_1 \lor \neg x_2$ is inductive
Relative Induction

\[ \neg x_1 \text{ is inductive relative to } x_1 \lor \neg x_2 \]
Shortcoming of Relative Induction

\[ P = (x_1 \lor x_2 \lor x_3) \land (\neg x_1 \lor \neg x_2 \lor x_3) \]
\[ \varphi = (x_1 \lor x_2) \land (\neg x_1 \lor \neg x_2) \]
Shortcoming of Relative Induction

\[(x_1 \lor x_2) \land P \land T \not\Rightarrow (x_1' \lor x_2')\]
Shortcoming of Relative Induction

\[(\neg x_1 \lor \neg x_2) \land P \land T \not\Rightarrow (\neg x'_1 \lor \neg x'_2)\]
Shortcoming of Relative Induction

\[(x_1 \lor x_2) \land (\neg x_1 \lor \neg x_2) \land P \land T \Rightarrow (x'_1 \lor x'_2) \land (\neg x'_1 \lor \neg x'_2)\]
Shortcoming of Relative Induction

\[(x_1 \lor x_2) \text{ and } (\neg x_1 \lor \neg x_2) \text{ are } \text{mutually inductive}\]
Outline

1. Proving Invariants by Induction
   - Induction for Transition Systems
   - Strengthening
   - Relative Induction

2. Proving Safety Properties with IC3
   - Basic Algorithm
   - Examples
   - Efficiency

3. Proving Progress Properties
   - FAIR
   - Examples

4. Proving Branching Time Properties
   - IICTL
   - Example
What Does IC3 Stand for?

- **Incremental** Construction of
- **Inductive Clauses** for
- **Indubitable Correctness**
Basic Tenets

- Approximate reachability assumptions
  - $F_i$: contains at least all the states reachable in $i$ steps or less
  - If $S \models P$, $F_i$ eventually becomes inductive for some $i$
  - Approximation is desirable: IC3 does not attempt to get the most precise $F_i$’s

- Stepwise relative induction
  - Learn useful facts via induction relative to reachability assumptions

- Clausal representation
  - Learn clauses from CTIs
  - A form of abstract interpretation
IC3 Invariants

- The four main invariants of IC3:
  \[ I \Rightarrow F_0 \]
  \[ F_i \Rightarrow F_{i+1} \quad 0 \leq i < k \]
  \[ F_i \Rightarrow P \quad 0 \leq i \leq k \]
  \[ F_i \wedge T \Rightarrow F_{i+1}' \quad 0 \leq i < k \]

- Established if no counterexamples of length 0 or 1 are found

- The implicit invariant of IC3’s outer loop: no counterexamples of length \( k \) or less
Reasonable Invariants

- $I \Rightarrow F_0$: $F_0$ overapproximates the initial condition. (In practice, $I = F_0$.)
- $F_i \Rightarrow F_{i+1}$: a state believed to be reachable in $i$ steps or less is also believed to be reachable in $i + 1$ steps or less.
- $F_i \Rightarrow P$: no state believed to be reachable in $i$ steps or less violates $P$.
- $F_i \land T \Rightarrow F'_{i+1}$: all the immediate successors of a state believed to be reachable in $i$ steps or less are believed to be reachable in $i + 1$ steps or less.
Pseudo-Pseudocode

```java
bool IC3 {
    if (I \not\Rightarrow P \text{ or } I \land T \not\Rightarrow P')
        return ⊥
    F_0 = I; F_1 = P; k = 1
    repeat {
        while (there are CTIs in F_k) {
            either find a counterexample and return ⊥
            or refine F_1, ..., F_k
        }
        k ++
        set F_k = P and propagate clauses
        if (F_i = F_{i+1} for some 0 < i < k)
            return ⊤
    }
}
```
Example: Passing Property

No counterexamples of length 0 or 1

\[ I = \neg x_1 \land \neg x_2 \]
\[ P = \neg x_1 \lor x_2 \]

\[ I \Rightarrow F_0 \]
\[ F_i \Rightarrow F_{i+1} \quad 0 \leq i < k \]
\[ F_i \Rightarrow P \quad 0 \leq i \leq k \]
\[ F_i \land T \Rightarrow F_{i+1}' \quad 0 \leq i < k \]
Example: Passing Property

Does $F_1 \land T \Rightarrow P'$?

- $F_0 = I = \neg x_1 \land \neg x_2$
- $F_1 = P = \neg x_1 \lor x_2$

Transitions:

- $I \Rightarrow F_0$
- $F_i \Rightarrow F_{i+1}$ for $0 \leq i < k$
- $F_i \Rightarrow P$ for $0 \leq i \leq k$
- $F_i \land T \Rightarrow F_{i+1}'$ for $0 \leq i < k$
Example: Passing Property

Found CTI \( s = x_1 \land x_2 \)

\[ F_0 = I = \neg x_1 \land \neg x_2 \]
\[ F_1 = P = \neg x_1 \lor x_2 \]

\[ I \Rightarrow F_0 \]
\[ F_i \Rightarrow F_{i+1} \quad 0 \leq i < k \]
\[ F_i \Rightarrow P \quad 0 \leq i \leq k \]
\[ F_i \land T \Rightarrow F'_i \quad 0 \leq i < k \]
Example: Passing Property

Is $\neg s = \neg x_1 \lor \neg x_2$ inductive relative to $F_1$?

\[
\begin{array}{l}
I \Rightarrow F_0 \\
F_i \Rightarrow F_{i+1} \\
F_i \Rightarrow P \\
F_i \land T \Rightarrow F'_i \\
\end{array}
\]

\[
F_0 = I = \neg x_1 \land \neg x_2 \\
F_1 = P = \neg x_1 \lor x_2
\]

\[
0 \leq i < k
\]

\[
0 \leq i \leq k
\]

\[
0 \leq i < k
\]
Example: Passing Property

No. Is \( \neg s = \neg x_1 \lor \neg x_2 \) inductive relative to \( F_0 \)?

\[ F_0 = I = \neg x_1 \land \neg x_2 \]
\[ F_1 = P = \neg x_1 \lor x_2 \]

\[ I \Rightarrow F_0 \]
\[ F_i \Rightarrow F_{i+1} \quad 0 \leq i < k \]
\[ F_i \Rightarrow P \quad 0 \leq i \leq k \]
\[ F_i \land T \Rightarrow F'_i \quad 0 \leq i < k \]
Example: Passing Property

Yes. Generalize $\neg s$ at level 0 in one of the two possible ways: either $\neg x_1$ or $\neg x_2$

$F_0 = I = \neg x_1 \land \neg x_2$

$F_1 = P = \neg x_1 \lor x_2$

$I \Rightarrow F_0$

$F_i \Rightarrow F_{i+1}$

$0 \leq i < k$

$F_i \Rightarrow P$

$0 \leq i \leq k$

$F_i \land T \Rightarrow F_i'$

$0 \leq i < k$
Example: Passing Property

Update $F_1$

$I \Rightarrow F_0$

$F_i \Rightarrow F_{i+1}$

$F_i \Rightarrow P$

$F_i \land T \Rightarrow F'_i$

$F_0 = I = \neg x_1 \land \neg x_2$

$F_1 = (\neg x_1 \lor x_2) \land \neg x_2$

$0 \leq i < k$

$0 \leq i \leq k$

$0 \leq i < k$
Example: Passing Property

No more CTIs in $F_1$. No counterexamples of length 2. Instantiate $F_2$

\[
\begin{align*}
F_0 &= I = \neg x_1 \land \neg x_2 \\
F_1 &= (\neg x_1 \lor x_2) \land \neg x_2 \\
F_2 &= P = \neg x_1 \lor x_2 \\
\end{align*}
\]

\[
\begin{align*}
I &\Rightarrow F_0 \\
F_i &\Rightarrow F_{i+1}, 0 \leq i < k \\
F_i &\Rightarrow P, 0 \leq i \leq k \\
F_i \land T &\Rightarrow F_{i+1}', 0 \leq i < k \\
\end{align*}
\]
Example: Passing Property

Propagate clauses from $F_1$ to $F_2$

\[ F_0 = I = \neg x_1 \land \neg x_2 \]
\[ F_1 = (\neg x_1 \lor x_2) \land \neg x_2 \]
\[ F_2 = (\neg x_1 \lor x_2) \land \neg x_2 \]

\[ I \Rightarrow F_0 \]
\[ F_i \Rightarrow F_{i+1} \quad 0 \leq i < k \]
\[ F_i \Rightarrow P \quad 0 \leq i \leq k \]
\[ F_i \land T \Rightarrow F_i' \quad 0 \leq i < k \]
Example: Passing Property

\( F_1 \) and \( F_2 \) are identical. Property proved

\[
\begin{align*}
F_0 &= I = \neg x_1 \land \neg x_2 \\
F_1 &= (\neg x_1 \lor x_2) \land \neg x_2 \\
F_2 &= (\neg x_1 \lor x_2) \land \neg x_2
\end{align*}
\]

\[
\begin{align*}
l \Rightarrow F_0 \\
F_i \Rightarrow F_{i+1} & \quad 0 \leq i < k \\
F_i \Rightarrow P & \quad 0 \leq i \leq k \\
F_i \land T \Rightarrow F_i' & \quad 0 \leq i < k
\end{align*}
\]
Example: Passing Property

What happens if we generalize $\neg s = \neg x_1 \lor \neg x_2$ at level 0 in the other way ($\neg x_1$)?

$$F_0 = l = \neg x_1 \land \neg x_2$$
$$F_1 = \neg x_1 \lor x_2$$

$$I \Rightarrow F_0$$
$$F_i \Rightarrow F_{i+1}$$
$$0 \leq i < k$$

$$F_i \Rightarrow P$$
$$0 \leq i \leq k$$

$$F_i \land T \Rightarrow F_i'$$
$$0 \leq i < k$$
Example: Passing Property

Update $F_1$

\[ F_0 = I = \neg x_1 \land \neg x_2 \]

\[ F_1 = (\neg x_1 \lor x_2) \land \neg x_1 \]

\[ I \Rightarrow F_0 \]

\[ F_i \Rightarrow F_{i+1} \quad 0 \leq i < k \]

\[ F_i \Rightarrow P \quad 0 \leq i \leq k \]

\[ F_i \land T \Rightarrow F'_i \quad 0 \leq i < k \]
Example: Passing Property

No more CTIs in $F_1$. No counterexamples of length 2. Instantiate $F_2$

\[ I \Rightarrow F_0 \]
\[ F_i \Rightarrow F_{i+1} \quad 0 \leq i < k \]
\[ F_i \Rightarrow P \quad 0 \leq i \leq k \]
\[ F_i \land T \Rightarrow F'_i \quad 0 \leq i < k \]

$F_0 = I = \neg x_1 \land \neg x_2$
$F_1 = (\neg x_1 \lor x_2) \land \neg x_1$
$F_2 = P = \neg x_1 \lor x_2$
Example: Passing Property

No clauses propagate from $F_1$ to $F_2$

$I \Rightarrow F_0$

$F_i \Rightarrow F_{i+1}$  $0 \leq i < k$

$F_i \Rightarrow P$  $0 \leq i \leq k$

$F_i \land T \Rightarrow F'_{i+1}$  $0 \leq i < k$

$F_0 = I = \neg x_1 \land \neg x_2$

$F_1 = (\neg x_1 \lor x_2) \land \neg x_1$

$F_2 = P = \neg x_1 \lor x_2$
Example: Passing Property

Remove subsumed clauses

\[ I \implies F_0 \]
\[ F_i \implies F_{i+1} \]
\[ F_i \implies P \]
\[ F_i \land T \implies F'_{i+1} \]

\[ F_0 = I = \neg x_1 \land \neg x_2 \]
\[ F_1 = \neg x_1 \]
\[ F_2 = P = \neg x_1 \lor x_2 \]

\[ 0 \leq i < k \]
\[ 0 \leq i \leq k \]
\[ 0 \leq i < k \]
Example: Passing Property

Does $F_2 \land T \Rightarrow P'$?

$$F_0 = I = \neg x_1 \land \neg x_2$$
$$F_1 = \neg x_1$$
$$F_2 = P = \neg x_1 \lor x_2$$

$I \Rightarrow F_0$
$F_i \Rightarrow F_{i+1}$
$F_i \Rightarrow P$
$F_i \land T \Rightarrow F'_i$

$0 \leq i \leq k$

$0 \leq i < k$
Example: Passing Property

Found CTI $s = x_1 \land x_2$ (same as before)

\[ F_0 = I = \neg x_1 \land \neg x_2 \]
\[ F_1 = \neg x_1 \]
\[ F_2 = P = \neg x_1 \lor x_2 \]

\[ I \Rightarrow F_0 \]
\[ F_i \Rightarrow F_{i+1} \quad 0 \leq i < k \]
\[ F_i \Rightarrow P \quad 0 \leq i \leq k \]
\[ F_i \land T \Rightarrow F_i' \quad 0 \leq i < k \]
Example: Passing Property

Is $\neg s = \neg x_1 \lor \neg x_2$ inductive relative to $F_1$?

\begin{align*}
F_0 &= I = \neg x_1 \land \neg x_2 \\
F_1 &= \neg x_1 \\
F_2 &= P = \neg x_1 \lor x_2
\end{align*}

\begin{align*}
I &\Rightarrow F_0 \\
F_i &\Rightarrow F_{i+1} & 0 \leq i < k \\
F_i &\Rightarrow P & 0 \leq i \leq k \\
F_i \land T &\Rightarrow F_i' & 0 \leq i < k
\end{align*}
Example: Passing Property

No. We know it is inductive at level 0.

\[
\begin{align*}
F_0 &= I = \neg x_1 \land \neg x_2 \\
F_1 &= \neg x_1 \\
F_2 &= P = \neg x_1 \lor x_2
\end{align*}
\]

\[
\begin{align*}
I &\Rightarrow F_0 \\
F_i &\Rightarrow F_{i+1} \\
F_i &\Rightarrow P \\
F_i \land T &\Rightarrow F_{i+1}'
\end{align*}
\]

\[
\begin{align*}
0 &\leq i < k \\
0 &\leq i \leq k \\
0 &\leq i < k
\end{align*}
\]
Example: Passing Property

If generalization produces $\neg x_1$ again, the CTI is not eliminated.

$$F_0 = I = \neg x_1 \land \neg x_2$$
$$F_1 = \neg x_1$$
$$F_2 = P = \neg x_1 \lor x_2$$

$$I \Rightarrow F_0$$
$$F_i \Rightarrow F_{i+1}$$
$$F_i \Rightarrow P$$
$$F_i \land T \Rightarrow F'_{i+1}$$

$$0 \leq i < k$$
$$0 \leq i \leq k$$
$$0 \leq i < k$$
Example: Passing Property

Find predecessor $t$ of CTI $x_1 \land x_2$ in $F_1 \setminus F_0$

$$F_0 = I = \neg x_1 \land \neg x_2$$
$$F_1 = \neg x_1$$
$$F_2 = P = \neg x_1 \lor x_2$$

$$I \Rightarrow F_0$$
$$F_i \Rightarrow F_{i+1}$$
$$F_i \Rightarrow P$$
$$F_i \land T \Rightarrow F'_i$$

$$0 \leq i < k$$
Example: Passing Property

Found \( t = \neg x_1 \land x_2 \)

\[
\begin{align*}
F_0 &= I = \neg x_1 \land \neg x_2 \\
F_1 &= \neg x_1 \\
F_2 &= P = \neg x_1 \lor x_2
\end{align*}
\]

\[
\begin{align*}
I &\Rightarrow F_0 \\
F_i &\Rightarrow F_{i+1} \\
F_i &\Rightarrow P \\
F_i \land T &\Rightarrow F_{i+1}'
\end{align*}
\]
Example: Passing Property

The clause $\neg t = x_1 \lor \neg x_2$ is inductive at all levels

$$F_0 = I = \neg x_1 \land \neg x_2$$

$$F_1 = \neg x_1$$

$$F_2 = P = \neg x_1 \lor x_2$$

$$I \Rightarrow F_0$$

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

$$0 \leq i < k$$

$$F_i \Rightarrow P$$

$$0 \leq i \leq k$$

$$F_i \land T \Rightarrow F_i'$$

$$0 \leq i < k$$
Example: Passing Property

Generalization of \( \neg t = x_1 \lor \neg x_2 \) produces \( \neg x_2 \)

\[
\begin{align*}
F_0 & = I = \neg x_1 \land \neg x_2 \\
F_1 & = \neg x_1 \\
F_2 & = P = \neg x_1 \lor x_2
\end{align*}
\]

\[
\begin{align*}
I & \Rightarrow F_0 \\
F_i & \Rightarrow F_{i+1} \\
F_i & \Rightarrow P \\
F_i \land T & \Rightarrow F'_{i+1}
\end{align*}
\]

\[0 \leq i < k\]
Example: Passing Property

Update $F_1$ and $F_2$

$I \Rightarrow F_0$
$F_i \Rightarrow F_{i+1}$
$F_i \Rightarrow P$
$F_i \land T \Rightarrow F_i'$

$F_0 = I = \neg x_1 \land \neg x_2$
$F_1 = \neg x_1 \land \neg x_2$
$F_2 = (\neg x_1 \lor x_2) \land \neg x_2$

$0 \leq i < k$
Example: Passing Property

\( F_1 \) and \( F_2 \) are equivalent. Property (almost) proved

\[
\begin{align*}
F_0 &= I = \neg x_1 \land \neg x_2 \\
F_1 &= \neg x_1 \land \neg x_2 \\
F_2 &= (\neg x_1 \lor x_2) \land \neg x_2
\end{align*}
\]

\[
\begin{align*}
I &\Rightarrow F_0 \\
F_i &\Rightarrow F_{i+1} \quad 0 \leq i < k \\
F_i &\Rightarrow P \quad 0 \leq i \leq k \\
F_i \land T &\Rightarrow F'_{i+1} \quad 0 \leq i < k
\end{align*}
\]
“Principled” IC3

No counterexamples of length 0 or 1

\[ I \Rightarrow F_0 \]
\[ F_i \Rightarrow F_{i+1} \]
\[ F_i \Rightarrow P \]
\[ F_i \wedge T \Rightarrow F'_{i+1} \]

\[ I = \neg x_1 \land \neg x_2 \]
\[ P = \neg x_1 \lor x_2 \]

\[ 0 \leq i < k \]

\[ 0 \leq i \leq k \]

\[ 0 \leq i < k \]
“Principled” IC3

Does $F_1 \land T \Rightarrow P'$?

$I \Rightarrow F_0$

$F_i \Rightarrow F_{i+1}$

$F_i \Rightarrow P$

$F_i \land T \Rightarrow F'_{i+1}$

$F_0 = I = \neg x_1 \land \neg x_2$

$F_1 = P = \neg x_1 \lor x_2$
“Principled” IC3

Found CTI \( s = x_1 \land x_2 \)

\[
\begin{align*}
F_0 &= I = \neg x_1 \land \neg x_2 \\
F_1 &= P = \neg x_1 \lor x_2 \\
I &\Rightarrow F_0 \\
F_i &\Rightarrow F_{i+1} \\
F_i &\Rightarrow P \\
F_i \land T &\Rightarrow F_{i+1}' \\
0 \leq i < k &
\end{align*}
\]
"Principled" IC3

Does $\neg s = \neg x_1 \lor \neg x_2$ contain a subclause that is inductive relative to $F_1$?

$$F_0 = I = \neg x_1 \land \neg x_2$$
$$F_1 = P = \neg x_1 \lor x_2$$

$$I \Rightarrow F_0$$
$$F_i \Rightarrow F_{i+1}$$
$$F_i \Rightarrow P$$
$$F_i \land T \Rightarrow F_{i+1}'$$

$0 \leq i < k$
“Principled” IC3

Yes, $\neg x_2$

\[
F_0 = I = \neg x_1 \land \neg x_2 \quad F_1 = P = \neg x_1 \lor x_2
\]

\[
I \Rightarrow F_0
\]

\[
F_i \Rightarrow F_{i+1} \quad 0 \leq i < k
\]

\[
F_i \Rightarrow P \quad 0 \leq i \leq k
\]

\[
F_i \land T \Rightarrow F'_i \quad 0 \leq i < k
\]
“Principled” IC3

Update $F_1$

\[ F_0 = I = \neg x_1 \land \neg x_2 \]
\[ F_1 = (\neg x_1 \lor x_2) \land \neg x_2 \]

\[ I \Rightarrow F_0 \]
\[ F_i \Rightarrow F_{i+1} \quad 0 \leq i < k \]
\[ F_i \Rightarrow P \quad 0 \leq i \leq k \]
\[ F_i \land T \Rightarrow F_i' \quad 0 \leq i < k \]
“Principled” IC3

No more CTIs in \( F_1 \). No counterexamples of length 2

\[
\begin{align*}
F_0 &= I = \neg x_1 \land \neg x_2 \\
F_1 &= (\neg x_1 \lor x_2) \land \neg x_2
\end{align*}
\]

\[
\begin{align*}
I &\Rightarrow F_0 \\
F_i &\Rightarrow F_{i+1} \\
F_i &\Rightarrow P \\
F_i \land T &\Rightarrow F_{i+1}'
\end{align*}
\]

\[
\begin{align*}
0 &\leq i < k \\
0 &\leq i \leq k \\
0 &\leq i < k
\end{align*}
\]
“Principled” IC3

And so on

\[ F_0 = I = \neg x_1 \land \neg x_2 \]
\[ F_1 = (\neg x_1 \lor x_2) \land \neg x_2 \]

\[ I \Rightarrow F_0 \]
\[ F_i \Rightarrow F_{i+1} \]
\[ F_i \Rightarrow P \]
\[ 0 \leq i < k \]
\[ F_i \land T \Rightarrow F'_{i+1} \]
\[ 0 \leq i < k \]
Failing Property

No counterexamples of length 0 or 1

\[
I = \neg x_1 \land \neg x_3 \land \neg x_3
\]

\[
P = \neg x_1 \lor \neg x_2 \lor \neg x_3
\]

\[
F_i \Rightarrow F_0
\]

\[
F_i \Rightarrow F_{i+1}
\]

\[
F_i \Rightarrow P
\]

\[
F_i \land T \Rightarrow F'_{i+1}
\]

\[
0 \leq i < k
\]

\[
0 \leq i \leq k
\]

\[
0 \leq i < k
\]
Failing Property

Does \( F_1 \land T \Rightarrow P' \)?

\[
F_0 = I = \neg x_1 \land \neg x_3 \land \neg x_3
\]

\[
F_1 = P = \neg x_1 \lor \neg x_2 \lor \neg x_3
\]

\[
I \Rightarrow F_0
\]

\[
F_i \Rightarrow F_{i+1}
\]

\[
F_i \Rightarrow P
\]

\[
F_i \land T \Rightarrow F'_i
\]

\[
0 \leq i < k
\]

\[
0 \leq i \leq k
\]

\[
0 \leq i < k
\]
Failing Property

Found CTI $s = \neg x_1 \land x_2 \land x_3$

\[
F_0 = I = \neg x_1 \land \neg x_3 \land \neg x_3
\]
\[
F_1 = P = \neg x_1 \lor \neg x_2 \lor \neg x_3
\]

\[
I \Rightarrow F_0
\]
\[
F_i \Rightarrow F_{i+1}
\]
\[
F_i \Rightarrow P
\]
\[
F_i \land T \Rightarrow F'_i
\]
\[
0 \leq i < k
\]
\[
0 \leq i \leq k
\]
\[
0 \leq i < k
\]
Failing Property

The clause $\neg s = x_1 \lor \neg x_2 \lor \neg x_3$ generalizes to $\neg x_2$ at level 0

\[
F_0 = I = \neg x_1 \land \neg x_3 \land \neg x_3
\]
\[
F_1 = (\neg x_1 \lor \neg x_2 \lor \neg x_3) \land \neg x_2
\]

\[
I \Rightarrow F_0
\]
\[
F_i \Rightarrow F_{i+1} \quad 0 \leq i < k
\]
\[
F_i \Rightarrow P \quad 0 \leq i \leq k
\]
\[
F_i \land T \Rightarrow F'_i \quad 0 \leq i < k
\]
Failing Property

No CTI left: no counterexample of length 2. $F_2$ instantiated, but no clause propagated

\[ F_0 = I = \neg x_1 \land \neg x_3 \land \neg x_3 \]
\[ F_1 = \neg x_2 \]
\[ F_2 = P = \neg x_1 \lor \neg x_2 \lor \neg x_3 \]

\[ I \Rightarrow F_0 \]
\[ F_i \Rightarrow F_{i+1} \quad 0 \leq i < k \]
\[ F_i \Rightarrow P \quad 0 \leq i \leq k \]
\[ F_i \land T \Rightarrow F'_i \quad 0 \leq i < k \]
Failing Property

The clause \( \neg s \) generalizes again to \( \neg x_2 \) at level 0

\[
F_0 = I = \neg x_1 \land \neg x_3 \land \neg x_3
\]

\[
F_1 = \neg x_2
\]

\[
F_2 = P = \neg x_1 \lor \neg x_2 \lor \neg x_3
\]

\[
I \Rightarrow F_0
\]

\[
F_i \Rightarrow F_{i+1}
\]

\[
F_i \Rightarrow P
\]

\[
F_i \land T \Rightarrow F'_{i+1}
\]

\[
0 \leq i < k
\]

\[
0 \leq i \leq k
\]

\[
0 \leq i < k
\]
Failing Property

Suppose IC3 recurs on \( t = \neg x_1 \land \neg x_2 \land x_3 \) in \( F_1 \setminus F_0 \)

\[
\begin{align*}
F_0 &= I = \neg x_1 \land \neg x_3 \land \neg x_3 \\
F_1 &= \neg x_2 \\
F_2 &= P = \neg x_1 \lor \neg x_2 \lor \neg x_3 \\
I &\Rightarrow F_0 \\
F_i &\Rightarrow F_{i+1}  \quad 0 \leq i < k \\
F_i &\Rightarrow P  \quad 0 \leq i \leq k \\
F_i \land T &\Rightarrow F'_{i+1}  \quad 0 \leq i < k
\end{align*}
\]
Failing Property

Clause $\neg t = x_1 \lor x_2 \lor \neg x_3$ is not inductive at level 0: the property fails

\[
F_0 = I = \neg x_1 \land \neg x_3 \land \neg x_3
\]
\[
F_1 = \neg x_2
\]
\[
F_2 = P = \neg x_1 \lor \neg x_2 \lor \neg x_3
\]

\[
I \Rightarrow F_0
\]
\[
F_i \Rightarrow F_{i+1}
\]
\[
F_i \Rightarrow P
\]
\[
F_i \land T \Rightarrow F'_{i+1}
\]
Failing Property

Suppose now IC3 recurs on \( t = x_1 \land \neg x_2 \land x_3 \) in \( F_1 \setminus F_0 \)

\[
\begin{align*}
F_0 &= I = \neg x_1 \land \neg x_3 \land \neg x_3 \\
F_1 &= \neg x_2 \\
F_2 &= P = \neg x_1 \lor \neg x_2 \lor \neg x_3 \\
I \Rightarrow F_0 \\
F_i \Rightarrow F_{i+1} & \quad 0 \leq i < k \\
F_i \Rightarrow P & \quad 0 \leq i \leq k \\
F_i \land T \Rightarrow F'_{i+1} & \quad 0 \leq i < k
\end{align*}
\]
Failing Property

Clause $\neg t = \neg x_1 \lor x_2 \lor \neg x_3$ is inductive at level 1

$F_0 = I = \neg x_1 \land \neg x_3 \land \neg x_3$

$F_1 = \neg x_2$

$F_2 = P = \neg x_1 \lor \neg x_2 \lor \neg x_3$

$I \Rightarrow F_0$

$F_i \Rightarrow F_{i+1}$

$F_i \Rightarrow P$

$F_i \land T \Rightarrow F'_i$

$0 \leq i < k$

$0 \leq i \leq k$

$0 \leq i < k$
Failing Property

Generalization of $\neg t$ adds $\neg x_1$ to $F_1$ and $F_2$

\[
F_0 = I = \neg x_1 \land \neg x_3 \land \neg x_3
\]
\[
F_1 = \neg x_2 \land \neg x_1
\]
\[
F_2 = (\neg x_1 \lor \neg x_2 \lor \neg x_3) \land \neg x_1
\]

\[
I \Rightarrow F_0
\]
\[
F_i \Rightarrow F_{i+1}
\]
\[
F_i \Rightarrow P
\]
\[
F_i \land T \Rightarrow F'_i
\]

\[
0 \leq i < k
\]

\[
0 \leq i \leq k
\]

\[
0 \leq i < k
\]
Failing Property

Only $t = \neg x_1 \land \neg x_2 \land x_3$ remains in $F_1 \setminus F_0$

$I \Rightarrow F_0$

$F_i \Rightarrow F_{i+1} \quad 0 \leq i < k$

$F_i \Rightarrow P \quad 0 \leq i \leq k$

$F_i \land T \Rightarrow F_i' \quad 0 \leq i < k$

$F_0 = I = \neg x_1 \land \neg x_3 \land \neg x_3$

$F_1 = \neg x_2 \land \neg x_1$

$F_2 = (\neg x_1 \lor \neg x_2 \lor \neg x_3) \land \neg x_1$
Failing Property

The same counterexample as before is found

\[
\begin{align*}
F_0 &= I = \neg x_1 \land \neg x_3 \land \neg x_3 \\
F_1 &= \neg x_2 \land \neg x_1 \\
F_2 &= (\neg x_1 \lor \neg x_2 \lor \neg x_3) \land \neg x_1
\end{align*}
\]

\[
\begin{align*}
I &\Rightarrow F_0 \\
F_i &\Rightarrow F_{i+1} \\
F_i &\Rightarrow P \\
F_i \land T &\Rightarrow F_{i+1}'
\end{align*}
\]

\[
0 \leq i < k
\]

\[
0 \leq i \leq k
\]

\[
0 \leq i < k
\]
Clause Generalization

- A CTI is a **cube**
  - e.g., \( s = x_1 \land \neg x_2 \land x_3 \)
- The negation of a CTI is a **clause**
  - e.g., \( \neg s = \neg x_1 \lor x_2 \lor \neg x_3 \)
- Conjoining \( \neg s \) to a reachability assumption \( F_i \) excludes the CTI from it
- Generalization extracts a **subclause** from \( \neg s \) that excludes more states that are “like the CTI”
  - e.g., \( \neg x_3 \) may be a subclause of \( \neg s \) that excludes states that, like the CTI, are not reachable in \( i \) steps
  - Every literal dropped **doubles** the number of states excluded by a clause
  - Generalization is time-consuming, but critical to performance
Generalization

- Crucial for efficiency
- Generalization in IC3 produces a minimal inductive clause (MIC)
- The MIC algorithm is based on DOWN and UP.
- DOWN extracts the (unique) maximal subclause
- UP finds a small, but not necessarily minimal subclause
- MIC recurs on subclauses of the result of UP
Minimal Inductive Clause
Minimal Inductive Clause
Minimal Inductive Clause
Minimal Inductive Clause
Minimal Inductive Clause

![Minimal Inductive Clause Diagram]

- 1234
- 123
- 124
- 134
- 234
- 12
- 13
- 14
- 23
- 24
- 34
- 2
- 3
- 4
- ⊥
Maximal Inductive Subclause (DOWN)

\[ \neg x_1 \lor x_2 \lor \neg x_3 \]
Maximal Inductive Subclause (DOWN)

\[ \neg x_1 \lor x_2 \lor \neg x_3 \]
Maximal Inductive Subclause (DOWN)

\[ x_2 \lor \neg x_3 \]
Maximal Inductive Subclause (DOWN)

\[ x_2 \lor \neg x_3 \]
Maximal Inductive Subclause (DOWN)
Use of UNSAT Cores

- \( \neg s \land F_i \land T \Rightarrow \neg s' \) if and only if \( \neg s \land F_i \land T \land s' \) is unsatisfiable
- The literals of \( s' \) are (unit) clauses in the SAT query
- If the implication holds, the SAT solver returns an unsatisfiable core
- Any literal of \( s' \) not in the core can be removed from \( s' \) because it does not contribute to the implication ...
- and from \( \neg s \) because strengthening the antecedent preserves the implication
Use of UNSAT Core Example

- $\neg s \land F_0 \land T \Rightarrow \neg s'$ with
  
  $\neg s = \neg x_1 \lor \neg x_2$
  
  $F_0 = \neg x_1 \land \neg x_2$
  
  $T = (\neg x_1 \land \neg x_2 \land \neg x'_1 \land \neg x'_2) \lor \cdots$

- The SAT query, after some simplification, is
  
  $\neg x_1 \land \neg x_2 \land \neg x'_1 \land \neg x'_2 \land x'_1 \land x'_2$

- Two UNSAT cores are
  
  $\neg x'_1 \land x'_1$
  
  $\neg x'_2 \land x'_2$
  
  from which the two generalizations we saw before follow
Clause Clean-Up

• As IC3 proceeds, clauses may be added to some $F_i$s that subsume other clauses
• The weaker, subsumed clauses no longer contribute to the definition of $F_i$
• However, a weaker clause may propagate to $F_{i+1}$ when the stronger clause does not
• Weak clauses are eliminated by subsumption only between major iterations and after propagation
More Efficiency-Related Issues

- State encoding determines what clauses are derived
- Incremental vs. monolithic
  - Reachability assumptions carry global information
  - ... but are built incrementally
- Semantic vs. syntactic approach
  - Generalization “jumps over large distances”
- Long counterexamples at low $k$
  - Typically more efficient than increasing $k$
- Consequences of no unrolling
  - Many cheap (incremental) SAT calls
- Ability to parallelize
  - Clauses are easy to exchange
Outline

1. Proving Invariants by Induction
   Induction for Transition Systems
   Strengthening
   Relative Induction

2. Proving Safety Properties with IC3
   Basic Algorithm
   Examples
   Efficiency

3. Proving Progress Properties
   FAIR
   Examples

4. Proving Branching Time Properties
   IICTL
   Example
FAIR: Finding Reachable Fair Cycles

- Checking progress (non-safety) property $\varphi$ can be reduced to checking language nonemptiness of the composition of structure $S$ and generalized Büchi automaton for $\neg \varphi$
- Generalized means that multiple acceptance conditions (aka fairness constraints may be given: each must be satisfied
- FAIR looks for a reachable fair cycle
FAIR: Finding Reachable Fair Cycles

- Checking progress (non-safety) property $\varphi$ can be reduced to checking language nonemptiness of the composition of structure $S$ and generalized Büchi automaton for $\neg \varphi$
- Generalized means that multiple acceptance conditions (aka fairness constraints) may be given: each must be satisfied
- FAIR looks for a reachable fair cycle
FAIR: Finding Reachable Fair Cycles

- Checking progress (non-safety) property $\varphi$ can be reduced to checking language nonemptiness of the composition of structure $S$ and generalized Büchi automaton for $\neg \varphi$
- Generalized means that multiple acceptance conditions (aka fairness constraints) may be given: each must be satisfied
- FAIR looks for a reachable fair cycle
A counterexample to a progress property is a lasso-shaped path that satisfies fairness constraints.

- A lasso’s cycle is contained in a strongly connected component (SCC) of the state graph.
- A nonempty set of states is SCC-closed if every SCC is either contained in it or disjoint from it.
- A partition of the states into SCC-closed sets is a coarser partition than the SCC partition; hence, . . .
- Every cycle of a graph is contained in some SCC-closed set.
- Maintain a partition of the states into SCC-closed set.
  - Start with the trivial partition (all states in one set).
  - Refine it until a reachable fair cycle is found or none is proved to exist.
Strongly Connected Components

- A counterexample to a progress property is a lasso-shaped path that satisfies fairness constraints.
- A lasso’s cycle is contained in a strongly connected component (SCC) of the state graph.
  - A nonempty set of states is SCC-closed if every SCC is either contained in it or disjoint from it.
  - A partition of the states into SCC-closed sets is a coarser partition than the SCC partition; hence, …
  - Every cycle of a graph is contained in some SCC-closed set.
  - Maintain a partition of the states into SCC-closed set:
    - Start with the trivial partition (all states in one set).
    - Refine it until a reachable fair cycle is found or none is proved to exist.
Strongly Connected Components

- A counterexample to a progress property is a lasso-shaped path that satisfies fairness constraints.
- A lasso’s cycle is contained in a strongly connected component (SCC) of the state graph.
- A nonempty set of states is SCC-closed if every SCC is either contained in it or disjoint from it.
  - A partition of the states into SCC-closed sets is a coarser partition than the SCC partition; hence, ... 
  - Every cycle of a graph is contained in some SCC-closed set.
  - Maintain a partition of the states into SCC-closed set 
    - Start with the trivial partition (all states in one set)
    - Refine it until a reachable fair cycle is found or none is proved to exist.
Strongly Connected Components

- A counterexample to a progress property is a lasso-shaped path that satisfies fairness constraints.
- A lasso’s cycle is contained in a strongly connected component (SCC) of the state graph.
- A nonempty set of states is SCC-closed if every SCC is either contained in it or disjoint from it.
- A partition of the states into SCC-closed sets is a coarser partition than the SCC partition; hence, ...
  - Every cycle of a graph is contained in some SCC-closed set.
  - Maintain a partition of the states into SCC-closed set:
    - Start with the trivial partition (all states in one set).
    - Refine it until a reachable fair cycle is found or none is proved to exist.
Strongly Connected Components

- A counterexample to a progress property is a lasso-shaped path that satisfies fairness constraints.
- A lasso’s cycle is contained in a strongly connected component (SCC) of the state graph.
- A nonempty set of states is SCC-closed if every SCC is either contained in it or disjoint from it.
- A partition of the states into SCC-closed sets is a coarser partition than the SCC partition; hence, . . .
- Every cycle of a graph is contained in some SCC-closed set.
  - Maintain a partition of the states into SCC-closed set
    - Start with the trivial partition (all states in one set)
    - Refine it until a reachable fair cycle is found or none is proved to exist.
Strongly Connected Components

- A counterexample to a progress property is a lasso-shaped path that satisfies fairness constraints.
- A lasso’s cycle is contained in a strongly connected component (SCC) of the state graph.
- A nonempty set of states is SCC-closed if every SCC is either contained in it or disjoint from it.
- A partition of the states into SCC-closed sets is a coarser partition than the SCC partition; hence, . . .
- Every cycle of a graph is contained in some SCC-closed set.
- Maintain a partition of the states into SCC-closed set:
  - Start with the trivial partition (all states in one set).
  - Refine it until a reachable fair cycle is found or none is proved to exist.
FAIR: Finding Reachable Fair Cycles

Reduce search for reachable fair cycle to a set of safety problems:

- **Skeleton:**
  
  States of skeleton together satisfy all fairness constraints.

- **Task:** Connect states to form lasso.
Reach Queries

Each connection task is a reach query.

- **Stem query**: Connect initial condition to a state:

![Stem query diagram]

- **Cycle query**: Connect one state to another:

![Cycle query diagram]

(To itself if skeleton has only one state.)
Witness to Nonemptiness

If all queries are answered positively:

Witness to nonemptiness of $C$. 
Global Reachability

If a stem query is answered negatively: new **inductive** global reachability information.

- Constrains subsequent selection of skeletons.
- Constrains subsequent reach (stem and cycle) queries.
- Improve proof by strengthening (using ideas from IC3).
Barriers: Discovering SCC-Closed Sets

If a cycle query is answered negatively: new information about SCC structure of state graph.

- **Inductive** proof: “one-way barrier”
- Each “side” of the proof is SCC-closed.
- Constrains subsequent selections of skeletons: all states on one side.
Example: Empty Language

```
000 001
101100
010
110
011
111
100 101
```

Diagonal arrows indicate branching time.
Example: Empty Language

\[
\begin{array}{c}
s_{0} \quad 010 \\
s_{1} \quad 110
\end{array}
\]
Example: Empty Language

\[ s_0 \quad s_1 \]

sk1 \quad 010 \quad 110

\[
\begin{array}{c}
000 \\
001 \\
010 \\
011 \\
100 \\
101 \\
110 \\
111
\end{array}
\]

stem query produces \( x_1 \lor \neg x_2 \)
Example: Empty Language

```
000 001
101 100
110 111
```

```
010
```

```
011
```

```
011
```

```
111
```

```
110
```

```
101
```

```
100
```

```
001
```

```
000
```

```
010
```

```
011
```

```
110
```

```
111
```

```
101
```

```
100
```

```
001
```

```
000
```

```
010
```

```
011
```
Example: Empty Language

\[ s_0 \quad s_1 \]

sk2 101 110

states satisfy

\[ x_1 \lor \neg x_2 \]
Example: Empty Language

\[ s_0 \quad s_1 \]

\[
\begin{array}{c}
\text{sk2} \\
\text{101} \\
\text{110}
\end{array}
\]

states satisfy

\[ x_1 \lor \neg x_2 \]

stem query passes
Example: Empty Language

\[
s_0 \quad s_1
\]

states satisfy
\[
x_1 \lor \neg x_2
\]

\[\text{reach}(S, (x_1 \lor \neg x_2), s_0, s_1) \text{ passes}\]
Example: Empty Language

\[ s_0 \quad s_1 \]

\[ sk2 \quad 101 \quad 110 \]

states satisfy \( x_1 \lor \lnot x_2 \)

\[ \text{reach}(S, (x_1 \lor \lnot x_2), s_1, s_0) \text{ produces } x_2 \]
Example: Empty Language

\[ s_0 \quad s_1 \]
\[ sk2 \quad 101 \quad 110 \]

states satisfy
\[ x_1 \lor \neg x_2 \]

because \[ x_1 \land x_2 \land \neg x_3 \Rightarrow x_2 \ldots \]
Example: Empty Language

\[ s_0 \quad s_1 \]
\[ sk2 \quad 101 \quad 110 \]

states satisfy
\[ x_1 \lor \neg x_2 \]

and \[ x_2 \land (x_1 \lor \neg x_2) \land T \Rightarrow x'_2 \]
Example: Empty Language

```
000 001
101 100
110 111
```
Example: Empty Language

\[ s_0, s_1 \]

\begin{align*}
\text{states satisfy} & \\
(x_1 \lor \neg x_2) \land \neg x_2
\end{align*}
Example: Empty Language

 states satisfy
 \((x_1 \lor \neg x_2) \land \neg x_2\)

stem query passes
Example: Empty Language

states satisfy
\((x_1 \lor \neg x_2) \land \neg x_2\)

reach\((S, (x_1 \lor \neg x_2) \land \neg x_2 \land \neg x_2', s_0, s_1))\) produces \(x_3\)

\((\neg x_2 \land \neg x_2'\) can be simplified to \(\neg x_2'\))
Example: Empty Language

no skeletons left
Example: Single-State Skeleton

![Diagram of a single-state skeleton](image)
Example: Single-State Skeleton

$\text{sk1} \quad 101 \quad 101$

$S_0 = S_1$

Diagram:

States: 000, 001, 010, 011, 100, 101, 110, 111

Transitions:
- 000 → 001
- 001 → 010
- 010 → 011
- 011 → 100
- 100 → 101
- 101 → 110
- 110 → 111

$S_0 = S_1$
Example: Single-State Skeleton

\[ s_0 = s_1 \]

stem query passes
Example: Single-State Skeleton

\[ s_0 = s_1 \]

\[ \text{reach}(S, \top, \text{post}(S, s_0), s_0) \text{ produces } x_1 \wedge x_2 \]
\[ \text{and } (\neg x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_3) \]
Example: Single-State Skeleton
Example: Single-State Skeleton

\[
\begin{align*}
& s_0 & s_1 \\
& sk2 & 001 & 100 \\
\text{states satisfy} & \left( \neg x_1 \lor \neg x_2 \right) \land \\
& \left( \neg x_1 \lor \neg x_3 \right)
\end{align*}
\]
Example: Single-State Skeleton

\[ s_0 \quad s_1 \]

sk2 001 100

states satisfy
\[
(\neg x_1 \lor \neg x_2) \land \\
(\neg x_1 \lor \neg x_3)
\]

stem query passes
Example: Single-State Skeleton

\[ s_0 \quad s_1 \]

\[ \text{sk2} \quad 001 \quad 100 \]

States satisfy

\[ (\neg x_1 \lor \neg x_2) \land (\neg x_1 \lor \neg x_3) \]

reach\( (S, (\neg x_1' \lor \neg x_2') \land (\neg x_1' \lor \neg x_3'), s_0, s_1) \) produces \( x_2 \lor x_3 \)
Example: Single-State Skeleton
Example: Single-State Skeleton

\[ s_0 \quad s_1 \]

\[ \text{sk3} \quad 001 \quad 011 \]

states satisfy

\[ (\neg x_1 \lor \neg x_2) \land \]

\[ (\neg x_1 \lor \neg x_3) \land \]

\[ (x_2 \lor x_3) \]
Example: Single-State Skeleton

\[
s_0 \quad s_1
\]

\[
\text{sk3 001 011}
\]

states satisfy
\[
(\neg x_1 \lor \neg x_2) \land \\
(\neg x_1 \lor \neg x_3) \land \\
(x_2 \lor x_3)
\]

stem query passes
Example: Single-State Skeleton

\[ s_0 \quad s_1 \]

\[ sk3 \quad 001 \quad 011 \]

states satisfy

\[ \neg x_1 \lor \neg x_2 \land \neg x_1 \lor \neg x_3 \land x_2 \lor x_3 \]

reach\((S, (\neg x'_1 \lor \neg x'_2) \land (\neg x'_1 \lor \neg x'_3) \land (x_2 \lor x_3), s_0, s_1)\) produces

\[ \neg x_2 \]
Example: Single-State Skeleton
Example: Single-State Skeleton

\[ s_0 \quad s_1 \]
\[ \text{sk4} \quad 010 \quad 011 \]

states satisfy
\[ (\neg x_1 \lor \neg x_2) \land (\neg x_1 \lor \neg x_3) \land (x_2 \lor x_3) \land x_2 \]

\[ \begin{array}{c}
000 \quad 001 \\
010 \quad 011 \\
100 \quad 101 \\
110 \quad 111 \\
\end{array} \]
Example: Single-State Skeleton

$S_0$  $S_1$
sk4  010  011

states satisfy
$(\neg x_1 \lor \neg x_2) \land$
$(\neg x_1 \lor \neg x_3) \land$
$(x_2 \lor x_3) \land x_2$

stem query produces $x_1 \lor \neg x_2$
Example: Single-State Skeleton

no skeletons left
Key Insights

- Inductive assertions describe SCC-closed sets.
- Arena: Set of states all on the same side of each barrier.
- Unlike previous symbolic methods:
  
  Barrier constraints on the transition relation combined with the over-approximating nature of IC3 enable the simultaneous (symbolic) consideration of all arenas.

- A proof can provide information about many arenas even though the motivating skeleton comes from one arena.
Outline

1. Proving Invariants by Induction
   - Induction for Transition Systems
   - Strengthening
   - Relative Induction

2. Proving Safety Properties with IC3
   - Basic Algorithm
   - Examples
   - Efficiency

3. Proving Progress Properties
   - FAIR
   - Examples

4. Proving Branching Time Properties
   - IICTL
   - Example
IICTL: Incremental Inductive CTL Model Checking

- Task-directed strategy (local model checking)
- Maintains upper and lower bounds on states satisfying each subformula
- States in between the bounds are undecided
- Typically don’t need to decide all states to decide the property (Traditional symbolic CTL algorithms do)
- Decide states by executing appropriate query:
  - EX: SAT query
  - EU: Safety model checker (e.g., IC3)
  - EG: Fair cycle finder (e.g., FAIR)
- Generalizing decisions (proofs or counterexamples) to other states and refining the bounds
Example: Resetability

\[
\begin{align*}
S &\models \varphi \text{ because } \neg x_1 \land \neg x_2 \land \neg x_3 \Rightarrow x_1 \lor \neg x_2 \\
\end{align*}
\]
Example: Resetability

Initialize bounds ignoring transitions

\begin{align*}
U_0 &= \top \\
L_0 &= \bot \\
U_1 &= \top \\
L_1 &= \bot \\
U_2 &= \neg x_1 \lor \neg x_2 \lor x_3 \\
L_2 &= \bot \\
U_3 &= \top \\
L_3 &= x_1 \land x_2 \land \neg x_3 \\
U_4 &= x_1 \land x_2 \land \neg x_3 \\
L_4 &= x_1 \land x_2 \land \neg x_3 
\end{align*}
Example: Resetability

State \( \neg x_1 \land \neg x_1 \land \neg x_3 \) (000) is undecided for node 0 (and for node 1 too)
Example: Resetability

Can \( \neg x_1 \land \neg x_2 \land \neg x_3 \) reach \( L_2 \)? No
Example: Resetability

Can \( \neg x_1 \land \neg x_2 \land \neg x_3 \) reach \( U_2 \)? Yes, it can reach itself
Example: Resetability

\[-x_1 \land \neg x_2 \land \neg x_3 \text{ is undecided for nodes 2 and 3}\]
Example: Resetability

Can $\neg x_1 \land \neg x_2 \land \neg x_3$ reach $L_4 = U_4$? Yes
Example: Resetability

Which other states? \( \neg x_2 \).

Update \( L_3 \) and \( U_2 \)

---

**Example: Resetability**

- **States:**
  - 000
  - 001
  - 010
  - 011
  - 100
  - 101
  - 110
  - 111

- **Transitions:**
  - \( U_2 \)

- **Formulas:**
  - \( L_0 = \bot \)
  - \( U_0 = T \)
  - \( L_1 = \bot \)
  - \( U_1 = T \)
  - \( L_2 = \bot \)
  - \( U_2 = \neg x_1 \lor \neg x_2 \lor x_3 \)
  - \( L_3 = x_1 \land x_2 \land \neg x_3 \)
  - \( U_3 = T \)
  - \( L_4 = x_1 \land x_2 \land \neg x_3 \)
  - \( U_4 = x_1 \land x_2 \land \neg x_3 \)

**Notes:**
- \( U_0 = T \) and \( U_1 = T \) indicate the start states.
- \( U_2 \) and \( U_3 \) are update formulas for transitions.
- \( L_0 \) and \( L_1 \) are the initial states.
- \( L_2 \) and \( L_3 \) are the target states.

**Diagram:**
- The diagram represents the transition system with states and transitions defined by the formulas above.

**Questions:**
- Which other states?
- Update \( L_3 \) and \( U_2 \)
Example: Resetability

$U_2$ has changed. Can $\neg x_1 \land \neg x_2 \land \neg x_3$ still reach it?
Yes, it can reach $x_1 \land x_2 \land x_3$
Example: Resetability

Can \( x_1 \land x_2 \land x_3 \) reach \( L_4 = U_4 \)? Yes
Example: Resetability

Which other states? $x_1$.
Update $L_3$ and $U_2$
Example: Resetability

Can $\neg x_1 \land \neg x_2 \land \neg x_3$ still reach $U_2$?

No. Inductive proof is $x_1 \lor \neg x_2$. Update $U_1$ and $L_0$. 
Example: Resetability

The initial state is now decided. $S \models \varphi$
Generalization in IICTL

- Generalization required of both \textit{proofs} and counterexample \textit{traces}.
Generalization in IICCTL

- Generalization required of both proofs and counterexample traces

Is 110 is reachable from 000?
Generalization in IICTL

- Generalization required of both proofs and counterexample traces

IC3 returns a trace complete with input values
Generalization in IICTL

- Generalization required of both proofs and counterexample traces

On input 1, state 001 also reaches 101
Add it to the “trace”
Generalization in ICTL

- Generalization required of both proofs and counterexample traces

Need to existentially quantify over inputs
Generalization in IICTL

- Generalization required of both proofs and counterexample traces

Technique leverages UNSAT cores
### Incremental Inductive Verification

<table>
<thead>
<tr>
<th></th>
<th>IC3</th>
<th>FAIR</th>
<th>IICTL</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Information:</strong></td>
<td>Over-approx. Sets</td>
<td>SCC-closed arenas</td>
<td>Over- &amp; under-approx. sets</td>
</tr>
<tr>
<td><strong>Objective:</strong></td>
<td>Inductive strengthening</td>
<td>All arenas skeleton-free</td>
<td>All init. states in underapprox.</td>
</tr>
<tr>
<td><strong>Seed:</strong></td>
<td>CTI</td>
<td>Skeleton</td>
<td>Task state</td>
</tr>
<tr>
<td><strong>Lemma:</strong></td>
<td>Inductive clause</td>
<td>Global reach. proof &amp; one-way barrier</td>
<td>Refinement of approximations</td>
</tr>
<tr>
<td><strong>Incremental:</strong></td>
<td>Relative to previously discovered lemmas.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
• One of many existing (industrial and academic) implementations
• Implements IC3 and FAIR (IICTL in next release)
• http://iimc.colorado.edu
Bibliography I


- A. R. Bradley, “SAT-based model checking without unrolling,” in *Verification, Model Checking, and Abstract Interpretation (VMCAI’11)*, Austin, TX, 2011, pp. 70–87, LNCS 6538.


Bibliography II


Bibliography III

Bibliography IV

