IC3 and Beyond: Incremental, Inductive Verification

Aaron R. Bradley

ECEE, CU Boulder &

Summit Middle School

Induction

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

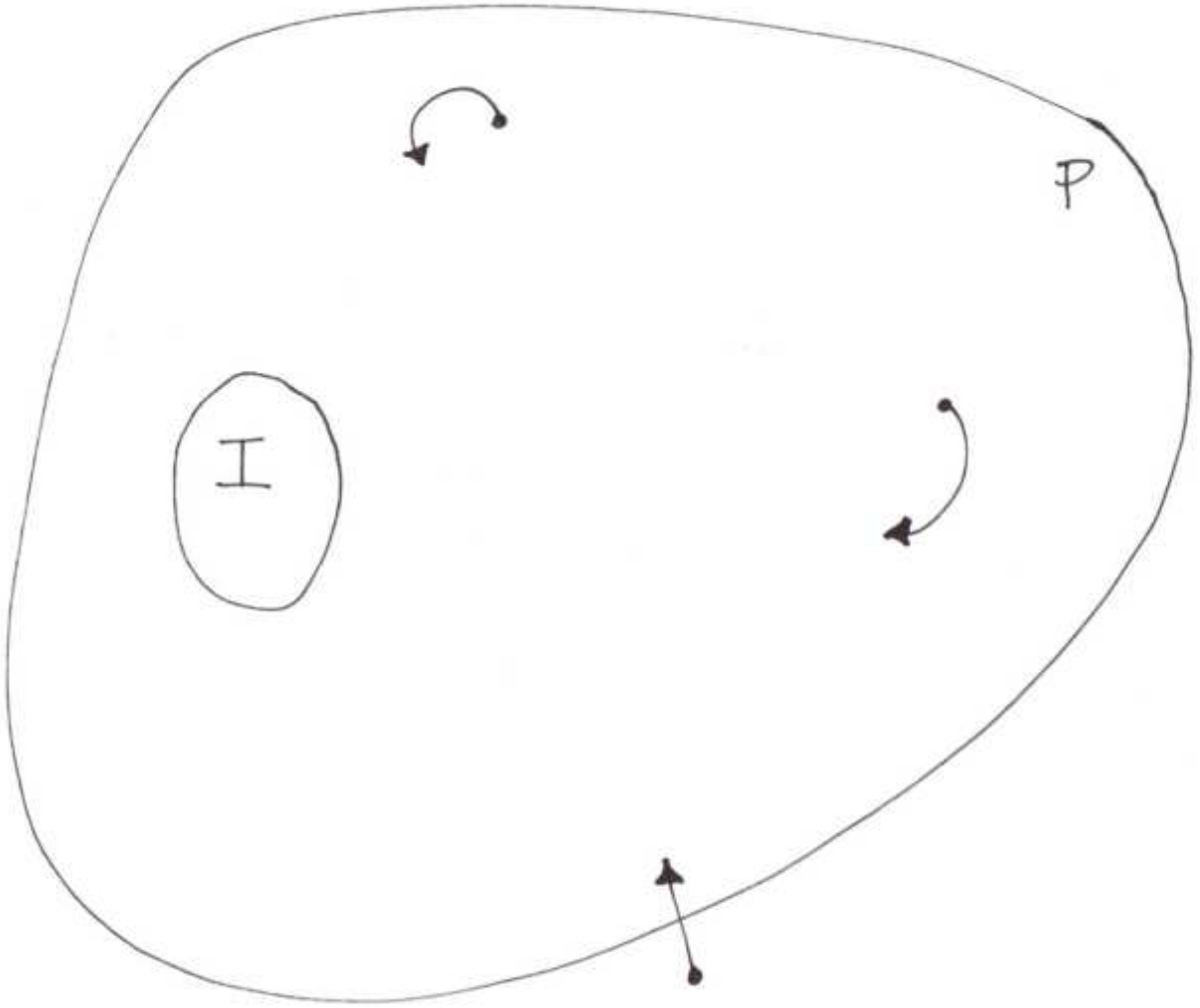
To prove that $S : (I, T)$ has safety property P, prove:

- Base case (**initiation**):

$$I \Rightarrow P$$

- Inductive case (**consecution**):

$$P \wedge T \Rightarrow P'$$

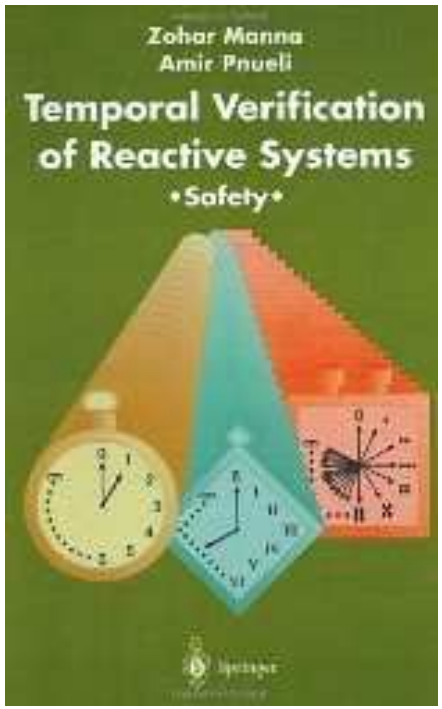


P is inductive

When Induction Fails

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 incremental vs. monolithic methods
2. SAT-based model checkers
3. Understanding IC3
4. FAIR: for ω-regular properties
5. Recent work

Two Transition Systems

S_1:

```
x, y := 1, 1                    1
while *:                        2
   x, y := x + 1, y + x         3
```

S_2:

```
x, y := 1, 1                    1
while *:                        2
   x, y := x + y, y + x         3
```

$$P: \; y \geq 1$$

Induction on System 1

S_1:

```
x, y := 1, 1                    1
while *:                        2
   x, y := x + 1, y + x         3
```

- Initiation:

$$\underbrace{x = 1 \wedge y = 1}_{\text{initial condition}} \Rightarrow \underbrace{y \geq 1}_{P}$$

- Consecution (fails):

$$\underbrace{y \geq 1}_{P} \wedge \underbrace{x' = x + 1 \wedge y' = y + x}_{\text{transition relation}} \not\Rightarrow \underbrace{y' \geq 1}_{P'}$$

Incremental Proof

S_1:

```
x, y := 1, 1                1
while *:                    2
   x, y := x + 1, y + x     3
```

Problem: y decreases if x is negative. But...

$\varphi_1 : \; x \geq 0$

- Initiation:

$$x = 1 \wedge y = 1 \Rightarrow x \geq 0$$

- Consecution:

$$\underbrace{x \geq 0}_{\varphi_1} \wedge \underbrace{x' = x + 1 \wedge y' = y + x}_{\text{transition relation}} \Rightarrow \underbrace{x' \geq 0}_{\varphi_1'}$$

Back to P

S_1:

```
x, y := 1, 1                    1
while *:                        2
    x, y := x + 1, y + x        3
```

Consecution:

$$\underbrace{x \geq 0}_{\varphi_1} \wedge \underbrace{y \geq 1}_{P} \wedge \underbrace{x' = x + 1 \wedge y' = y + x}_{\text{transition relation}} \Rightarrow \underbrace{y' \geq 1}_{P'}$$

P is inductive **relative to** φ_1.

Induction on System 2

S_2:

```
x, y := 1, 1                    1
while *:                        2
   x, y := x + y, y + x         3
```

Induction fails for P as in System 1.
Additionally,

$$x \geq 0 \wedge x' = x + y \wedge y' = y + x \not\Rightarrow x' \geq 0$$

$x \geq 0$ is not inductive, either.

Monolithic Proof

S_2:

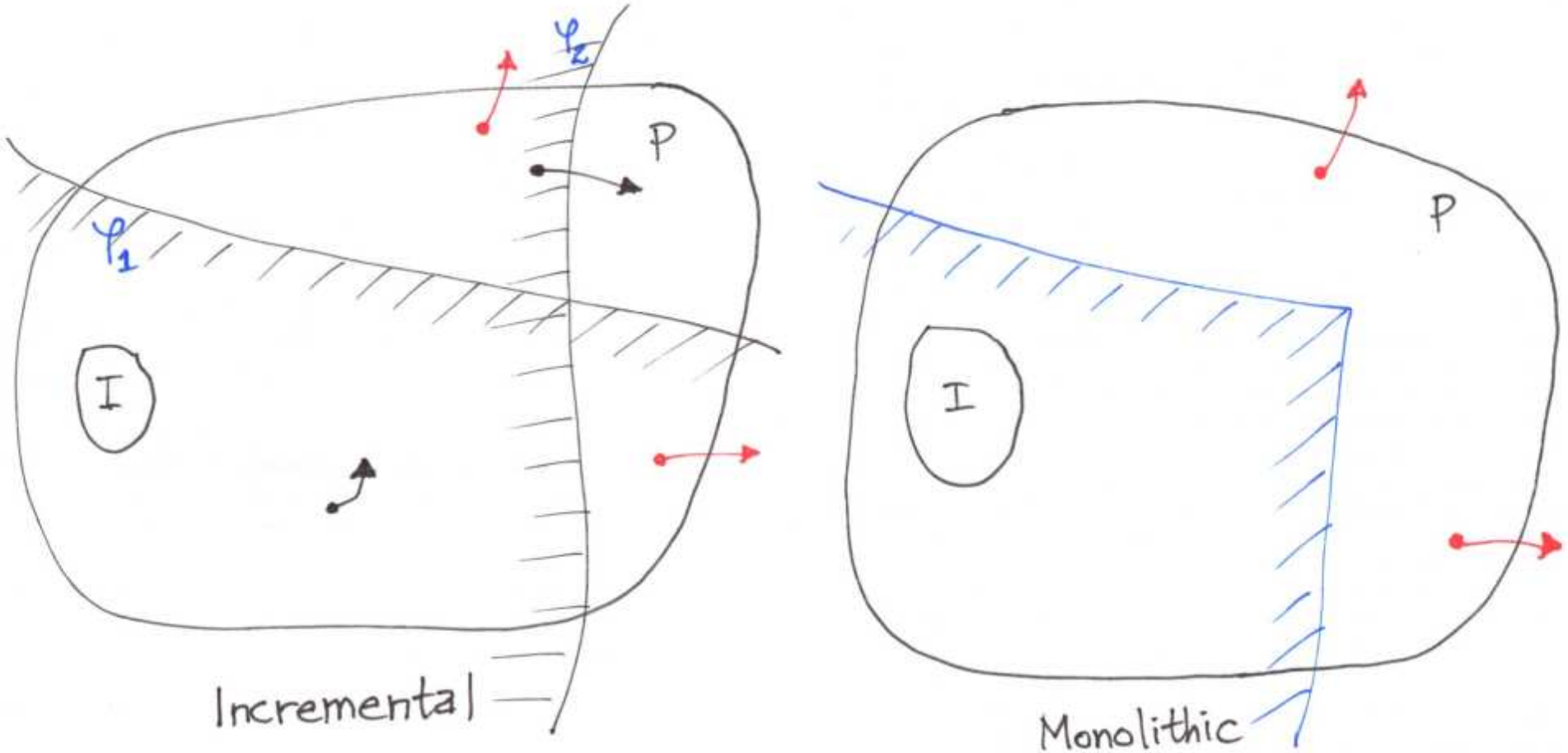
```
x, y := 1, 1                    1
while *:                        2
    x, y := x + y, y + x        3
```

Invent strengthening all at once:

$$\widehat{P}: \quad x \geq 0 \wedge y \geq 1$$

Consecution:

$$\underbrace{x \geq 0 \wedge y \geq 1}_{\widehat{P}} \wedge x' = x + y \wedge y' = y + x \Rightarrow \underbrace{x' \geq 0 \wedge y' \geq 1}_{\widehat{P}'}$$


φ2
P
φ1
I
Incremental
P
I
Monolithic

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:

$$S: \quad (\bar{i}, \bar{x}, I(\bar{x}), T(\bar{x}, \bar{i}, \bar{x}'))$$

Cube s:

- Conjunction of literals, e.g.,

$$x_1 \wedge \neg x_2 \wedge \neg x_3 \wedge x_4 \wedge \cdots$$

- 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:

$$P \wedge T \Rightarrow P'$$

   If none, property holds.

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

3. $P := P \wedge \neg \bar{s}$

Inductive Generalization

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

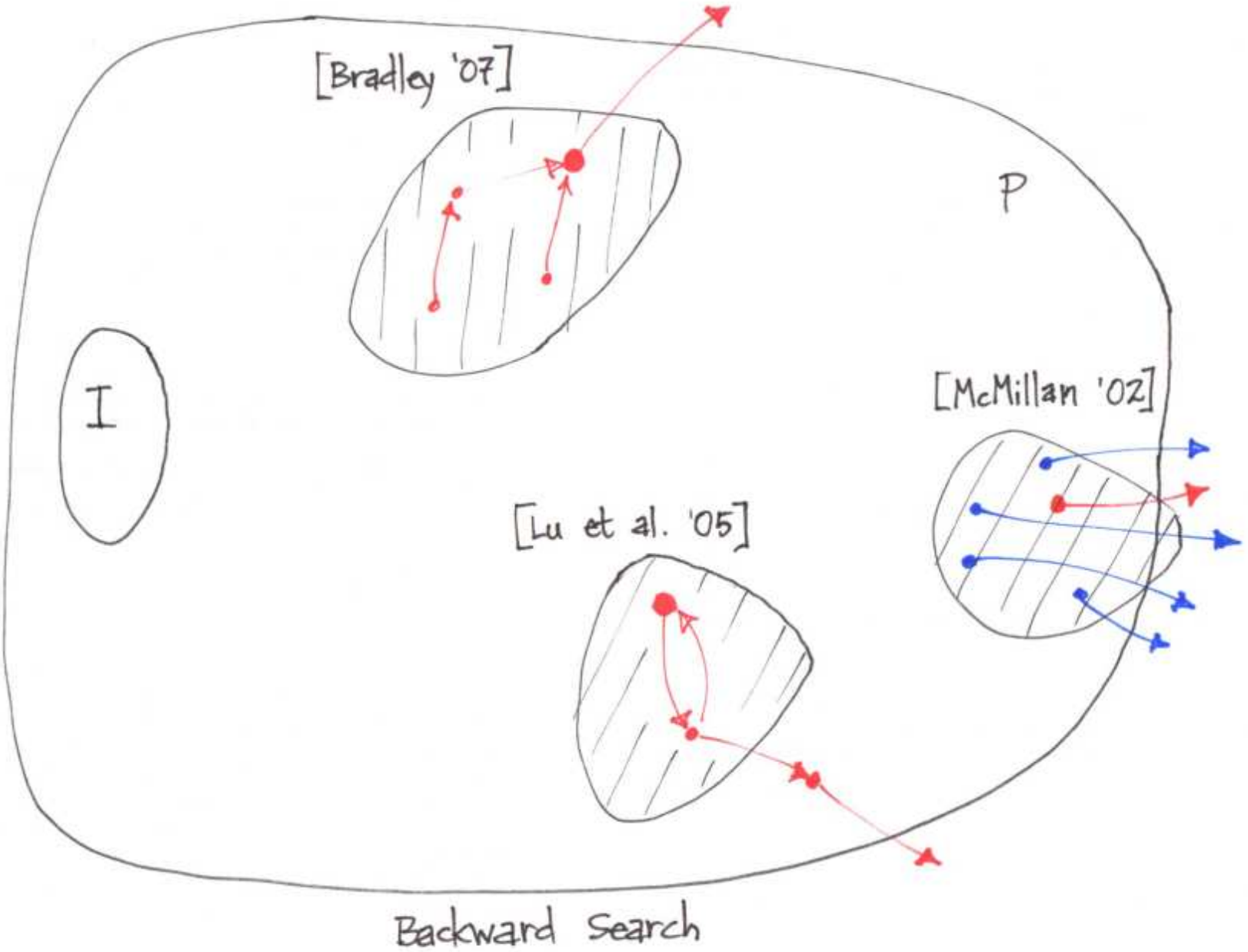
- Initiation:

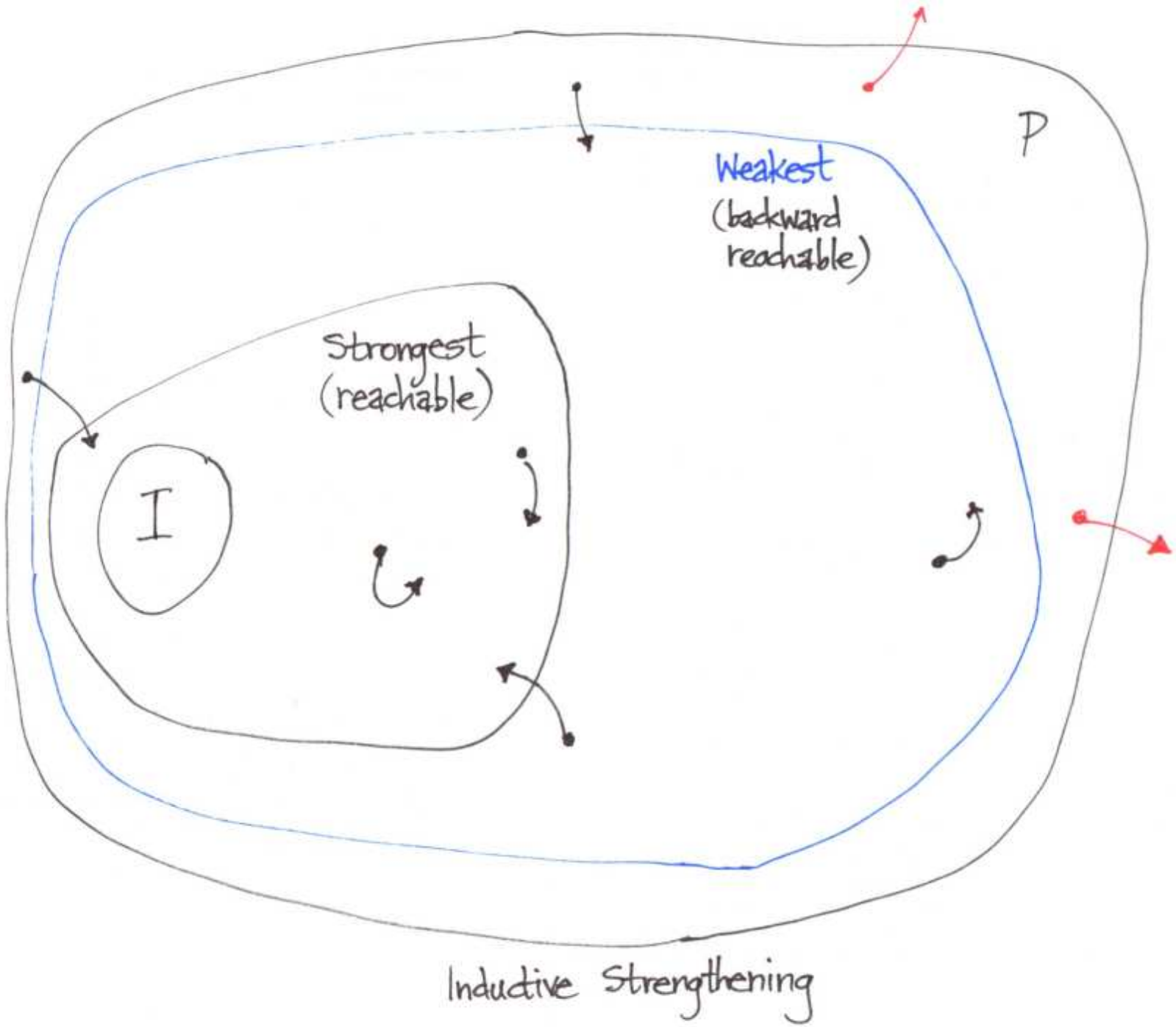
$$I \Rightarrow c$$

- Consecution (relative to information P):

$$P \wedge c \wedge T \Rightarrow c'$$

- No strict subclause of c is inductive relative to P


[Bradley '07]
P
I
[McMillan '02]
[Lu et al. '05]
Backward Search



Inductive Strengthening

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:

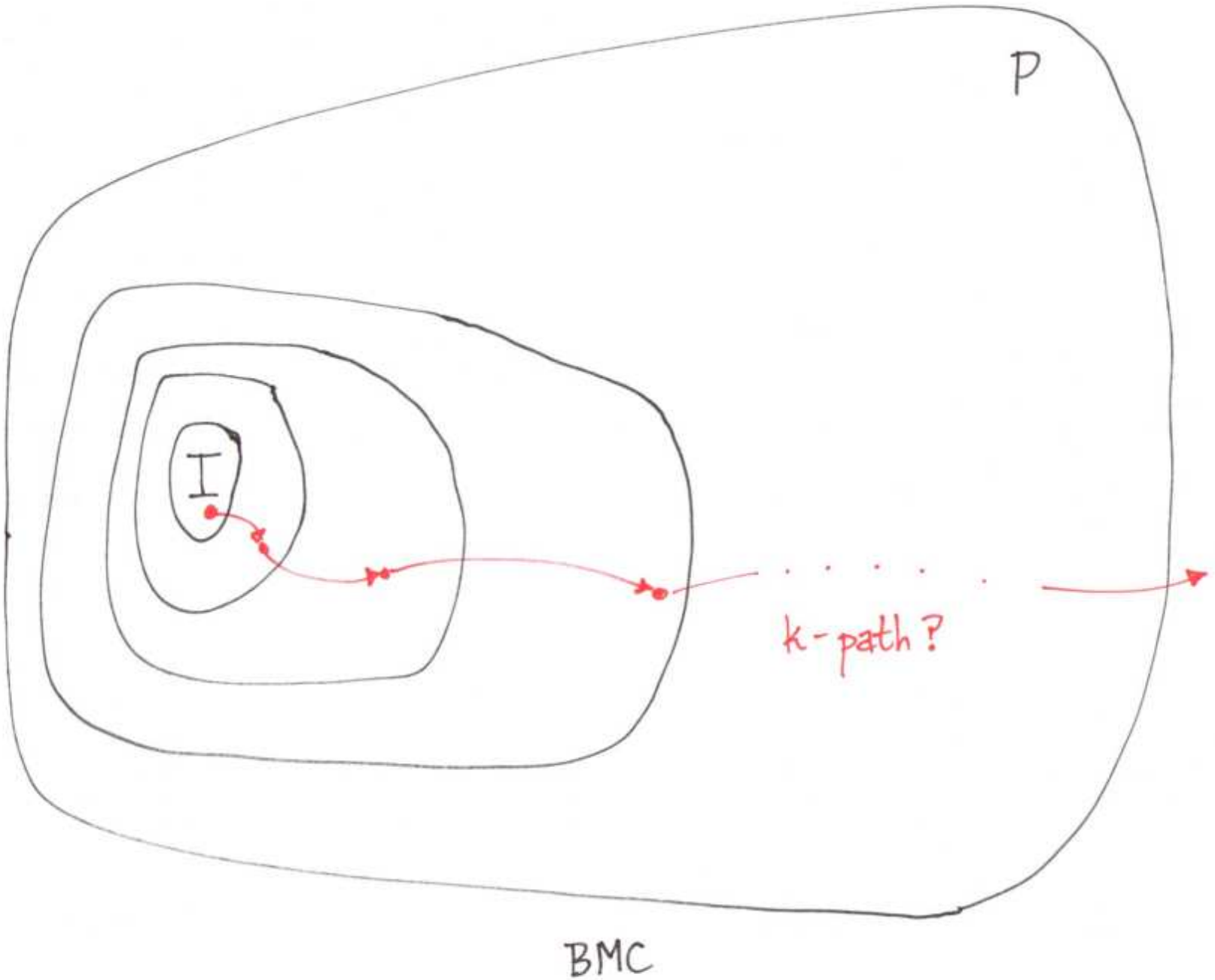
- Like others when inductive generalization fails

BMC [Biere et al. 1999]

Compared to backward search:

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

$$I \wedge \bigwedge_{i=0}^{k-1} (P^{(i)} \wedge T^{(i)}) \wedge \neg P^{(k)}$$


P
I
k-path?
BMC

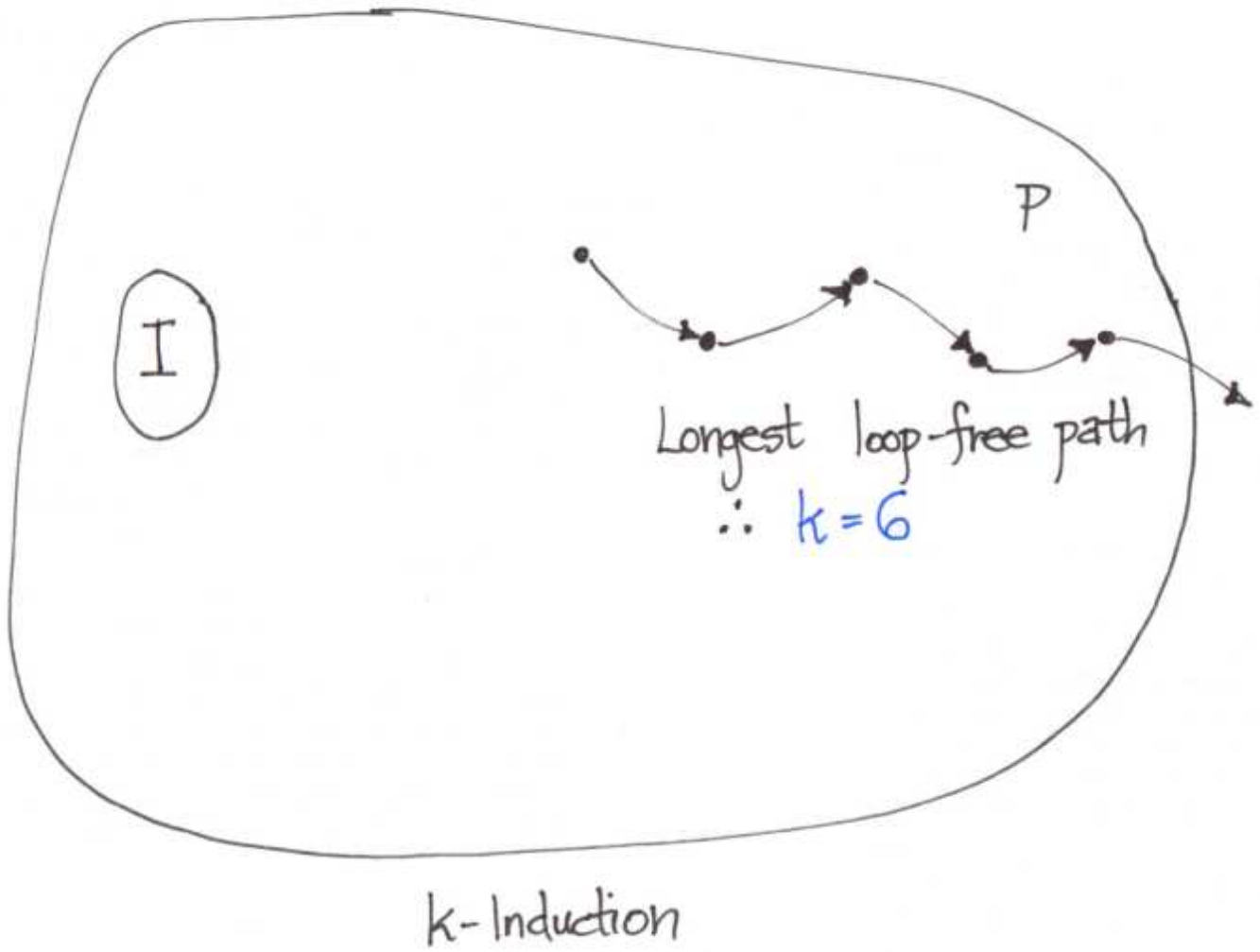
k-Induction [Sheeran et al. 2000]

Addresses practical incompleteness of BMC:

- Initiation: BMC
- Consecution:

$$\bigwedge_{i=0}^{k-1} (P^{(i)} \wedge T^{(i)}) \Rightarrow P^{(k)}$$

(plus extra constraints to consider loop-free paths)


P
I
Longest loop-free path
∴ k = 6
k-Induction

ITP [McMillan 2003]

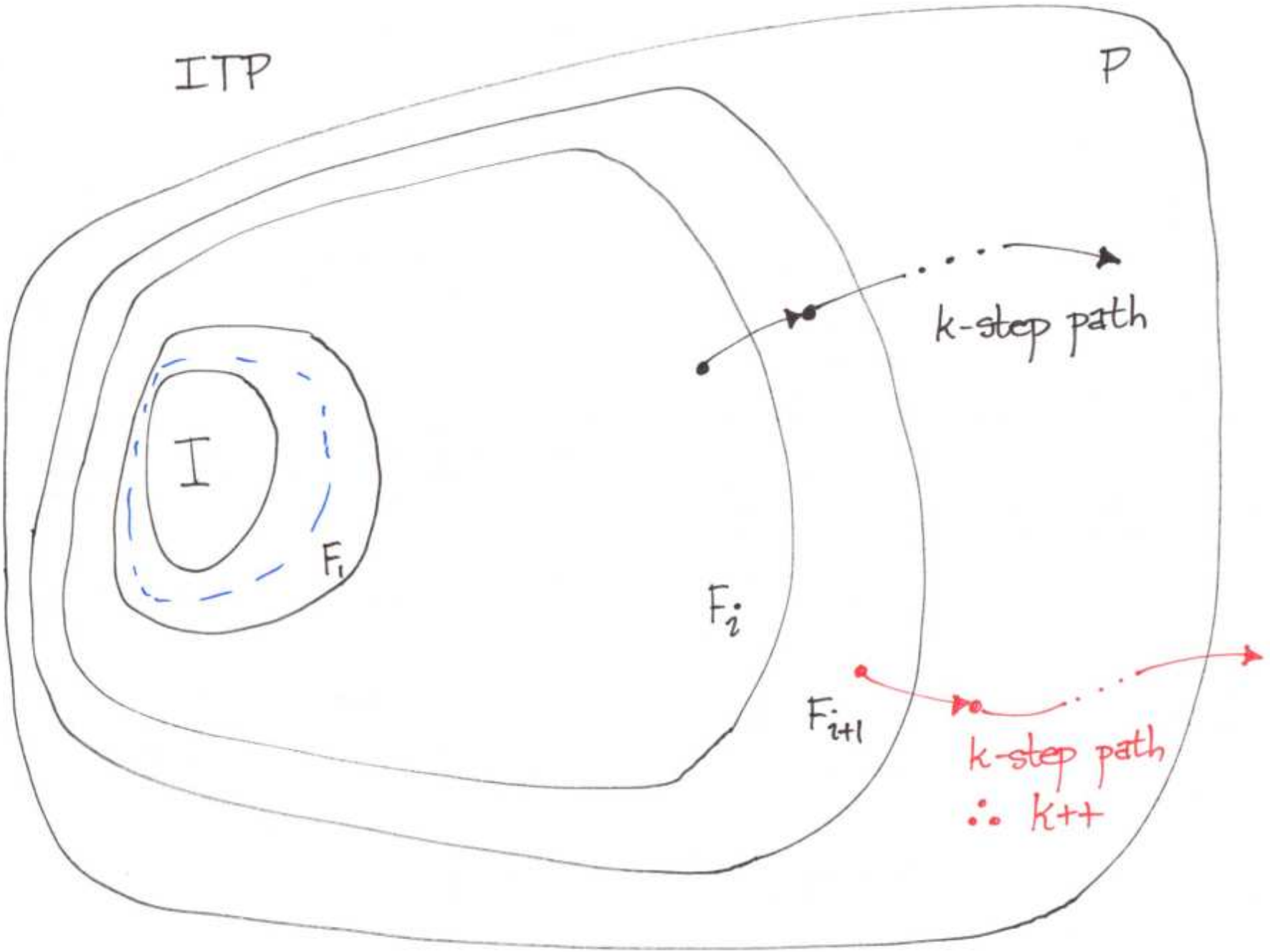
Property-focused over-approximating post-image:

$$F_i \wedge \bigwedge_{i=0}^{k-1} (P^{(i)} \wedge T^{(i)}) \Rightarrow P^{(k)}$$

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

$$F_i \wedge T \Rightarrow F'_{i+1} \qquad F'_{i+1} \wedge \bigwedge_{i=1}^{k-1} (P^{(i)} \wedge T^{(i)}) \Rightarrow P^{(k)}$$

- If fails, increases k


ITP
P
k-step path
I
F_1
F_i
F_{i+1}
k-step path
∴ k++

BMC $\rightarrow$ k-Induction $\rightarrow$ ITP

- Completeness from unrolling transition relation
- Evolution: reduce max k 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)

IC3: A Prover

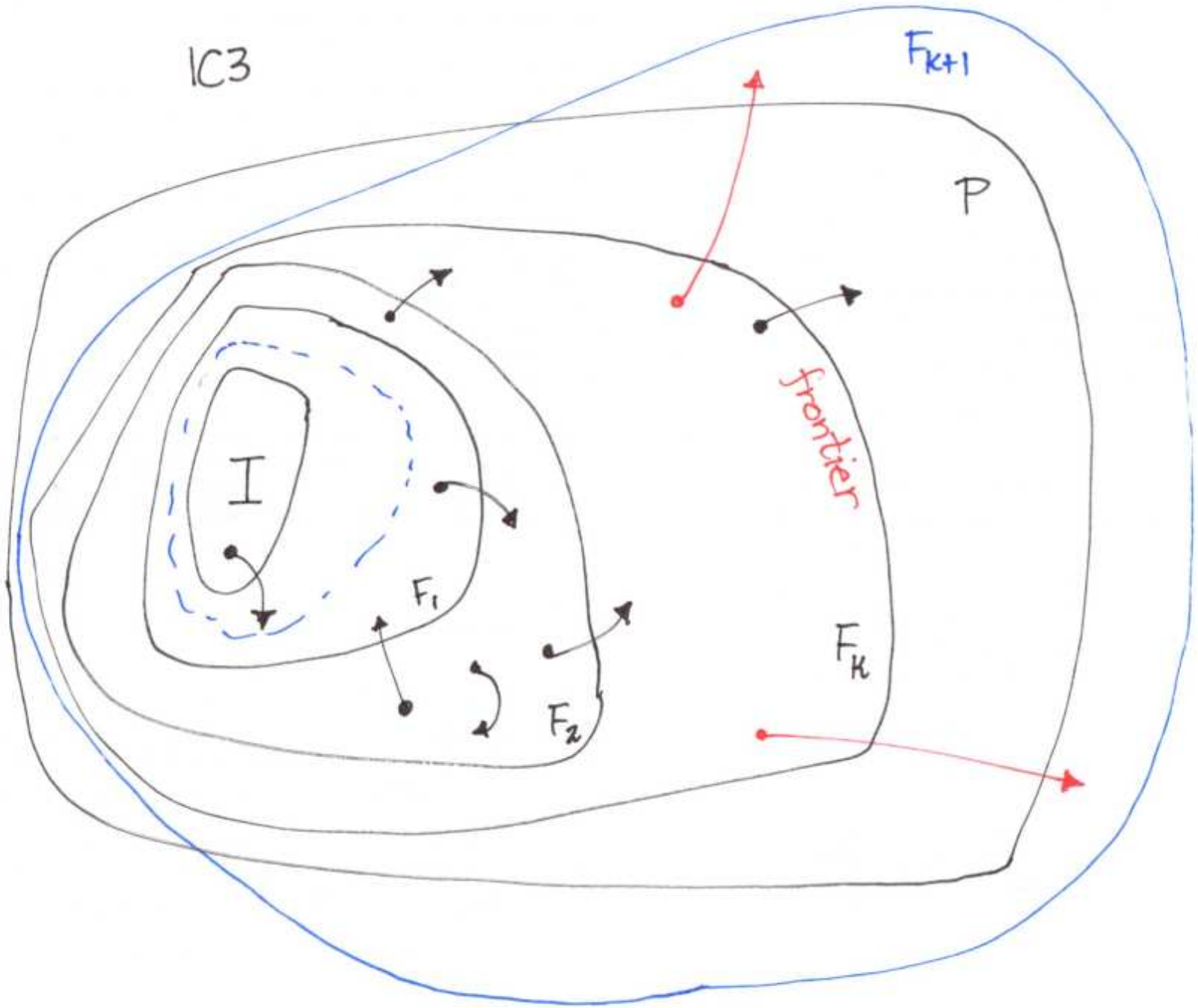
Stepwise sets $F_0, F_1, \ldots, F_k, F_{k+1}$ (CNF):

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

Four invariants:

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

$\therefore$ if ever $F_i = F_{i+1}$, F_i is inductive & P is invariant


IC3
F_{k+1}
P
frontier
I
F_1
F_2
F_k

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 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

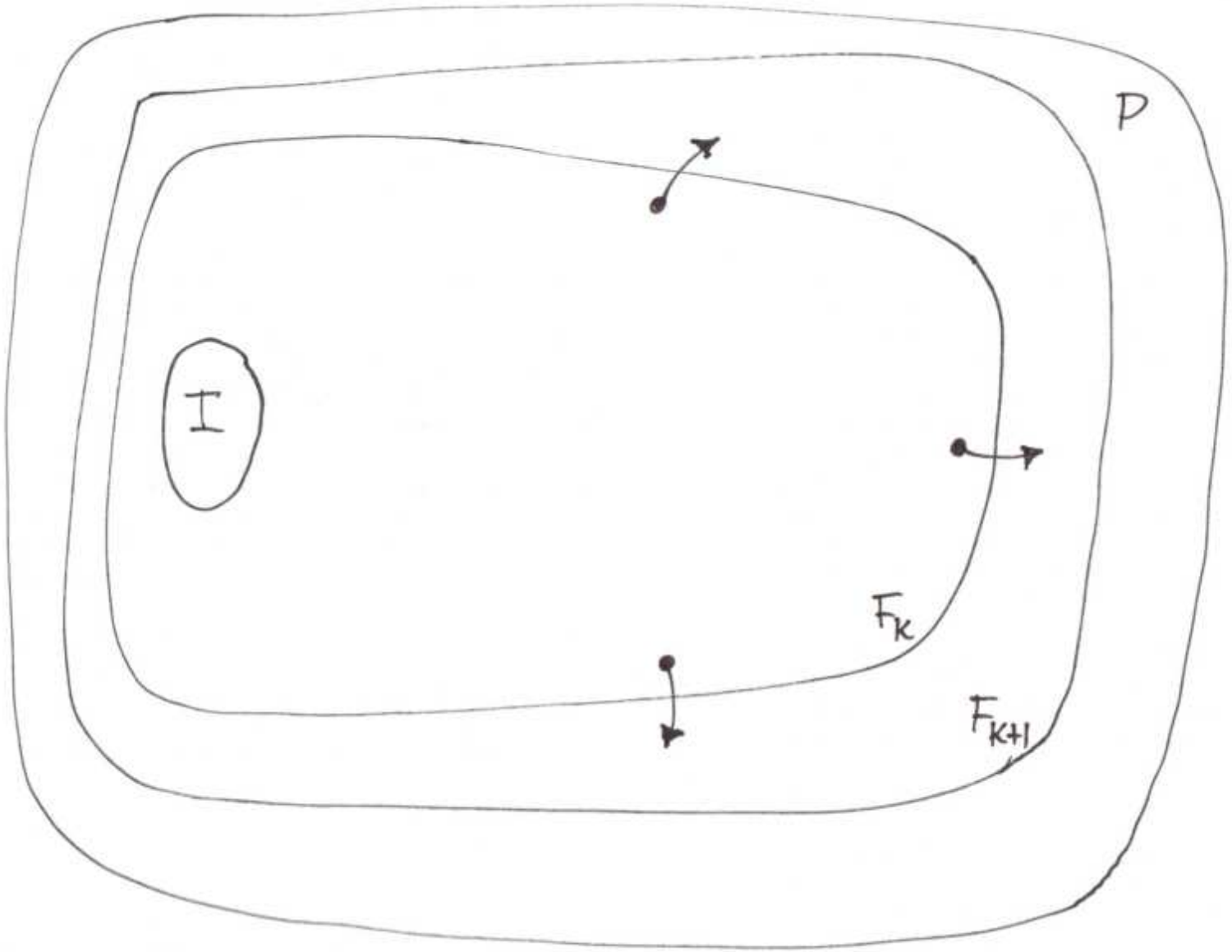
Induction at Top Level

Is P inductive relative to F_k?

$$F_k \wedge T \Rightarrow P'$$

(Recall: $F_k \Rightarrow P$)

- Possibility #1: Yes
- Conclusion: P is inductive relative to F_k



$$F_k \wedge P \wedge T \Rightarrow P'$$

Induction at Top Level

Monolithic behavior (predicate abstraction):

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

$$F_i \wedge T \Rightarrow C'$$

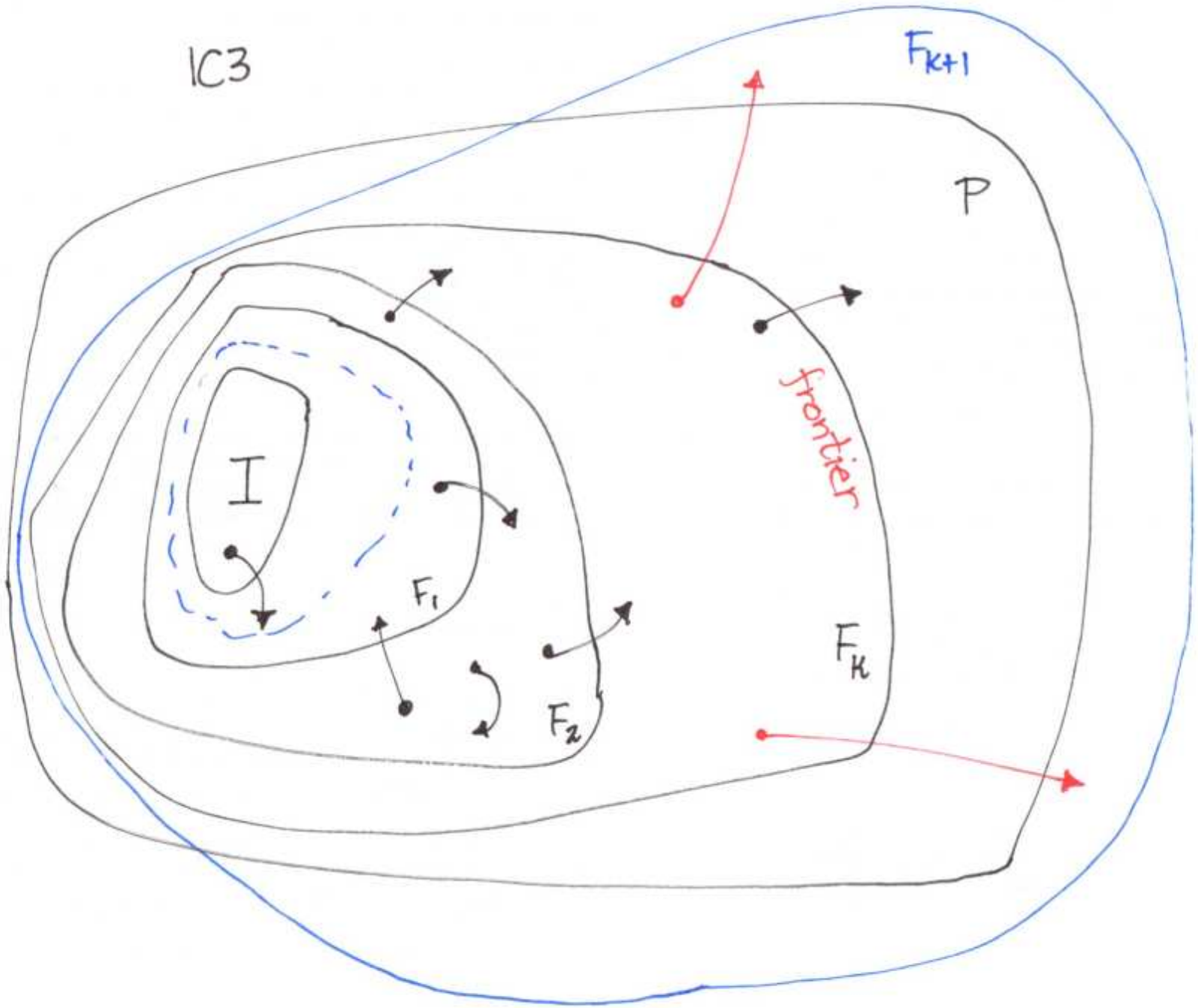
  F_{i+1} := $F_{i+1} \wedge C$
- F_{k+1} := $F_{k+1} \wedge P$
- New frontier: F_{k+1}

If ever $F_i = F_{i+1}$, done: P is invariant.

Counterexample To Induction (CTI)

$$F_k \wedge T \Rightarrow P'$$

- Possibility #2: No
- Conclusion: $\exists$ F_k-state s with error successor
- If s is an initial state, done: P is not invariant
- Otherwise...


IC3
F_{k+1}
P
frontier
I
F_1
F_2
F_k

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):

$$F_i \wedge c \wedge T \Rightarrow c'$$

  - No strict subclause of c is inductive relative to F_i

Induction at Low Level

Is c an interpolant?

$$I' \vee (F_i \wedge c \wedge T) \Rightarrow c' \qquad c' \Rightarrow \neg s'$$

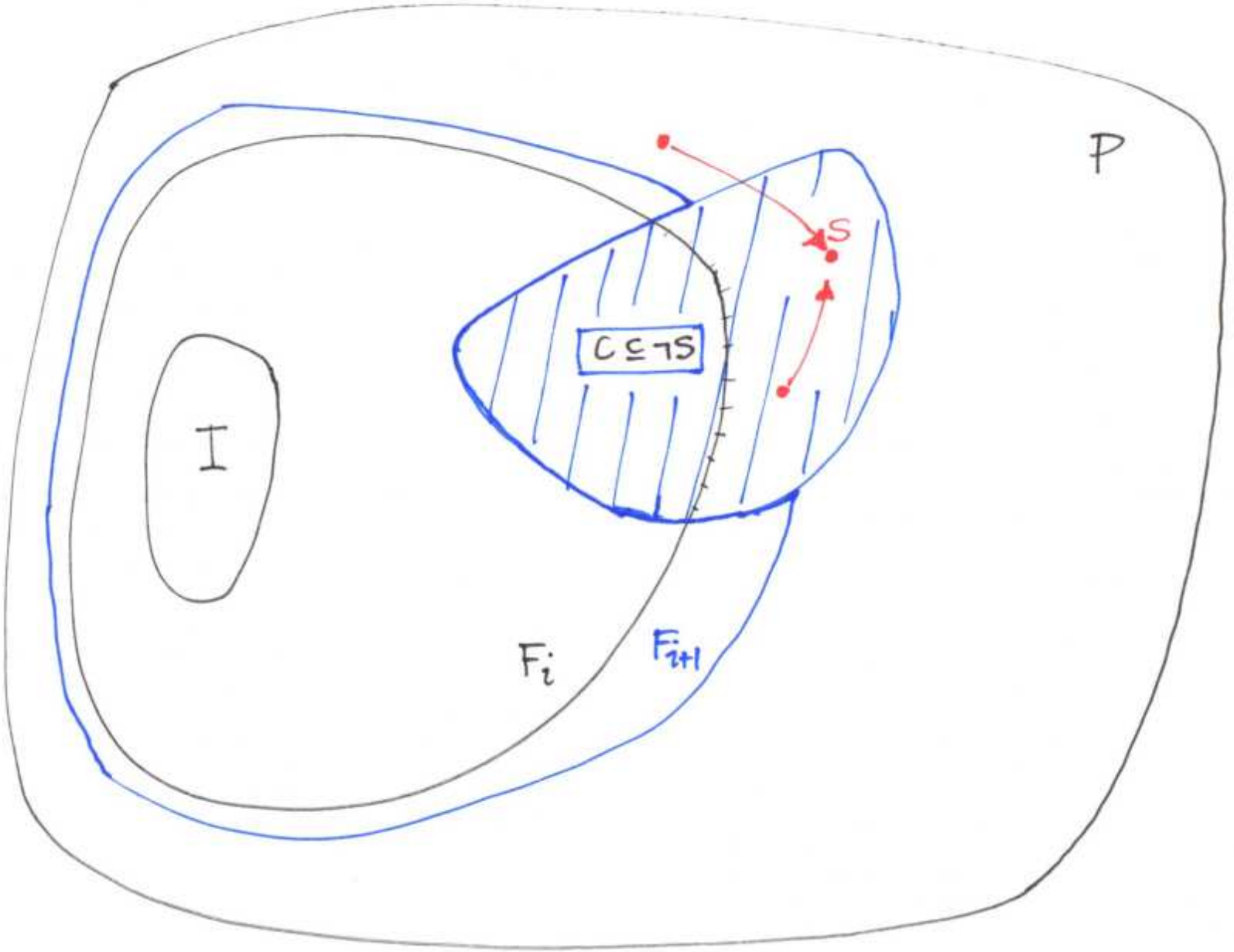
$$I' \vee (F_i \wedge T) \Rightarrow c' \qquad c' \Rightarrow \neg s'$$

Not necessarily:

- An interpolant is inductive but. . .
- An inductive assertion need not be an interpolant.

In practice:

- Quality vs. speed trade-off per clause
- Quality wins in experiments: faster overall



Inductive Generalization

Addressing CTI s

- Find highest i such that

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

- Apply inductive generalization:

$$c \subseteq \neg s \qquad I \Rightarrow c \qquad F_i \wedge c \wedge T \Rightarrow c'$$

- $\therefore F_{i+1} := F_{i+1} \wedge c$ (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:

$$F_j \wedge \neg t \wedge T \Rightarrow \neg t'$$

If UNSAT:

- Inductive generalization must succeed:

$$c \subseteq \neg t \qquad I \Rightarrow c \qquad F_j \wedge c \wedge T \Rightarrow c'$$

- F_{j+1} := $F_{j+1} \wedge c$
- Updated proof obligation (if $j < k$): $(t,\ j+1)$

Addressing Proof Obligation (t, j)

SAT query:

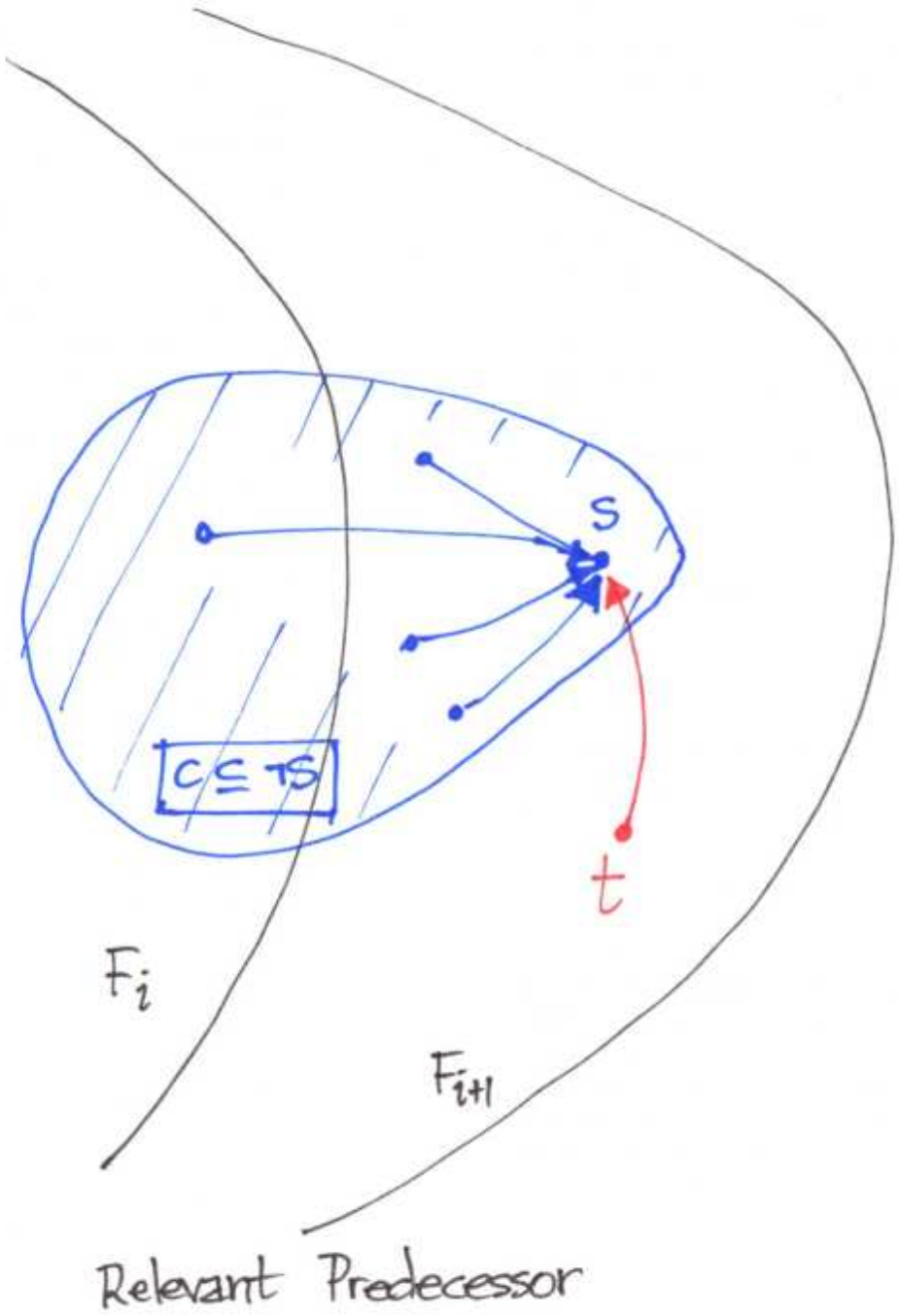
$$F_j \wedge \neg t \wedge T \Rightarrow \neg t'$$

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}.


s
$c \subseteq \neg s$
t
F_i
F_{i+1}
Relevant Predecessor

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.

IC3: A Bug Finder

Suppose:

- $u \rightarrow t \rightarrow s \rightarrow$ Error
- Proof obligations:

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

That is,

- s: inductively generalize relative to F_{k-1}
- t: inductively generalize relative to F_{k-2}
- u: 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

$$u \to t \to s \to \text{Error}$$

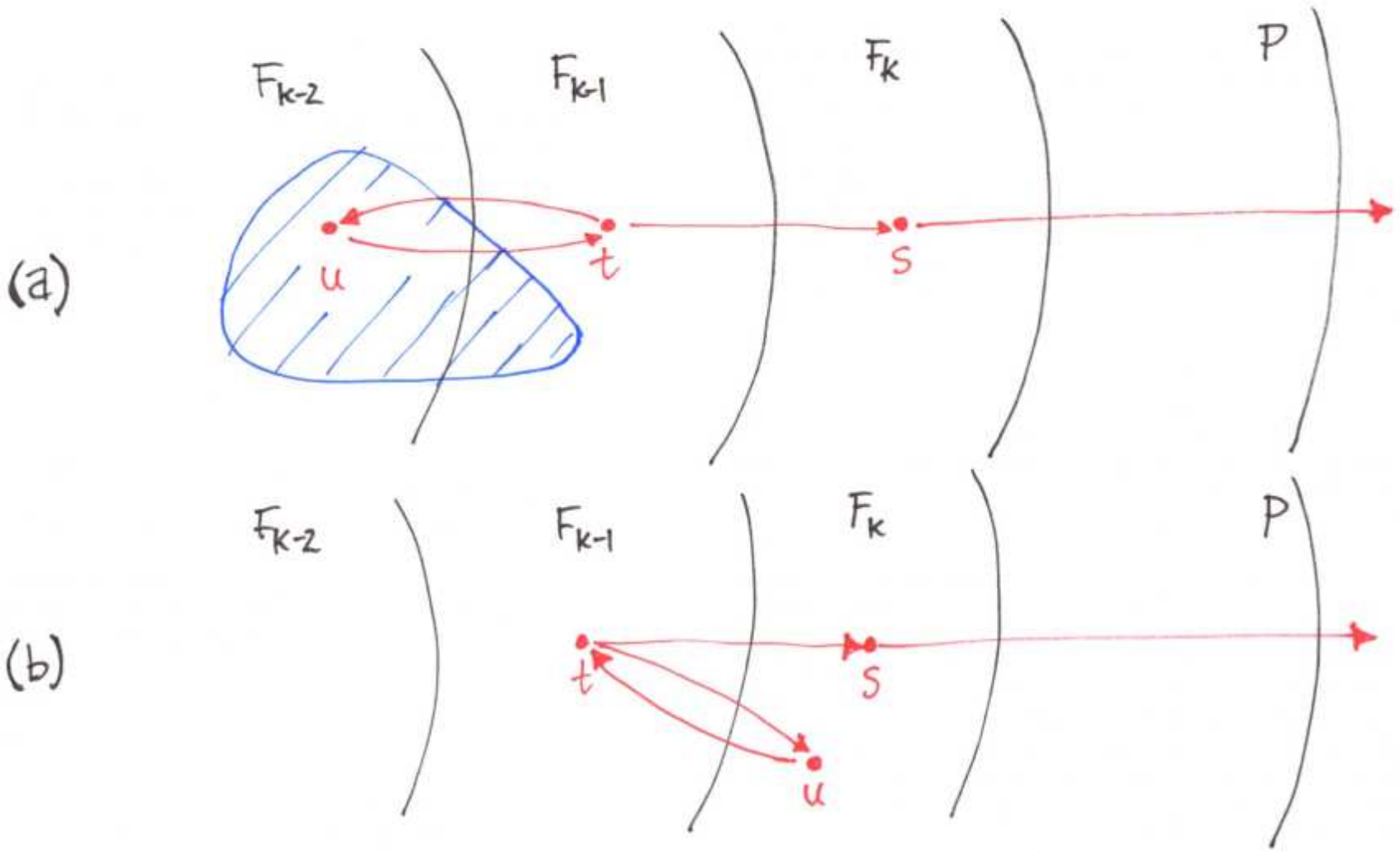
- t is the state that IC3 **considers as likeliest to be closest to an initial state**.

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

"Proximity metric"

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

(It also happens to be the correct choice [Bradley 2011].)



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

Proof Obligations: Guided Search

Recent Work: 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!)

Temporal Logics

- FAIR [Bradley et al., FMCAD'11]
  - For ω-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

Infinite-state Systems

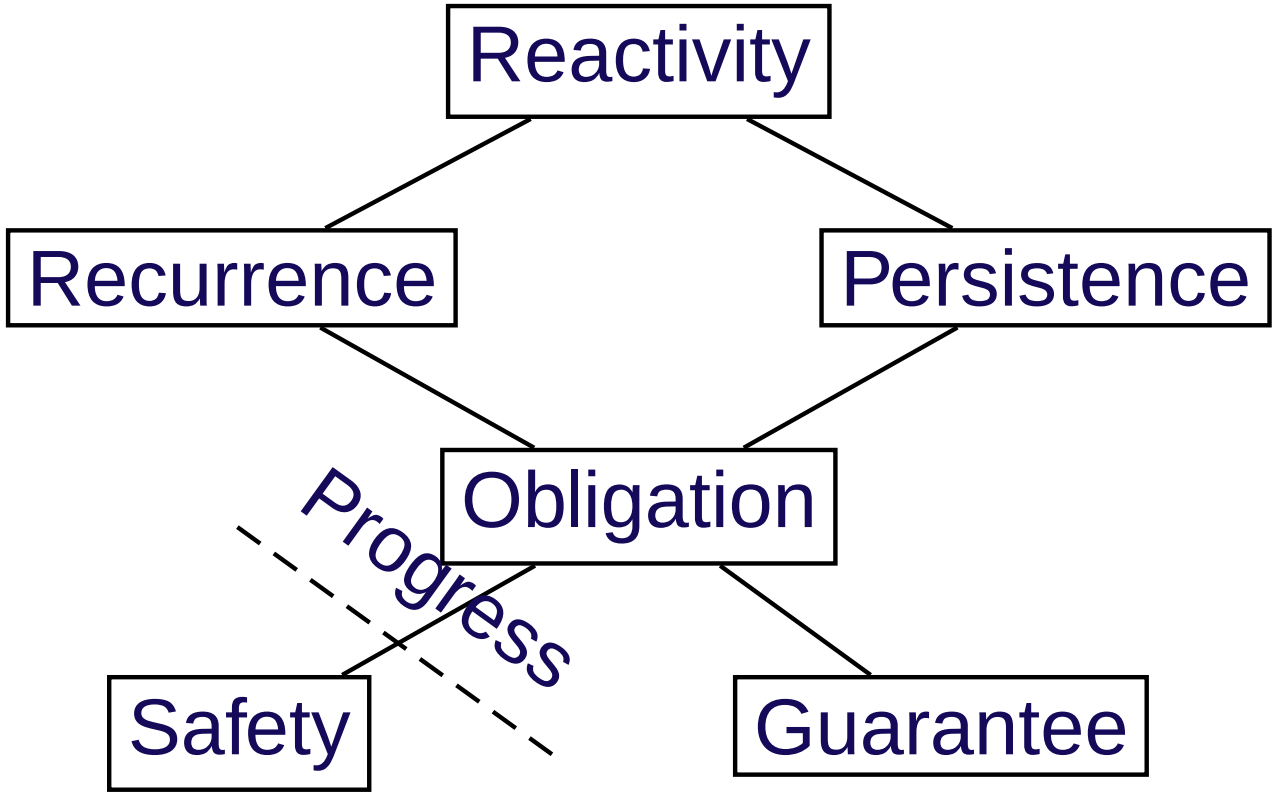
- **SMT-based Induction Methods for Timed Systems** [Kindermann et al., arXiv'12]
- **Generalized Property Directed Reachability** [Hoder et al., SAT'12]
  - Boolean push-down systems
  - Linear real arithmetic
- **Software Model Checking via IC3** [Cimatti et al., CAV'12]
  - Explicit handling of CFG
  - Applies IC3 techniques to McMillan's "Lazy Abstraction with Interpolants" [McMillan, CAV'06]

Handling Proof Obligations

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

Linear Time Hierarchy



Safety: IC3

Progress: FAIR over IC3

Generalized Büchi Automata

- Given:
  - Fair Transition System (FTS) $\mathcal{S}$
  - LTL property P
- Compute **generalized Büchi automaton** $\mathcal{C} = \mathcal{A}_{\neg P} \parallel \mathcal{S}$.
- If $\mathcal{S}$ is finite state, nonemptiness of $\mathcal{C}$ corresponds to the existence of a **reachable fair cycle**, aka **lasso**.

Strongly Connected Components

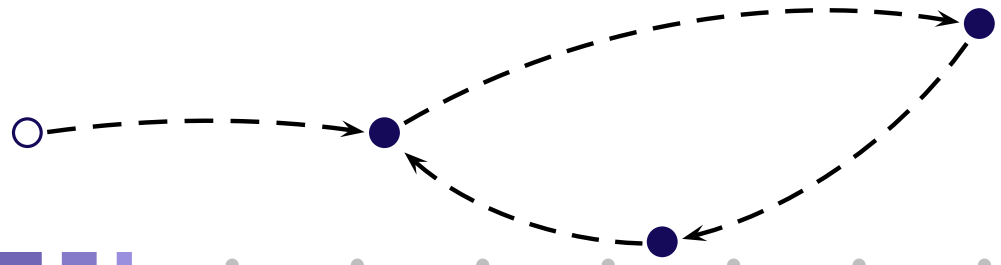
- A lasso's cycle is contained in a **strongly connected component** (SCC) of the state graph.
- A nonempty set of states is **SCC-closed** if every SCC is either contained in it or disjoint from it.
- A partition of the states into SCC-closed sets is a coarser partition than the SCC partition.
- ∴ Every cycle of a graph is contained in some SCC-closed set.

FAIR: Reachable Fair Cycles

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

- Skeleton:

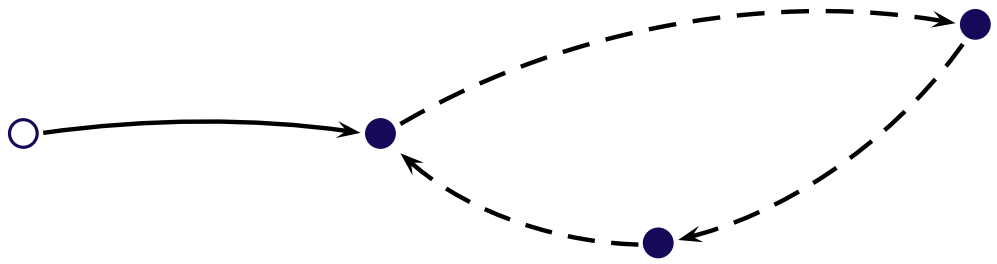
  Together satisfy all fairness constraints.
- Task: Connect states to form lasso.



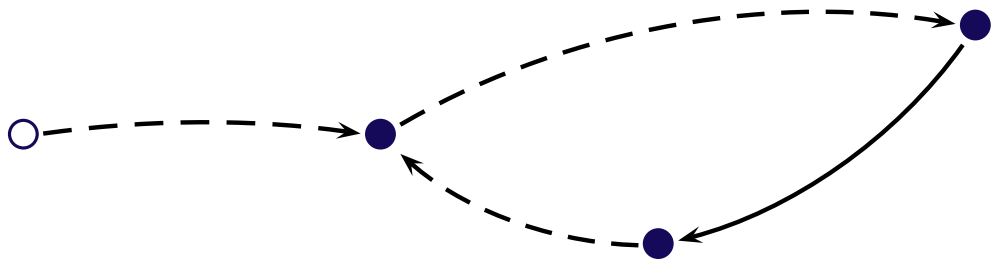
Reach Queries

Each connection task is a reach query.

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



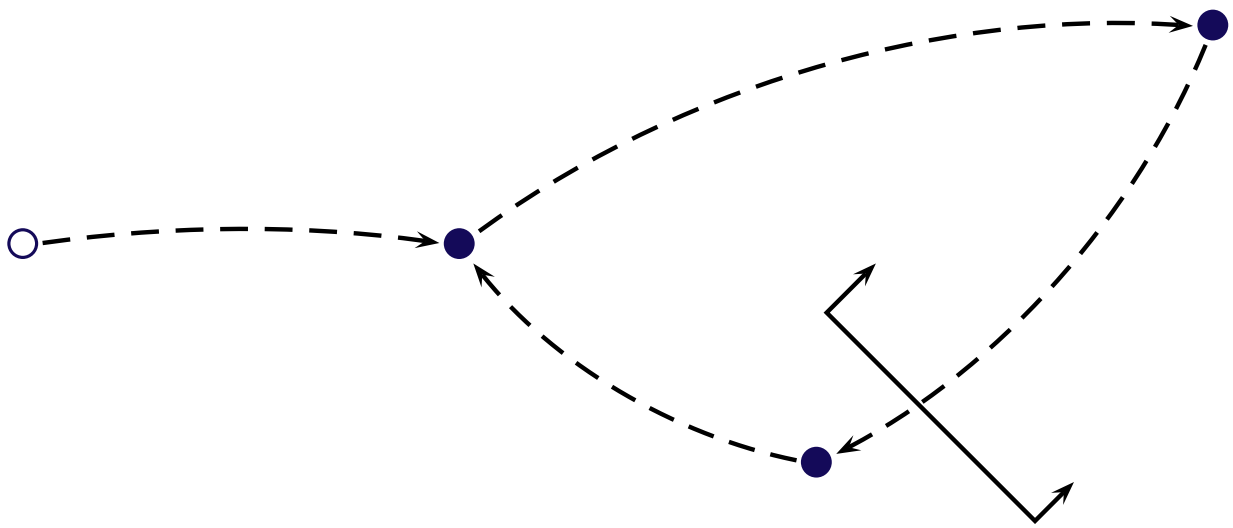
- **Cycle query**: Connect one state to another:



(To itself if skeleton has only one state.)

Discovering SCC-Closed Sets

Negative cycle query $\Rightarrow$ knowledge of SCC structure



- **Inductive** proof: “one-way barrier”
- Each “side” of the proof is SCC-closed
- Subsequent skeletons: all states on one side

Key Insight

- **Inductive assertions** describe **SCC-closed sets**
- **Arena**: States all on same side of each barrier
- Unlike previous symbolic methods:

$$\begin{array}{rr} & \text{Barrier constraints on } T \\ + & \text{Over-approximating prover (IC3)} \\ \hline & \text{Simultaneous consideration of all arenas} \end{array}$$

- In other words:

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

Incremental, Inductive Verification

	IC3	FAIR	IICTL
Hypothesis	CTI	skeleton	task state
Lemma	clause	barrier	refinement
Induction	clause	barrier	EU (IC3), EG (FAIR)
Generalization	MIC	proof improvement	
			trace generalization

Conclusions

- Attempted to explain why IC3 works:
  - As a compromise between the **incremental** and **monolithic** strategies
  - In terms of characteristics of previous SAT-based MC
  - As a prover
  - As a bug finder
- IIV:
  - State-driven, inductive generation of lemmas
  - Simple but many SAT queries
  - Parallelizable