Understanding IC3

SAT-Based Model Checking Without Unrolling

Predicate abstraction

- A state is variables assignment $x_1 = v_1 \land x_2 = v_2 \land \cdots \land x_n = v_n$.
- A state space is $X_1 \times X_2 \times \cdots \times X_n$.
- Given a logic formula φ such that $\mathrm{Var}(\varphi) \in \{x_1, \cdots, x_n\}$, then it can split the state space into two distinct parts φ and $\neg\varphi$, i.e., $\varphi = x_1 > 0$ and $\neg\varphi = x_1 < 0$.
- When n is large, the state space is huge, $\mathrm{i.e.}, |D|^n$ where D is minimum domain of variables. We can not analyze huge state space due to time and machine.
- Thus we have to model the state space to a proper abstract state space, which we can run our analysis on it and preserve the correctness.

Predicate abstraction

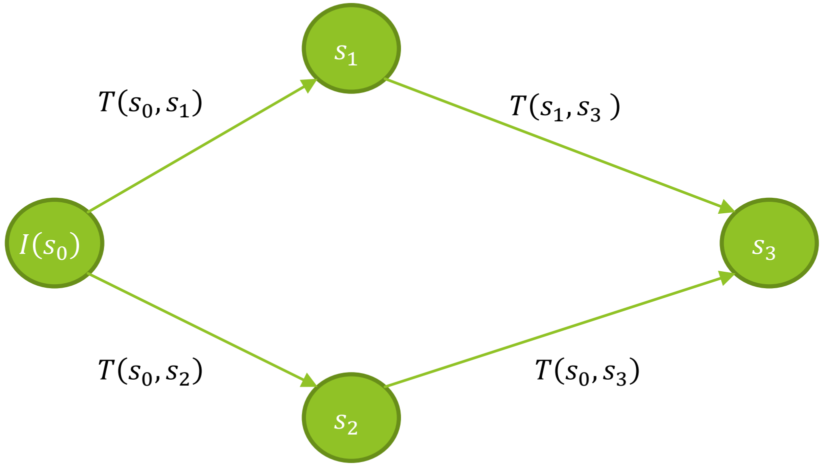
- A Boolean formula consists of Boolean variables which can be assigned to true or false.
- Given some logic formulas $\varphi_1, \varphi_2, \cdots, \varphi_m$ over state space $X_1 \times X_2 \times \cdots \times X_n$. We can construct different abstract states by combine $\varphi_1, \varphi_2, \cdots, \varphi_m$ with logic connectives i.e., $\wedge, \vee, \neg$. For example, let $\varphi_1 = x_1 > 0$, $\varphi_2 = x_2 > 0$, then $s_1 = \varphi_1 \wedge \varphi_2$, $s_2 = \varphi_1 \wedge \neg\varphi_2$.
- In formal, we often call $\varphi_1, \varphi_2, \cdots, \varphi_m$ as predicates, an abstract state space consists of abstract states.
- True is whole state space, false means contradiction.

Predicate abstraction

- We can control the predicates to balance the correctness and effectiveness.
- Some examples:
  - Safety property: Given an state space, we want to verify that all states satisfied $x_1 > 1$. We can use predicate $\varphi = x_1 > 2$ to model original state space, if we verified all states satisfy φ, then we know all states also satisfy $x_1 > 1$.
  - Liveness property: Given an state space, we want to verify that there is a state satisfied $x_1 = 1$. We can also use predicate $\varphi = x_1 > 2$ to model original state space, if we verified all states satisfy φ, the we know there is no state satisfied $x_1 = 1$.

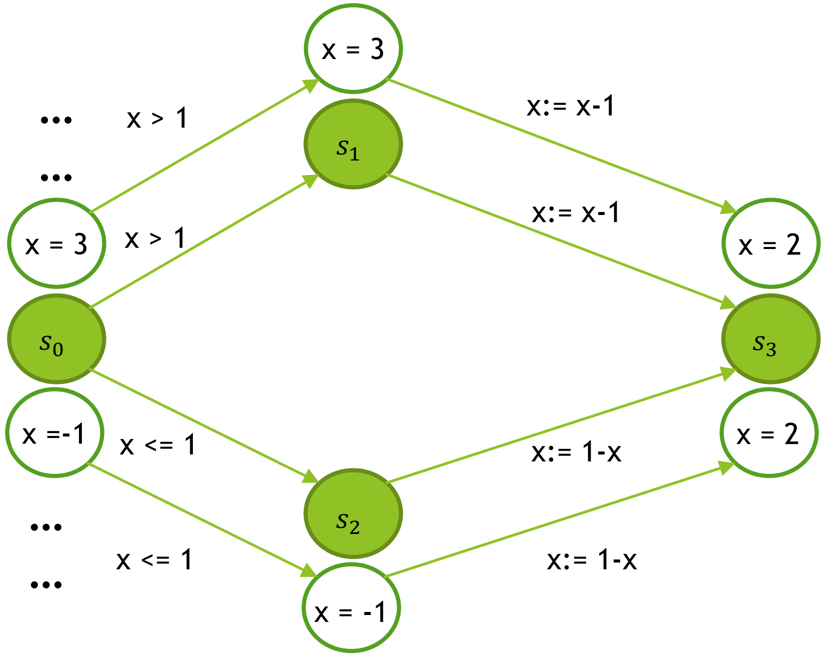
Transition system or automata

- $\mathcal{T} = (S, I, T)$ where S is state space, I is initial state predicate, T is states transition relation predicate.



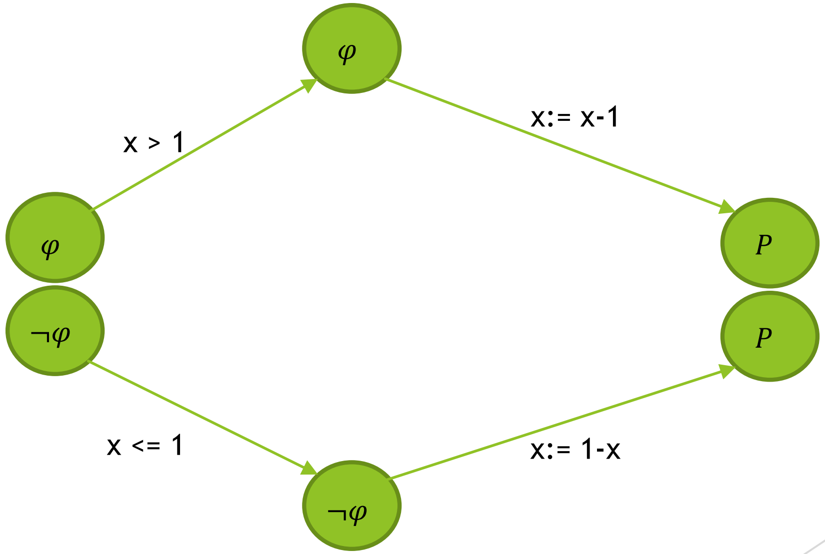
Transition system with predicate abstraction

Original



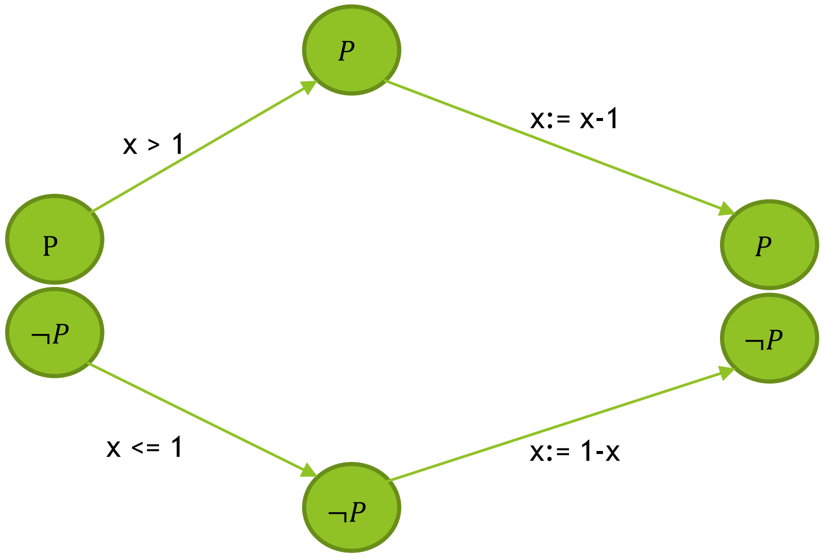
Abstract

Let $\varphi = \ x \ > 1, \, P \ = x^{s_3} \geq 0.$

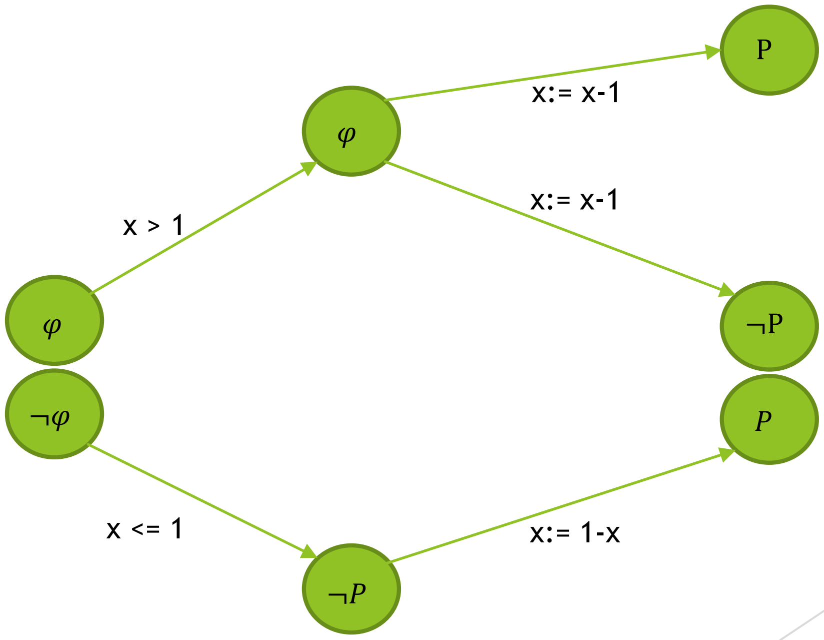


Counterexample

Let $P = x^{S_3} \geq 0$.



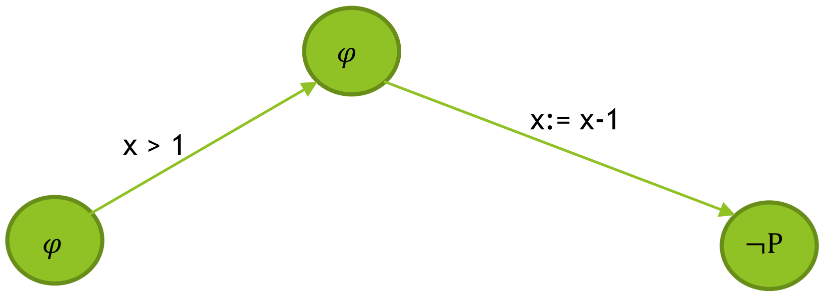
Let $\varphi = x \geq 0$, $P = x^{S_3} \geq 0$.



Refinement

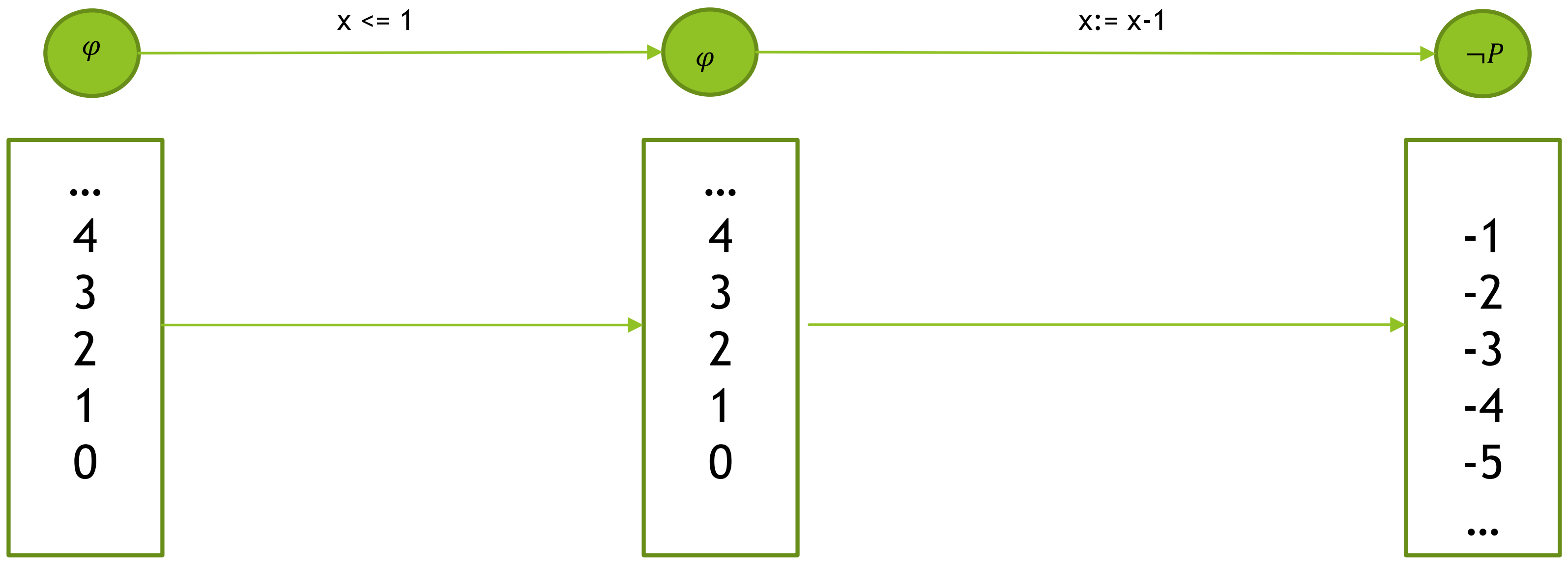
- Reabstract state space to eliminate counterexamples.
- Extract predicates from transition system.
- Learn from counterexamples.

$\varphi = x \geq 0,\ P = x^{S_3} \geq 0$



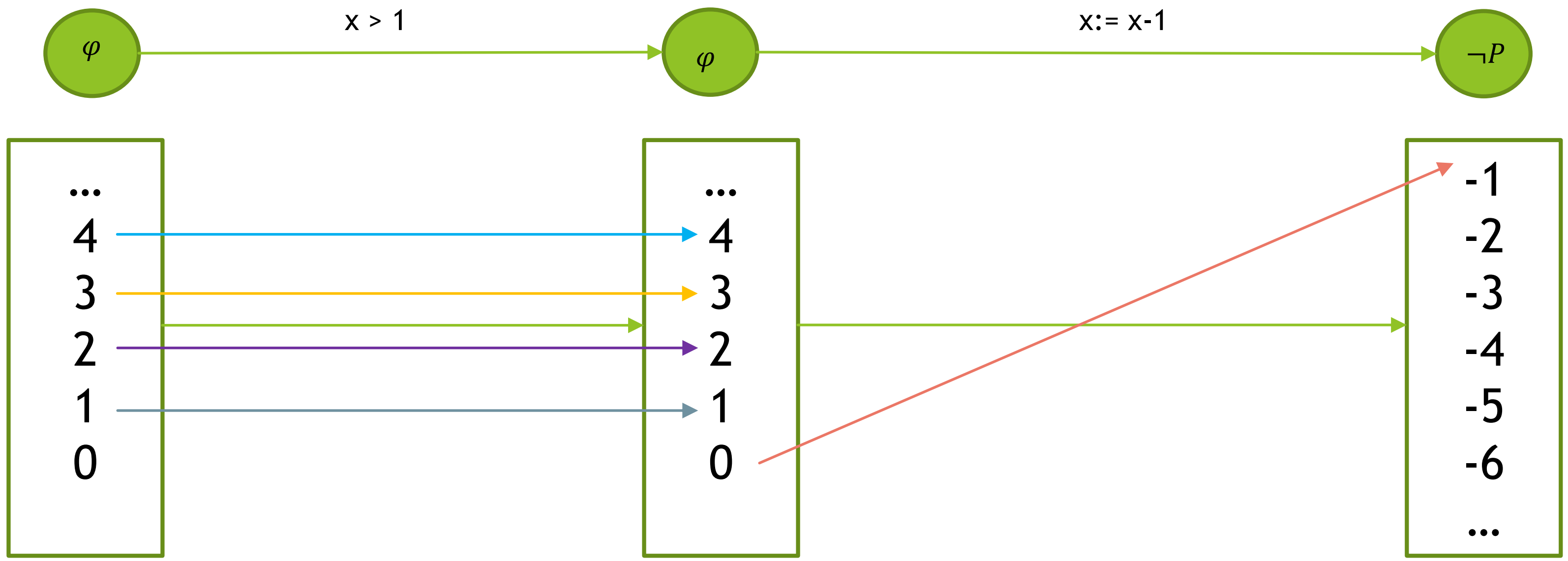
Counterexample-guided abstraction refinement (CEGAR)

$\varphi = x \geq 0,\ P = x^{S_3} \geq 0$



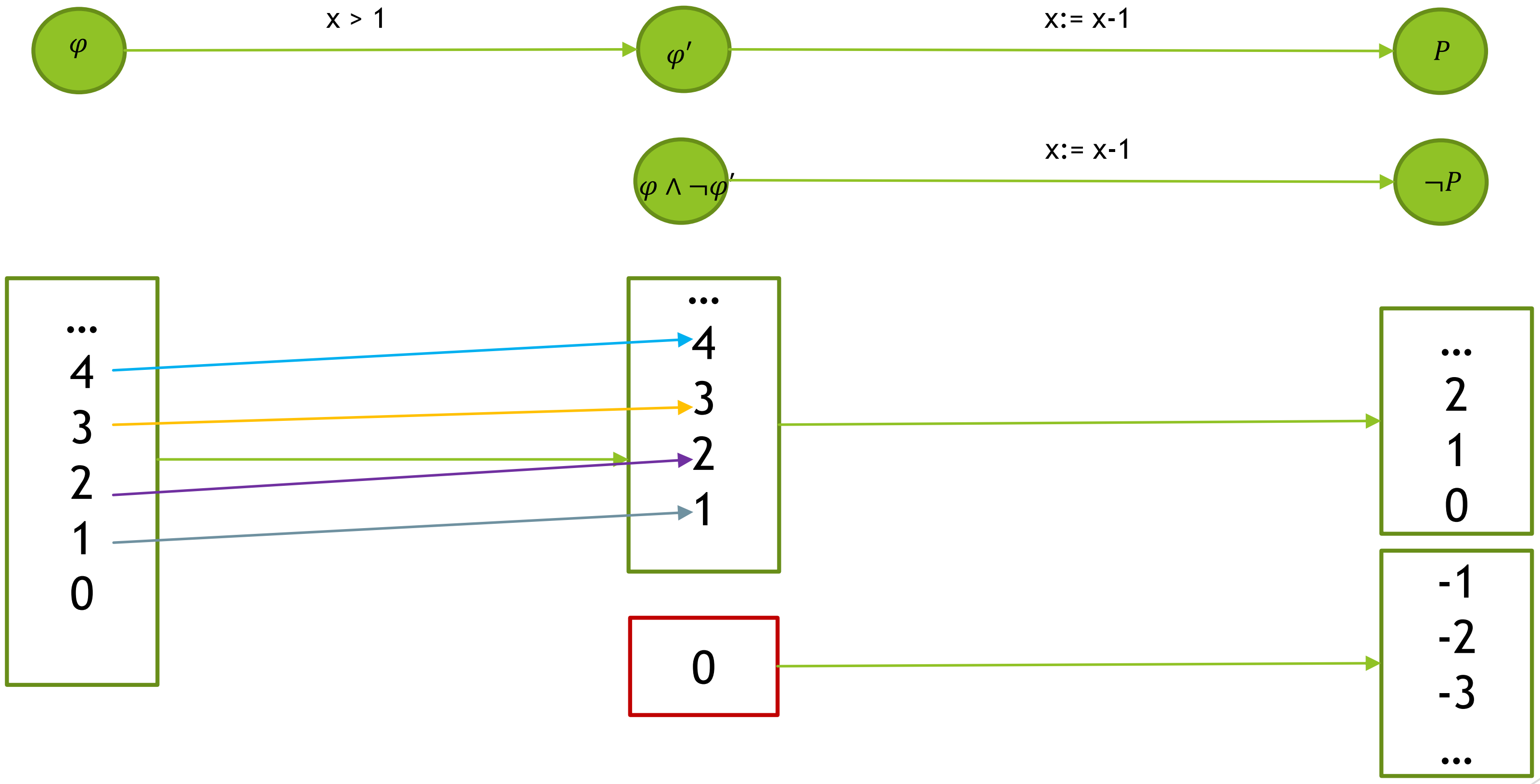
Counterexample-guided abstraction refinement (CEGAR)

$\varphi = x \geq 0,\ P = x^{S_3} \geq 0$



Counterexample-guided abstraction refinement (CEGAR)

$\varphi = \ x \geq 0,\ \varphi' = \ x > 0,\ P \ = x^{S_3} \geq 0$



SAT-Based Model Checking

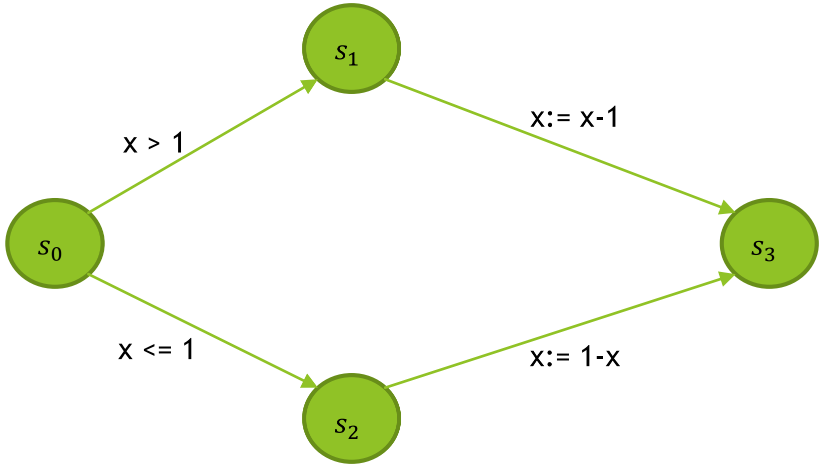
- A formula P is true $\equiv \neg P$ is unsatisfiable. For example, $x+1 > x$ is always true, that is we can not find x satisfies $x + 1 \leq x$.
- Given a transition system $\mathcal{T} = (S, I, T)$ and safety property P. Our goal is

$$\mathrm{G} = I(s_0) \wedge \left(\bigwedge_{i=0}^{k-1} T(s_i, s_{i+1}) \right) \wedge \left(\bigvee_{j=0}^{k} \neg P(s_j) \right)$$

- When $k = 0$, $G = I(s_0) \wedge \neg P(s_0)$.
- When $k = 1$, $G = I(s_0) \wedge T(s_0, s_1) \wedge (\neg P(s_0) \vee \neg P(s_1))$.
- K-bounded model checking.

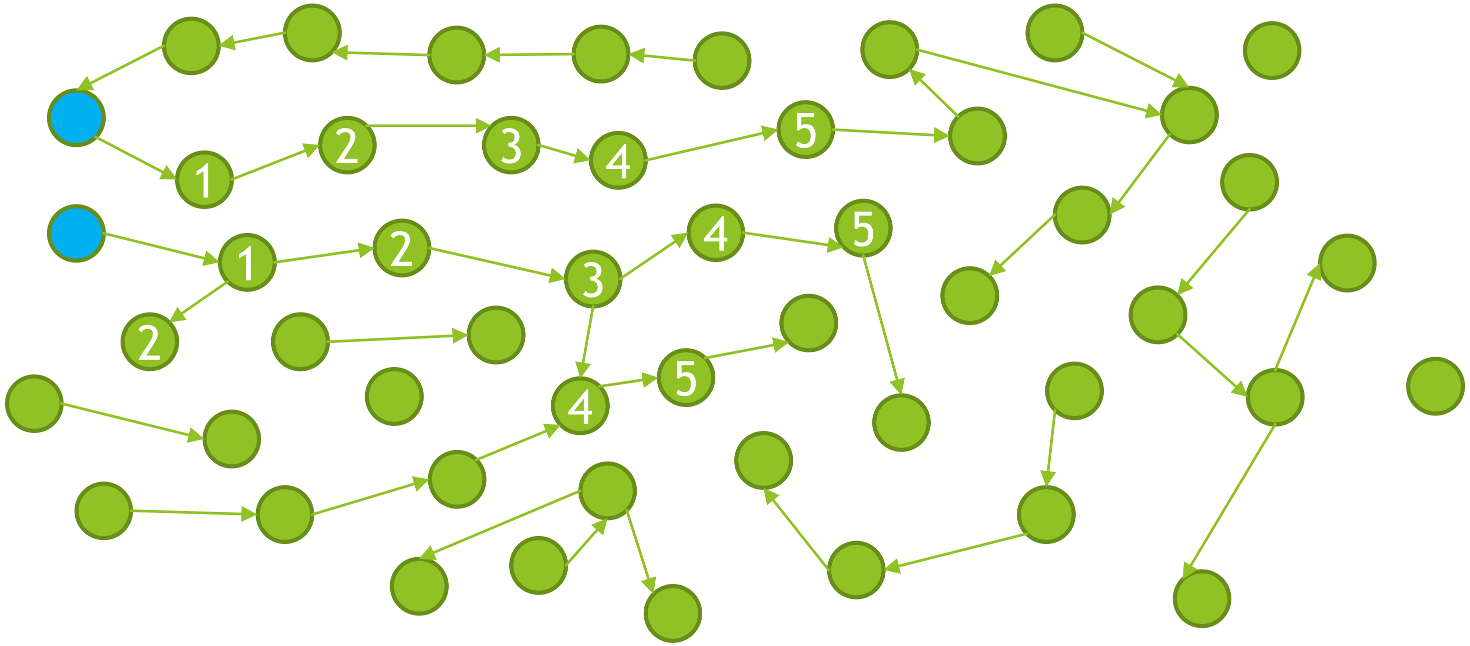
SAT-Based Model Checking

- $G = \big(I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_3) \wedge \neg P(s_3)\big) \vee \big(I(s_0) \wedge T(s_0, s_2) \wedge T(s_2, s_3) \wedge \neg P(s_3)\big)$ where $I : x \in \mathbb{Z}$, $P : x \geq 0$.
- G is unsatisfiable.



Interpolation and SAT-Based MC

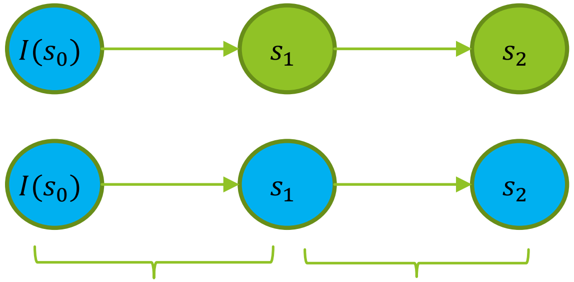
- Real world is difficult, we need approximation !
- We hope the k as small as possible.



Interpolation and SAT-Based MC

- Suppose $k = 2$, then $\mathrm{G} = I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2) \wedge (\neg P(s_0) \vee \neg P(s_1) \vee \neg P(s_2))$.
- Can we just use G model over whole state space ? Craig interpolation theorem is a excellent method!
- If $A \wedge B$ is unsatisfiable, then there is a C satisfies
  - $\mathrm{Atoms}(C) \subseteq \mathrm{Atoms}(A) \cap \mathrm{Atoms}(B)$.
  - $A \Rightarrow C$.
  - $C \wedge B$ is unsatisfiable.

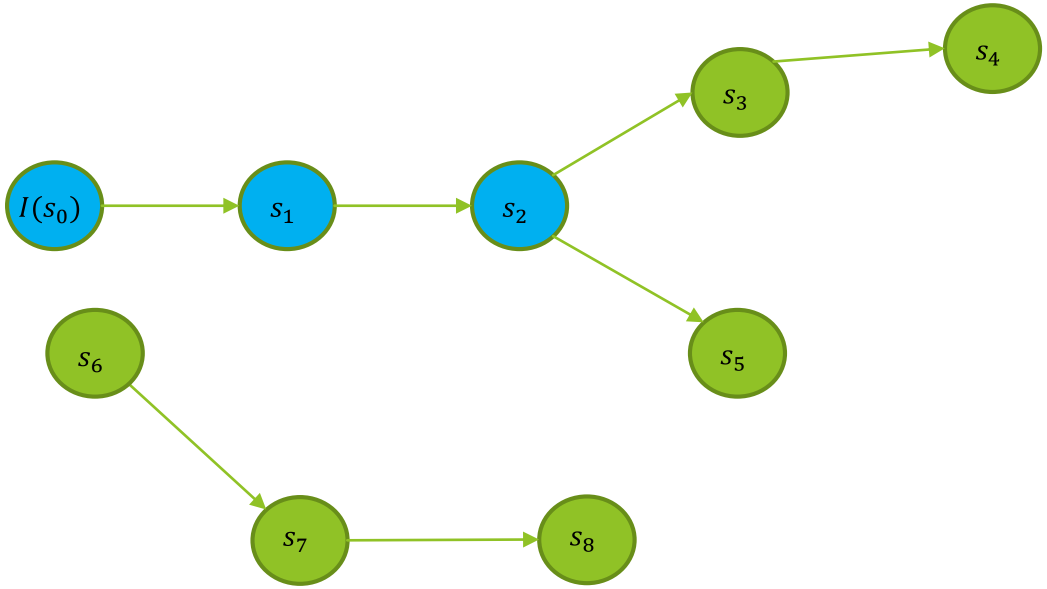
Interpolation and SAT-Based MC



$$G = I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2) \wedge (\neg P(s_0) \vee \neg P(s_1) \vee \neg P(s_2)).$$

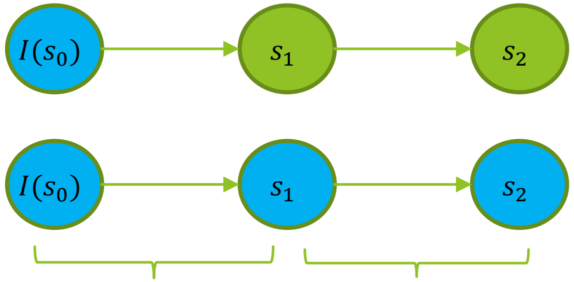
$A \Rightarrow C$, $C \wedge B$ is unsat

$A = I(s_0) \wedge T(s_0, s_1)$ $\quad B = T(s_1, s_2) \wedge (\neg P(s_0) \vee \neg P(s_1) \vee \neg P(s_2)).$



1. C is true in every state reachable from the initial state in one step.
2. no states satisfying C can reach a final sate in 1 (k-1) steps

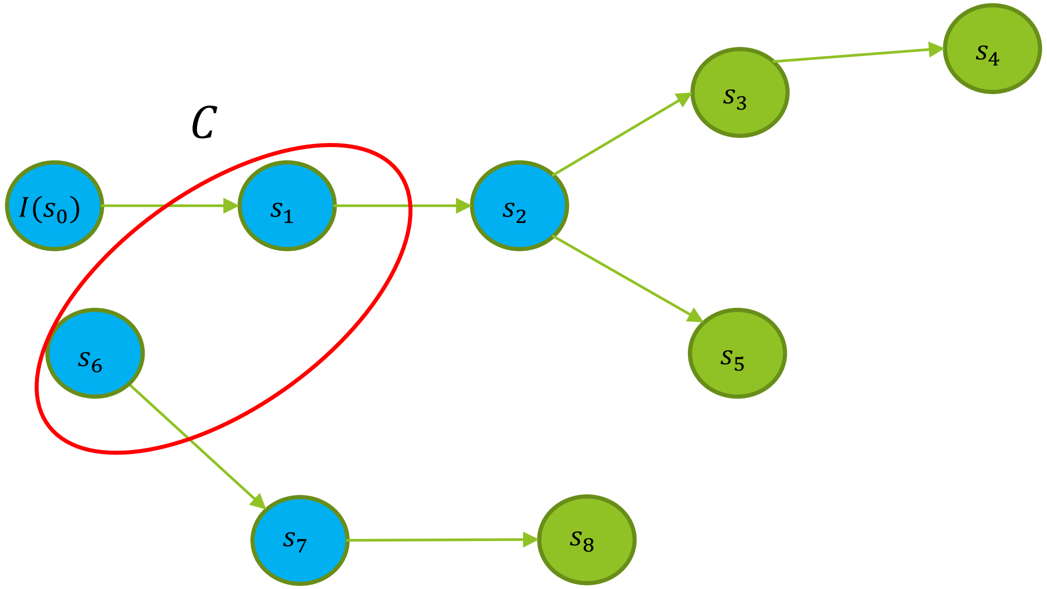
Interpolation and SAT-Based MC



$G = I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2) \wedge (\neg P(s_0) \vee \neg P(s_1) \vee \neg P(s_2)).$

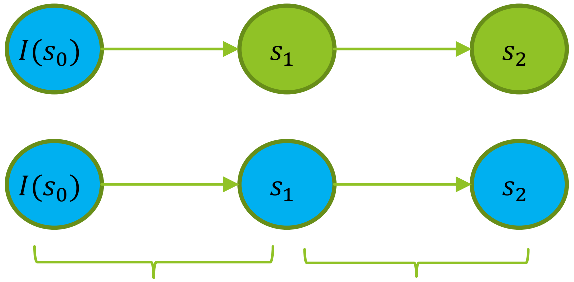
$A \Rightarrow C$, $C \wedge B$ is unsat

$A = I(s_0) \wedge T(s_0, s_1)$ $\quad$ $B = T(s_1, s_2) \wedge (\neg P(s_0) \vee \neg P(s_1) \vee \neg P(s_2)).$



1. C is true in every state reachable from the initial state in one step.
2. no states satisfying C can reach a final sate in 1 (k-1) steps

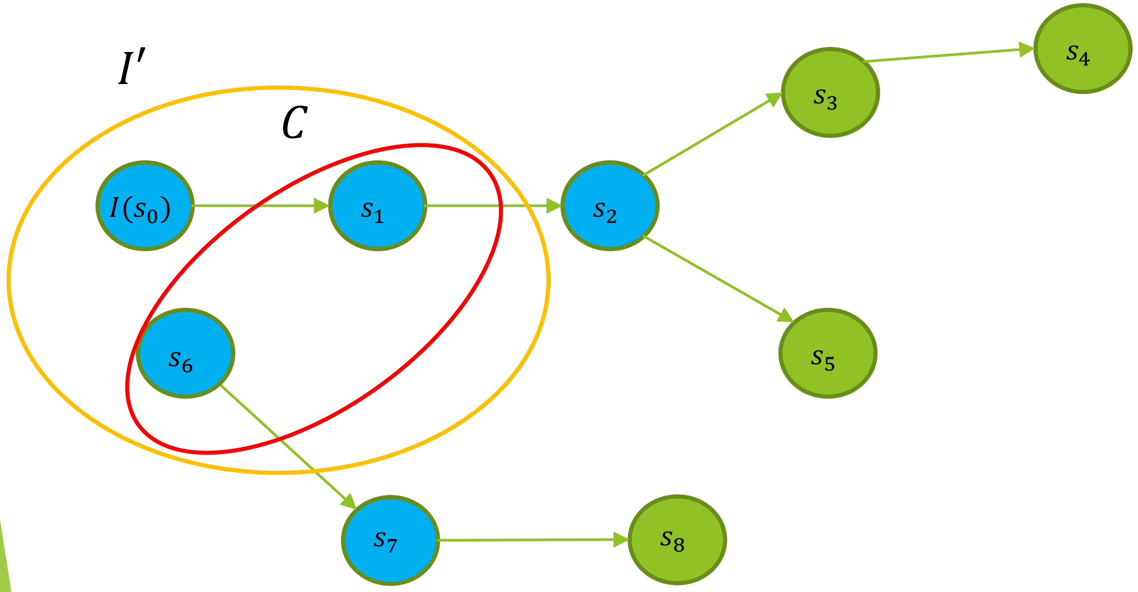
Interpolation and SAT-Based MC



$\mathrm{G} = I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2) \wedge (\neg P(s_0) \vee \neg P(s_1) \vee \neg P(s_2)).$

$A \Rightarrow C$, $C \wedge B$ is unsat

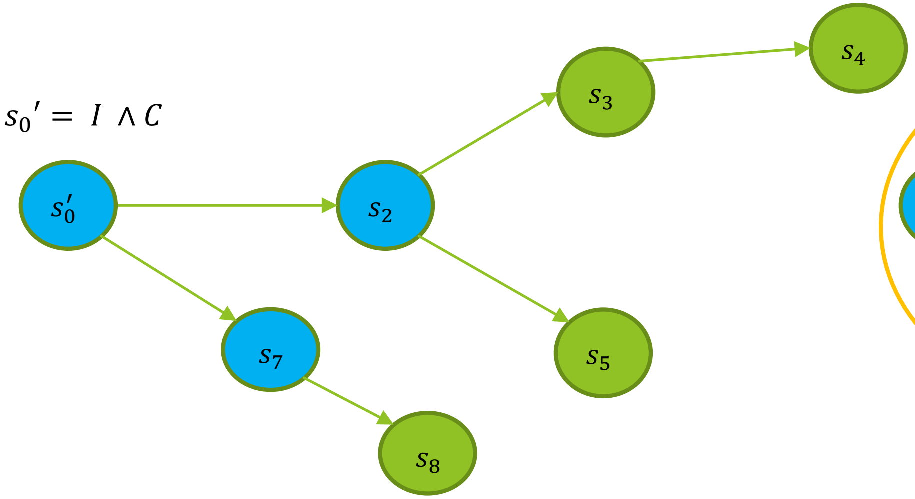
$A = I(s_0) \wedge T(s_0, s_1)$ $\quad$ $B = T(s_1, s_2) \wedge (\neg P(s_0) \vee \neg P(s_1) \vee \neg P(s_1)).$

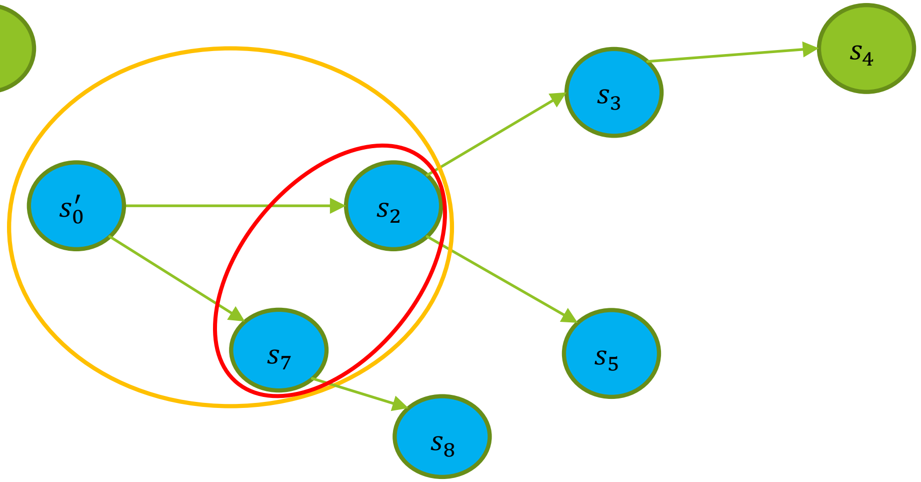


1. C is true in every state reachable from the initial state in one step.
2. no states satisfying C can reach a final sate in 1 (k-1) steps

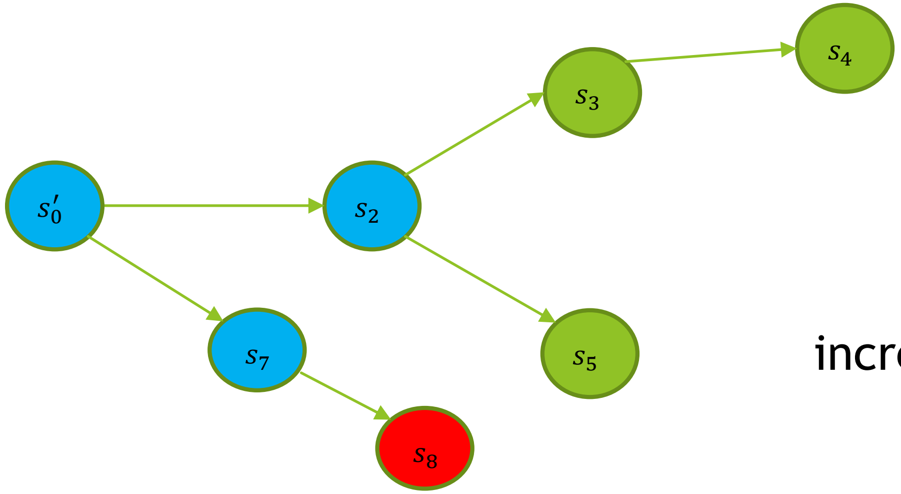
Interpolation and SAT-Based MC

$s_0{}' = I \wedge C$





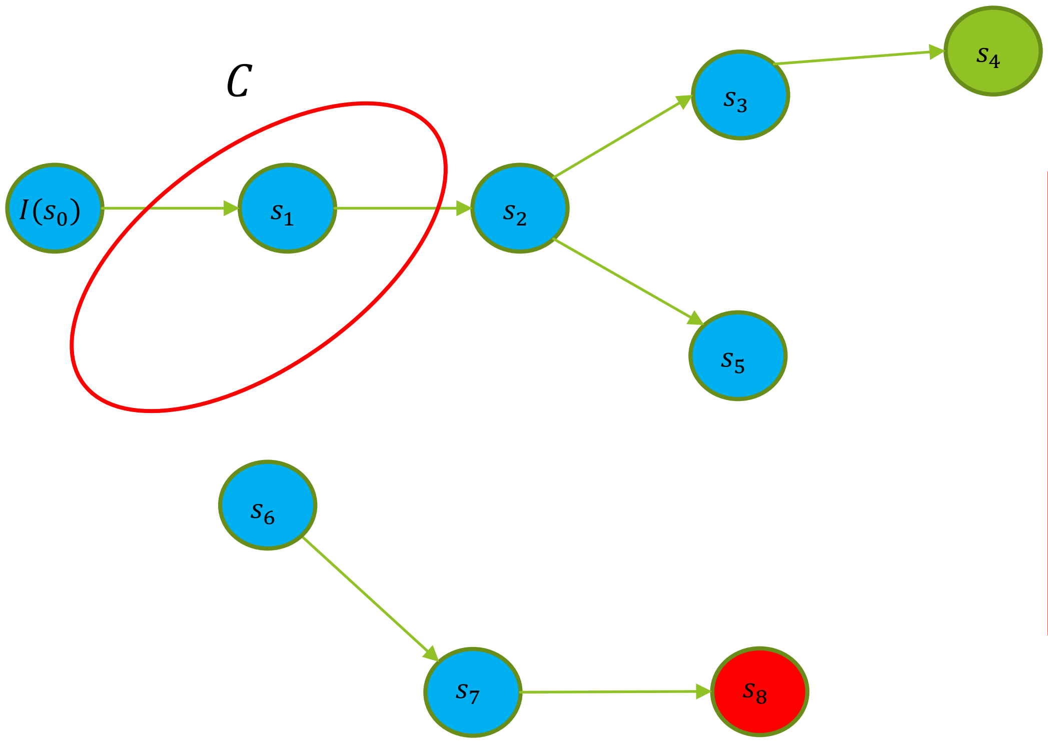
keep going



increase k

Interpolation and SAT-Based MC

K = 3



1. C is true in every state reachable from the initial state in one step.
2. no states satisfying C can reach a final sate in 2 steps

IC3: without unrolling

- Incremental Construction of Inductive Clauses for Indubitable Correctness.
- Unrolling is the goal

$$G = I(s_0) \wedge \left(\bigwedge_{i=0}^{k-1} T(s_i, s_{i+1}) \right) \wedge \left(\bigvee_{j=0}^{k} \neg P(s_j) \right)$$

with $k > 1$.

- Is there a way only use k = 1 ? That is only give SAT solver one step constraint like $F \wedge T \Rightarrow S$ is true $\equiv F \wedge T \wedge \neg S$ is unsat .

IC3: Inductive invariant

- Given a transition system $\mathcal{T} = (S, I, T)$ and formula F. We say F is inductive invariant of T, if F satisfies the following conditions
  - $I \Rightarrow F$
  - $F \wedge T \Rightarrow F'$
- Given a safety property P, if P is inductive invariant of T, then we are done.
- Unfortunately, P does not often satisfy $P \wedge T \Rightarrow P'$.

IC3: inductive strengthening

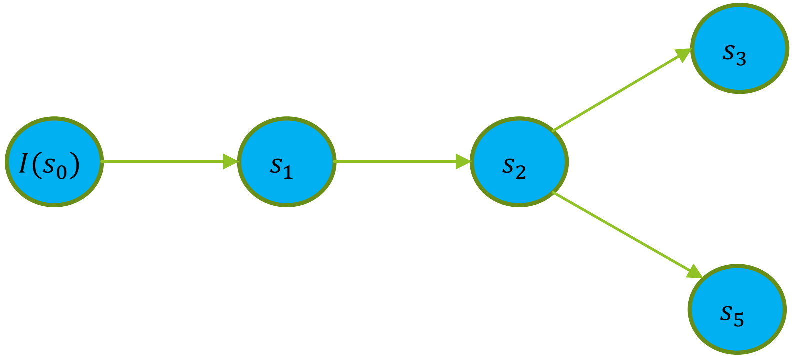
- Given a formula F. We say F is inductive relative to P, if F satisfies the following condition
  - $I \Rightarrow F$
  - $F \wedge P \wedge T \Rightarrow F'$
- If F is inductive relative to P, and $F \wedge P \wedge T \Rightarrow P'$. Then $F \wedge P$ is inductive invariant of T. That is every reachable state satisfies $F \wedge P$, it clearly satisfies P.

IC3: inductive strengthening

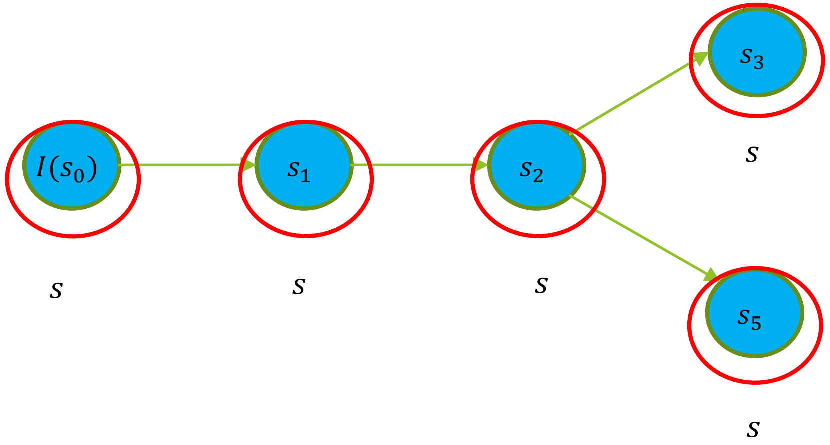
```
x, y := 1, 1
while * :
    y = y + x
    x = x + 1
```

- Let $P = y \geq 1$.
- Initial: $x = 1 \land y = 1 \Rightarrow y \geq 1$.
- 1-step: $y \geq 1 \land y' = y + x \nRightarrow y' \geq 1$.
- Then we add predicate $\varphi_1 = x \geq 0$.
  - Initial $x = 1 \land y = 1 \Rightarrow x \geq 0$.
  - 1-step: $x \geq 0 \land y \geq 1 \land y' = y + x \Rightarrow x' \geq 0$.
  - 2-step: $x \geq 0 \land y \geq 1 \land x' = x + 1 \Rightarrow x' \geq 0$.
- It is obvious that φ_1 is inductive relative to P and $\varphi_1 \land P$ is a inductive invariant.

IC3: inductive strengthening

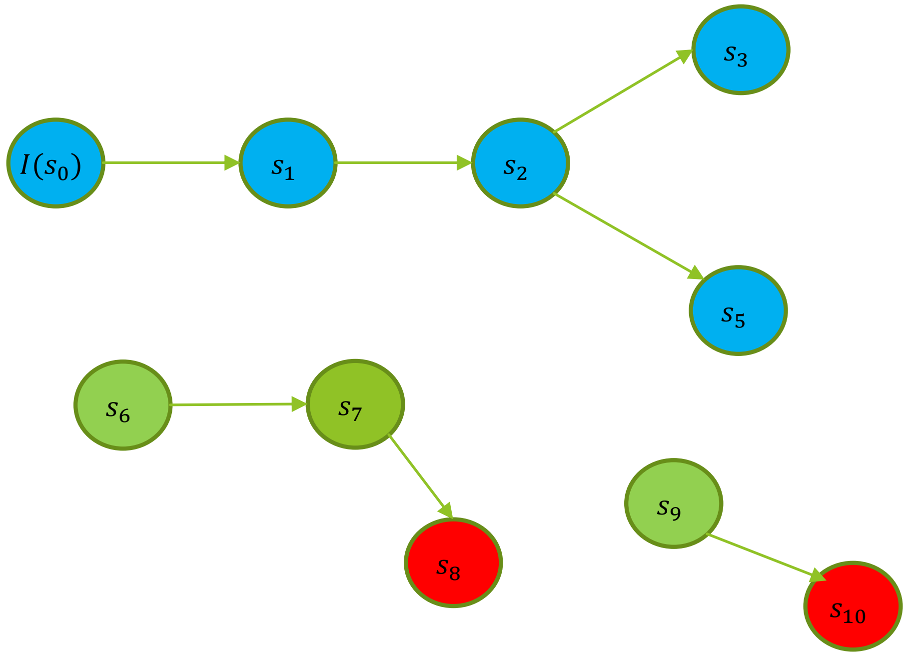


$$s_0 \Rightarrow P, s_1 \Rightarrow P, s_2 \Rightarrow P, s_3 \Rightarrow P, s_5 \Rightarrow P$$



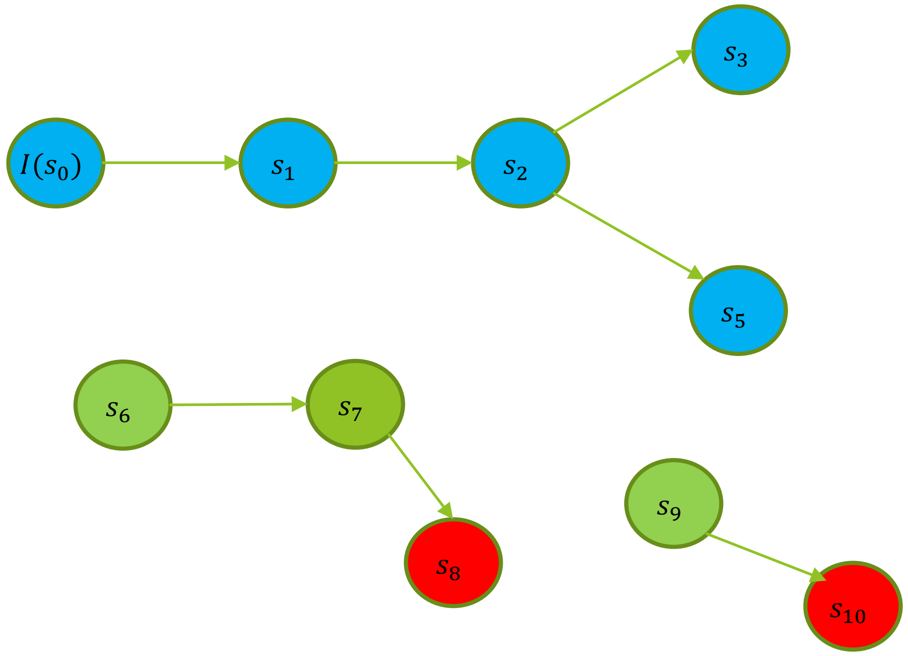
$$s \Rightarrow P$$

IC3: inductive strengthening



s = ?

IC3: inductive strengthening



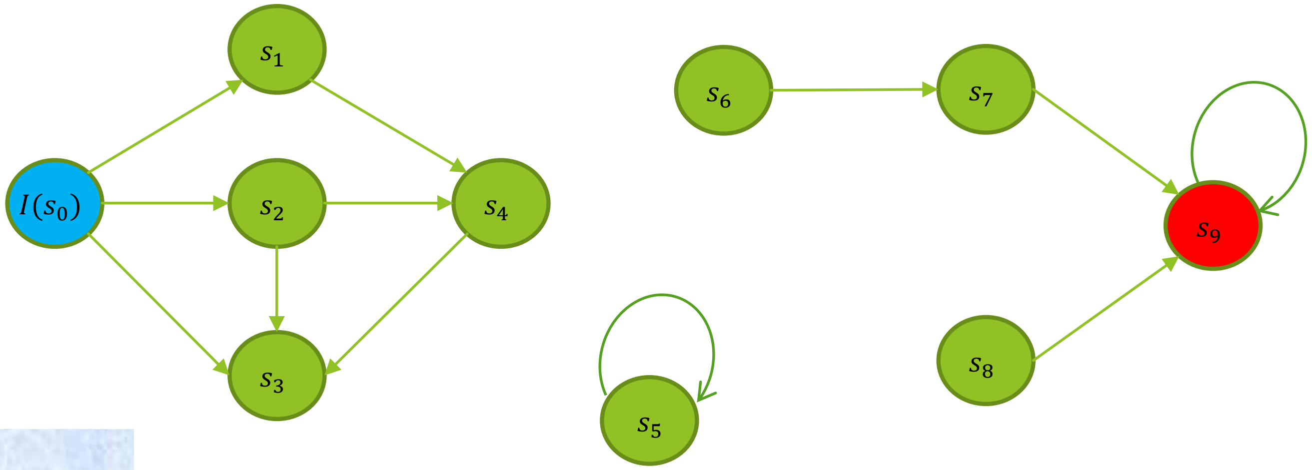
$$s = \neg s_6 \land \neg s_7 \land \neg s_8 \land \neg s_9 \land \neg s_{10}$$

IC3

- Construct $F_0, F_1, F_2, \cdots, F_k$ satisfy the following conditions
  - $F_0 = I$.
  - $F_i \wedge T \Rightarrow F_{i+1}$ for $0 \leq i < k$.
  - $F_i \Rightarrow P$ for $0 \leq i < k$.
- If $F_i = F_{i+1}$, then we are done.

IC3: example

$$F_0 = I$$

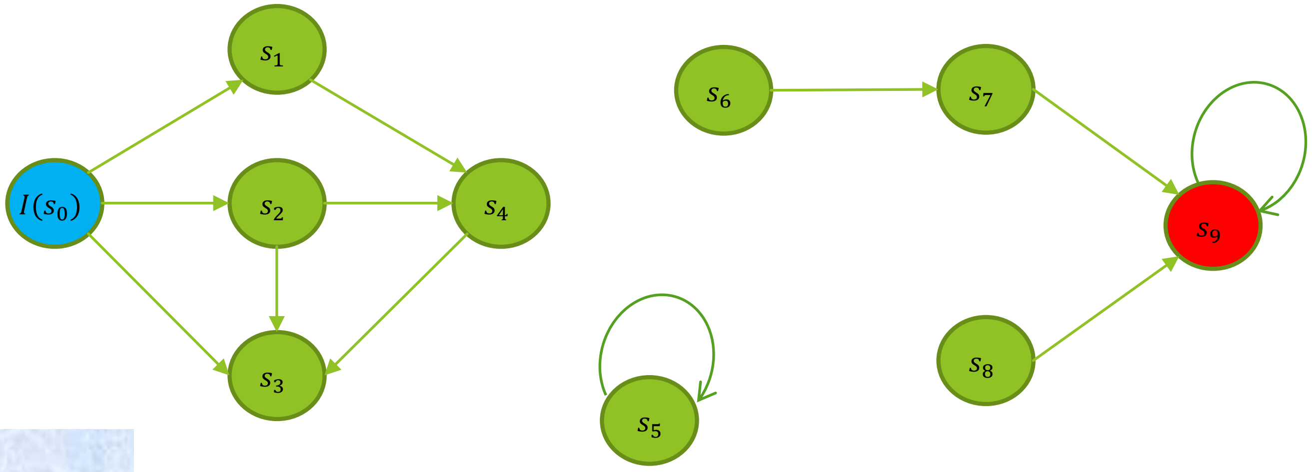


- $F_0 = I$.
- $F_i \wedge T \Rightarrow F_{i+1}$ for $0 \le i < k$.
- $F_i \Rightarrow P$ for $0 \le i < k$.

IC3: example

$F_1 \nRightarrow$ P ($F_1 \wedge \neg P$ is sat), s_9 is the counterexample.

$$F_0 = I, F_1 = true$$

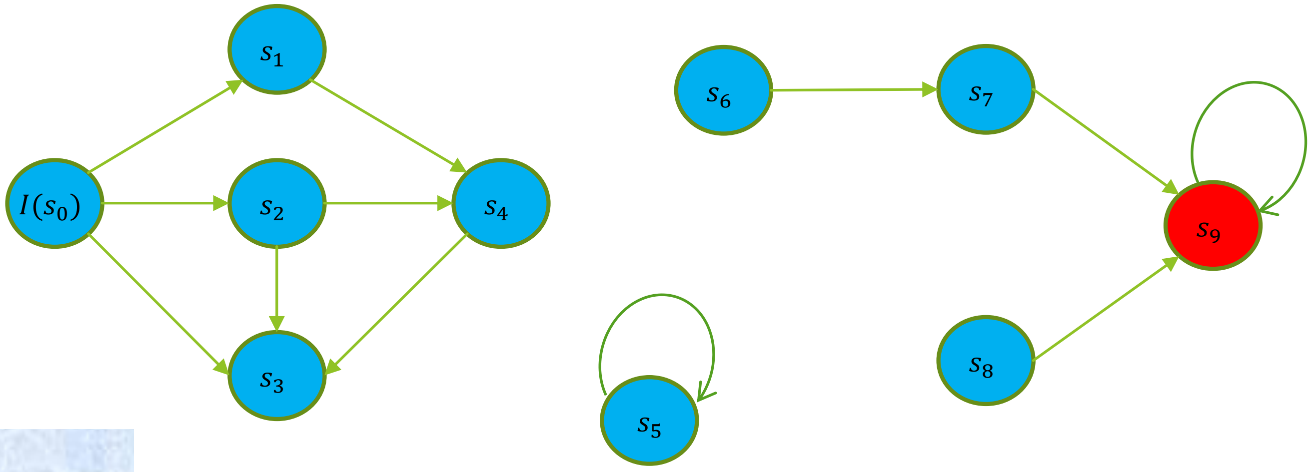


- $F_0 = I$.
- $F_i \wedge T \Rightarrow F_{i+1}$ for $0 \leq i < k$.
- $F_i \Rightarrow P$ for $0 \leq i < k$.

IC3: example

$F_0 \wedge T \Rightarrow F_1$ ($F_0 \wedge T \wedge \neg F_1$ is unsat) and $F_1 \Rightarrow P$ (we assume only s_9 violates P).

$$F_0 = I, F_1 = \neg s_9$$

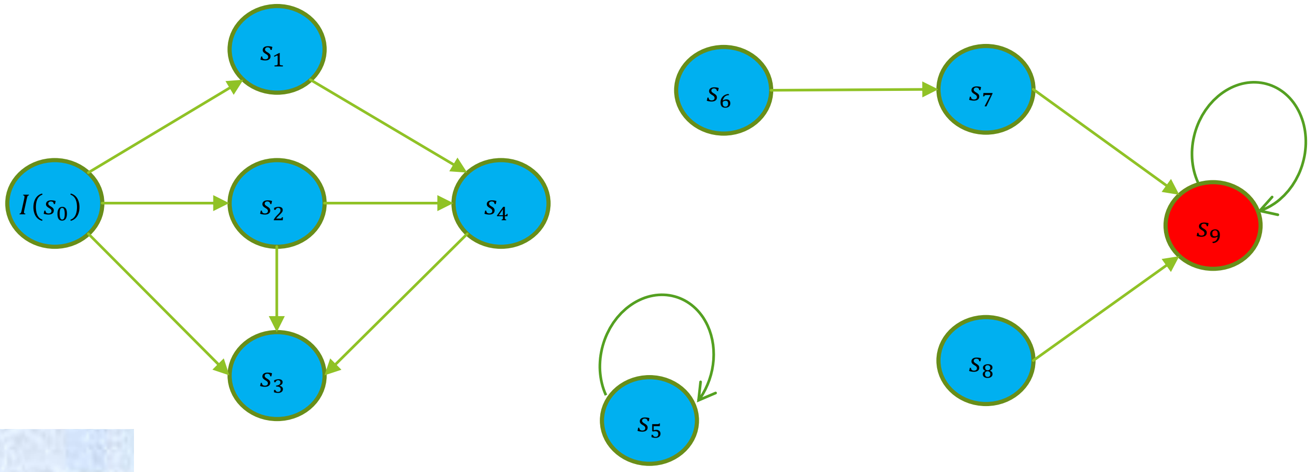


- $F_0 = I.$
- $F_i \wedge T \Rightarrow F_{i+1}$ for $0 \leq i < k.$
- $F_i \Rightarrow P$ for $0 \leq i < k.$

IC3: example

$F_2 \not\Rightarrow \mathrm{P}$ ($F_2 \wedge \neg P$ is sat), s_9 is the counterexample.

$$F_0 = I, F_1 = \neg s_9, F_2 = true$$

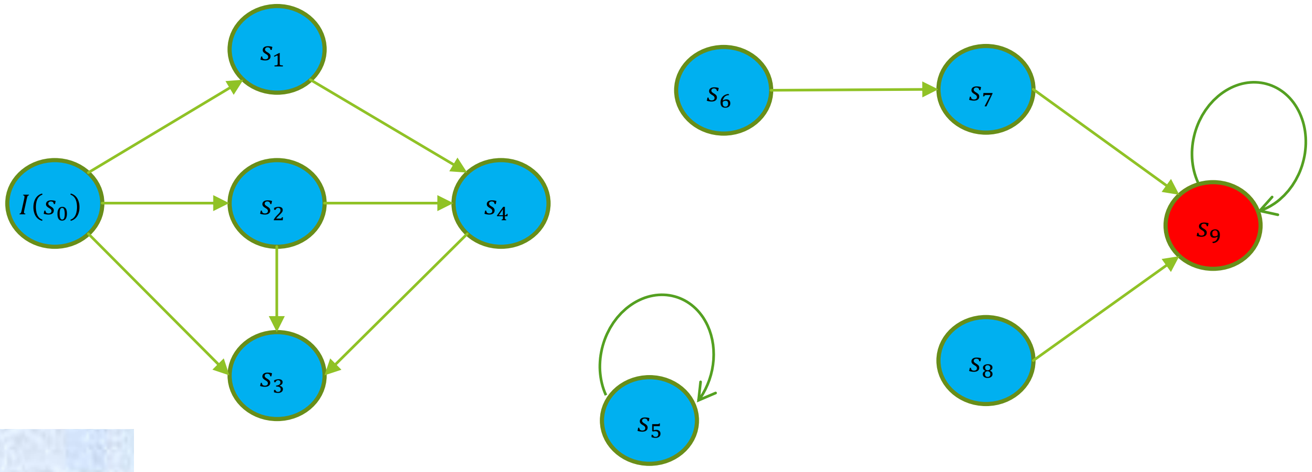


- $F_0 = I$.
- $F_i \wedge T \Rightarrow F_{i+1}$ for $0 \leq i < k$.
- $F_i \Rightarrow P$ for $0 \leq i < k$.

IC3: example

$F_1 \wedge T \not\Rightarrow F_2$, because s_7, s_8 are counterexamples.

$$F_0 = I, F_1 = \neg s_9, F_2 = \neg s_9$$

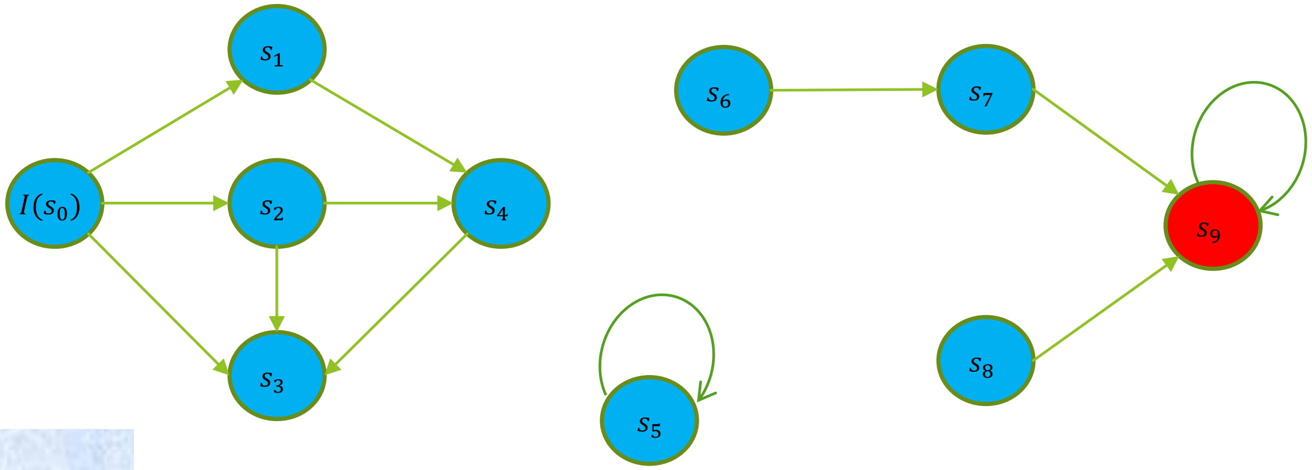


- $F_0 = I$.
- $F_i \wedge T \Rightarrow F_{i+1}$ for $0 \leq i < k$.
- $F_i \Rightarrow P$ for $0 \leq i < k$.

IC3: example

$F_0 \wedge T \Rightarrow F_1,\ F_1 \wedge T \Rightarrow F_2$ are all ok.

$$F_0 = I, F_1 = \neg s_9 \wedge \neg s_7 \wedge \neg s_8, F_2 = \neg s_9$$

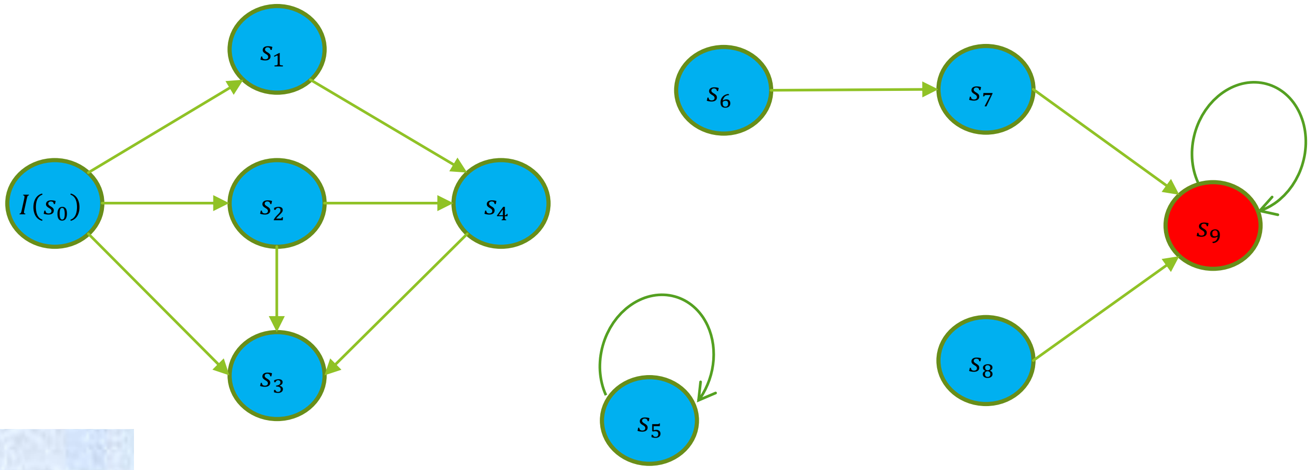


- $F_0 = I.$
- $F_i \wedge T \Rightarrow F_{i+1}$ for $0 \leq i < k.$
- $F_i \Rightarrow P$ for $0 \leq i < k.$

IC3: example

$F_3 \not\Rightarrow \mathrm{P}$, s_9 is the counterexample.

$$F_0 = I, F_1 = \neg s_9 \wedge \neg s_7 \wedge \neg s_8, F_2 = \neg s_9, F_3 = true.$$

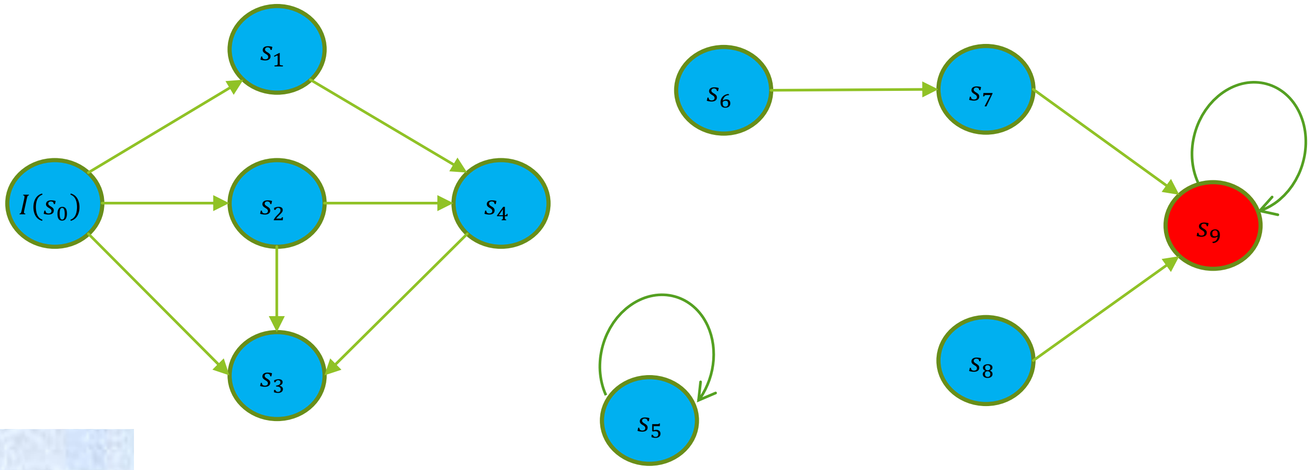


- $F_0 = I$.
- $F_i \wedge T \Rightarrow F_{i+1}$ for $0 \leq i < k$.
- $F_i \Rightarrow P$ for $0 \leq i < k$.

IC3: example

$F_2 \wedge T \not\Rightarrow \mathrm{F}_3$, s_7, s_8 is the counterexample.

$$F_0 = I, F_1 = \neg s_9 \wedge \neg s_7 \wedge \neg s_8, F_2 = \neg s_9, F_3 = \neg s_9.$$

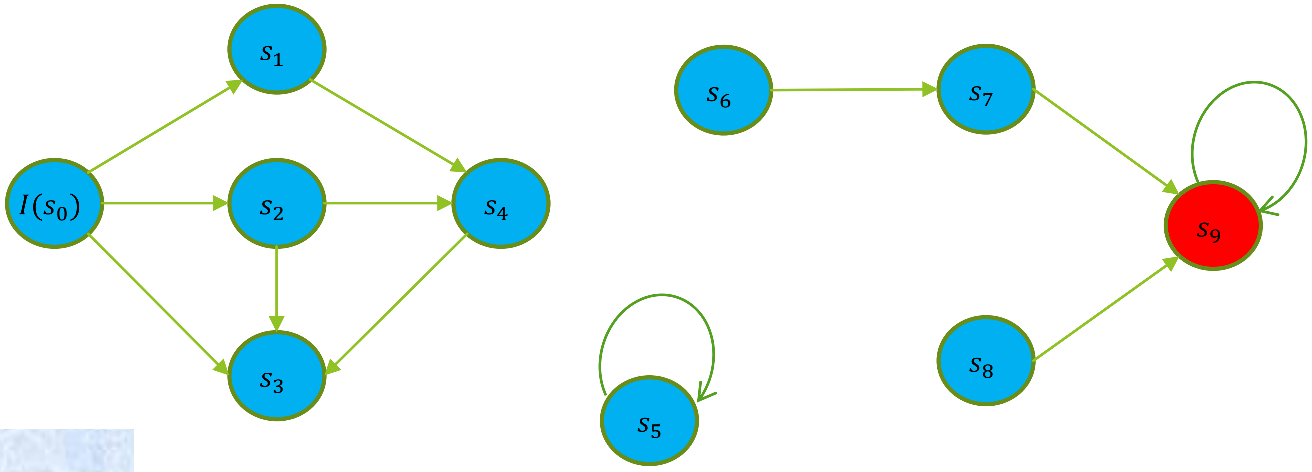


- $F_0 = I$.
- $F_i \wedge T \Rightarrow F_{i+1}$ for $0 \leq i < k$.
- $F_i \Rightarrow P$ for $0 \leq i < k$.

IC3: example

$F_1 \wedge T \not\Rightarrow \mathrm{F}_2$, s_6 is the counterexample.

$$F_0 = I, F_1 = \neg s_9 \wedge \neg s_7 \wedge \neg s_8, F_2 = \neg s_9 \wedge \neg s_7 \wedge \neg s_8, F_3 = \neg s_9.$$

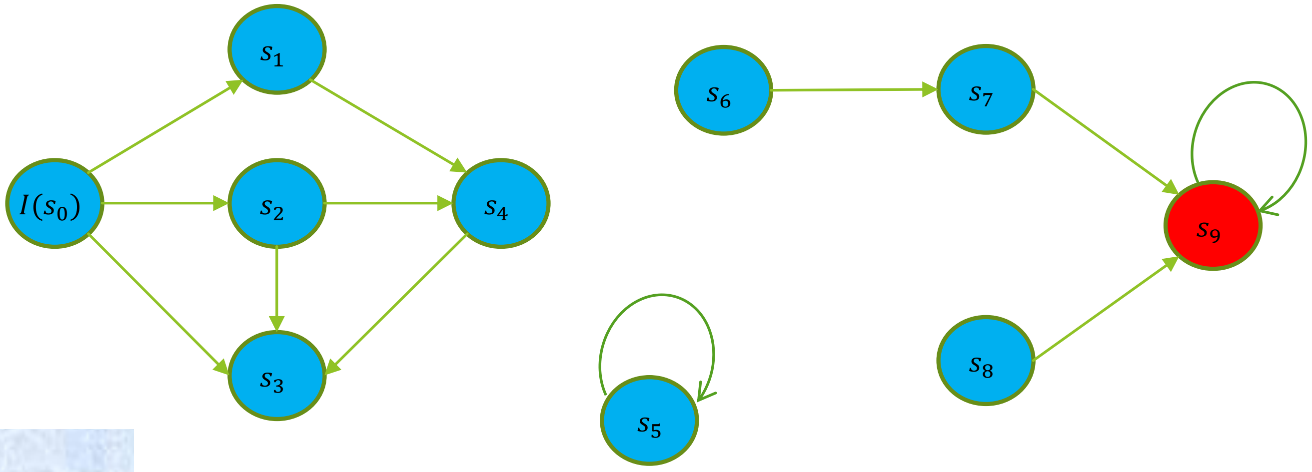


- $F_0 = I$.
- $F_i \wedge T \Rightarrow F_{i+1}$ for $0 \leq i < k$.
- $F_i \Rightarrow P$ for $0 \leq i < k$.

IC3: example

Remove s_6 from F_1, $\mathrm{F}_0 \wedge T \Rightarrow F_1$, $F_1 \wedge T \Rightarrow F_2$, $F_2 \wedge T \Rightarrow F_3$ are all ok.

$$F_0 = I, F_1 = \neg s_9 \wedge \neg s_7 \wedge \neg s_8 \wedge \neg s_6, F_2 = \neg s_9 \wedge \neg s_7 \wedge \neg s_8, F_3 = \neg s_9.$$

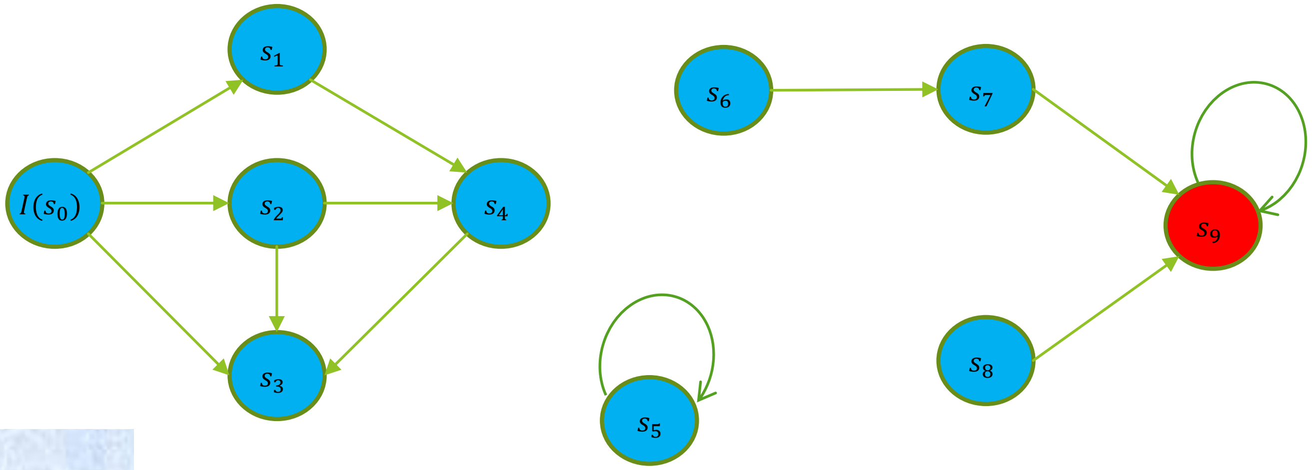


- $F_0 = I$.
- $F_i \wedge T \Rightarrow F_{i+1}$ for $0 \leq i < k$.
- $F_i \Rightarrow P$ for $0 \leq i < k$.

IC3: example

$F_4 \nRightarrow \mathrm{P}$, s_9 is the counterexample.

$$F_0 = I, F_1 = \neg s_9 \wedge \neg s_7 \wedge \neg s_8 \wedge \neg s_6, F_2 = \neg s_9 \wedge \neg s_7 \wedge \neg s_8, F_3 = \neg s_9, F_4 = \neg s_9 \ .$$

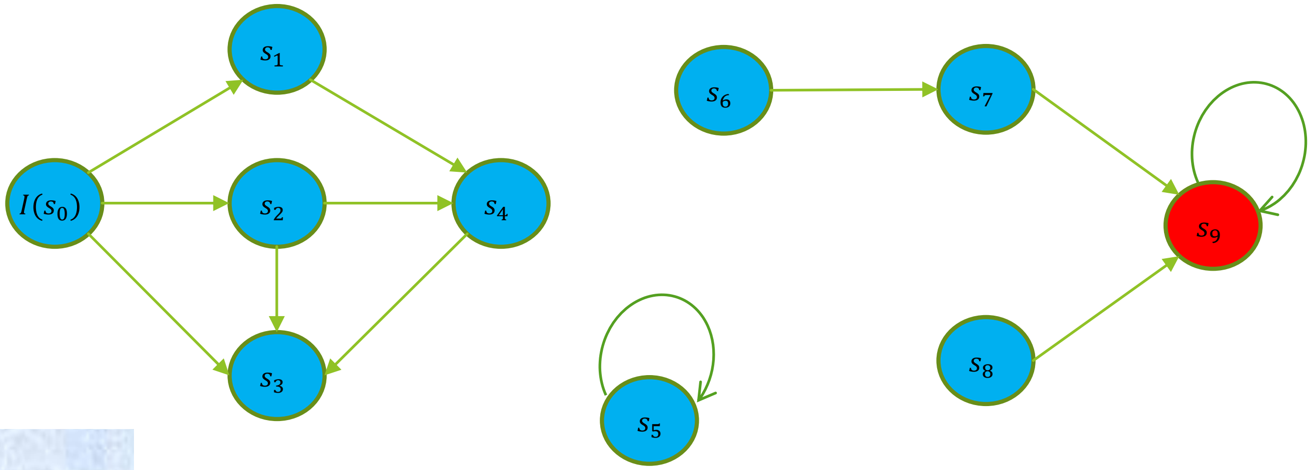


- $F_0 = I$.
- $F_i \wedge T \Rightarrow F_{i+1}$ for $0 \leq i < k$.
- $F_i \Rightarrow P$ for $0 \leq i < k$.

IC3: example

$F_1 = F_2$, we are done!

$$F_0 = I, F_1 = \neg s_9 \wedge \neg s_7 \wedge \neg s_8 \wedge \neg s_6, F_2 = \neg s_9 \wedge \neg s_7 \wedge \neg s_8 \wedge \neg s_6,$$
$$F_3 = \neg s_9 \wedge \neg s_7 \wedge s_8, F_4 = \neg s_9.$$



- $F_0 = I.$
- $F_i \wedge T \Rightarrow F_{i+1}$ for $0 \leq i < k.$
- $F_i \Rightarrow P$ for $0 \leq i < k.$

IC3: summary

- It decomposes a big problem into small problems that are cheap to be solved.
- The method is very friendly to theorem solvers.
- The method can be implemented in parallel.

> It is easier to write an incorrect program than to understand a correct one.
>
> — Alan Perlis
> Epigrams on Programming, 1982