Understanding IC3

Aaron R. Bradley

ECEE, CU Boulder &

Summit Middle School

Further Reading

This presentation is based on

> Bradley, A. R. “Understanding IC3.” In *SAT*, June 2012.

`http://theory.stanford.edu/~arbrad`

Induction

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

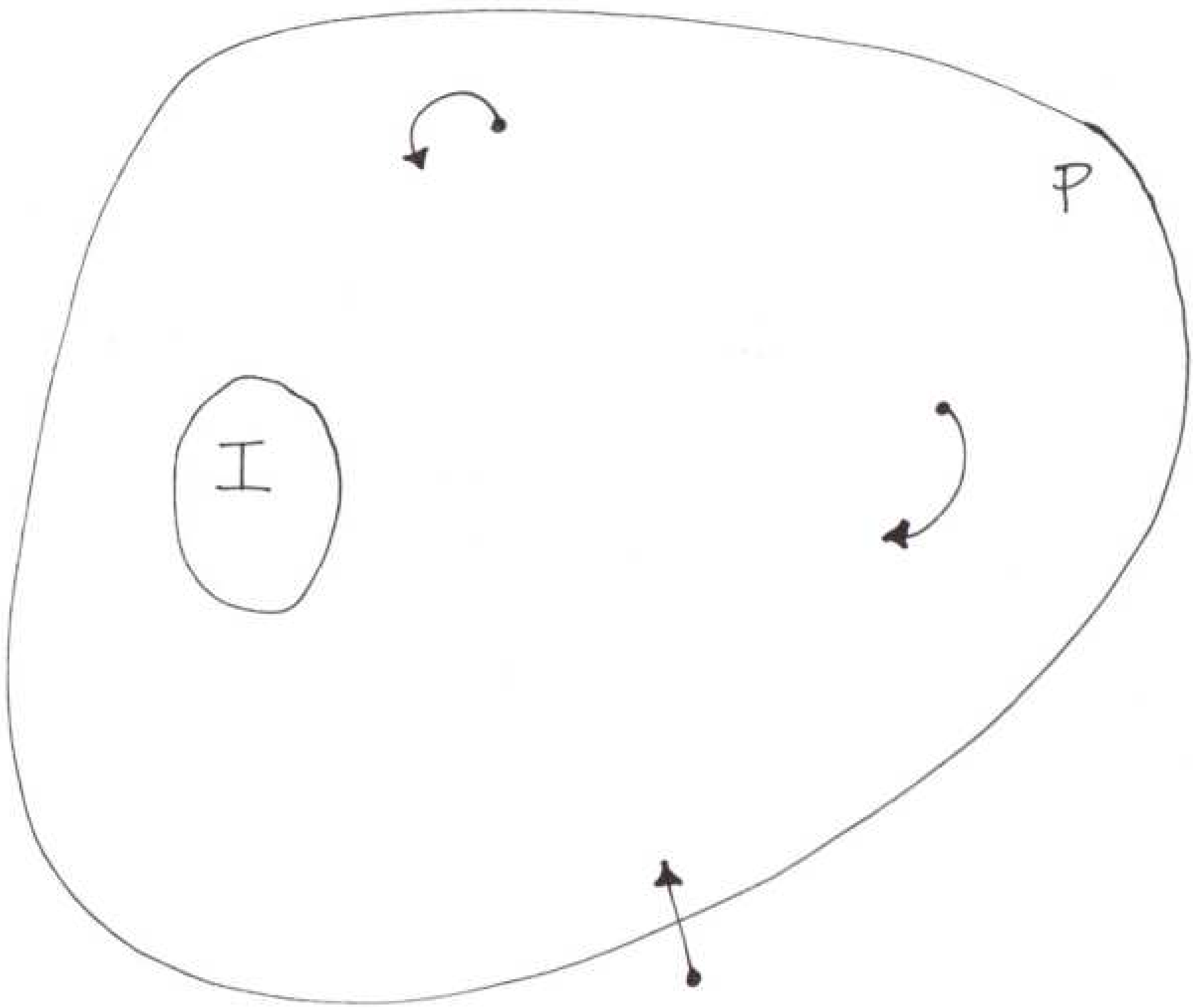
To prove that $S : (I, T)$ has safety property P, prove:

- Base case (**initiation**):

$$I \Rightarrow P$$

- Inductive case (**consecution**):

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

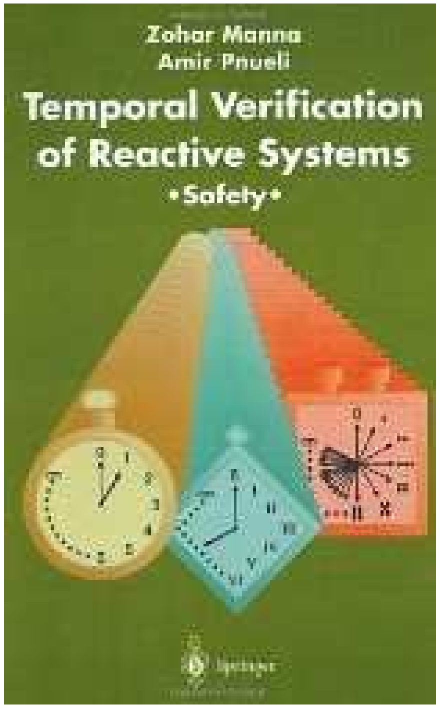


P is inductive

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 the two methods
2. SAT-based model checkers
3. Understanding IC3 as a prover
4. Understanding IC3 as a bug finder
5. Beyond IC3: Incremental, inductive verification

Two Transition Systems

S_1:

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

S_2:

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

$$P: \ y \geq 1$$

Induction on System 1

S_1:

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

- 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
while *:
  x, y := x + 1, y + x
```

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

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
while *:
  x, y := x + y, y + x
```

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:

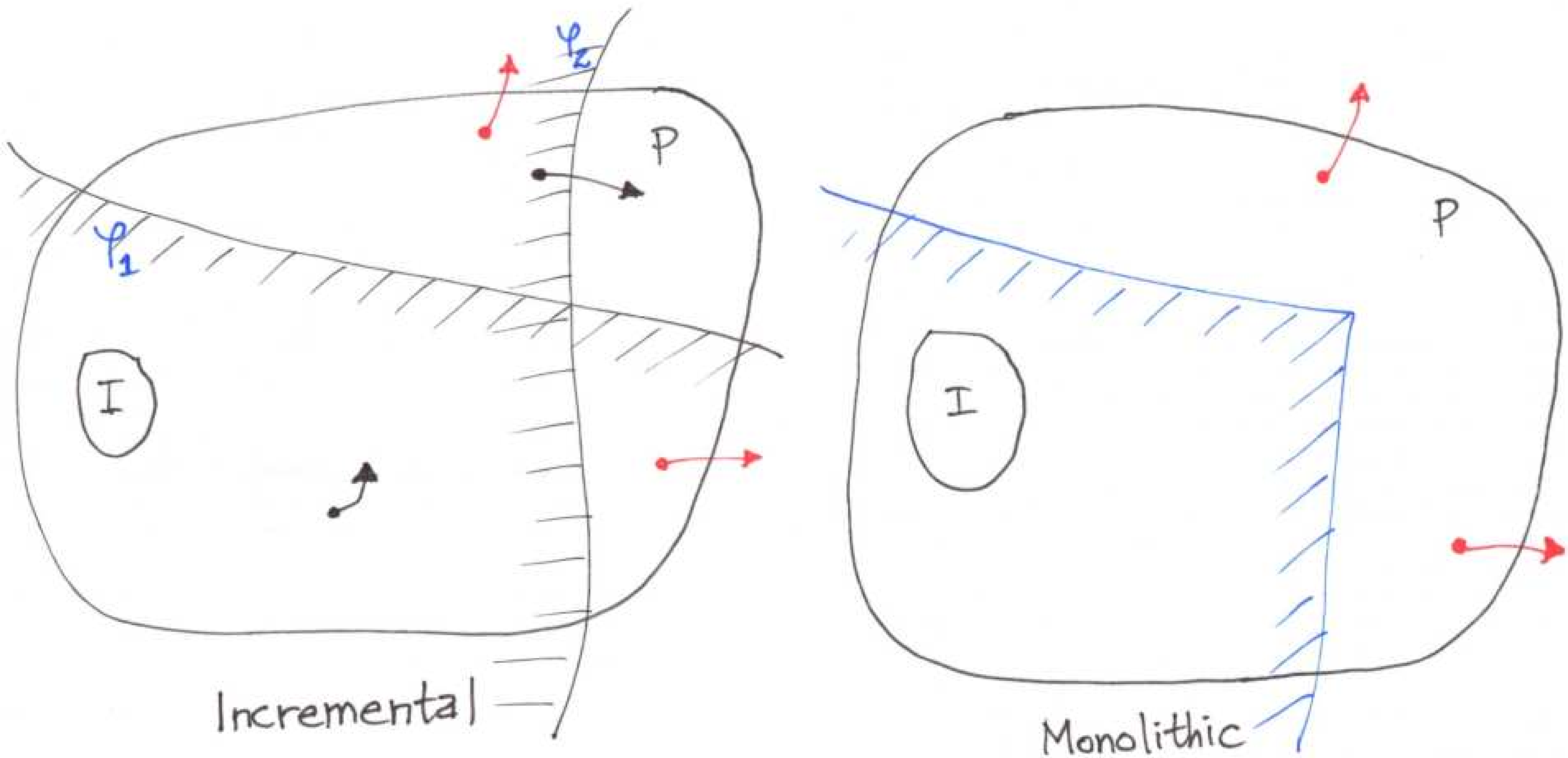
```
x, y := 1, 1
while *:
  x, y := x + y, y + x
```

Invent strengthening all at once:

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

Consecution:

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



φ_2
P
φ_1
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.,

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

- 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

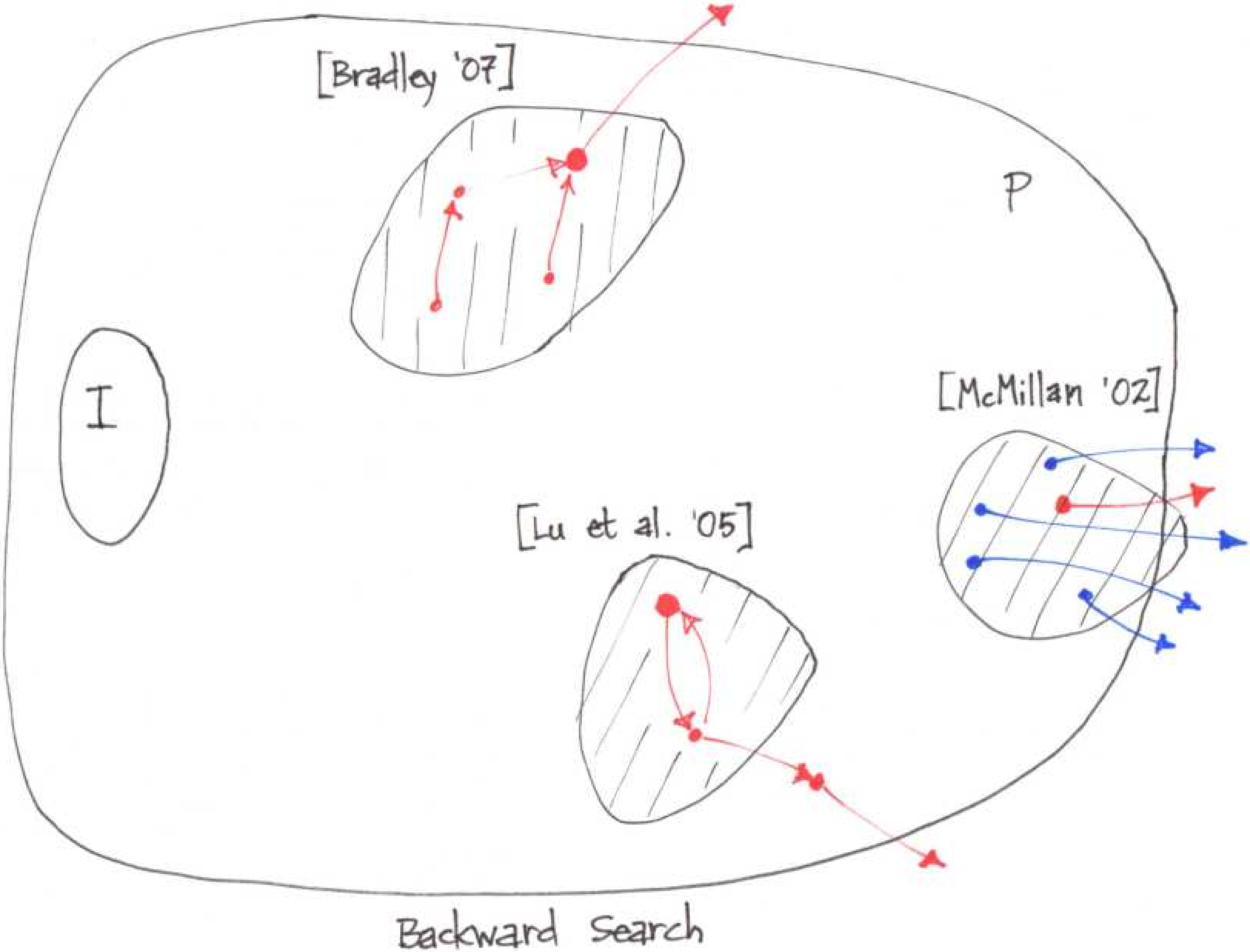
- Initiation:

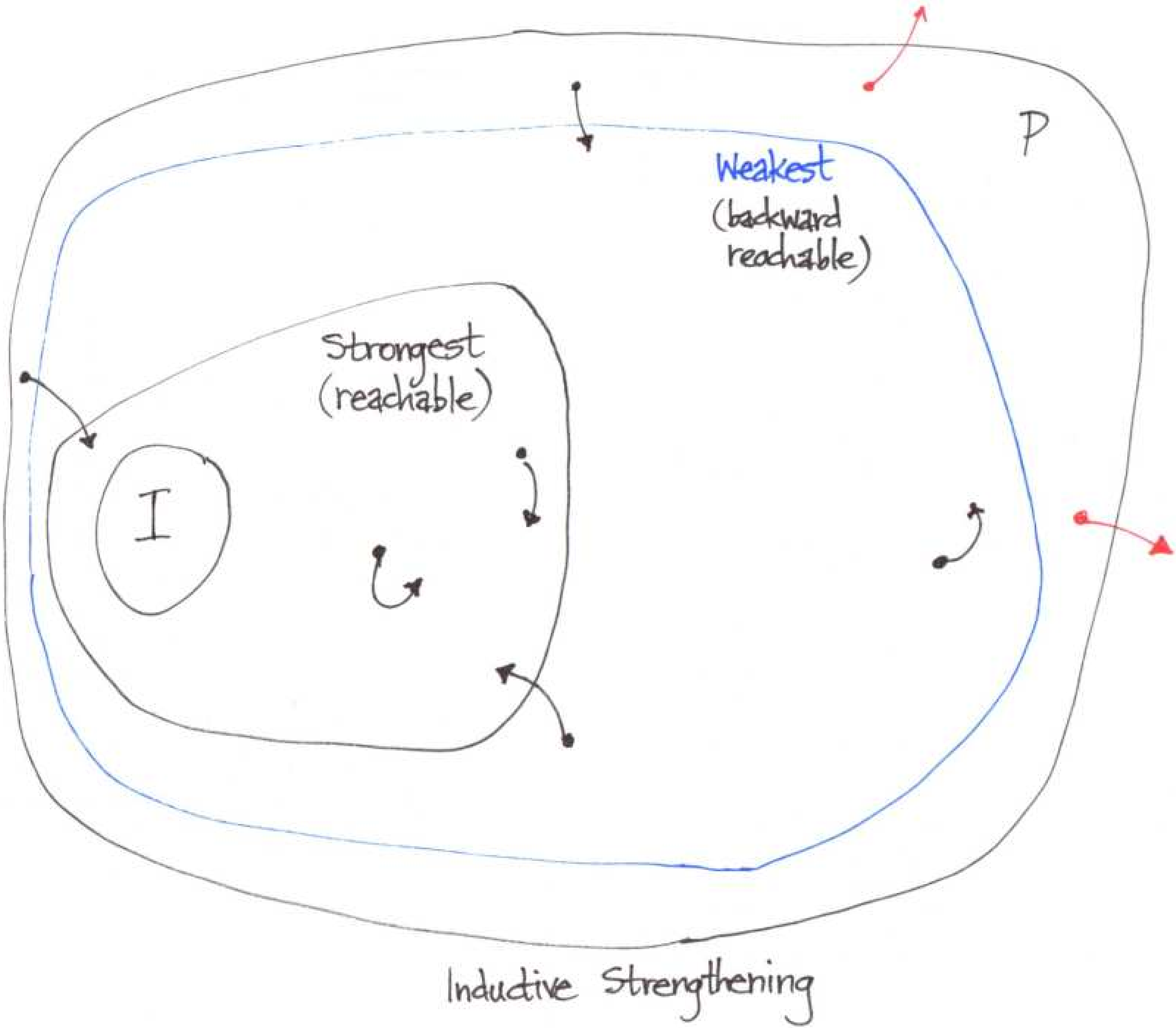
$$I \Rightarrow c$$

- 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


P
Weakest
(backward
reachable)
Strongest
(reachable)
I
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:

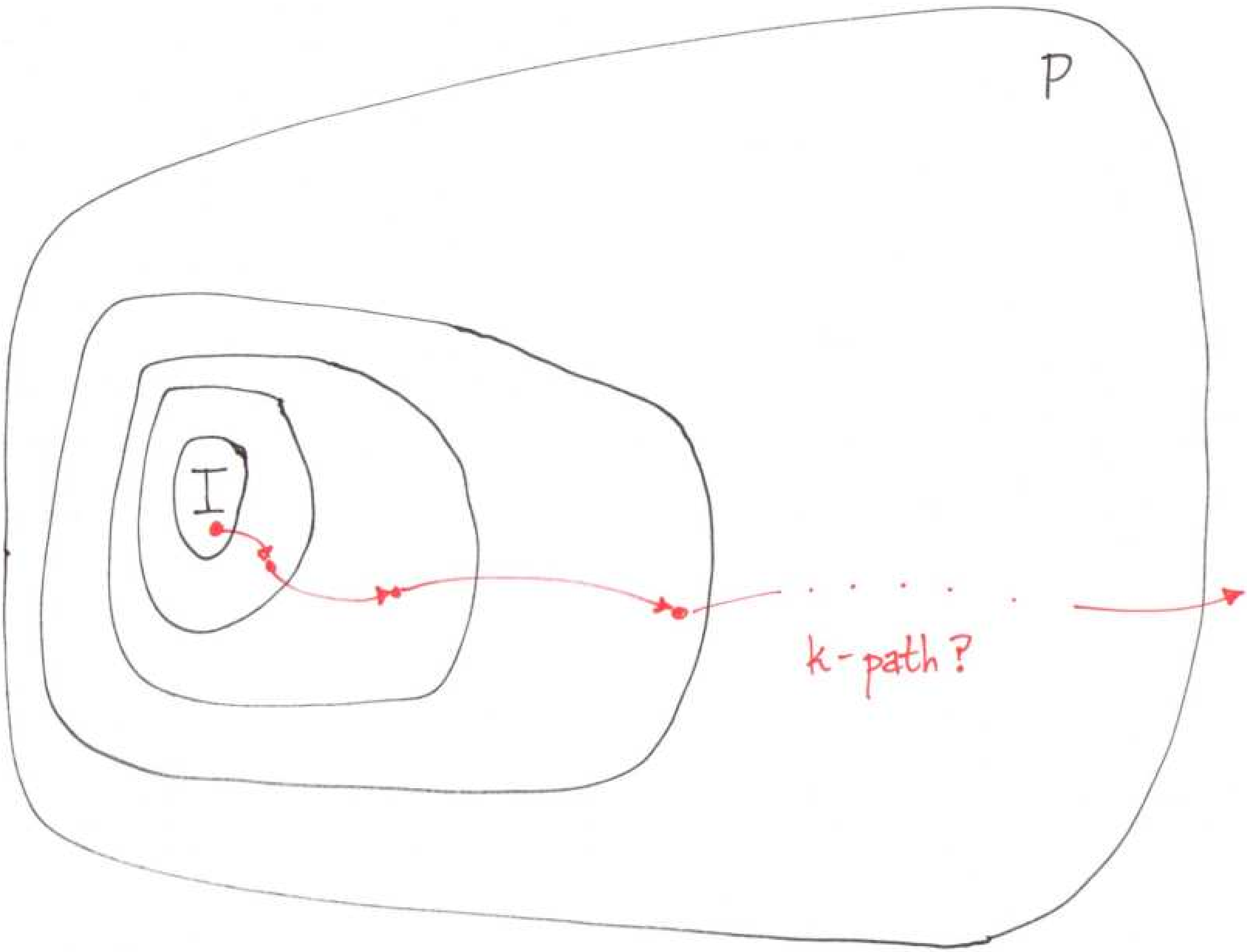
- Like others when inductive generalization fails

BMC [Biere et al. 1999]

Compared to backward search:

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

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



BMC

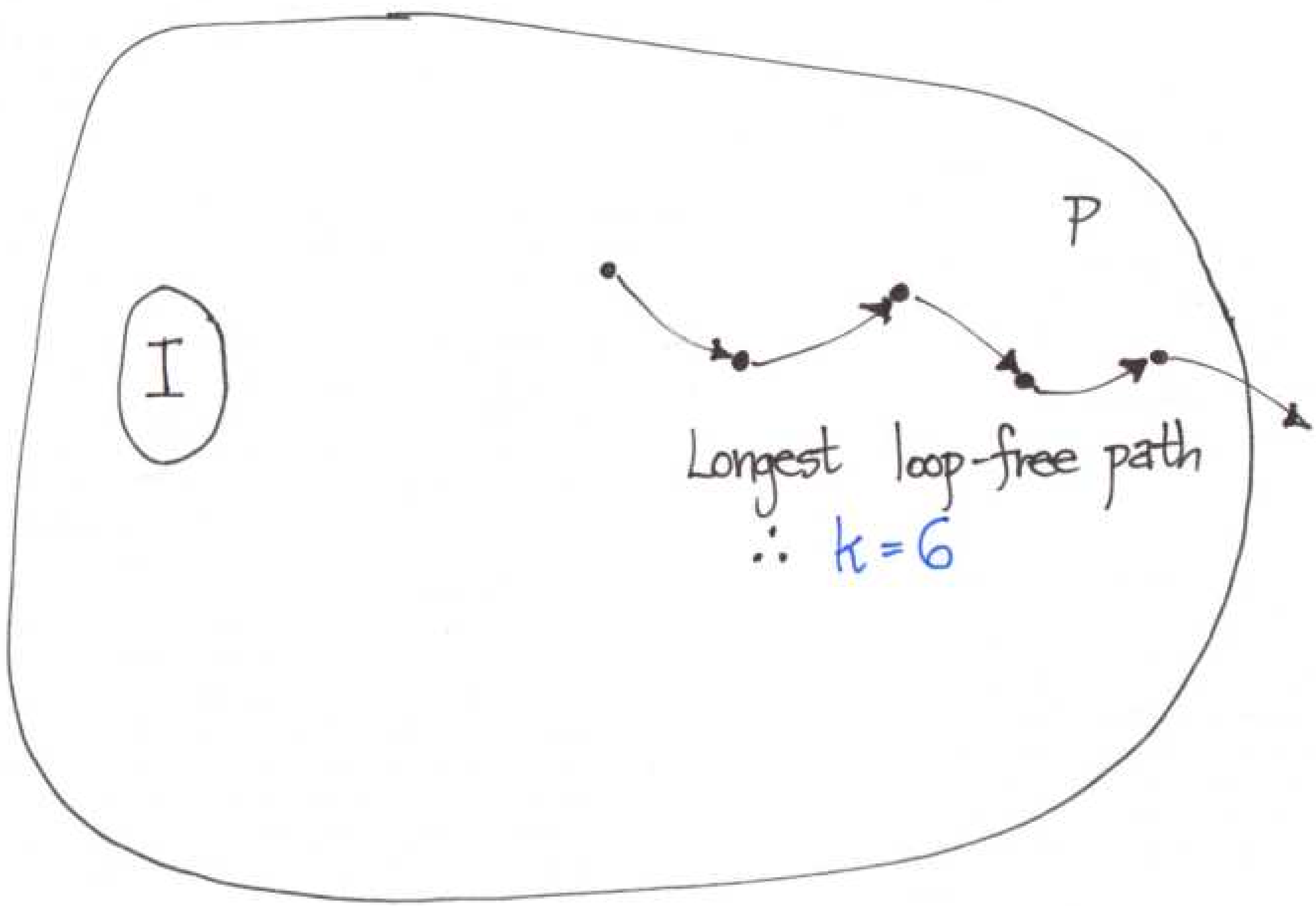
k-Induction [Sheeran et al. 2000]

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)



k-Induction

ITP [McMillan 2003]

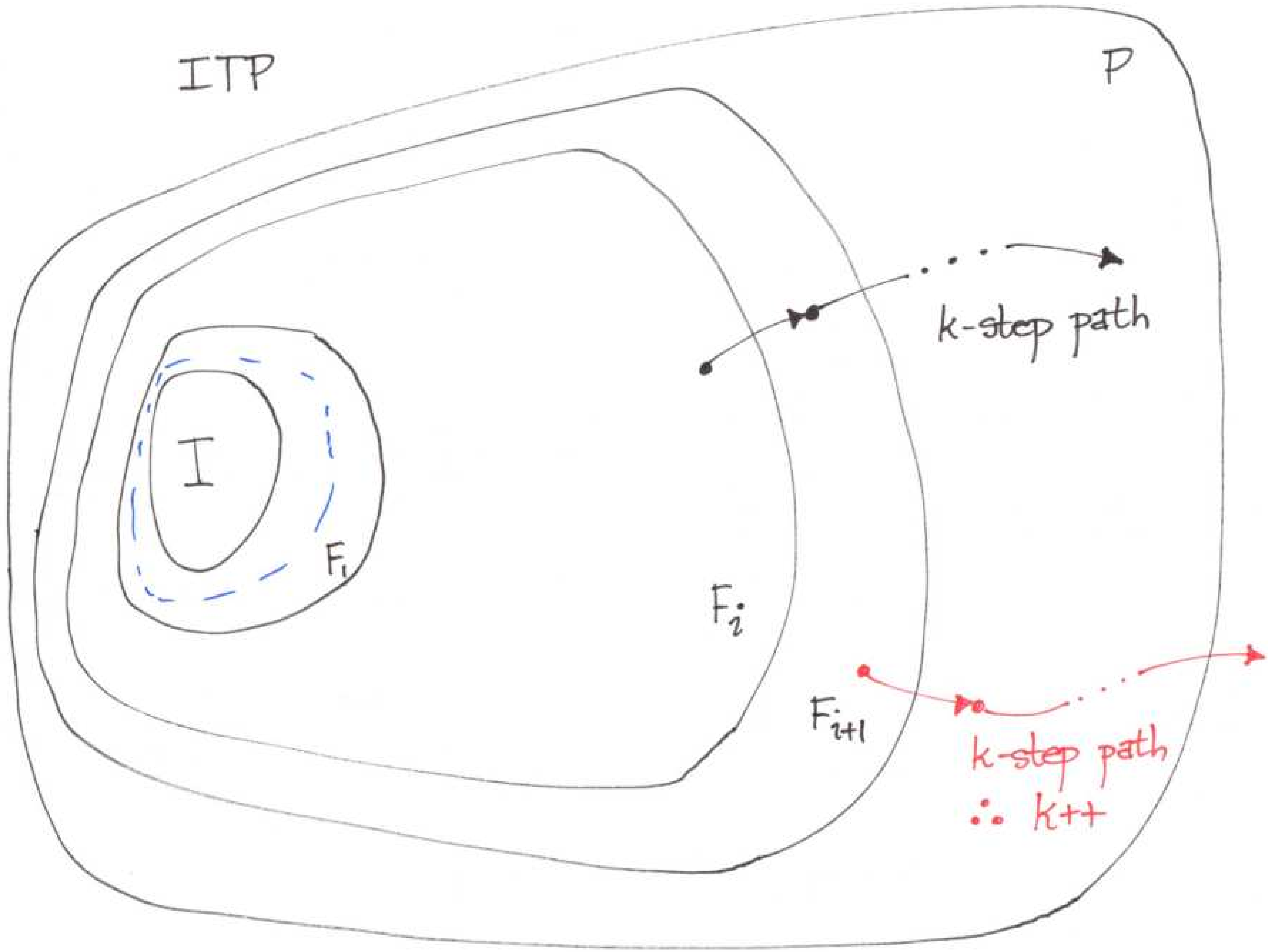
Property-focused over-approximating post-image:

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

- $\{$states $\leq i$ 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

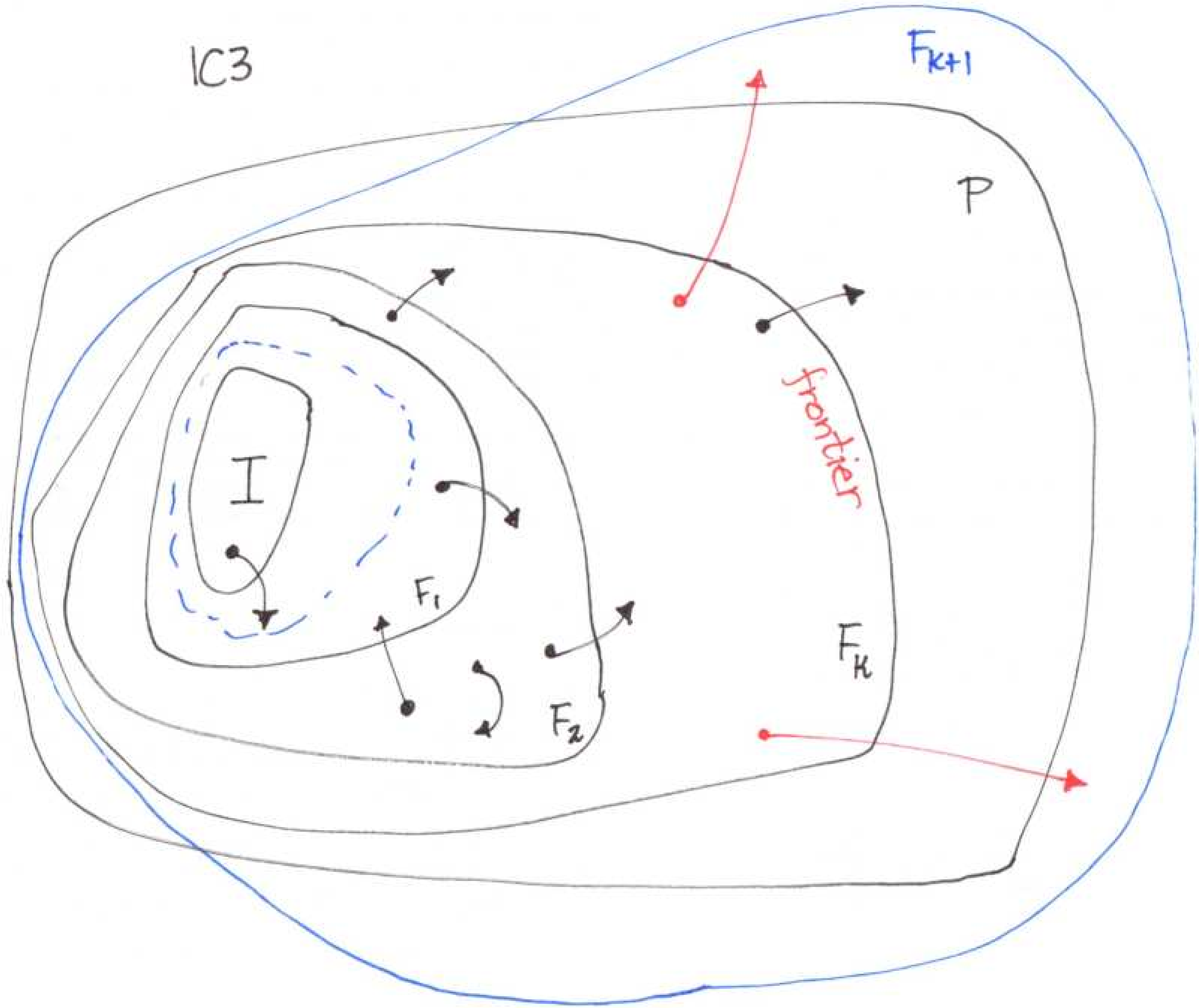
Stepwise sets $F_0, F_1, \ldots, F_k, F_{k+1}$ (CNF):

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

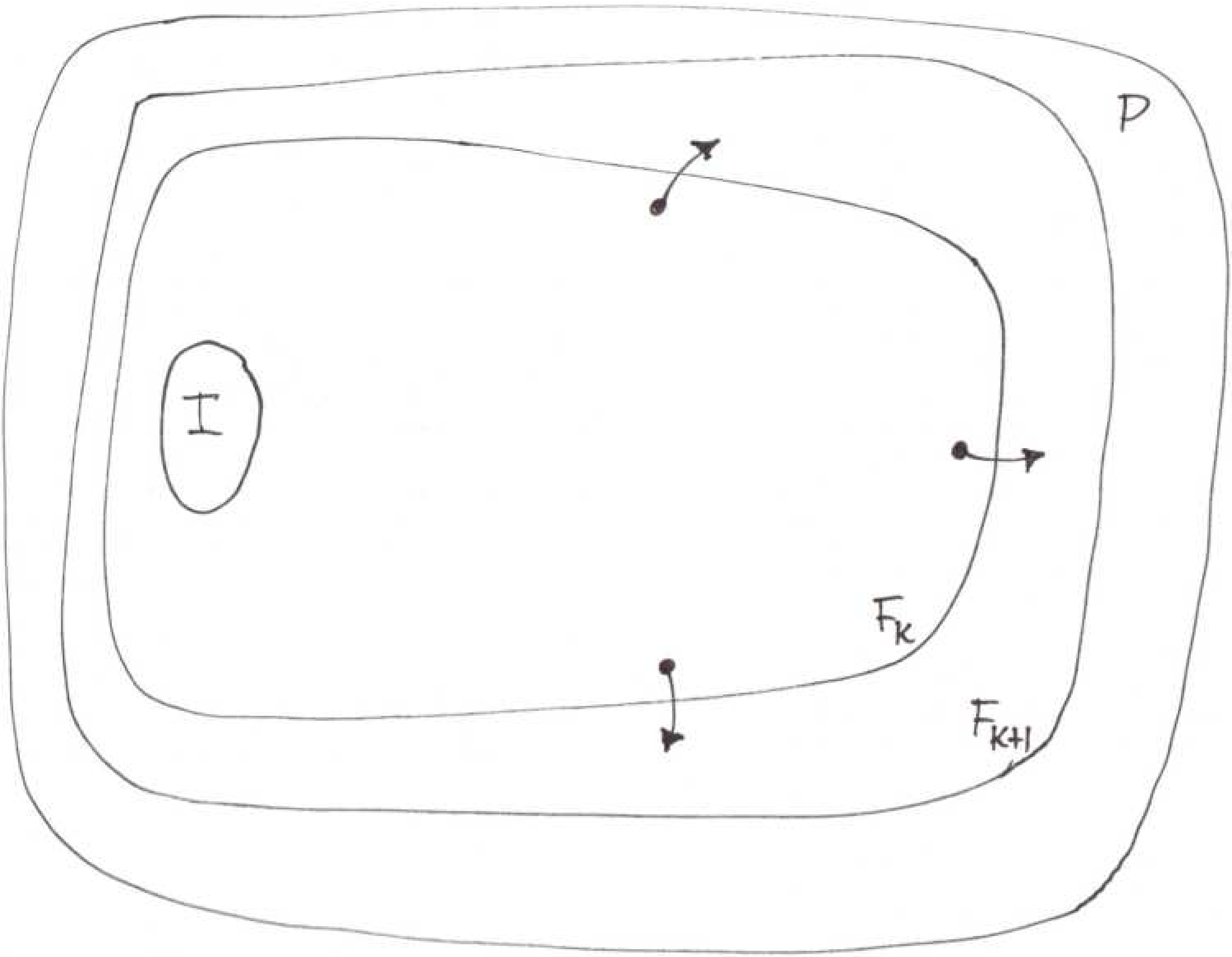
Induction at Top Level

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.

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

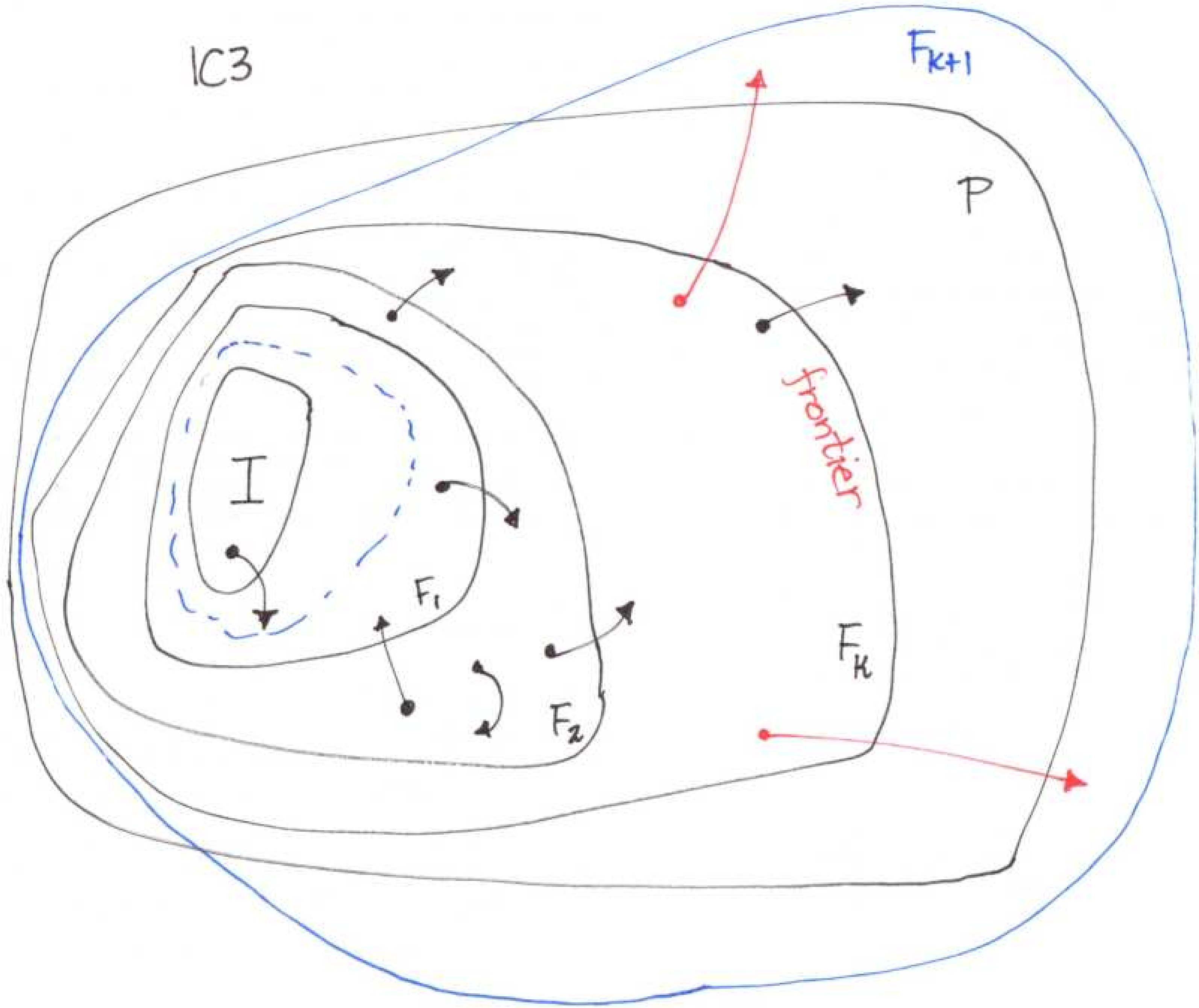
  $F_{i+1} := F_{i+1} \wedge C$
- $F_{k+1} := F_{k+1} \wedge P$
- New frontier: F_{k+1}

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

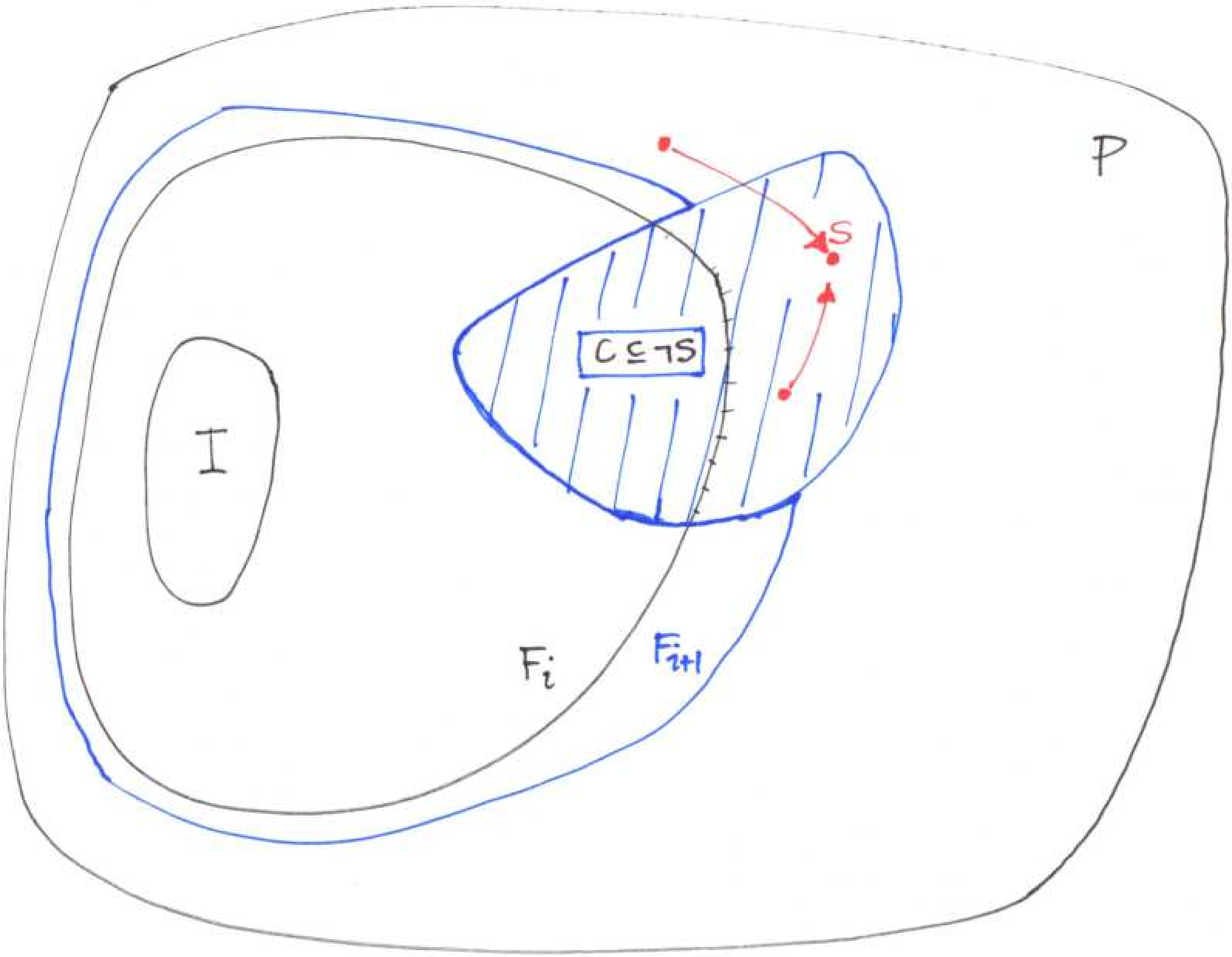
- **Given**: cube s
- **Find**: $c \subseteq \neg s$ such that
  - Initiation:

$$I \Rightarrow c$$

  - Consecution (relative to F_i):

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

  - No strict subclause of c is inductive relative to F_i



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

- Find highest i s.t. $\neg u$ is inductive relative to F_i
- Inductively generalize ($c \subseteq \neg u$): F_{i+1} := $F_{i+1} \wedge c$
- New proof obligation (if $i < k$): $(u,\ i+1)$

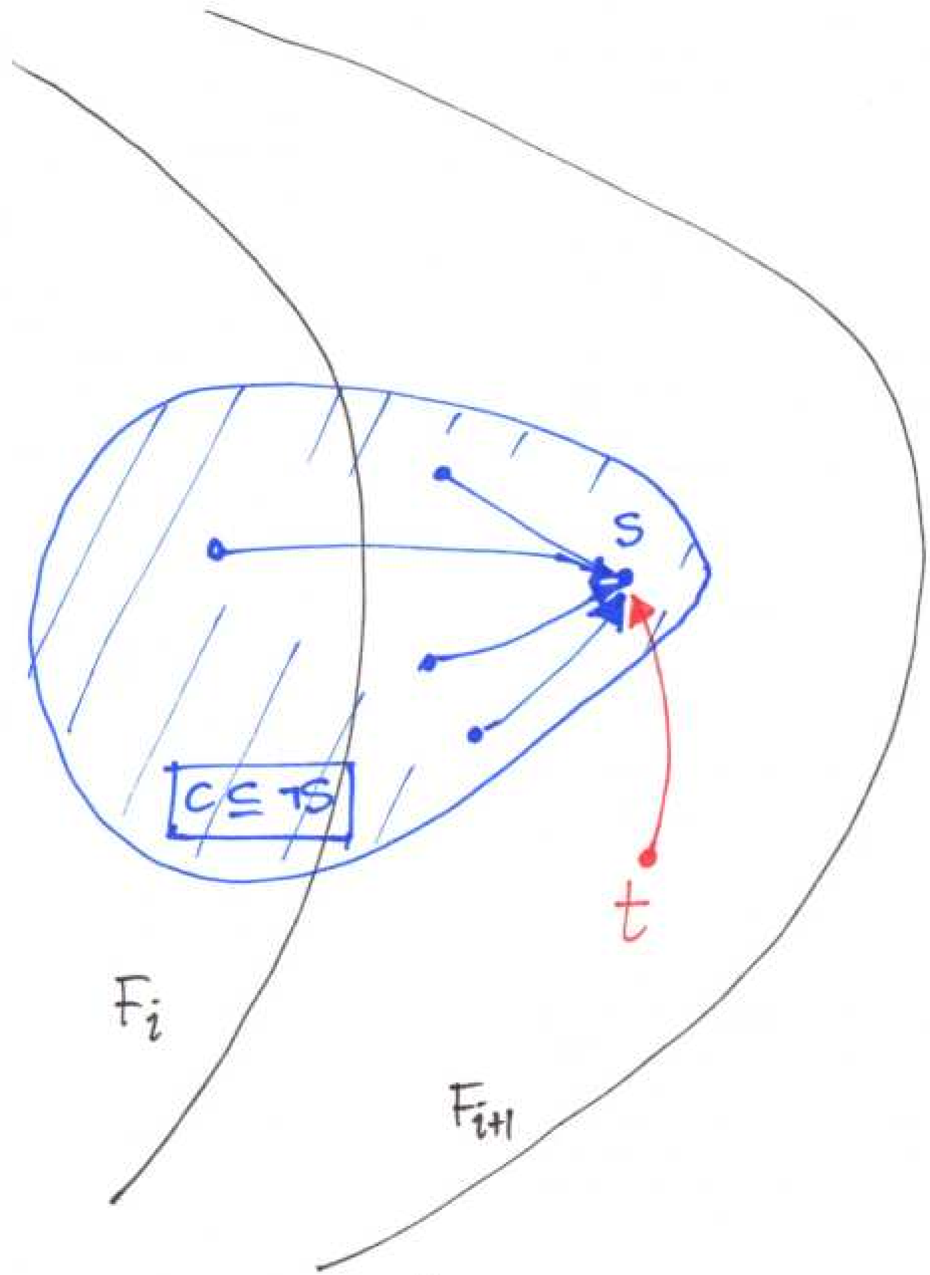
One of IC3's Insights

- Suppose CTI s was inductively generalized at F_i
  - F_{i+1} := $F_{i+1} \wedge c$
  - Removed s and some predecessors from F_{i+1}
  - Updated proof obligation: $(s,\ i+1)$

One of IC3's Insights

- Suppose CTI s was inductively generalized at F_i
  - $F_{i+1} := F_{i+1} \wedge c$
  - Removed s and some predecessors from F_{i+1}
  - Updated proof obligation: $(s,\ i+1)$
- Suppose $F_{i+1} \wedge \neg s \wedge T \not\Rightarrow \neg s'$
  - $\exists$ s-predecessor F_{i+1}-state t
  - But t was not a F_i-state
  - t is a **relevant predecessor**: the difference between F_i and F_{i+1}

Inductive generalization at F_i focuses IC3's choice of predecessors at F_{i+1}.



Relevant Predecessor

Meeting Obligations

IC3 pursues proof obligation (t, j) until $j = k$ — even if the original CTI has been addressed. Why?

- Supporting lemmas for this frontier can be useful at next
- During “predicate abstraction” phase, supporting clauses propagate forward together
- Allows IC3 to find mutually (relatively) inductive lemmas, addressing a key weakness of FSIS
- More...

IC3: A Prover

- Based on CTIs from frontier and predecessors, IC3 generates stepwise-relative inductive clauses.
- IC3 propagates clauses forward in preparing a new frontier.
  - Some clauses may be too specific.
  - Their loss can break mutual support.
- But 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:

- $u \to t \to s \to$ 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

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

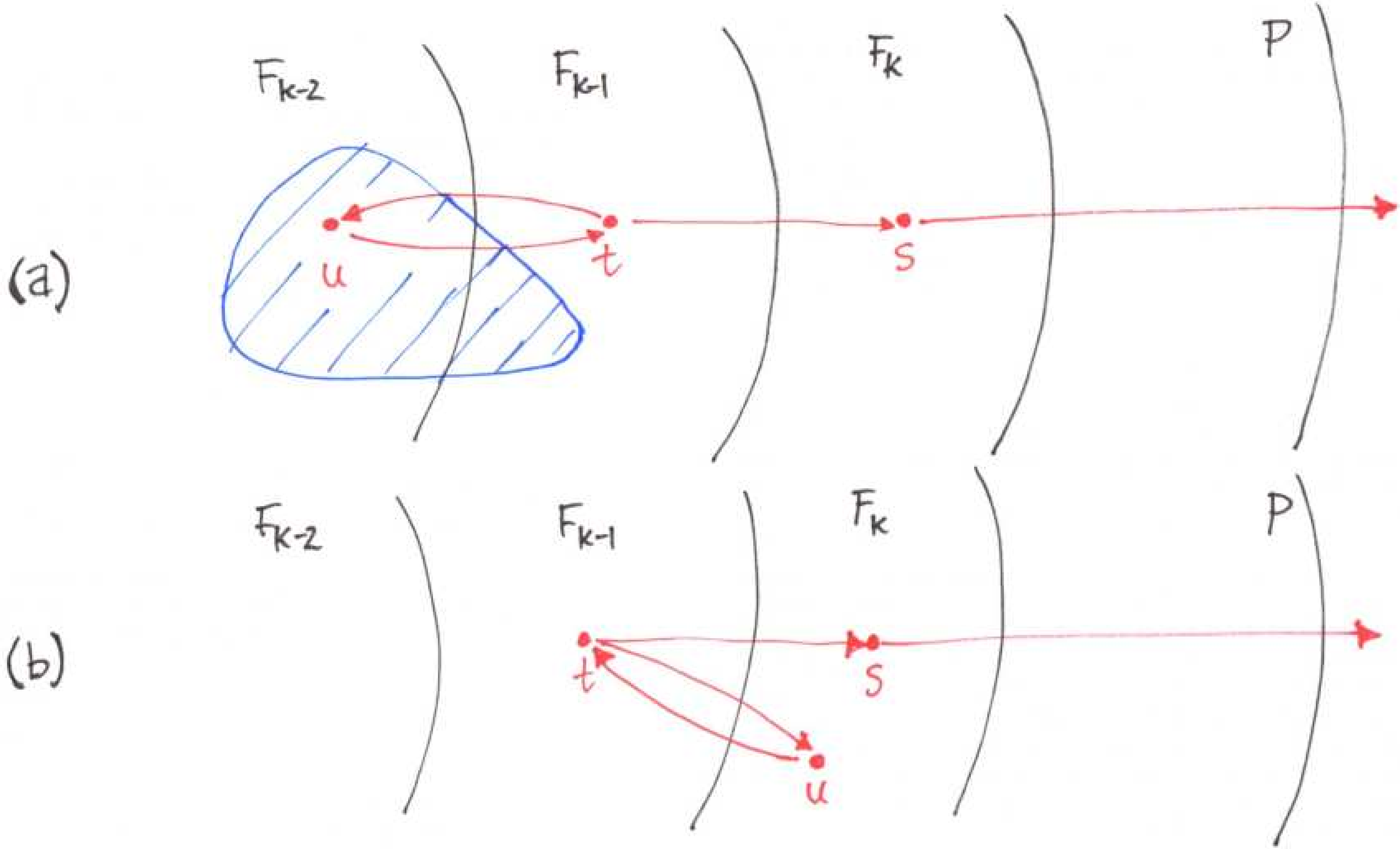
- t is the state that IC3 **considers as likeliest to be closest to an initial state**.

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

"Proximity metric"

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

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



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

Proof Obligations: Guided Search

IC3: A Bug Finder

IC3 executes a guided search.

- Proximity metric: j of (t, j)
- IC3 pursues obligation with minimal proximity
- A new clause updates the proximity metric for many states
- Same conclusion as proof perspective:
  - Pursue all proof obligations (t, j) until $j = k$
  - Now: To gain important heuristic information
  - Additionally: Allows IC3 to search deeply even for small k

Incremental, Inductive Verification

IIV Algorithm:

- Constructs concrete **hypotheses**
- Generates intermediate lemmas **incrementally**
- Applies **induction** many times
- **Generalizes** from hypotheses to strong lemmas

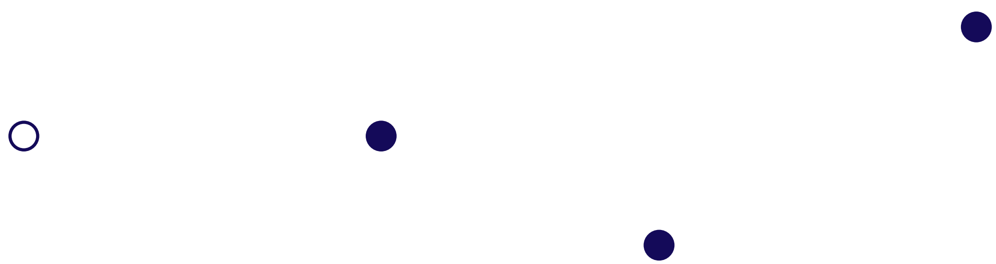
After IC3

- FAIR [Bradley et al. 2011]
  - For ω-regular properties, e.g., LTL
  - Insight: SCC-closed regions can be characterized inductively
- IICTL [Hassan et al. 2012]
  - 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

FAIR: Reachable Fair Cycles

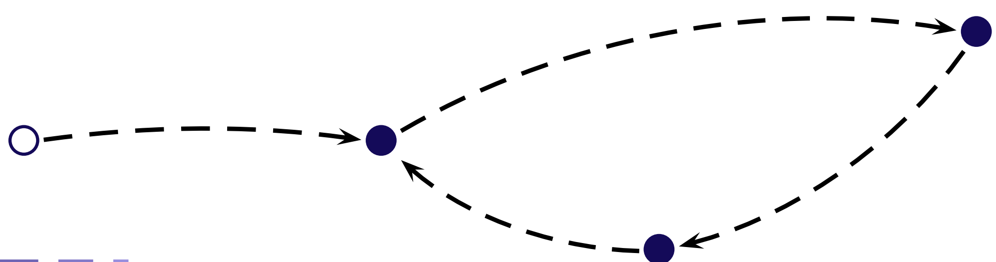
Reduce search for reachable fair cycle to a set of safety problems:

- Skeleton:



  Together satisfy all fairness constraints.

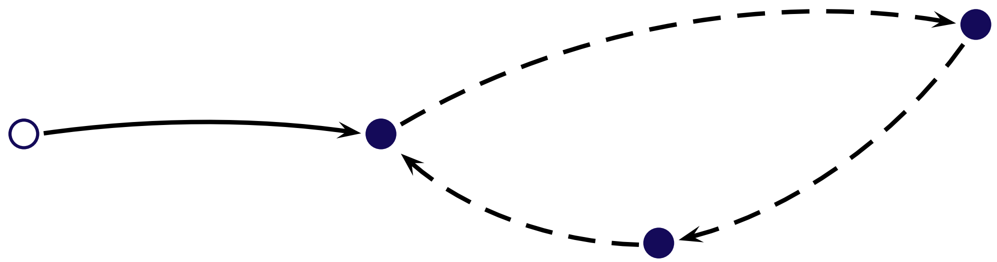
- Task: Connect states to form lasso.



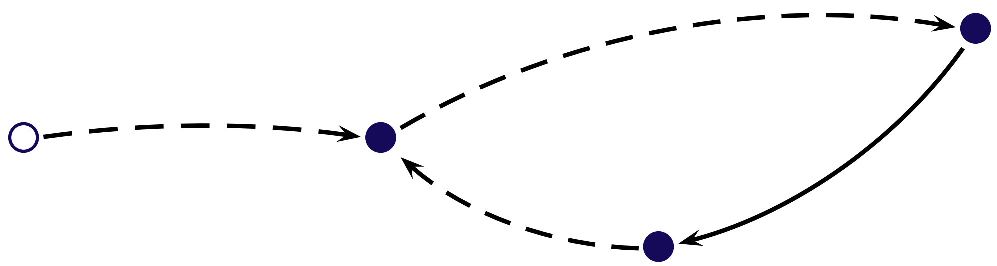
Reach Queries

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.)

IIV

	IC3	FAIR	IICTL
Hypothesis	CTI	"lasso" skeleton	task state
Lemma	clause	barrier	refinement
Induction	$\uparrow$	$\uparrow$	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 best and worst qualities of previous SAT-based model checkers
  - As a prover
  - As a bug finder
- Other IIV algorithms:
  - FAIR and IICTL
  - An indication that IC3's characteristics work in other contexts