SAT-Based Verification with IC3: Foundations and Demands

Aaron R. Bradley

ECEE, CU Boulder & Summit Middle School
Induction

Foundation of verification for 40+ years (Floyd, Hoare)

To prove that $S : (I, T)$ has safety property $P$, prove:

- Base case (initiation):
  \[ I \Rightarrow P \]

- Inductive case (consecution):
  \[ P \land T \Rightarrow P' \]
P is inductive
We present two solutions. . .
1. Use a stronger assertion, or
2. Construct an incremental proof, using previously established invariants.

– Manna and Pnueli

*Temporal Verification of Reactive Systems: Safety*

1995

Method 1 = “Monolithic”
Method 2 = “Incremental”
Outline

1. Illustration of the two methods
2. SAT-based model checkers
3. Understanding IC3
4. After IC3: Temporal Logics, SMT
5. Challenges for SAT/SMT
Two Transition Systems

$S_1: \begin{align*}
x, y &:= 1, 1 \\
\textbf{while} \ &*: \\
x, y &:= x + 1, y + x
\end{align*}$

$S_2: \begin{align*}
x, y &:= 1, 1 \\
\textbf{while} \ &*: \\
x, y &:= x + y, y + x
\end{align*}$

$P: y \geq 1$
Induction on System 1

$S_1$:

\[
\begin{align*}
\text{while } \ast : & \\
\text{x, y := x + 1, y + x}
\end{align*}
\]

- **Initiation:**

\[x = 1 \land y = 1 \implies y \geq 1\]

initial condition

- **Consecution (fails):**

\[y \geq 1 \land x' = x + 1 \land y' = y + x \not\implies y' \geq 1\]

transition relation
**Incremental Proof**

\[ S_1: \]

\[
\begin{align*}
\text{x, y := 1, 1} \\
\text{while } *: \\
\text{x, y := x + 1, y + x}
\end{align*}
\]

Problem: \( y \) decreases if \( x \) is negative. But...

\( \varphi_1: x \geq 0 \)

- **Initiation:**
  \[
  x = 1 \land y = 1 \implies x \geq 0
  \]

- **Consecution:**
  \[
  x \geq 0 \land x' = x + 1 \land y' = y + x \implies x' \geq 0
  \]

**transition relation**
Back to $P$

\[
\begin{align*}
S_1: & \quad x, y := 1, 1 \\
& \textbf{while} \quad \ast:\ 
\begin{array}{l}
x, y := x + 1, y + x
\end{array}
\end{align*}
\]

Consecution:

\[
x \geq 0 \land y \geq 1 \land x' = x + 1 \land y' = y + x \Rightarrow y' \geq 1
\]

$P$ is inductive \textbf{relative to} $\varphi_1$. 

SAT-Based Verification with IC3: Foundations and Demands – 9/55
Induction on System 2

\[
S_2:
\begin{align*}
  & x, \ y := 1, \ 1 \\
  \text{while } & *:\ \\
  & x, \ y := x + y, \ y + x
\end{align*}
\]

Induction fails for \( P \) as in System 1. Additionally,

\[
x \geq 0 \land x' = x + y \land y' = y + x \nRightarrow x' \geq 0
\]

\( x \geq 0 \) is not inductive, either.
Monolithic Proof

\[ S_2: \]
\[
x, y := 1, 1
\]
while *:
\[
x, y := x + y, y + x
\]

Invent strengthening all at once:

\[ \hat{P}: \ x \geq 0 \land y \geq 1 \]

Consecution:

\[ \begin{align*}
\hat{P} & : \ x \geq 0 \land y \geq 1 \\
\land x' = x + y \land y' = y + x & \Rightarrow x' \geq 0 \land y' \geq 1
\end{align*} \]
Incremental vs. Monolithic Methods

- Incremental: does not always work
- Monolithic: relatively complete
- Incremental: apply induction iteratively ("modular")
- Monolithic: invent one strengthening formula

We strongly recommend its use whenever applicable. Its main advantage is that of modularity.

– Manna and Pnueli

*Temporal Verification of Reactive Systems: Safety*

1995
Finite-state System

Transition system:

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

Cube \( s \):

- Conjunction of literals, e.g.,

\[ x_1 \land \neg x_2 \land \neg x_3 \land x_4 \land \cdots \]

- Like any formula, represents set of states (that satisfy it)

Clause: \( \neg s \)
SAT-Based Backward Model Checking:

1. Search for predecessor $s$ to some error state:

$$P \land T \Rightarrow P'$$

If none, property holds.

2. Reduce cube $s$ to $\overline{s}$:
   - Expand to others with bad successors [McMillan 2002], [Lu et al. 2005]
   - If $P \land \neg s \land T \Rightarrow \neg s'$, reduce by implication graph [Lu et al. 2005]
   - Apply inductive generalization [Bradley 2007]

3. $P := P \land \neg \overline{s}$
Inductive Generalization

Given: cube $s$
Find: $c \subseteq \neg s$ such that

- Initiation:
  \[ I \Rightarrow c \]

- Consecution (relative to information $P$):
  \[ P \land c \land T \Rightarrow c' \]

- No strict subclause of $c$ is inductive relative to $P$
Analysis of Backward Search

Strengths:

• Easy SAT queries, low memory
• Property focused
• Some are approximating, computing neither strongest nor weakest strengthening

Weaknesses:

• Essentially undirected search (bad for bug finding)
• Ignore initial states
Analysis of FSIS [Bradley 2007]

Strengths (essentially, great when it works):

- Can significantly reduce backward search
- Can find strong lemmas with induction

Weaknesses:

- Like others when inductive generalization fails
Compared to backward search:

- Considers initial and final states
- Requires solving hard SAT queries
- Practically incomplete (UNSAT case)

\[ I \land \bigwedge_{i=0}^{k-1} (P(i) \land T(i)) \land \neg P(k) \]
**$\kappa$-Induction** [Sheeran et al. 2000]

 Addresses practical incompleteness of BMC:

- **Initiation:** BMC
- **Consecution:**

\[
\bigwedge_{i=0}^{k-1} (P(i) \land T(i)) \implies P^{(k)}
\]

(plus extra constraints to consider loop-free paths)
k-Induction

Longest loop-free path

\[ k = 6 \]
Property-focused over-approximating post-image:

\[ F_i \land \bigwedge_{i=0}^{k-1} (P^{(i)} \land T^{(i)}) \Rightarrow P^{(k)} \]

- \{\text{states } \leq i \text{ steps from initial states}\} \subseteq F_i
- If holds, finds interpolant \( F_{i+1} \):

\[ F_i \land T \Rightarrow F'_{i+1} \land \bigwedge_{i=1}^{k-1} (P^{(i)} \land T^{(i)}) \Rightarrow P^{(k)} \]

- If fails, increases \( k \)
BMC $\rightarrow \kappa$-Induction $\rightarrow$ ITP

- Completeness from unrolling transition relation
- Evolution: reduce max $\kappa$ in practice (UNSAT case)
- Monolithic:
  - hard SAT queries
  - induction at top-level only
- Consider both initial and final states
Best of Both?

Desire:

• Stable behavior (backward search)
  • Low memory, reasonable queries
  • Can just let it run
• Consideration of initial and final states (BMC)
• Modular reasoning (incremental method)

Avoid:

• Blind search (backward search)
• Queries that overwhelm the SAT solver (BMC)
Stepwise sets $F_0, F_1, \ldots, F_k, F_{k+1}$ (CNF):

- $\{\text{states } \leq i \text{ steps from initial states}\} \subseteq F_i$
- $F_i \subseteq \{\text{states } \geq k - i + 1 \text{ steps from error}\}$

Four invariants:

- $F_0 = I$
- $F_i \Rightarrow F_{i+1}$
- $F_i \land T \Rightarrow F'_{i+1}$
- Except $F_{k+1}$, $F_i \Rightarrow P$

∴ if ever $F_i = F_{i+1}$, $F_i$ is inductive & $P$ is invariant
Essence of IC3

- Continual refinement of over-approximating stepwise sets
  - Until one is inductive
  - Monolithic use of induction

- Generation of clauses as response to backward reachable states
  - Inductive generalization: \( c \subseteq \neg s \)
    \((c \text{ is inductive relative to a stepwise set})
  - Incremental use of induction
Two Views of IC3

- **Prover**: Generates predicates from counterexamples
  - From $s$: state that can reach error
  - To $c \subseteq \neg s$: inductive relative to $F_i$
  - $c$ proves that $s$ is unreachable in $\leq i + 1$ steps
- **Bug finder**: Guided backward search
  - Stepwise sets: proximity estimate to initial state
Induction at Top Level

Is $P$ inductive relative to $F_k$?

$$F_k \land T \Rightarrow P'$$

(Recall: $F_k \Rightarrow P$)

- Possibility #1: Yes
- Conclusion: $P$ is inductive relative to $F_k$
\[ F_k(\land P) \land T \Rightarrow P' \]
Induction at Top Level

Monolithic behavior (predicate abstraction):

• For $i$ from 1 to $k$: find largest $C \subseteq F_i$ s.t.

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

$$F_{i+1} := F_{i+1} \land C$$

• $F_{k+1} := F_{k+1} \land P$

• New frontier: $F_{k+1}$

If ever $F_i = F_{i+1}$, done: $P$ is invariant.
Counterexample To Induction (CTI):

\[ F_k \land T \Rightarrow P' \]

- Possibility #2: No
- Conclusion: \( \exists F_k\)-state \( s \) with error successor
- If \( s \) is an initial state, done: \( P \) is not invariant
- Otherwise...
Induction at Low Level

Inductive Generalization in IC3

• **Given**: cube \( s \)

• **Find**: \( c \subseteq \neg s \) such that
  
  • Initiation:
    
    \[ I \Rightarrow c \]
  
  • Consecution (relative to \( F_i \)):
    
    \[ F_i \land c \land T \Rightarrow c' \]

• No strict subclause of \( c \) is inductive relative to \( F_i \)
Inductive Generalization
Addressing CTI $s$

- Find highest $i$ such that

$$F_i \land \neg s \land T \Rightarrow \neg s'$$

- Apply inductive generalization:

$$c \subseteq \neg s \quad I \Rightarrow c \quad F_i \land c \land T \Rightarrow c'$$

- $\therefore F_{i+1} := F_{i+1} \land c$ (also update $F_j, j \leq i$)

- If $i < k$, new proof obligation:

$$(s, i + 1)$$

“Inductively generalize $s$ relative to $F_{i+1}$”
Addressing Proof Obligation \((t, j)\):

SAT query:

\[ F_j \land \neg t \land T \Rightarrow \neg t' \]

If UNSAT:

- Inductive generalization must succeed:
  
  \[ c \subseteq \neg t \quad I \Rightarrow c \quad F_j \land c \land T \Rightarrow c' \]

- \( F_{j+1} := F_{j+1} \land c \)

- Updated proof obligation (if \( j < k \)): \((t, j + 1)\)
Addressing Proof Obligation \((t, j)\) :

SAT query:

\[ F_j \land \neg t \land T \Rightarrow \neg t' \]

If SAT: New CTI \(u\), treat as before
One of IC3’s Insights

Identification of relevant predecessors:

- Why did inductive generalization of \( s \) succeed relative to \( F_i \) but fail relative to \( F_{i+1} \)?
- Because of some \( F_{i+1} \)-state \( s \)-predecessor \( t \).
- Analysis at \( F_i \) focuses IC3’s choice of predecessors at \( F_{i+1} \).
IC3: A Prover

- Based on CTIs ($s$), IC3 generates $F_i$-relative inductive clauses ($c \subseteq \neg s$) to refine $F_i$’s.
- IC3 propagates clauses to prepare new frontier.
  - Some clauses may be too specific.
  - Their loss can break mutual support.
- As the frontier advances, IC3 considers ever more general situations.
- It eventually finds the real reasons (as truly inductive clauses) that $P$ is invariant.
Suppose:

- $u \rightarrow t \rightarrow s \rightarrow \text{Error}$
- Proof obligations:

$$\{(s, k - 1), (t, k - 2), (u, k - 1)\}$$

That is,

- $s$ must be inductively generalize relative to $F_{k-1}$
- $t$ must be inductively generalize relative to $F_{k-2}$
- $u$ must be inductively generalize relative to $F_{k-1}$

Which proof obligation should IC3 address next?
Guided Search

Two observations:

- \( u \) is the “deepest” of the states

\[ u \rightarrow t \rightarrow s \rightarrow \text{Error} \]

- \( t \) is the state that IC3 considers as likeliest to be closest to an initial state.

\[ \{(s, k - 1), (t, k - 2), (u, k - 1)\} \]

“Proximity metric”

Conclusion: Pursue \((t, k - 2)\) next.

(It also happens to be the correct choice [Bradley 2011].)
\{(s, k-1), (t, k-2), (u, k-1)\}

Proof Obligations: Guided Search
Incremental, Inductive Verification

IIV Algorithm:

• Constructs concrete hypotheses
• Generates intermediate lemmas incrementally
• Applies induction many times
• Generalizes from hypotheses to strong lemmas
After IC3: Refinements

- New heuristic: ternary simulation cube reduction
  [Een et al., FMCAD’11]
- Industrial setting: incremental verification
  [Chockler et al., FMCAD’11]

Oh, yeah, and a name change: PDR
(Thanks, Niklas!)
PDR: Temporal Logics

- **FAIR** [Bradley et al., FMCAD’11]
  - For $\omega$-regular properties, e.g., LTL
  - Insight: SCC-closed regions can be characterized inductively

- **IICTL** [Hassan et al., CAV’12]
  - For CTL properties
  - Insight: EX (SAT), EU (IC3), EG (FAIR)
  - Standard traversal of CTL property’s parse tree
    - Over- and under-approximating sets
    - Task state-driven refinement
PDR: Infinite-state Systems

- SMT-based Induction Methods for Timed Systems [Kindermann et al., arXiv’12]

- Generalized Property Directed Reachability [Hoder et al., SAT’12]
  - Boolean push-down systems
  - Linear real arithmetic

- Software Model Checking via IC3 [Cimatti et al., CAV’12]
  - Explicit handling of CFG
  - Applies IC3 techniques to McMillan’s “Lazy Abstraction with Interpolants” [McMillan, CAV’06]
Some presentations use LIFO ordering:

- Trivial correctness; easier to understand
- [Hoder et al., SAT’12], [Cimatti et al., CAV’12]
- Downside: not quite as good?
  - PSPACE-complete (finite-state), so...
  - But: fixed-length counterexamples for $K$
  - And: not aggressive about mutual induction
Challenges for SAT/SMT

1. Emphasizes incremental calls (100s-1000s/sec) (FAIR/IICLTL: even pushes/pops sets of clauses)
2. Understand effect of solver’s choices on PDR
3. Variable ordering:
   • Vital to practical performance
   • Direct core assumptions and lifting
4. IIV: proofs pushed to block (e.g., FAIR)
   • Solver should report whether proof is “useful”
5. Multi-threaded access over core constraints
Conclusions

• Attempted to explain why IC3 works:
  • As a compromise between the **incremental** and **monolithic** strategies
  • Characteristics of previous SAT-based MC
  • As a prover
  • As a bug finder

• Subsequent work: temporal logic, SMT

• Challenges for SAT/SMT