Incremental Inductive Verification

slides adapted from the ones by
Luca Geatti
University of Udine, Italy
FBK, Trento, Italy

Inductive Verification

In order to prove that $P(x)$ is $\mathcal{M}$-invariant, one possibility is to check if P is inductive. With 2 SAT-solver calls, we check the validity of:

$$\text{(initiation)} \quad I \Rightarrow P$$
$$\text{(consecution)} \quad P \wedge T \Rightarrow P'$$

Inductive Verification

In order to prove that $P(x)$ is $\mathcal{M}$-invariant, one possibility is to check if P is inductive. With 2 SAT-solver calls, we check the validity of:

$$\text{(initiation)} \quad I \Rightarrow P$$

$$\text{(consecution)} \quad P \land T \Rightarrow P'$$

It is a sufficient condition to prove invariance for P. It is not also a necessary condition.

Q: Why?

Inductive Verification

In order to prove that $P(x)$ is $\mathcal{M}$-invariant, one possibility is to check if P is inductive. With 2 SAT-solver calls, we check the validity of:

$$\text{(initiation)} \;\; I \Rightarrow P$$
$$\text{(consecution)} \;\; P \land T \Rightarrow P'$$

It is a sufficient condition to prove invariance for P. It is not also a necessary condition.

Q: Why?

Remember the discussion about k-induction we did on the 5^{th} lecture?

The set of states in P could be must larger than the ones in I and contain states unreachable from I via T.

Incremental vs Monolithic

If consecution fails, then:

Incremental vs Monolithic

If consecution fails, then:

- **Monolithic approach**: look for a stronger assertion F such that $F \wedge P$ is inductive. $F \wedge P$ is called an inductive strenghtening.

Incremental vs Monolithic

If consecution fails, then:

- **Monolithic approach**: look for a stronger assertion F such that $F \wedge P$ is inductive. $F \wedge P$ is called an inductive strenghtening.
- **Incremental proof**: look for a sequence of lemmata $\phi_1, \phi_2, \ldots, \phi_k = P$ such that ϕ_i is inductive relative to $\phi_1 \wedge \cdots \wedge \phi_{i-1}$, for all $1 < i \leq k$, *i.e.*,
  - $I \Rightarrow \phi_i$
  - $\phi_1 \wedge \cdots \wedge \phi_{i-1} \wedge \phi_i \wedge T \Rightarrow \phi_i'$

Incremental vs Monolithic

If consecution fails, then:

- **Monolithic approach**: look for a stronger assertion F such that $F \wedge P$ is inductive. $F \wedge P$ is called an inductive strenghtening.
- **Incremental proof**: look for a sequence of lemmata $\phi_1, \phi_2, \ldots, \phi_k = P$ such that ϕ_i is inductive relative to $\phi_1 \wedge \cdots \wedge \phi_{i-1}$, for all $1 < i \leq k$, *i.e.*,
  - $I \Rightarrow \phi_i$
  - $\phi_1 \wedge \cdots \wedge \phi_{i-1} \wedge \phi_i \wedge T \Rightarrow \phi_i'$

  It follows that $P \wedge \bigwedge_{i=1}^{k-1} \phi_i$ is an inductive strengthening.

Abstraction

Note that:

- both methods do *not* compute a formula R for the exact set of reachable states in $\mathcal{M}$;

Abstraction

Note that:

- both methods do *not* compute a formula R for the exact set of reachable states in $\mathcal{M}$;
- rather, they find a formula $F \wedge P$ that represents a larger set of states *all satisfying* $F \wedge P$:
  - $\Rightarrow$ this F is a much smaller formula than R.

Monolithic Approach - Naïve algorithm

Naïve algorithm for finding an inductive strengthening:

1. $IS := P$

Monolithic Approach - Naïve algorithm

Naïve algorithm for finding an inductive strengthening:

1. $IS := P$
2. if IS is inductive, then we have found an inductive strengthening; stop.

Monolithic Approach - Naïve algorithm

Naïve algorithm for finding an inductive strengthening:

1. $IS := P$
2. if IS is inductive, then we have found an inductive strengthening; stop.
3. else find a CTI err (counterexample to inductiveness):

$$err \models IS \wedge T \wedge \neg IS'$$

Monolithic Approach - Naïve algorithm

Naïve algorithm for finding an inductive strengthening:

1. $IS := P$
2. if IS is inductive, then we have found an inductive strengthening; stop.
3. else find a CTI err (counterexample to inductiveness):

$$err \models IS \wedge T \wedge \neg IS'$$

   3.1 if $err \wedge I$ is SAT, then stop: P is NOT invariant;

Monolithic Approach - Naïve algorithm

Naïve algorithm for finding an inductive strengthening:

1. $IS := P$
2. if IS is inductive, then we have found an inductive strengthening; stop.
3. else find a CTI err (counterexample to inductiveness):

$$err \models IS \wedge T \wedge \neg IS'$$

   3.1 if $err \wedge I$ is SAT, then stop: P is NOT invariant;
   3.2 else set $IS := IS \wedge \neg err$ and go to Item 2.

Monolithic Approach - Naïve algorithm

Naïve algorithm for finding an inductive strengthening:

1. $IS := P$
2. if IS is inductive, then we have found an inductive strengthening; stop.
3. else find a CTI err (counterexample to inductiveness):

$$err \models IS \wedge T \wedge \neg IS'$$

   3.1 if $err \wedge I$ is SAT, then stop: P is NOT invariant;
   3.2 else set $IS := IS \wedge \neg err$ and go to Item 2.

At the end, the inductive strengthening (if any) will be:

$$P \wedge \bigwedge_{err \in CTI} \neg err$$

Incremental Proof - Example

Consider the program $\mathcal{M}_1$:

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

Incremental Proof - Example

Consider the program $\mathcal{M}_1$:

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

We want to prove that $y \geq 1$ is $\mathcal{M}_1$-invariant:

Incremental Proof - Example

Consider the program $\mathcal{M}_1$:

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

We want to prove that $y \geq 1$ is $\mathcal{M}_1$-invariant:

- $\underbrace{x = 1 \wedge y = 1}_{I} \Rightarrow \underbrace{y \geq 1}_{P}$

Incremental Proof - Example

Consider the program $\mathcal{M}_1$:

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

We want to prove that $y \geq 1$ is $\mathcal{M}_1$-invariant:

- $\underbrace{x = 1 \land y = 1}_{I} \Rightarrow \underbrace{y \geq 1}_{P}$
- $\underbrace{y \geq 1}_{P} \land \underbrace{x' = x + 1 \land y' = y + x}_{T} \not\Rightarrow \underbrace{y' \geq 1}_{P'}$

Incremental Proof - Example

Consider the program $\mathcal{M}_1$:

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

We want to prove that $y \geq 1$ is $\mathcal{M}_1$-invariant:

- $\underbrace{x = 1 \wedge y = 1}_{I} \Rightarrow \underbrace{y \geq 1}_{P}$
- $\underbrace{y \geq 1}_{P} \underbrace{\wedge x' = x + 1 \wedge y' = y + x}_{T} \not\Rightarrow \underbrace{y' \geq 1}_{P'}$

Q: why?

Incremental Proof - Example

Consider the program $\mathcal{M}_1$:

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

We want to prove that $y \geq 1$ is $\mathcal{M}_1$-invariant:

- $\underbrace{x = 1 \wedge y = 1}_{I} \Rightarrow \underbrace{y \geq 1}_{P}$
- $\underbrace{y \geq 1}_{P} \underbrace{\wedge x' = x + 1 \wedge y' = y + x}_{T} \not\Rightarrow \underbrace{y' \geq 1}_{P'}$

Q: why?

We are not saying anything about x.

Incremental Proof - Example

We establish the first inductive incremental lemma $\phi_1 \coloneqq x \geq 0$:

- $x = 1 \wedge y = 1 \Rightarrow x \geq 0$
- $x \geq 0 \wedge x' = x + 1 \wedge y' = y + x \Rightarrow x' \geq 0$

Incremental Proof - Example

We establish the first inductive incremental lemma $\phi_1 \coloneqq x \geq 0$:

- $x = 1 \land y = 1 \Rightarrow x \geq 0$
- $x \geq 0 \land x' = x + 1 \land y' = y + x \Rightarrow x' \geq 0$

Now, $\phi_2 \coloneqq y \geq 1$ is inductive relative to ϕ_1:

- $x = 1 \land y = 1 \Rightarrow y \geq 1$
- $\underbrace{x \geq 0}_{\phi_1} \land y \geq 1 \land x' = x + 1 \land y' = y + x \Rightarrow y' \geq 1$

Incremental Proof - Example

We establish the first inductive incremental lemma $\phi_1 := x \geq 0$:

- $x = 1 \land y = 1 \Rightarrow x \geq 0$
- $x \geq 0 \land x' = x + 1 \land y' = y + x \Rightarrow x' \geq 0$

Now, $\phi_2 := y \geq 1$ is inductive relative to ϕ_1:

- $x = 1 \land y = 1 \Rightarrow y \geq 1$
- $\underbrace{x \geq 0}_{\phi_1} \land\ y \geq 1 \land x' = x + 1 \land y' = y + x \Rightarrow y' \geq 1$

We have found the inductive strengthening $\phi_1 \land \phi_2$, by means of an incremental proof.

Limitation of incremental proofs

Consider the program $\mathcal{M}_2$:

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

We want to prove $\phi_2 := y \geq 1$.

Limitation of incremental proofs

Consider the program $\mathcal{M}_2$:

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

We want to prove $\phi_2 := y \geq 1$.

Like before, ϕ_2 is not inductive on its own.

Limitation of incremental proofs

Consider the program $\mathcal{M}_2$:

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

We want to prove $\phi_2 \coloneqq y \geq 1$.

Like before, ϕ_2 is not inductive on its own.

$\cdots$ but now neither is $\phi \coloneqq x \geq 0$:

- $x = 1 \wedge y = 1 \Rightarrow x \geq 0$
- $x \geq 0 \wedge x' = x + y \wedge y' = y + x \not\Rightarrow x' \geq 0$

Limitation of incremental proofs

Consider the program $\mathcal{M}_2$:

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

We want to prove $\phi_2 \coloneqq y \geq 1$.

Like before, ϕ_2 is not inductive on its own.

$\cdots$ but now neither is $\phi \coloneqq x \geq 0$:

- $x = 1 \wedge y = 1 \Rightarrow x \geq 0$
- $x \geq 0 \wedge x' = x + y \wedge y' = y + x \not\Rightarrow x' \geq 0$

Monolithic approach = worst case of incremental proofs.

FSIS - Algorithm

- FSIS: Finite-State Inductive Strengthening. It follows the incremental methodology [BM07].

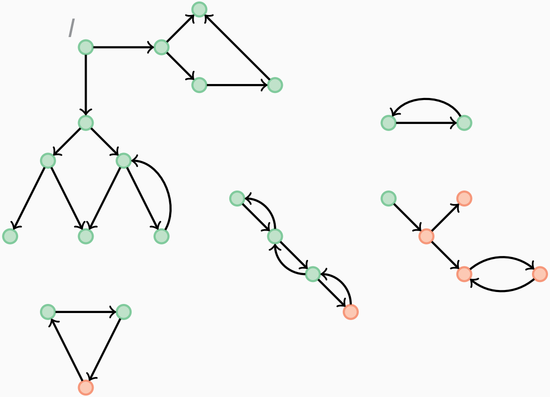
FSIS - Algorithm

- FSIS: Finite-State Inductive Strengthening. It follows the incremental methodology [BM07].
- “this algorithm is a result of asking the question: if the incremental method is often better for humans, might it be better for algorithms as well?" [Bra12];

FSIS - Algorithm

- FSIS: Finite-State Inductive Strengthening. It follows the incremental methodology [BM07].
- “this algorithm is a result of asking the question: if the incremental method is often better for humans, might it be better for algorithms as well?" [Bra12];
- the core of the algorithm is the generalization of an error state.

FSIS - Example



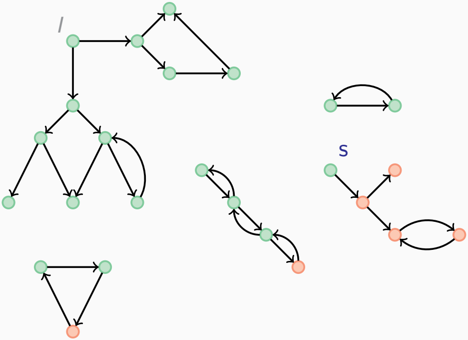
FSIS - Example - 1^{st} iteration

Check if P is inductive (relative to nobody). Check the validity of:

✓ $I \Rightarrow P$

✗ $P \wedge T \Rightarrow P'$

State s is a CTI.



FSIS - Example - 1^{st} iteration

- s is a cube returned by the SAT-solver; $\neg s$ is a clause encoding all states different from s;

FSIS - Example - 1^{st} iteration

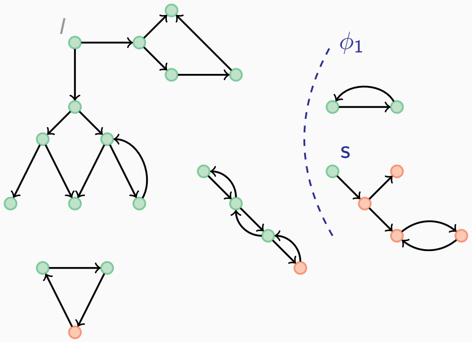
- s is a cube returned by the SAT-solver; $\neg s$ is a clause encoding all states different from s;
- generalization of error state s: find a clause ϕ_1 such that
  - $\phi_1 \subseteq \neg s$; *(it excludes s)*
  - ϕ_1 is inductive (relative to its own); *(it includes at least all the reachable states)*
  - ϕ_1 is minimal. *(it excludes the maximal number of non-reachable states)*

FSIS - Example - 1^{st} iteration

- s is a cube returned by the SAT-solver; $\neg s$ is a clause encoding all states different from s;
- generalization of error state s: find a clause ϕ_1 such that
  - $\phi_1 \subseteq \neg s$; *(it excludes s)*
  - ϕ_1 is inductive (relative to its own); *(it includes at least all the reachable states)*
  - ϕ_1 is minimal. *(it excludes the maximal number of non-reachable states)*
- if ϕ_1 does exist, it becomes the first incremental lemma.

FSIS - Example - 1^{st} iteration

ϕ_1 can be thought as a "boolean" cutting plane.



Which states are excluded by ϕ_1? (i) those who can reach s
(ii) states “similar" to s (they share with s the dropped literals).

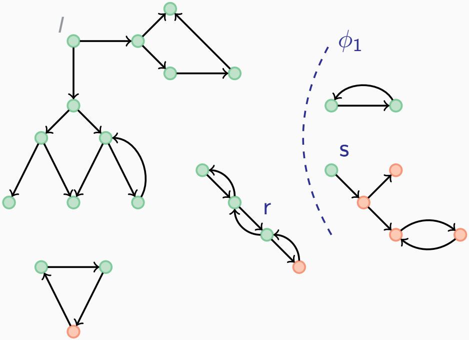
FSIS - Example - 2^{nd} iteration

Check if P is inductive relative to ϕ_1:

✓ $I \Rightarrow P$

✗ $\phi_1 \wedge P \wedge T \Rightarrow P'$

State r is a CTI.

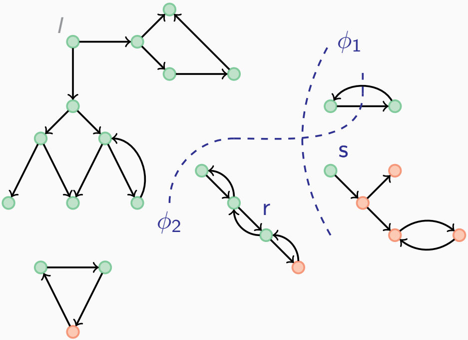


FSIS - Example - 2^{nd} iteration

- generalization of error state r:
  - $\phi_2 \subseteq \neg r$;
  - ϕ_2 is inductive relative to ϕ_1;
  - ϕ_2 is minimal;
- ϕ_2 is the second incremental lemma.

FSIS - Example - 2^{nd} iteration

- generalization of error state r:
  - $\phi_2 \subseteq \neg r$;
  - ϕ_2 is inductive relative to ϕ_1;
  - ϕ_2 is minimal;
- ϕ_2 is the second incremental lemma.



FSIS - Example - 2^{nd} iteration

Why "inductive relative to"?

- it would have been correct to generate ϕ_2 inductive (relative to its own), but it's more than what we need;
  - at the end we will consider the AND of all the lemmata;

FSIS - Example - 2^{nd} iteration

Why "inductive relative to"?

- it would have been correct to generate ϕ_2 inductive (relative to its own), but it's more than what we need;
    - at the end we will consider the AND of all the lemmata;
- in general, it is faster to generate "inductive relative to" clauses.
    - intuitively, we are considering many fewer states of the system.

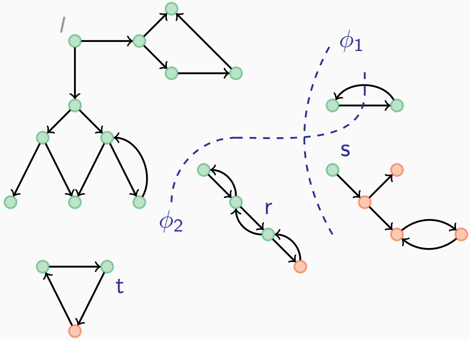
FSIS - Example - 3^{rd} iteration

Check if P is inductive relative to $\phi_1 \wedge \phi_2$:

✓ $I \Rightarrow P$

✗ $\phi_1 \wedge \phi_2 \wedge P \wedge T \not\Rightarrow P'$

State t is a CTI.

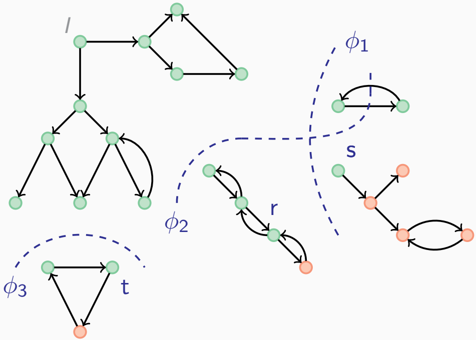


FSIS - Example - 3^{rd} iteration

- generalization of error state t:
  - $\phi_3 \subseteq \neg t$;
  - ϕ_3 is inductive relative to $\phi_1 \wedge \phi_2$;
  - ϕ_3 is minimal;
- ϕ_3 is the second incremental lemma.

FSIS - Example - 3^{rd} iteration

- generalization of error state t:
  - $\phi_3 \subseteq \neg t$;
  - ϕ_3 is inductive relative to $\phi_1 \wedge \phi_2$;
  - ϕ_3 is minimal;
- ϕ_3 is the second incremental lemma.



FSIS - Example - 4^{th} iteration

Check if P is inductive relative to $\phi_1 \wedge \phi_2 \wedge \phi_3$:

✓ $I \Rightarrow P$

✓ $\phi_1 \wedge \phi_2 \wedge \phi_3 \wedge P \wedge T \Rightarrow P'$

FSIS - Example - 4^{th} iteration

Check if P is inductive relative to $\phi_1 \wedge \phi_2 \wedge \phi_3$:

✓ $I \Rightarrow P$

✓ $\phi_1 \wedge \phi_2 \wedge \phi_3 \wedge P \wedge T \Rightarrow P'$

- $\phi_1 \wedge \phi_2 \wedge \phi_3 \wedge P$ is an inductive strengthening.
- P is $\mathcal{M}$-invariant.

Worst case

- suppose that an error state s does *not* have a minimal inductive generalization;

Worst case

- suppose that an error state s does *not* have a minimal inductive generalization;
- worst case: we proceed with the monolithic technique;

$$P := P \land \neg s$$

Worst case

- suppose that an error state s does *not* have a minimal inductive generalization;
- worst case: we proceed with the monolithic technique;

$$P := P \land \neg s$$

- eventually,
  - either $I \land \neg P$ is SAT: P is not invariant;
  - or we find an inductive strengthening $P \land \bigwedge_{i=0}^{n} \phi_i$;

Complexity and Parallelization

Complexity:

- it is on the convergence of the procedure, not on the calls to the SAT-solver as before;
- each SAT-solver call is relatively small compared to those made by BMC.

Complexity and Parallelization

Complexity:

- it is on the convergence of the procedure, not on the calls to the SAT-solver as before;
- each SAT-solver call is relatively small compared to those made by BMC.

Parallelization:

- straightforward; "by simply using a randomized decision procedure to obtain the CTIs, each process is likely to analyze a different part of the state-space." [BM07]

IC3

IC3 = **I**ncremental **C**onstruction of **I**nductive **C**lauses of **I**ndubitable **C**orrectness

IC3

IC3 = **I**ncremental **C**onstruction of **I**nductive **C**lauses of **I**ndubitable **C**orrectness

- FSIS sometimes enters a long search for the next relatively inductive clauses;

IC3

IC3 = **I**ncremental **C**onstruction of **I**nductive **C**lauses of **I**ndubitable **C**orrectness

- FSIS sometimes enters a long search for the next relatively inductive clauses;
- IC3 de-emphasized global information in favor of stepwise information: we will generate clauses that ensure that an error is unreachable up to some number of steps.

IC3 - Data Structures

Sequence of frames $F_0 (= I), F_1, F_2, \ldots, F_k$:

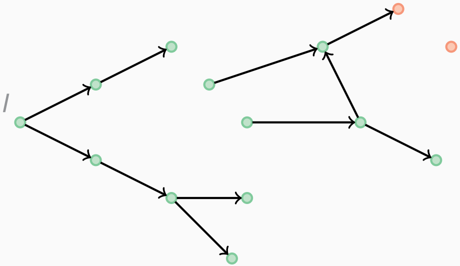
- each F_i is an over-approximation of the set of states reachable in at most k steps;
- each F_i is a set of clauses, *i.e.*, a CNF formula;
- the algorithm stops when $F_i \equiv F_{i+1}$. We will maintain the invariant that $\mathit{clauses}(F_{i+1}) \subseteq \mathit{clauses}(F_i)$: the equivalence check is simply a sintactic test: $F_i = F_{i+1}$.

IC3 - 1^{st} iteration

Check if there are counterexamples of length 0 or 1 with these two SAT-queries:

✗ $I \wedge \neg P$

✗ $F_0(= I) \wedge T \wedge \neg P'$



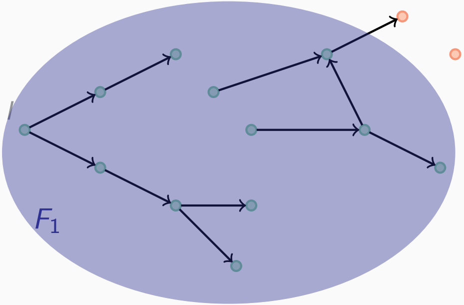
IC3 - 1^{st} iteration

Check if there are counterexamples of length 0 or 1 with these two SAT-queries:

✗ $I \wedge \neg P$

✗ $F_0(= I) \wedge T \wedge \neg P'$

Since $F_0 \wedge T \Rightarrow P'$, we set $F_1 := P$. (over-approximation)



IC3 - 2^{nd} iteration

At iteration k, check if $F_k \wedge T \wedge \neg P'$; in this case $(k = 1)$:

$$\checkmark \quad F_1 \wedge T \wedge \neg P'$$

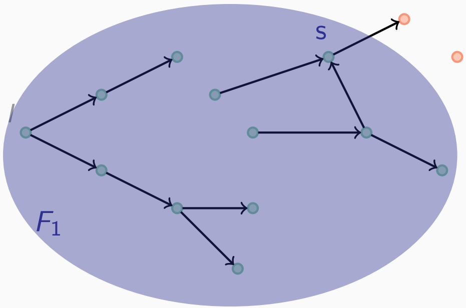
i.e., there exists an F_k-state that leads in one step to an error state?

IC3 - 2^{nd} iteration

At iteration k, check if $F_k \wedge T \wedge \neg P'$; in this case $(k = 1)$:

$$\checkmark \quad F_1 \wedge T \wedge \neg P'$$

i.e., there exists an F_k-state that leads in one step to an error state?



IC3 - 2^{nd} iteration (blocking phase)

$\neg s$ is inductive relative to $F_0 (= I)$: error state s is not reachable in at least $k = 1$ step.

IC3 - 2^{nd} iteration (blocking phase)

$\neg s$ is inductive relative to $F_0 (= I)$: error state s is not reachable in at least $k = 1$ step. We find a minimal $\phi_1 \subseteq \neg s$ such that ϕ_1 is inductive relative to $F_0 (= I)$.

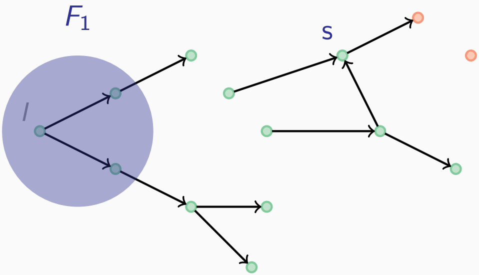
$\Rightarrow \phi_1$ excludes the error state s (and similar states) but contains at least all the states reachable in at most $k = 1$ steps.

IC3 - 2^{nd} iteration (blocking phase)

$\neg s$ is inductive relative to $F_0(= I)$: error state s is not reachable in at least $k = 1$ step. We find a minimal $\phi_1 \subseteq \neg s$ such that ϕ_1 is inductive relative to $F_0(= I)$.

$\Rightarrow$ ϕ_1 excludes the error state s (and similar states) but contains at least all the states reachable in at most $k = 1$ steps.

We add ϕ_1 to all the previous frames. In this case $F_1 := F_1 \wedge \phi_1$.



IC3 - Blocking phase

We have found a CTI s such that $s \models F_k \land T \land \neg P'$.

$\Rightarrow$ we want to generalize the error s or to prove that it's reachable from an initial state

IC3 - Blocking phase

We have found a CTI s such that $s \models F_k \wedge T \wedge \neg P'$.

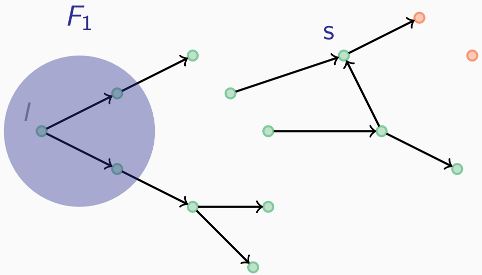
$\Rightarrow$ we want to generalize the error s or to prove that it's reachable from an initial state

if $\neg s$ is inductive relative to F_{k-1}, then generate a minimal subclause $c \subseteq \neg s$ inductive relative to F_{k-1}, *i.e.*, c holds for at least all states reachable in i steps.

$\Rightarrow$ add c to frames $F_0 \ldots F_{k+1}$, *i.e.*, refine the over-approximations.

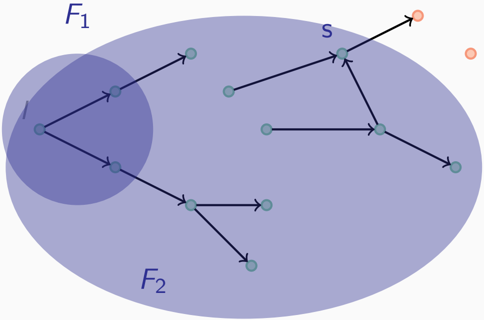
IC3 - 2^{nd} iteration

We create a new frame only when $F_k \land T \Rightarrow P'$ is valid.



IC3 - 2^{nd} iteration

We create a new frame only when $F_k \wedge T \Rightarrow P'$ is valid.
In this case, $F_1 \wedge T \Rightarrow P'$ is valid. We create a new frame $F_2 := P$.



IC3 - Propagation phase

Propagation phase: After creating a new frame $F_{k+1} := P$, we perform the propagation phase: we push forward the clause discovered in frame F_i for some i.

IC3 - Propagation phase

Propagation phase: After creating a new frame $F_{k+1} \coloneqq P$, we perform the propagation phase: we push forward the clause discovered in frame F_i for some i.

For all $0 \leq i \leq k$ and $c \in F_i$, check if

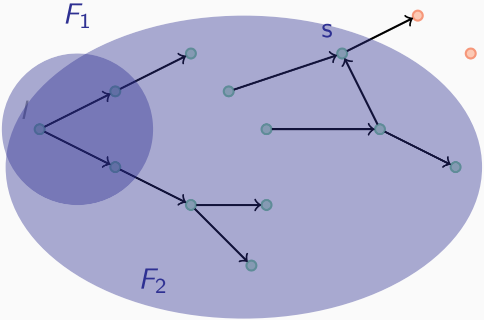
$$F_i \wedge T \Rightarrow c'$$

If $c \notin clauses(F_{i+1})$, then set $F_{i+1} \coloneqq F_{i+1} \cup \{c\}$

$\Rightarrow$ it propagates forward the errors
$\Rightarrow$ it helps the discovery of mutually inductive clauses

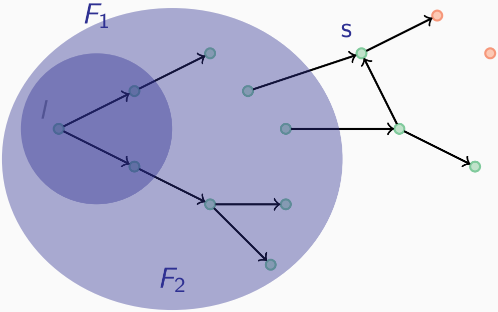
IC3 - 3^{rd} iteration

Check if $F_2 \land T \land \neg P'$ (✓). $\neg s$ is inductive relative to F_1: error state s is not reachable for at least $k = 2$ steps.



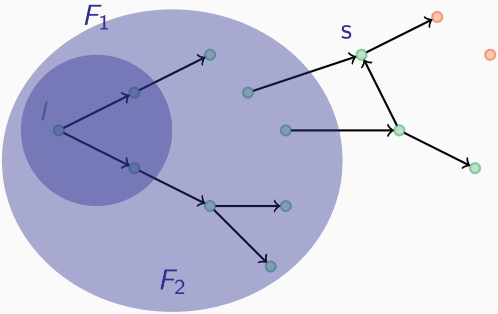
IC3 - 3^{rd} iteration

Check if $F_2 \land T \land \neg P'$ (✓). $\neg s$ is inductive relative to F_1: error state s is not reachable for at least $k = 2$ steps. Blocking phase: find minimal subclause $\phi_2 \subseteq \neg s$ inductive relative to F_1. Add ϕ_2 to frames F_0 and F_1.



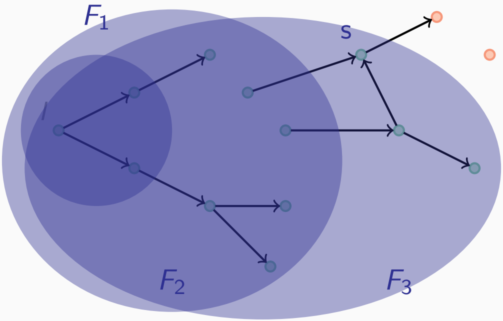
IC3 - 4^{rd} iteration

Since $F_2 \wedge T \Rightarrow P'$ is valid, we create a new frame $F_3 := P$.



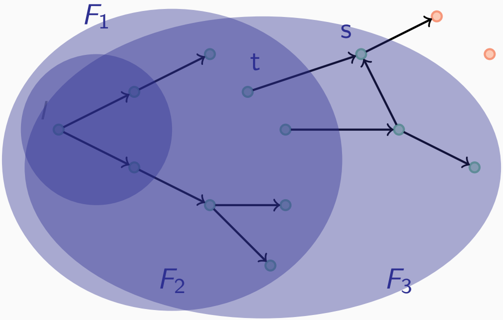
IC3 - 4^{rd} iteration

Since $F_2 \wedge T \Rightarrow P'$ is valid, we create a new frame $F_3 := P$.



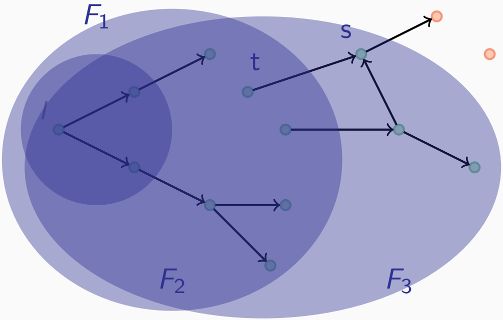
IC3 - 4^{rd} iteration

Again $F_3 \wedge T \wedge \neg P'$ (✓). But now $\neg s$ is not inductive relative to F_2



IC3 - 4^{rd} iteration

Again $F_3 \wedge T \wedge \neg P'$ (✓). But now $\neg s$ is not inductive relative to F_2: error state s could be reachable in $k = 3$ steps ...



IC3 - 4^{rd} iteration

Instead of generating a clause that excludes s (it is possible), we call the algorithm recursively on the predecessor t of s

... remember that t could still be reachable as far as we know ...

IC3 - 4^{rd} iteration

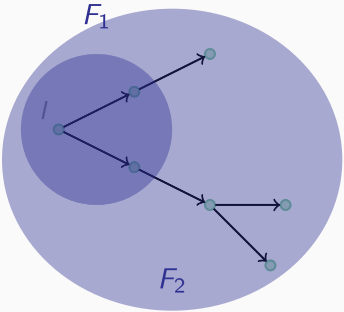
Instead of generating a clause that excludes s (it is possible), we call the algorithm recursively on the predecessor t of s

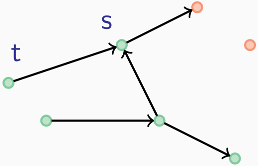
... remember that t could still be reachable as far as we know ...

"t is the new s" ;-)

IC3 - Recursion

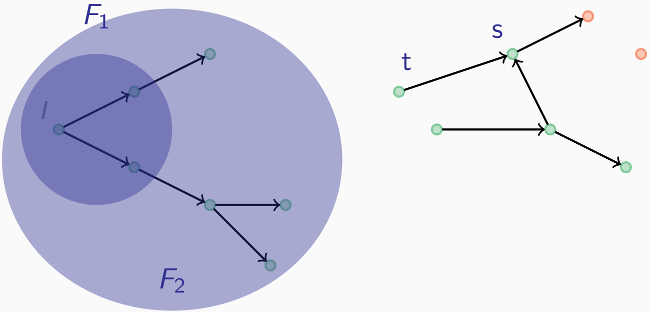
We want to remove error state t from F_2. $\neg t$ is inductive relative to F_1: find min subclause $\phi_4 \subseteq \neg t$ and add it to F_0,F_1,F_2.





IC3 - Recursion

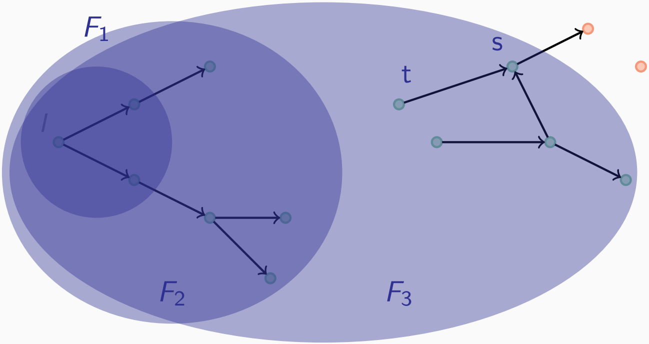
We want to remove error state t from F_2. $\neg t$ is inductive relative to F_1: find min subclause $\phi_4 \subseteq \neg t$ and add it to F_0,F_1,F_2.



If in this process we had gone back with recursion until an initial state, then we would have found a counterexample.

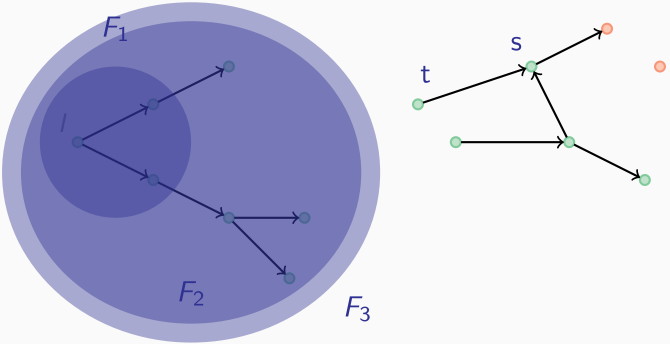
IC3 - Termination

Now error state s in frame F_3 can be generalized: find min clause $\phi_5 \subseteq \neg s$ inductive relative to F_2.



IC3 - Termination

Now error state s in frame F_3 can be generalized: find min clause $\phi_5 \subseteq \neg s$ inductive relative to F_2.



$F_2 = F_3$: IC3 terminates with True.

Beyond IC3

- FAIR: IC3 for ω-regular properties (*e.g.*, LTL) [Bra+11];
- IICTL: IC3 for CTL properties [HBS12];
- Infinite-state: software model checking via IC3 [CG12].

IC3 with nuXmv

- `go_msat` : initialise the system for infinite state model checking using the SMT solver mathsat5.
- `check_invar_ic3` : check invariant using IC3-based algorithm.
- `check_ltlspec_ic3` : check LTL using IC3-based algorithm.

But IC3 is for reachability, how do we apply it for checking LTL properties?

References

Aaron R Bradley and Zohar Manna. "Checking safety by inductive generalization of counterexamples to induction". In: *Formal Methods in Computer Aided Design (FMCAD'07)*. IEEE. 2007, pp. 173–180.

Aaron R Bradley, Fabio Somenzi, Zyad Hassan, and Yan Zhang. "An incremental approach to model checking progress properties". In: *2011 Formal Methods in Computer-Aided Design (FMCAD)*. IEEE. 2011, pp. 144–153.

Aaron R Bradley. "Understanding ic3". In: *International Conference on Theory and Applications of Satisfiability Testing*. Springer. 2012, pp. 1–14.

Alessandro Cimatti and Alberto Griggio. "Software model checking via IC3". In: *International Conference on Computer Aided Verification*. Springer. 2012, pp. 277–293.

Zyad Hassan, Aaron R Bradley, and Fabio Somenzi. "Incremental, inductive CTL model checking". In: *International Conference on Computer Aided Verification*. Springer. 2012, pp. 532–547.