SAT-Based Model Checking: IC3 and Lazy Abstraction

Verification course

Lecture 10, June 12, 2017

Part B

Incremental Construction of Inductive Clauses for Indubitable Correctness

or simply: IC3

A Simplified Description

"SAT-Based Model Checking without Unrolling", Aaron Bradley, VMCAI 2011

"Efficient Implementation of Property Directed Reachability",

Niklas Een, Alan Mishchenko, Robert Brayton, FMCAD 2011

Notations

- System is modeled as (V,I,T), where:
  - V is a finite set of variables
  - $I \subseteq 2^V$ is the set of initial states
  - $T \subseteq 2^V \times 2^V$ is the set of transitions

  Suitable for hardware: V is over {0, 1}

- A safety property of the form *AG* P
  - P is a propositional formula over V

Induction for proving AG P

- The simple case: P is an inductive invariant
  - $I \Rightarrow P$
  - $P \wedge T \Rightarrow P'$

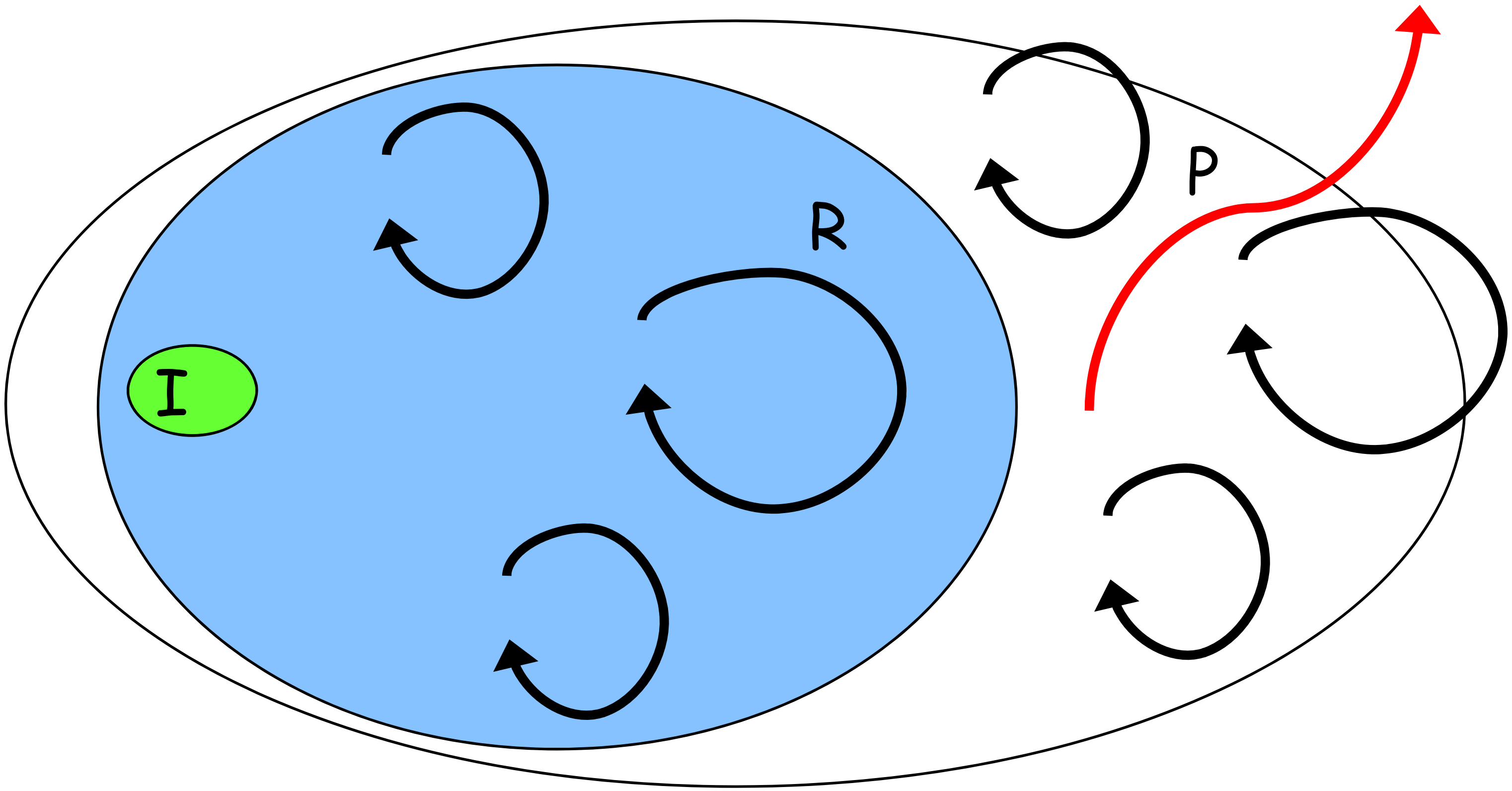
- Notation: P' – the value of P in the next state

- $I(V) \Rightarrow P(V)$
- $P(V) \wedge T(V, V') \Rightarrow P(V')$

Induction for proving AG P

- Usually, P is not an inductive invariant
- BUT – a stronger inductive invariant R may exist (strengthening)
  - I => R
  - $R \wedge T$ => R'
  - R => P
- R can be computed in various ways (BDDs, k-induction, Interpolation-Sequence,…)

Inductive invariant



IC3

- The Goal: Find an Inductive Invariant stronger than P by learning relatively inductive facts (incrementally)
  - Recall: F is inductive invariant if
    - I => F
    - $F \wedge T \Rightarrow F'$
  - If F is stronger than P, i.e., F => P, then
    - $F \wedge P \wedge T \Rightarrow F' \Rightarrow P'$

What Makes IC3 Special?

- No unrolling of the transition relation T is required

- All previous approaches require unrolling
  - Searching for an inductive invariant
  - Unrolling = A form of strengthening

- IC3 strengthens in a different way
  - Learning relatively inductive facts locally

IC3 Basics

- Iteratively compute Over-Approximated Reachability Sequence (OARS) $\langle F_0, F_1, \ldots, F_k \rangle$ s.t.
  - $F_0 = INIT$
  - $F_i \Rightarrow P$: P is an invariant up to k
  - $F_i \Rightarrow F_{i+1}$: $F_i \subseteq F_{i+1}$
  - $F_i \wedge T \Rightarrow F'_{i+1}$: Simulates one forward step

F_i - over-approximates the set of states reachable within i steps

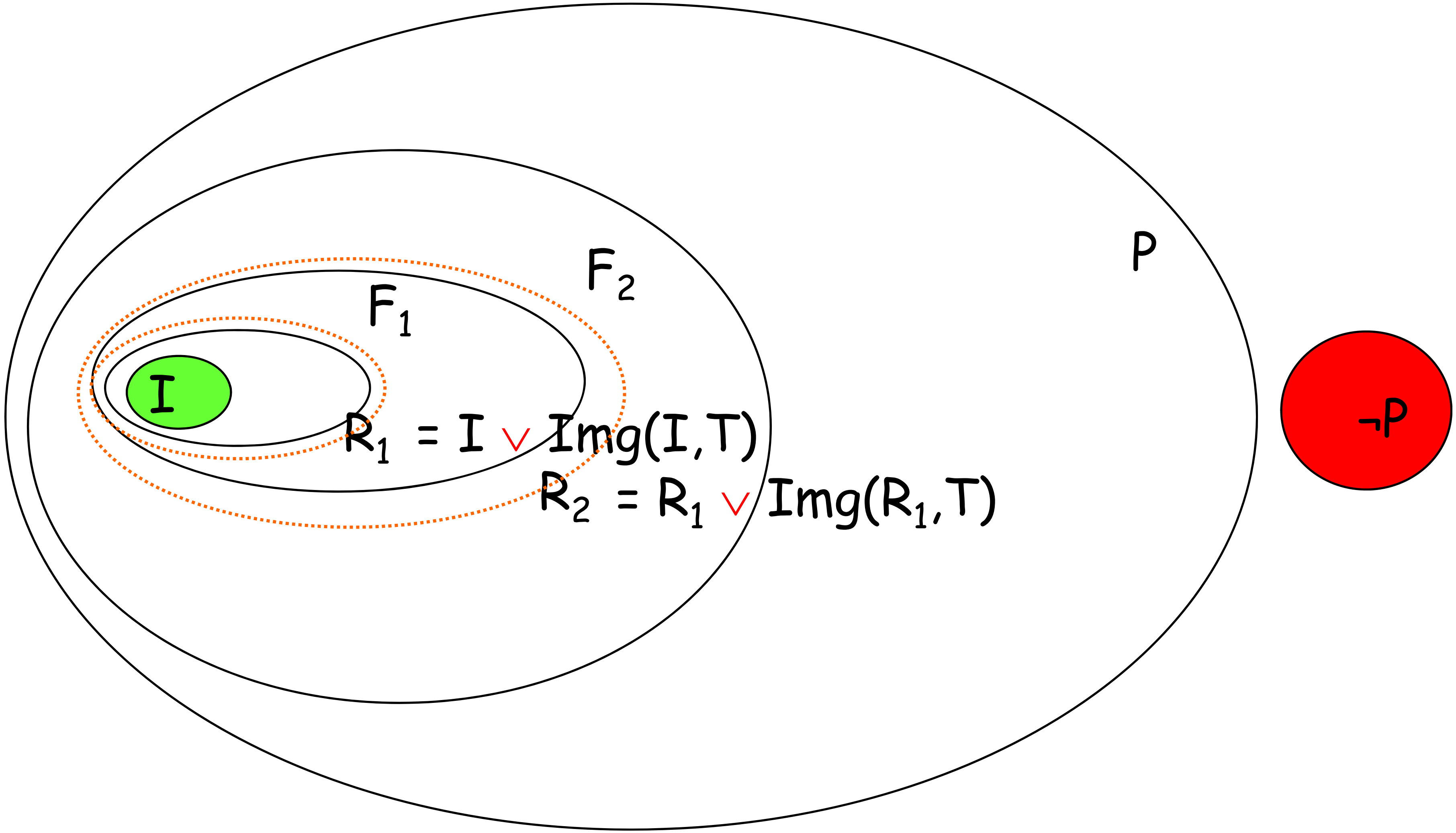
- If $F_{i+1} \Rightarrow F_i$ then fixpoint

IC3 Basics

- P is inductive relative to F if
  - $I \Rightarrow P$
  - $F \wedge P \wedge T \Rightarrow P'$

- Notations:
  - Cube s: conjunction of literals
    - $v_1 \wedge v_2 \wedge \neg v_3$ - Represents a state
  - s is a cube => **¬s is a clause** (DeMorgan)

OARS



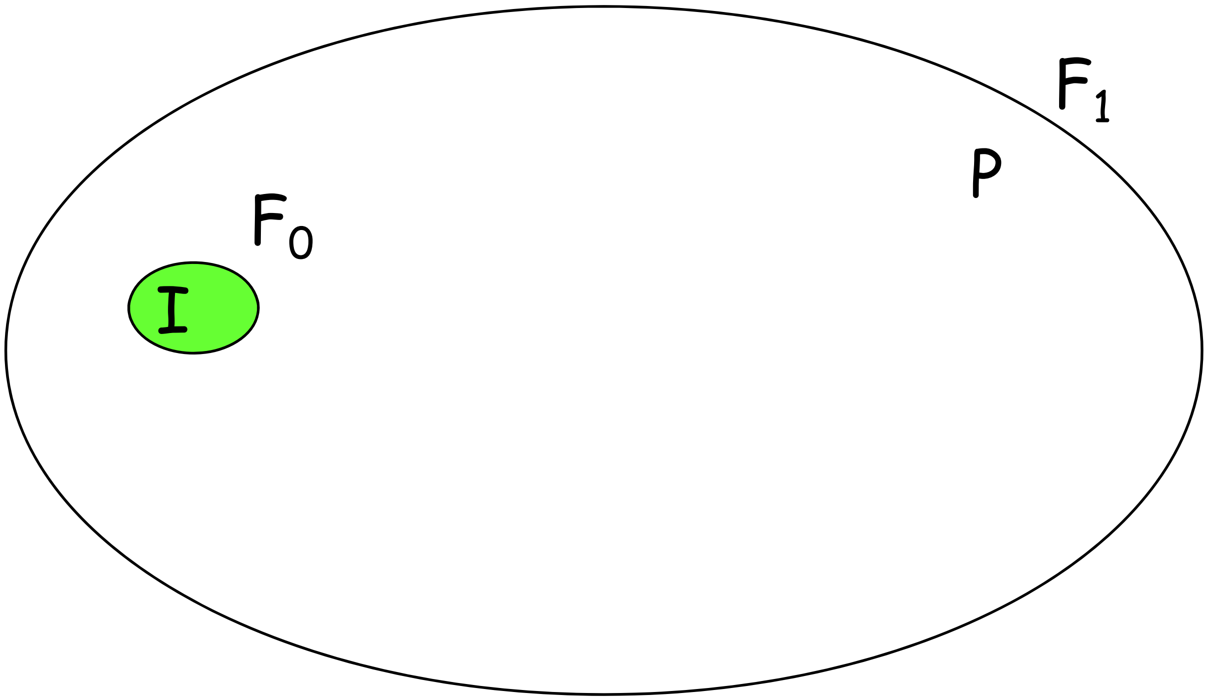
A Backward Search

- Search for a predecessor **s** to some error state: $P \wedge T \wedge \neg P'$
  - If none exists, property P holds:
    - $(P \wedge T \wedge \neg P')$ unsat IFF $(P \wedge T \Rightarrow P')$ valid
- Otherwise, try to block s
  - $P = P \wedge \neg s$
  - BUT, first need to show the s is not reachable

IC3 - Initialization

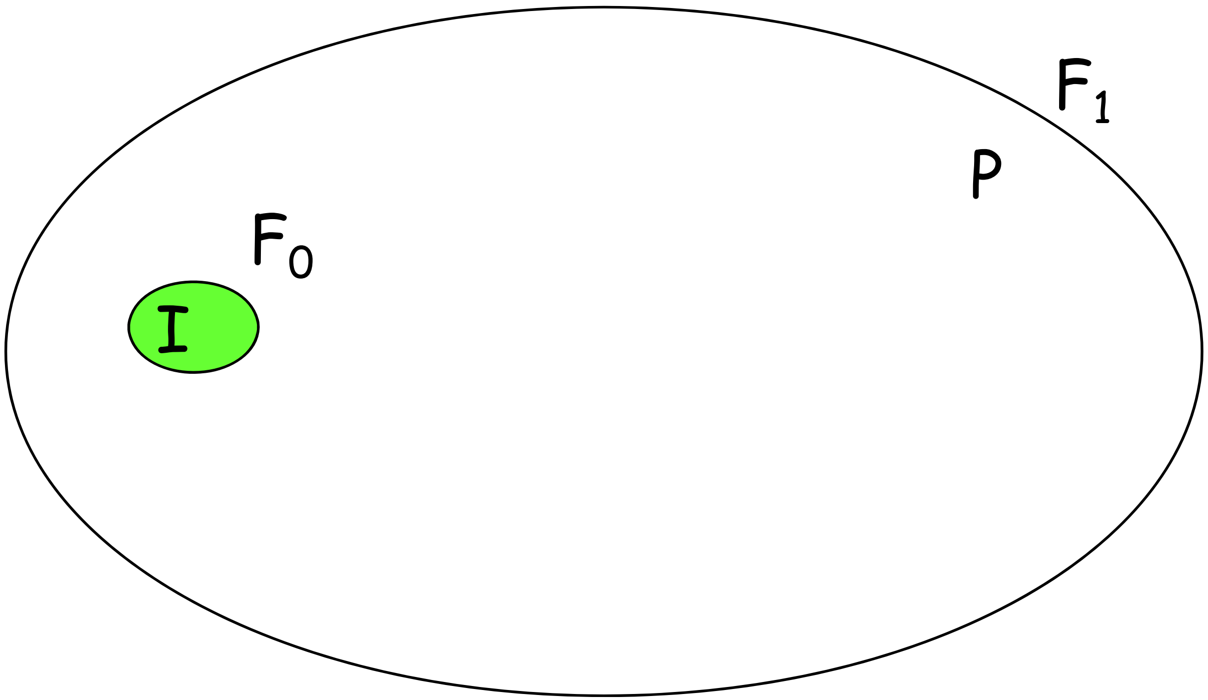
- Check satisfiability of the two formulas:
  - $I \wedge \neg P$
  - $I \wedge T \wedge \neg P'$
- If both are unsatisfiable then:
  - $I \Rightarrow P$
  - $I \wedge T \Rightarrow P'$
- Therefore
  - $F_0 = I$, $F_1 = P$
    - $\langle F_0, F_1 \rangle$ is OARS

IC3 - Initialization



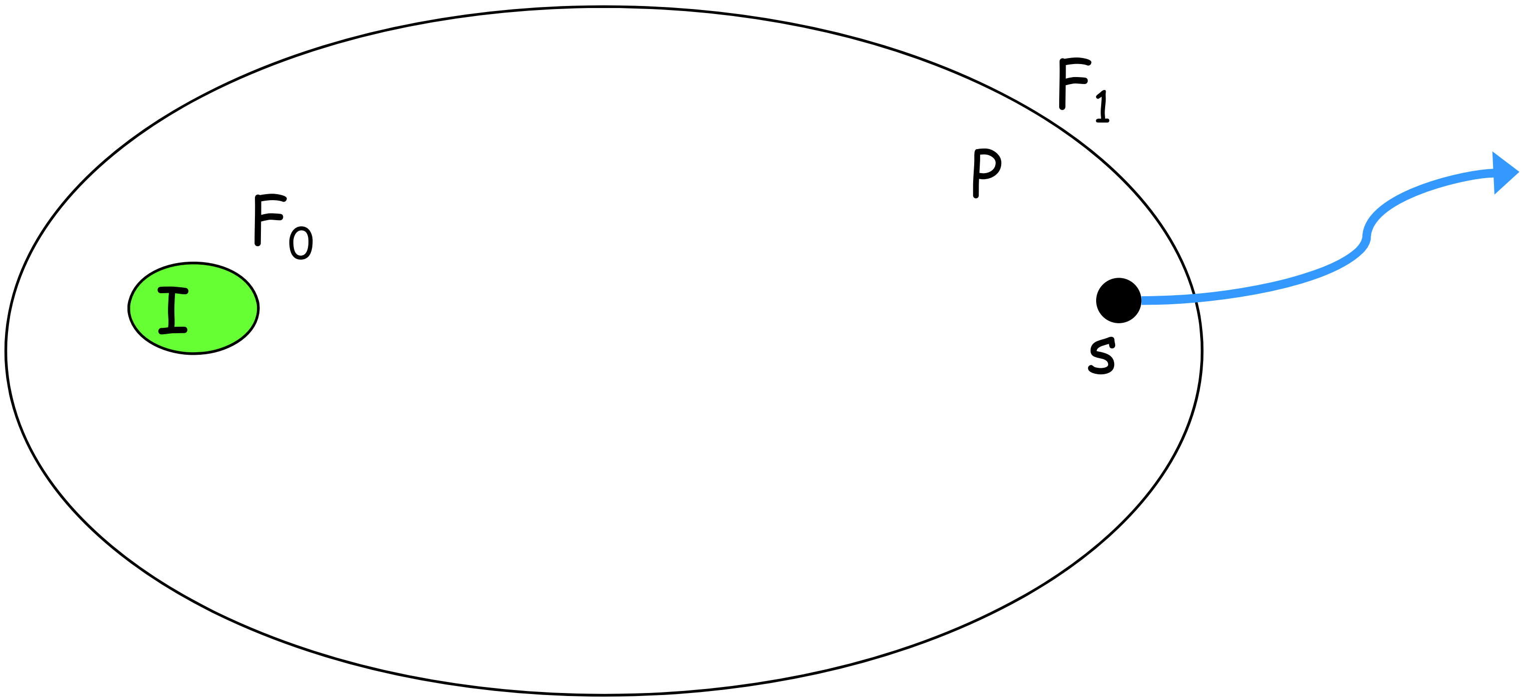
IC3 - Iteration

- Our OARS contains F_0 and F_1
  - If P is an inductive invariant - done! ☺
  - Otherwise:
    - F_1 should be strengthened



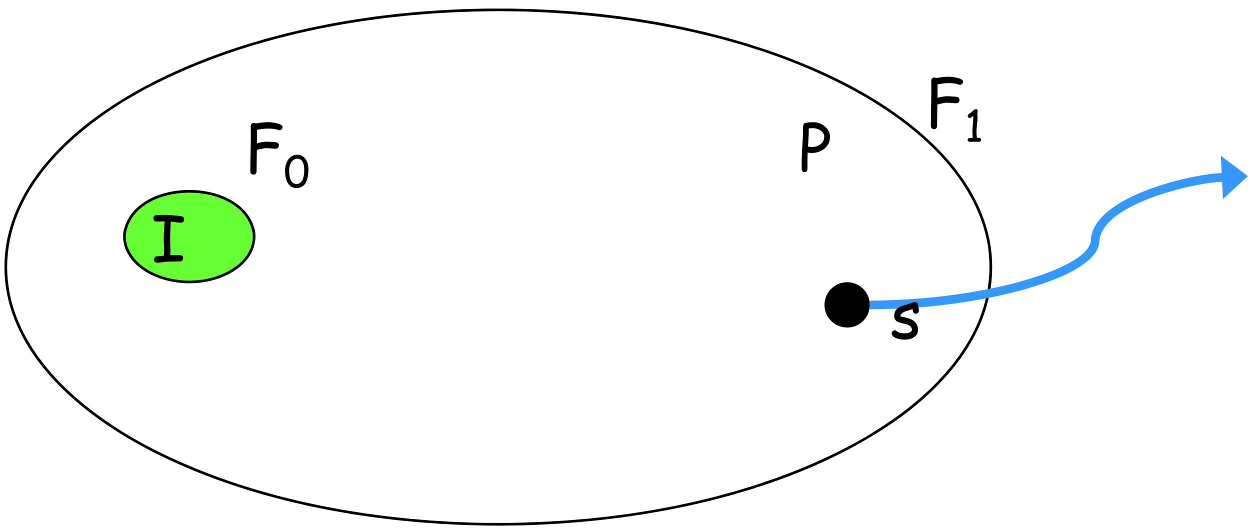
IC3 - Iteration

- P is not an inductive invariant
  - $F_1 \wedge T \wedge \neg P'$ is satisfiable
  - From the satisfying assignment get the state s that can reach the bad states



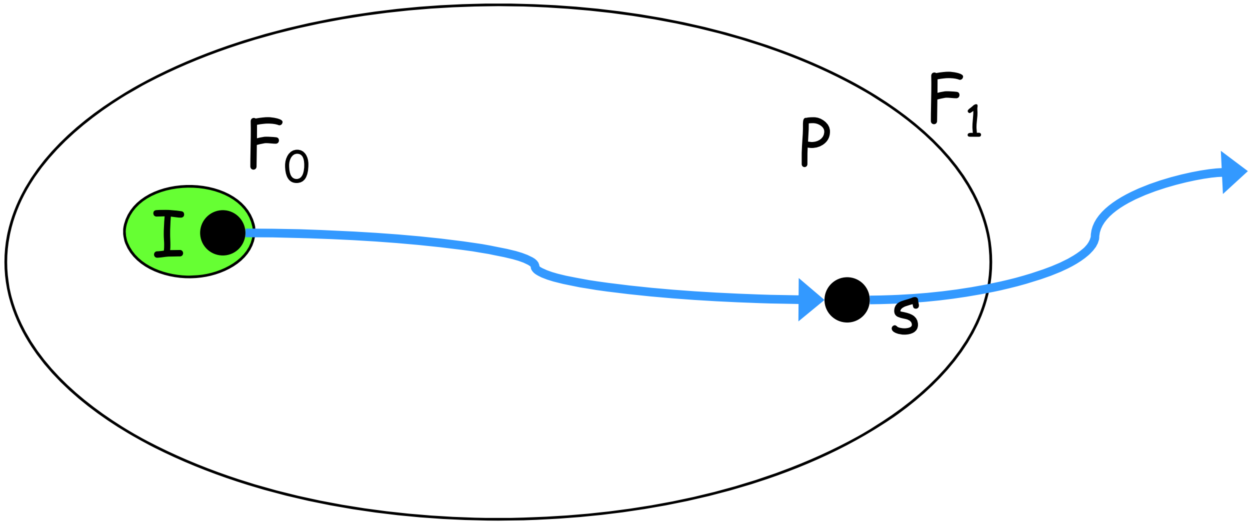
IC3 - Iteration

- Is s reachable or not?
  - Hard to know
  - If it is reachable a CEX exists
    - Why?



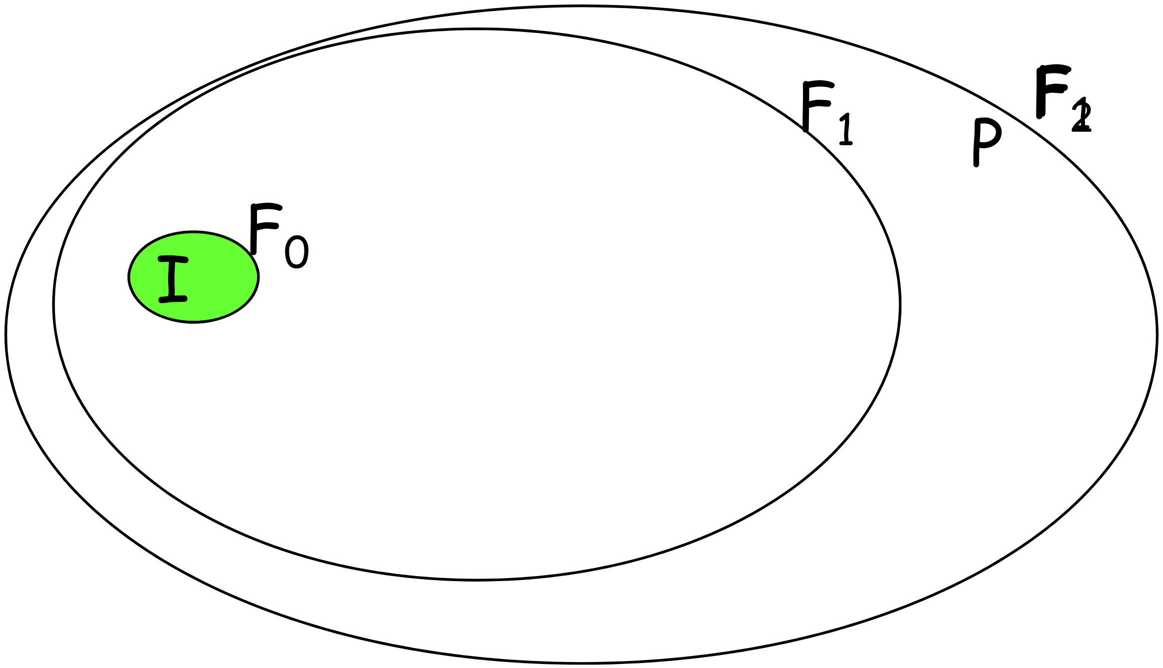
IC3 - Iteration

- Is s reachable in one transition from the previous set? (Bounded reachability)
  - Check $F_0 \wedge T \wedge s'$
  - If satisfiable, s is reachable from F_0 (CEX)
  - Otherwise, block it = remove it from F_1
    - $F_1 = F_1 \wedge \neg s$



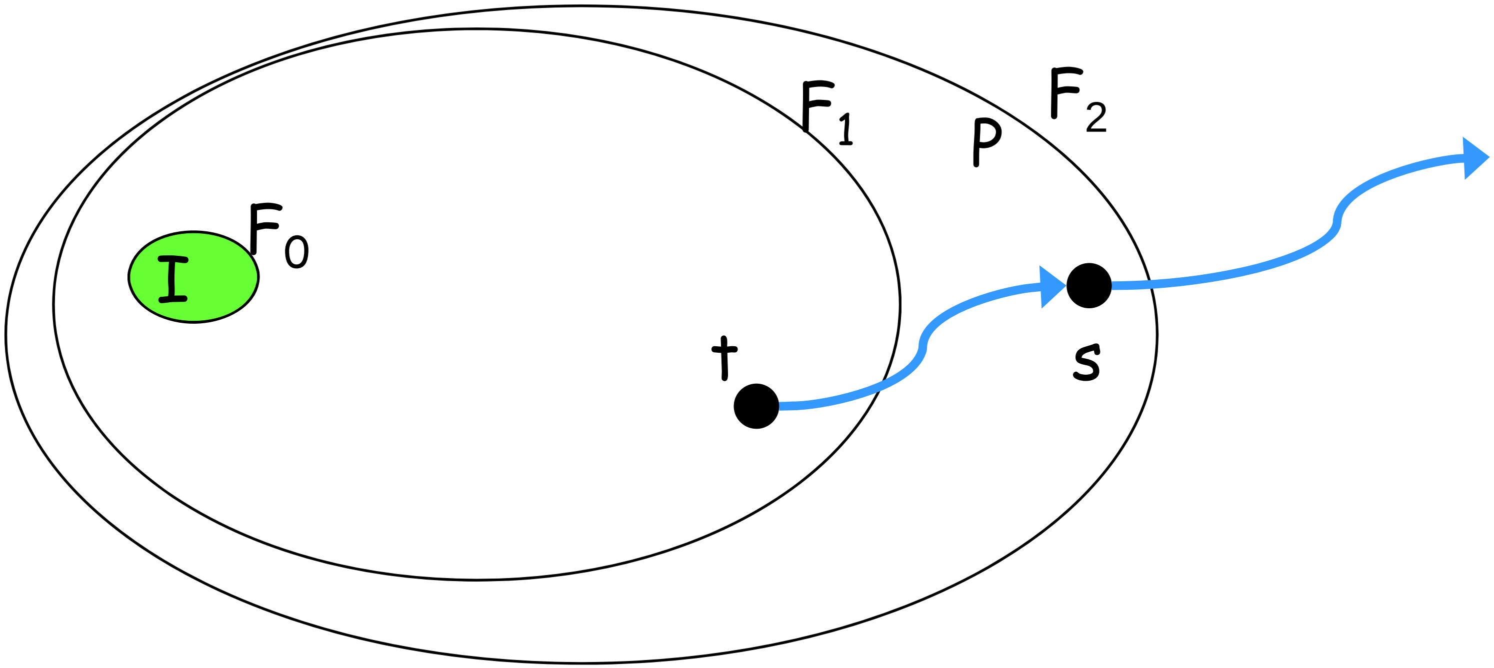
IC3 - Iteration

- Iterate this process until $F_1 \wedge T \wedge \neg P'$ becomes unsatisfiable
  - $F_1 \wedge T \Rightarrow P'$ holds
  - F_2 can be defined to be P
    - Any problems/issues with that?



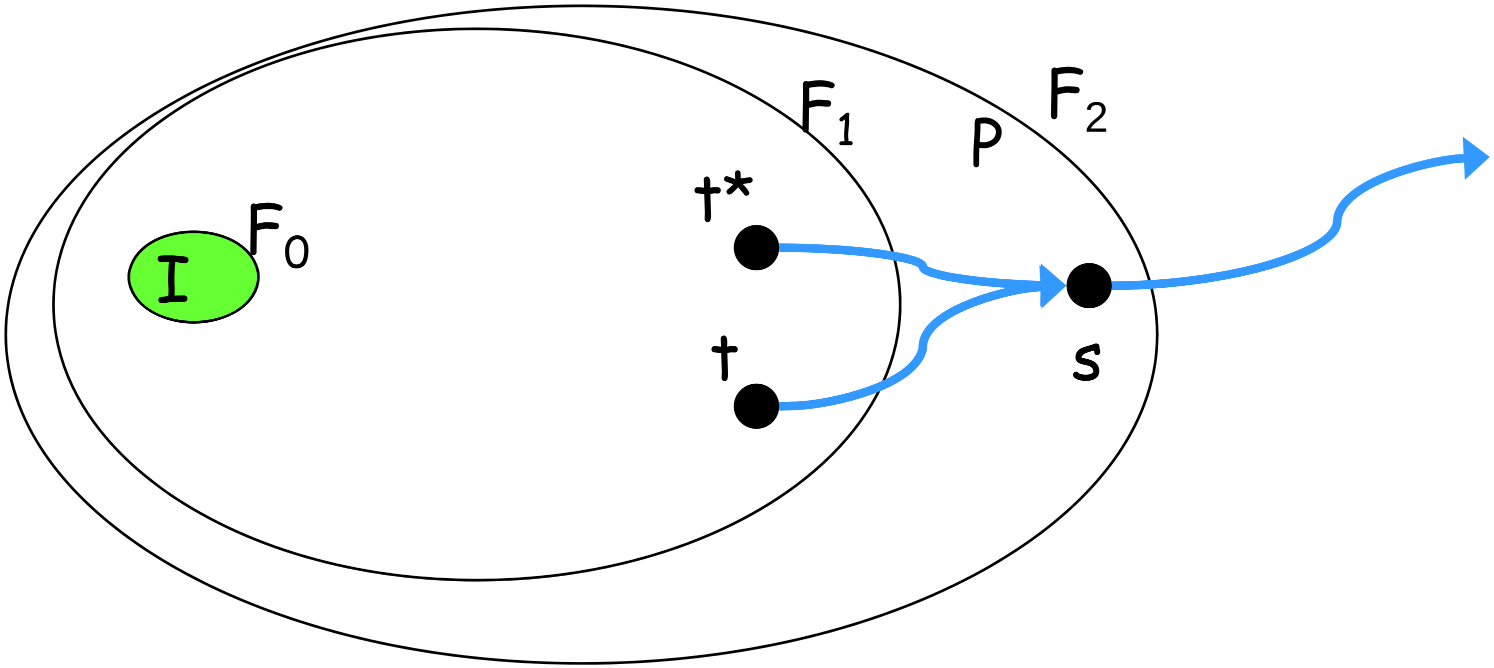
IC3 - Iteration

- New iteration, check $F_2 \wedge T \wedge \neg P'$
  - If satisfiable, get s that can reach ¬P
  - Now check if s can be reached from F_1 by $F_1 \wedge T \wedge s'$
  - **If it can be reached, get t and try to block it**

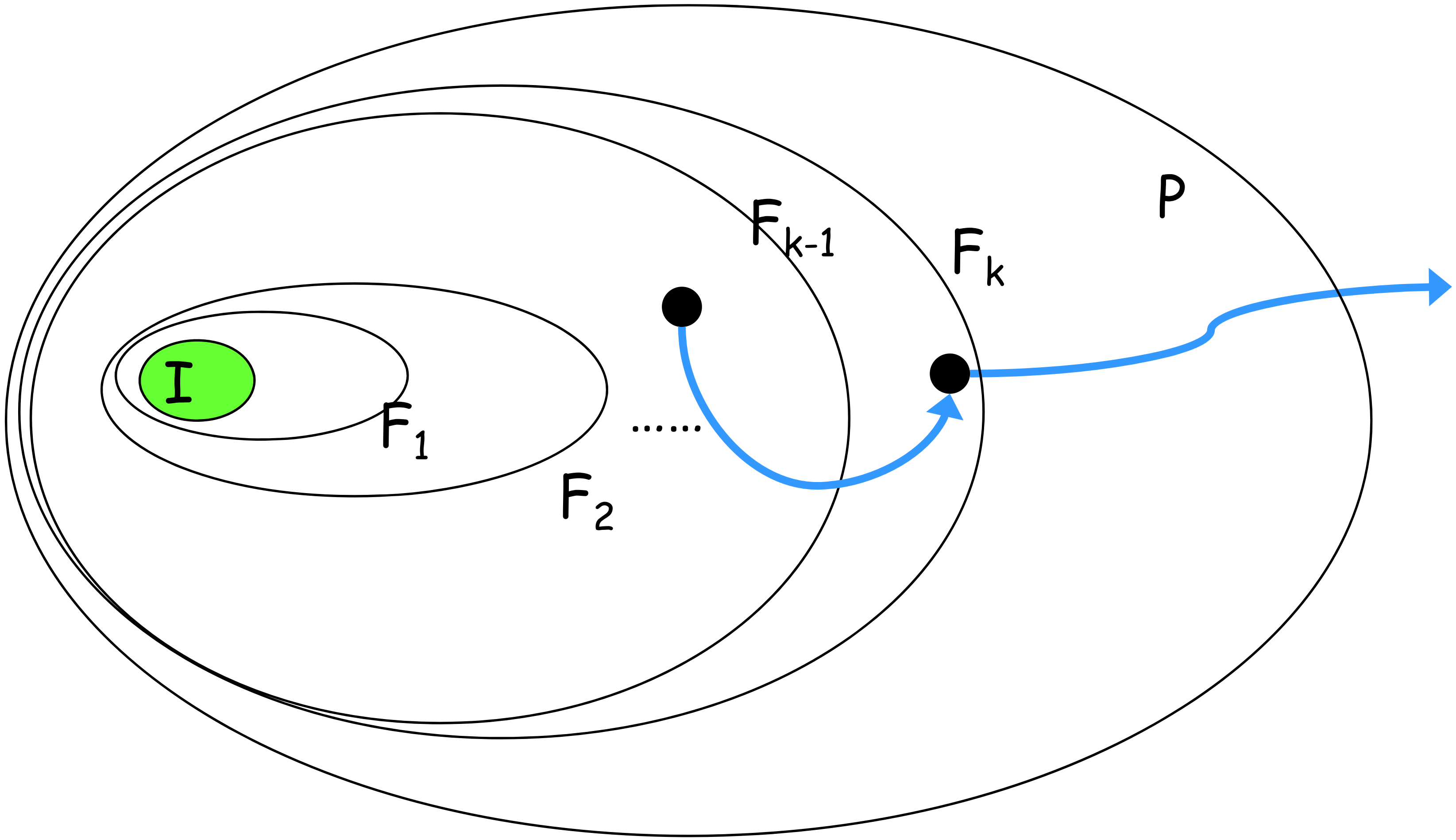


IC3 - Iteration

- To block t, check $F_0 \wedge T \wedge t'$
  - If satisfiable, a CEX
  - If not, t is blocked, get a "new" t by $F_1 \wedge T \wedge s'$
  - **If it can be reached, get t* and try to block it**
  - **……You get the picture ☺**



General Iteration



IC3 - Iteration

- Given an OARS $\langle F_0, F_1, \ldots, F_k \rangle$, define $F_{k+1}=P$
- Apply a backward search
  - Find predecessor s in F_k that can reach a bad state
    - Check $F_k \wedge T \wedge \neg P'$
  - If none exists ($F_k \wedge T \Rightarrow P'$), move to next iteration
  - If exists, try to find a predecessor t to s in F_{k-1}
    - $(F_{k-1} \wedge T \wedge s')$
  - If none exists ($F_{k-1} \wedge T \Rightarrow \neg s'$), s is removed from F_k
    - $F_k = F_k \wedge \neg s$
  - Otherwise: Recur on (t, F_{k-1})
    - We call $(t,k-1)$ a proof obligation
- If we can reach I, a CEX exists

That Simple?

- Looks simple
- But this "simple" solution does NOT work
- It amounts to States Enumeration
  - Too many states…
- Does IC3 enumerate states?
  - In general - No.
    It applies generalization for removing more than one state at a time
  - Sometimes, yes (when IC3 does not perform well)

Generalization

Consider the case:

- State s in F_k can reach a bad state in one transition
- s in not reachable (in k transitions):
  - Therefore $F_{k-1} \wedge T \Rightarrow \neg s'$ holds
- We want to generalize this fact
  - s is a single state
  - Goal: Find a set of states, unreachable in k transitions

Generalization

- We know $F_{k-1} \wedge T \Rightarrow \neg s'$
- And, $\neg s$ is a clause
- Generalization: Find a sub-clause $c \subseteq \neg s$ s.t. $\mathbf{F_{k-1} \wedge T \Rightarrow c'}$
  - Sub clause means less literals
  - Less literals implies less satisfying assignments
    - $(a \vee b \vee c)$ vs. $(a \vee b)$
  - $c \Rightarrow \neg s$ – c is a stronger fact
- $F_k = F_k \wedge c$
  - More states are removed from F_k, making it stronger/more precise (closer to R_k)

Generalization

- How do we find a sub-clause $c \subseteq \neg s$ s.t. $\mathbf{F_{k-1} \wedge T => c'}$?

Options:

1. Trial and Error
   - Try to remove literals from $\neg s$ while $\mathbf{F_{k-1}} \wedge \mathbf{T} \wedge \neg c'$ remains unsatisfiable
2. Use the UnSAT Core
   - $\mathbf{F_{k-1}} \wedge \mathbf{T} \wedge s'$ is unsatisfiable

Observation 1

- Assume a state s in F_k can reach a bad state in one transition
- Important Fact: **s is not in F_{k-1} (!!)**
  - $F_{k-1} \wedge T \Rightarrow F_k$
  - $F_k \Rightarrow P$
  - If s was in F_{k-1} we would have found it in an earlier iteration
- Therefore: $F_{k-1} \Rightarrow \neg s$

Inductive Generalization

- Assume a state s in F_k can reach a bad state in one transition
- Assume s is not reachable (in k transitions):
  - We get $F_{k-1} \wedge T \Rightarrow \neg s'$ holds
- BUT, this is equivalent: $F_{k-1} \wedge \neg s \wedge T \Rightarrow \neg s'$
  - Since $F_{k-1} \Rightarrow \neg s$
- This looks familiar!
  - $I \Rightarrow \neg s$
    - Otherwise, CEX! ($I \not\Rightarrow \neg s \Leftrightarrow$ s is in I)
  - $\neg s$ is inductive relative to F_{k-1}

Inductive Generalization

- Find $c \subseteq \neg s$ s.t.
  $F_{k-1} \wedge c \wedge T \Rightarrow c'$ and $I \Rightarrow c$ hold

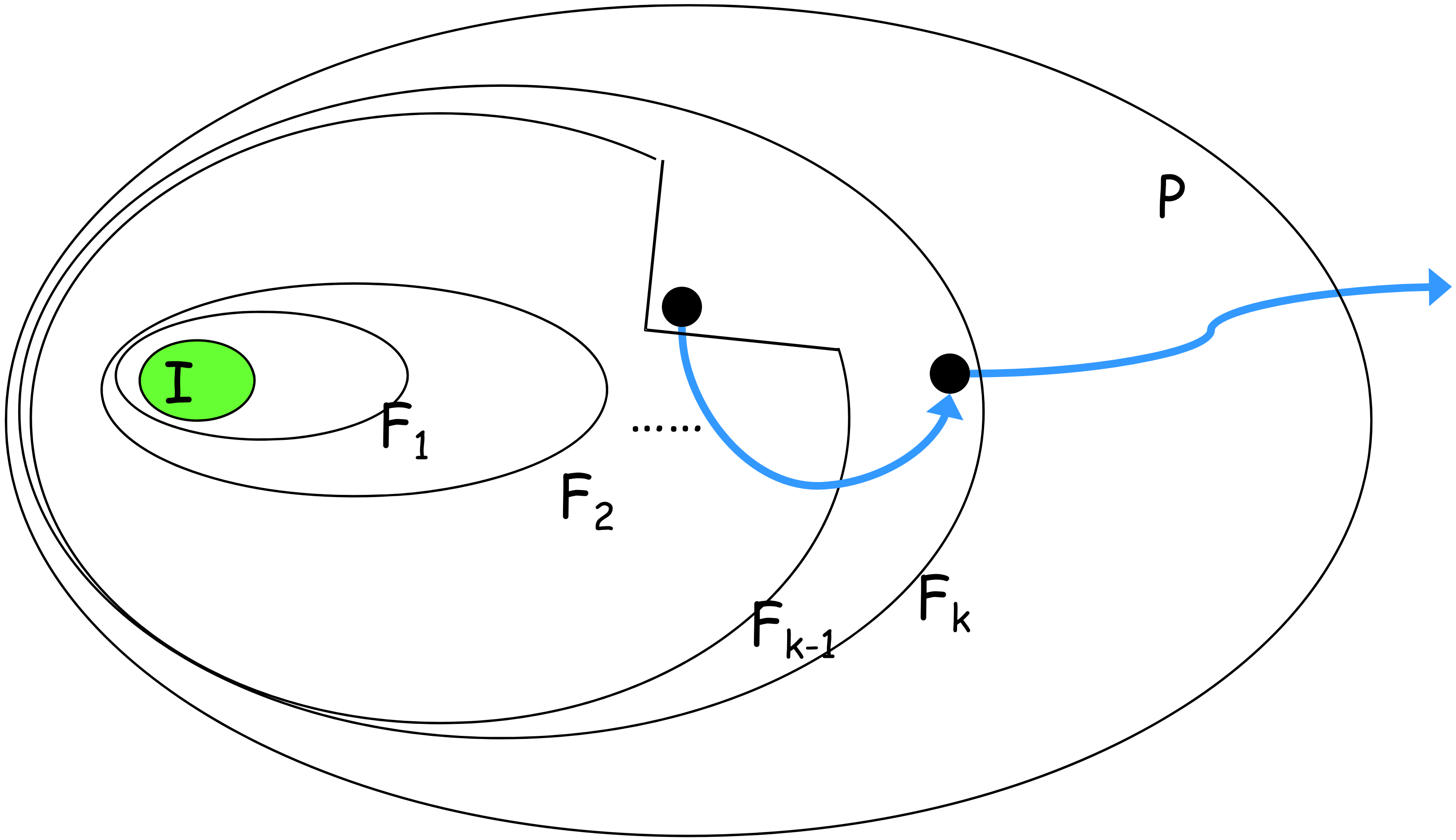
- Define $F_k^* = F_k \wedge c$

- Since $F_i \Rightarrow F_{i+1}$,
  c is inductive relative to $F_{k-1}, F_{k-2}, \ldots, F_0$
  - Add c to all of these sets
  - $F_i^* = F_i \wedge c$
- $F_i^* \wedge T \Rightarrow F_{i+1}^*$ hold

Observation 2

- Assume a state s in F_i can reach a bad state in a number of transitions
- s is also in F_j for j > i, since F_i => F_j
- a longer CEX may exist
  - s may not be reachable in i steps, but it may be reachable in j steps
- If s is blocked in F_i, it must be blocked in F_j for j > i
  - Otherwise, a CEX exists

Push Forward



Push Forward - summary

- s is removed from F_i
  - by conjoining a sub-clause c:
    $F_i = F_i \wedge c$
- c is a clause learnt at level i
  Try to push it forward to j >= i
  - If $F_j \wedge T \Rightarrow c'$ holds
    - c is implied by F_j in level j+1,
      $F_{j+1} = F_{j+1} \wedge c$
  - Else: s was not blocked at level j > i
    - Add a proof obligation (s,j)
    - If s is reachable from I, CEX!

IC3 – Key Ingredients

- Backward Search
  - Find a state s that can reach a bad state in a number of steps
  - s may not be reachable (over-approximations)
- Block a State
  - Do it efficient, block more than s
    - Generalization
- Push Forward
  - An inductive fact at frame i may also be inductive at higher frames
  - If not, a longer CEX is found

IC3 – High Level Algorithm

```
If I ∧ ¬P is SAT return false; // CEX
If I ∧ T ∧ ¬P' is SAT return false; // CEX
OARS = <I,P>;  // <F0,F1>
k=1
while (OARS.is_fixpoint() == false) do
    while (Fk ∧ T ∧ ¬P' is SAT) do
        s = get_state();
        If (block_state(s, k) == false) return cex; //
        recursive function
    extend(OARS);
    push_forward();
return valid;
```