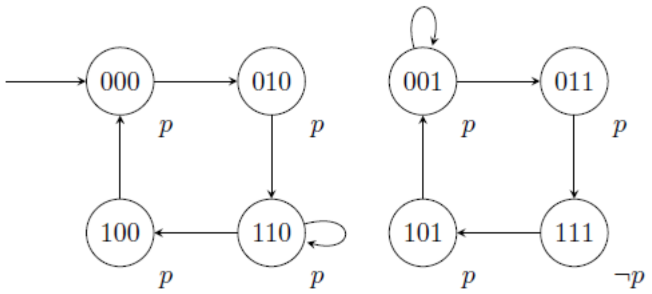
Consider the following Kripke structure $K$, with states $(x_1, x_2, x_3) \in \{0,1\}^3$ and atomic proposition $p$.




**Task 1.** [ **50 points** ] We want to use $k$-induction to prove that $p$ is always true.

1.1 Will $k$-induction succeed in proving the property? If so, what is the smallest $k$ such that $k$-induction proves the property to be true? [ **10 point** ]

1.2 Write the $k$ induction formulae, both base case and induction case, for $k = 2$. [ **20 points** ]

1.3 Are the formulae satisfiable? Explain. [ **20 points** ]

For task 1.2, you can use the formulas $R$, $S_0$, and $p$ for the transition relation, the initial states, and the property $p$, respectively, without explicitly stating the concrete expression of the formulas.

1.1 Yes. For $k = 3$, the following formula is UNSAT, so the system is correct

$$\bigwedge_{i=1}^{k+1} R(s_i, s_{i+1}) \wedge \bigwedge_{i=1}^{k+1} p(s_i) \wedge \neg p(s_{k+2})$$

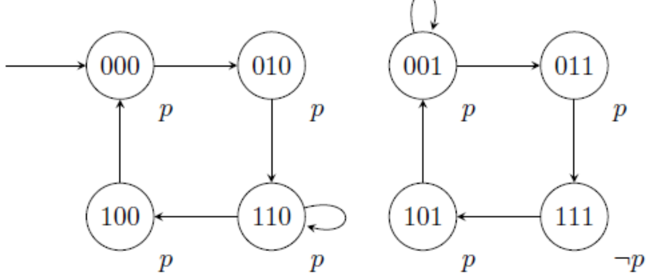
$$\wedge \bigwedge_{i=1}^{k+1} \bigwedge_{j=i+1}^{k+1} s_i \neq s_j.$$

1.2

Base: $S_0(s_1) \wedge \bigwedge_{i=1}^{k} R(s_i, s_{i+1}) \wedge \bigvee_{i=1}^{k+1} \neg p(s_i)$

Induction: $\bigwedge_{i=1}^{3} R(s_i, s_{i+1}) \wedge \bigwedge_{i=1}^{3} p(s_i)$
$\wedge \neg p(s_{k+2}) \wedge \bigwedge_{i=1}^{3} \bigwedge_{j=i+1}^{3} s_i \neq s_j$

1.3 The base formula is satisfiable if there is a path from S0 to $\neg p$ of length $k$, which there isn't, so it's unsat. The induction formula is SAT with path 101 – 001 – 011 – 111



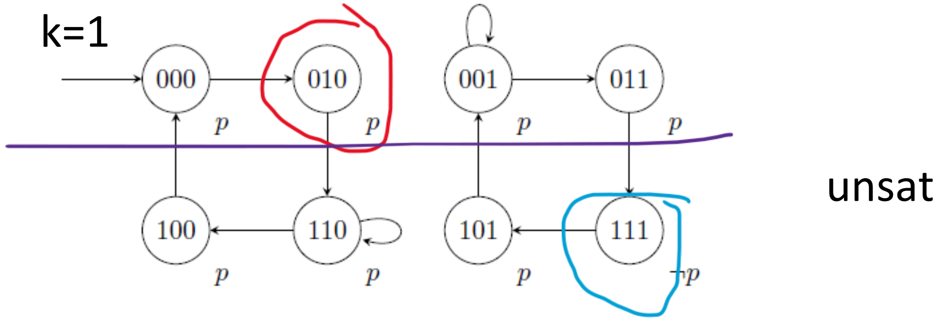

**Task 2.** [ **50 points** ] Use Model Checking with Craig Interpolants to prove that $p$ is always true.

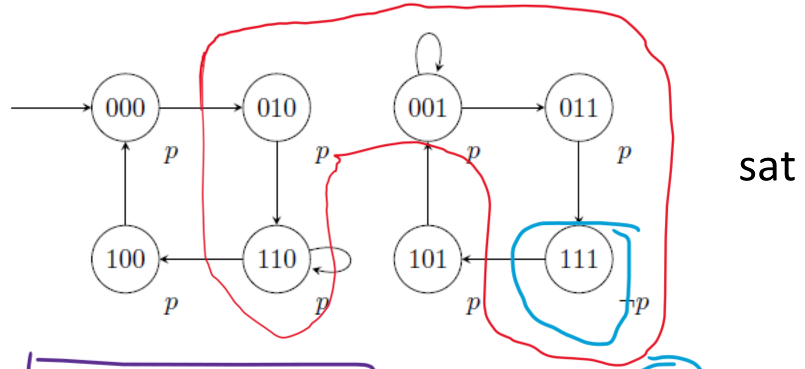
Clearly indicate the steps. Show the interpolants as formulas, for anything else, you can use set notation. You can also draw the sets, but use enough copies of the Kripke structure to make sure we can understand your steps, at least one for every $k$.

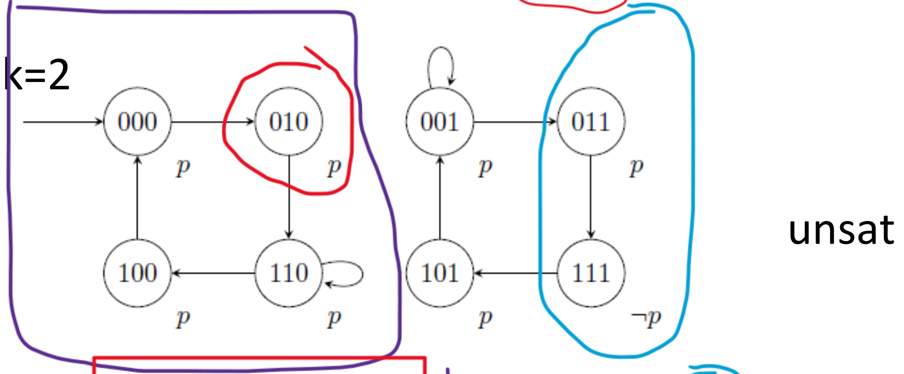
Use the same heuristic shown in class to find the interpolants. The heuristic shown in class is a hack, but it works in this example.

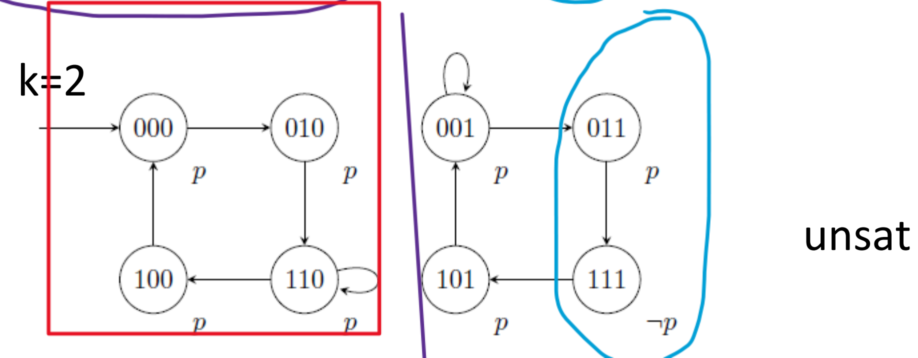
**procedure** CraigReachability(model $M$, $p \in AP$)
  if $S_0 \wedge \neg$p is SAT return "$M \not\models$ AG $p$ ";
  $k$ := 1;
  $Q := S_0(s_0)$;
  **while** true **do**
    $A := Q(s_0) \wedge R(s_0, s_1)$ ;
    $B$ =: $\bigvee_{i=1}^{k-1} R(s_i, s_{i+1}) \wedge \bigvee_{i=1}^{k} \neg p(s_i)$;
      **if** $A \wedge B$ is SAT **then**
        **if** $Q = S_0$ then return "$M \not\models$ AG $p$";
        Increase $k$
        $Q := S_0(s_0)$;
        **else**
        $I$ = interpolate $A$ and $B$;
        If $I(s_0) == Q$ then return "$M \models$ AG $p$";
        $Q := Q \vee I(s_0)$;
        **end if**
  **end while**
**end procedure**





# Property-Directed Reachability
# *or IC3*

(a simplified version)

Aaron Bradley

# PDR

Property-Directed Reachability or IC3

- Makes no copies of transition relation – memory efficient
- Overapproximate postimage (like interpolation)

# PDR Notation

Formula $X(V) \wedge R(V, V') \wedge Y(V')$ is shortened to $\boldsymbol{X} \wedge \boldsymbol{R} \wedge \boldsymbol{Y}'$

Meaning: there is an edge from $s$ to $s'$

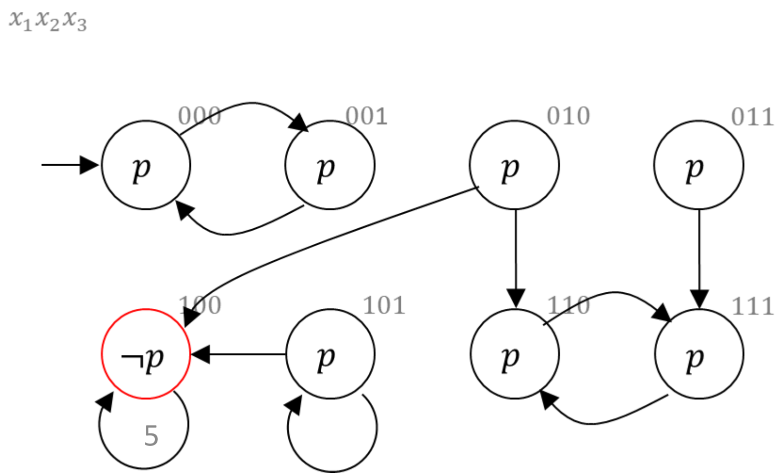
**s := SAT($\boldsymbol{A} \wedge \boldsymbol{R} \wedge \boldsymbol{B}'$):**

- $s :=$ $FALSE$ if ¬SAT(A $\wedge R \wedge B'$)
- $s$ := a state in $A$ with an edge to a state in $B$, otherwise

Example

$SAT(\mathrm{x}_1 \wedge \mathrm{R} \wedge \neg x_1)$

$s :=$ $SAT(\neg \mathrm{x}_1 \wedge R \wedge x_1')$



# PDR Notation

Formula $X(V) \wedge R(V, V') \wedge Y(V')$ is shortened to $\boldsymbol{X} \wedge \boldsymbol{R} \wedge \boldsymbol{Y}'$

Meaning: there is an edge from $s$ to $s'$

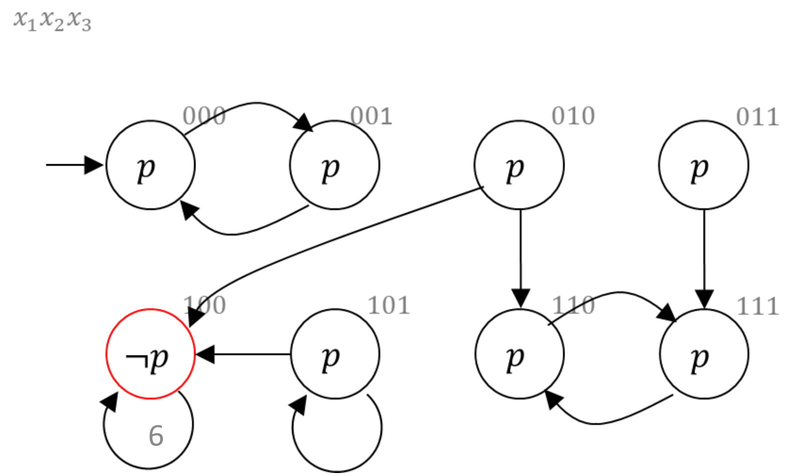
**s := SAT($\boldsymbol{A} \ \wedge \boldsymbol{R} \wedge \boldsymbol{B}'$):**

- $s :=\ FALSE$ if ¬SAT(A $\wedge$ $R \wedge B'$)
- $s\ \ :=$ a state in $A$ with an edge to a state in $B$, otherwise

## Example

$SAT(\mathrm{x}_1 \wedge \mathrm{R} \wedge \neg x_1) = FALSE.$

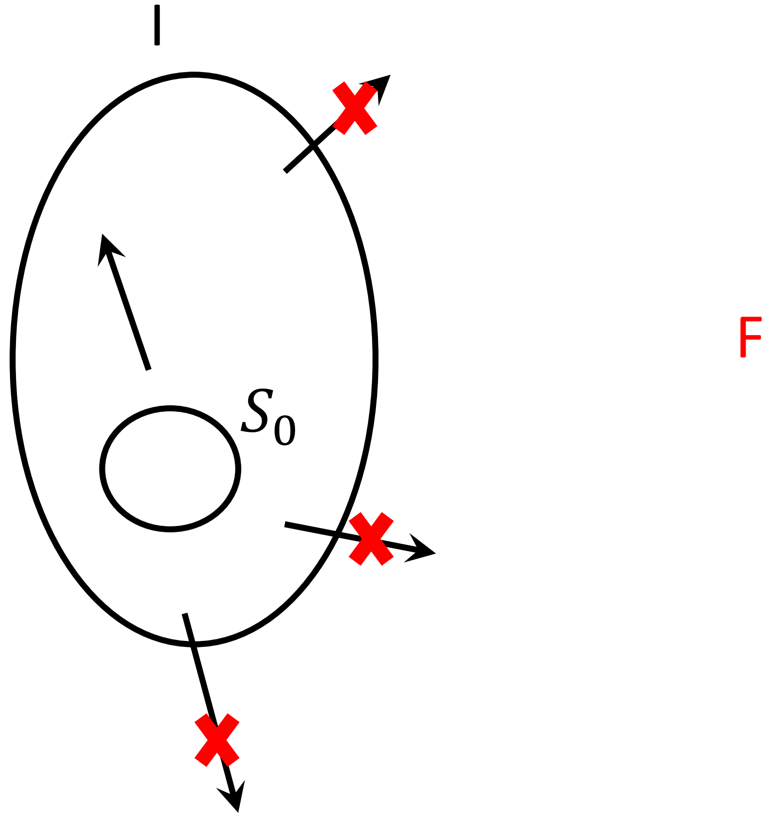
$s :=\ SAT(\neg \mathrm{x}_1 \wedge R \wedge x_1')$ can give $\neg x_1 \wedge x_2 \wedge \neg x_2$



# PDR: Notation

**Definition**

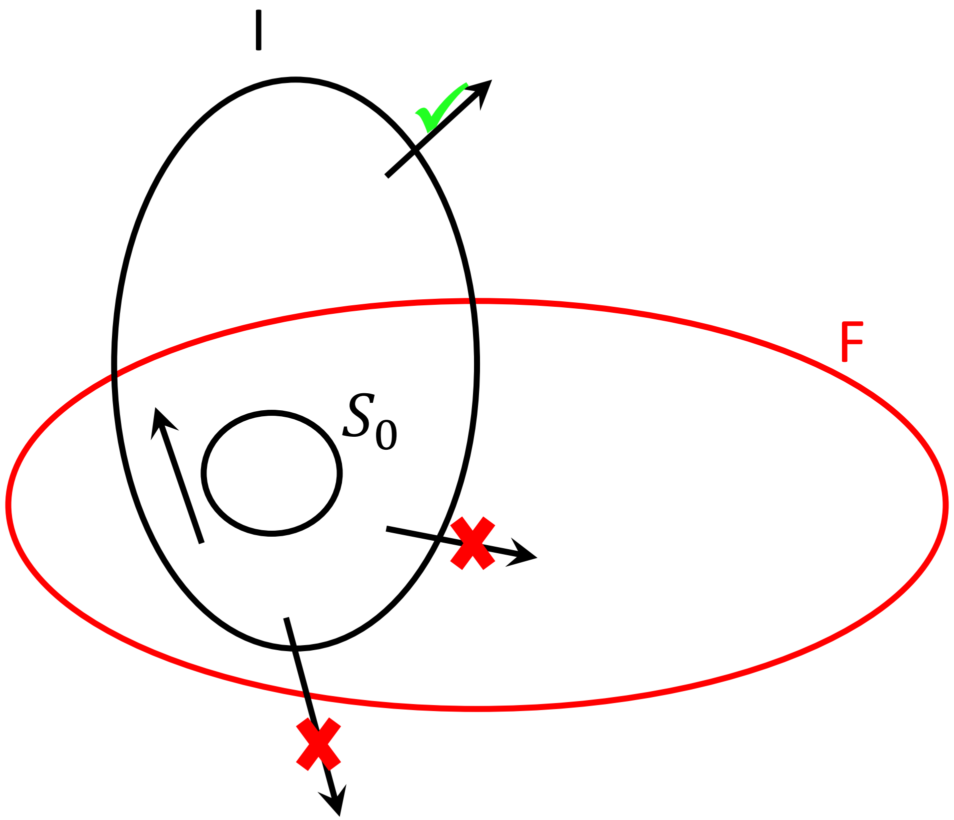
- $I \subseteq S$ is inductive if
  1. $S_0 \rightarrow I$ $(S_0 \subseteq I)$
  2. $I \wedge R \rightarrow I'$ $(postimage(I) \subseteq I)$



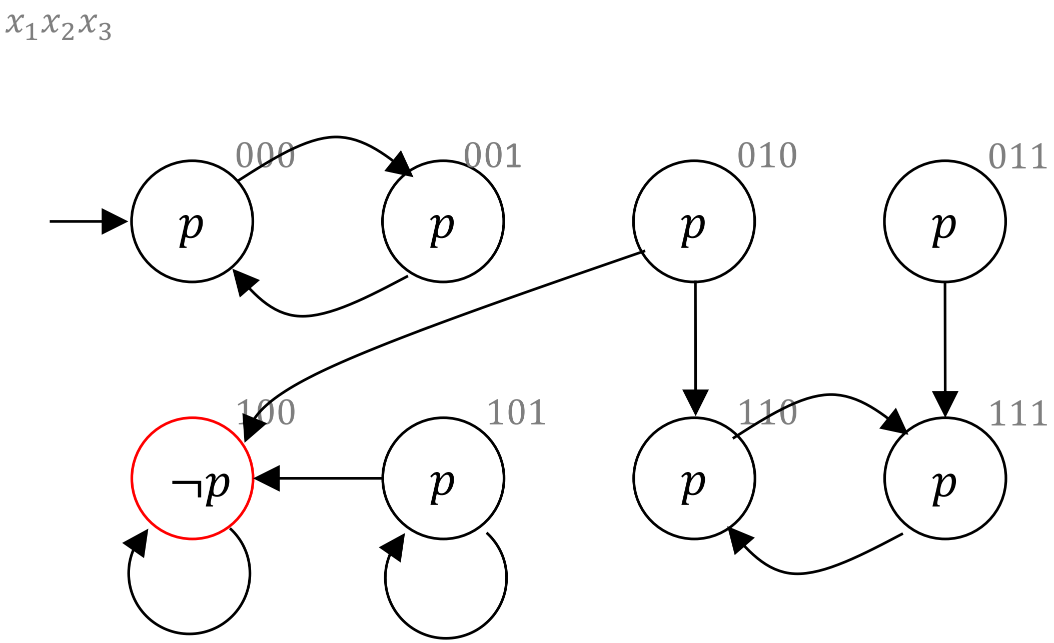
# PDR: Notation

**Definition**

- $I \subseteq S$ is inductive if
  1. $S_0 \rightarrow I \;\; (S_0 \subseteq I)$
  2. $I \wedge R \;\; \rightarrow I' \; (postimage(I) \subseteq I)$
- $I \subseteq S$ is inductive relative to F if
  1. $S_0 \rightarrow I \;\; (S_0 \subseteq I)$
  2. $I \wedge F \wedge R \;\; \rightarrow I' \; (postimage(F \cap I) \subseteq I)$



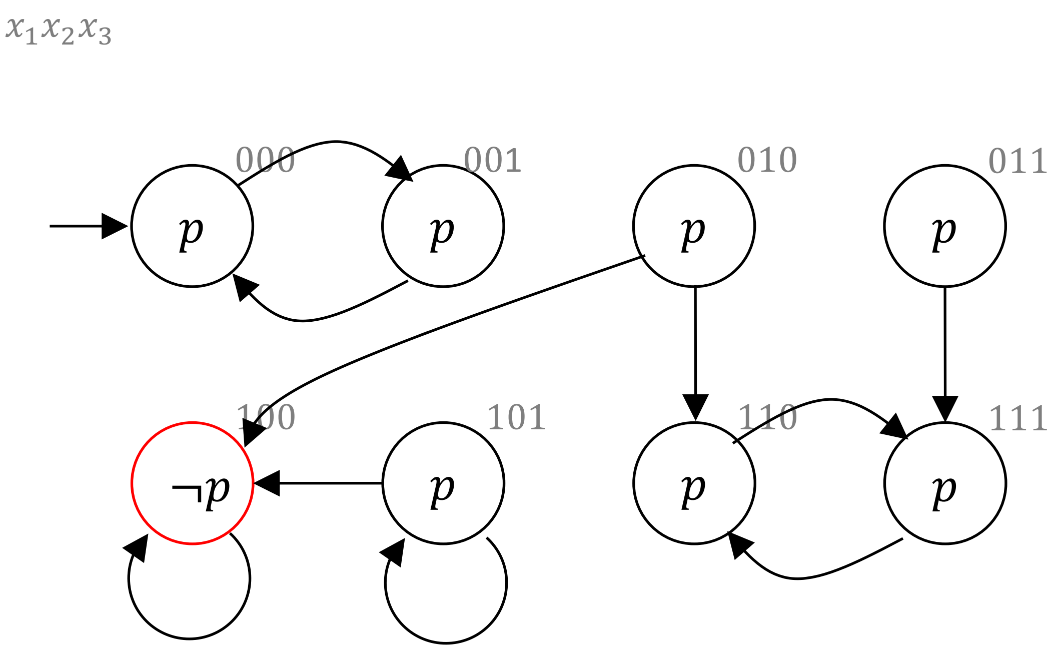
# Relative Inductiveness




Inductive facts

- Is $\neg x_1$ inductive?
- Is $\neg x_2$ inductive?
- Is $\neg x_1$ inductive relative to $x_2$?

# Relative Inductiveness




Inductive facts

- Is $\neg x_1$ inductive?
  - $S_0 \rightarrow \neg x_1$
  - $\neg x_1 \wedge R \rightarrow \neg x'_1$ is false
  - No!
- Is $\neg x_2$ inductive?
  - $S_0 \rightarrow \neg x_2$
  - $\neg x_2 \wedge R \rightarrow \neg x'_2$
  - Yes!
- Is $\neg x_1$ inductive relative to $\neg x_2$?
  - $S_0 \rightarrow \neg x_1$
  - $\neg x_2 \wedge \neg x_1 \wedge R \rightarrow \neg x'_1$
  - **Yes**!

**Idea: Find (relatively) inductive facts.**

# PDR: Data Structures & Invariants

**Data Structures**

Clause: Disjunction of literals. E.g. $\neg x_1 \vee x_2 \vee \neg x_2$

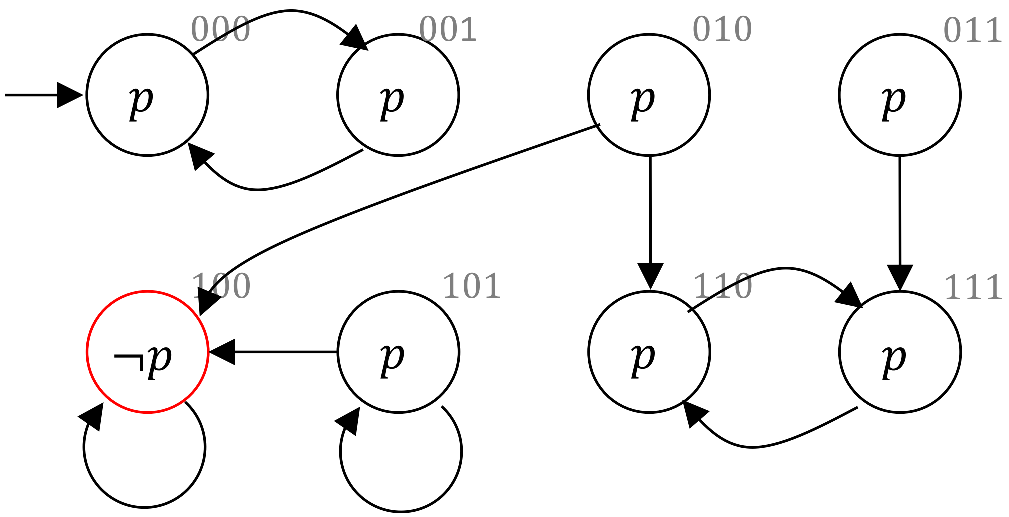
Cube: Conjunction of literals. E.g. $\neg x_1 \wedge x_2 \wedge \neg x_3$

- A state is a cube
- Clause and cubes are sets of states.

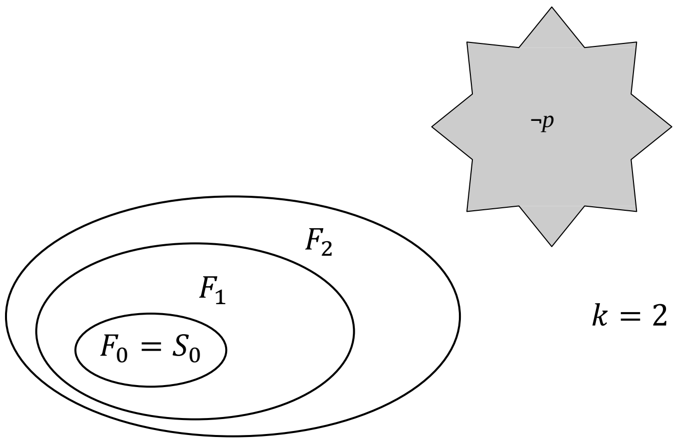
  Longer clauses – more states. Longer cubes – fewer states

Formulas $F_0, \ldots, F_k$ over $V$,

stored as sets of Clauses as CNF $(F_0, \ldots, F_k \subseteq S)$



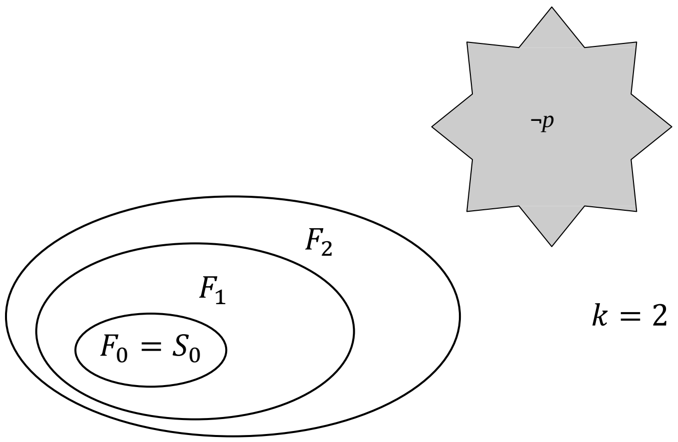
# PDR: Data Structures & Invariants




## Invariants

- I1: $S_0 \rightarrow F_0$. $(S_0 \subseteq F_0)$
- I2: $F_i \rightarrow F_{i+1}$. $(F_i \subseteq F_{i+1})$
  - I2': $F_i = F_{i+1} \wedge c_{i1} \wedge \cdots \wedge c_{in}$
- I3: $F_i \rightarrow P$. $(F_i \subseteq P)$
- I4: $F_i \wedge R \rightarrow F'_{i+1}$ $(postimg(F_i) \subseteq F_{i+1})$

# PDR: Data Structures & Invariants




**Invariants**

- I1: $S_0 = F_0$ ($S_0 = F_0$)
- I2: $F_i \rightarrow F_{i+1}$ ($F_i \subseteq F_{i+1}$)
  - I2’: $F_i = F_{i+1} \land c_{i1} \land \cdots \land c_{in}$
- I3: $F_i \rightarrow P$ ($F_i \subseteq P$)
- I4: $F_i \land R \rightarrow F'_{i+1}$ ($postimg(F_i) \subseteq F_{i+1}$)

**Facts.** Suppose we have frames $F_0$ through $F_k$

1. $\forall 0 < i \leq k$: There is no trace from $F_i$ to $\neg p$ of $k - i$ edges or less (I3,I4)
2. There is no counterexample with $k$ edges or less (with I1)
3. If $F_i = F_{i+1}$ then system is correct. (By I3, I4, $F_i$ is an inductive invariant)

# Storing Frames

I2’: $F_i = F_{i+1} \wedge c_{i1} \wedge \cdots \wedge c_{in}$

For each frame $F_i$, store only $\Delta_i$, the clauses that are new to that frame.

$F_i = F_{i+1}$ iff $\Delta_i = \emptyset$

# PDR, First Version

**function** PDR(Model $M$)

**if** SAT($S_0 \wedge \neg P$) **or** SAT($S_0 \wedge R \wedge \neg P'$) **then FAIL**

$F_0$ := $S_0$; $F_1$ := $P$; $k$ :=1;

**while**(true)

**while**($s$ := SAT($F_k \wedge R \wedge \neg P'$))

removeBad(k, $s$)

$k$++; $F_k$ := $P$

**if** $\exists 0 \leq i < k-1: F_i = F_{i+1}$ **then SUCCEED**

remove states in $F_k$ with edge to $\neg P$

// post: $\neg SAT(F_i \wedge s)$

**function** removeBad($i \in N$, state s)

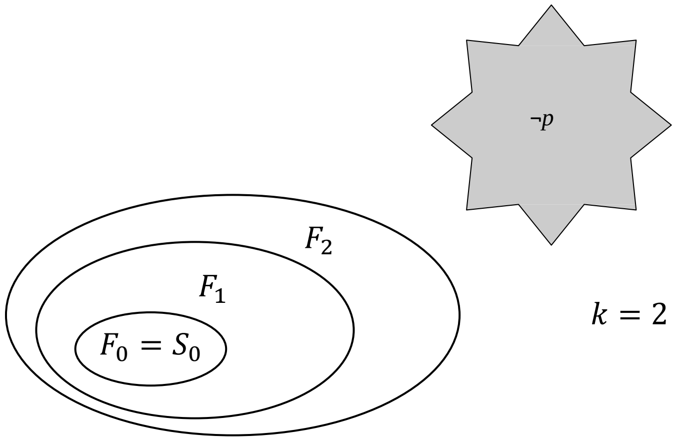
**if** SAT($S_0 \wedge s$) **then FAIL**

**while**($t$ := SAT($F_{i-1} \wedge R \wedge s'$))

removeBad($i-1, t$)

$\forall 0 < j \leq i: F_j \coloneqq F_j \wedge \neg s$

remove states in $F_i$ with path to $\neg P$ of length $k-i+1$



# PDR, First Version

**function** PDR(Model $M$)

  **if** SAT($S_0 \wedge \neg P$) **or** SAT($S_0 \wedge R \wedge \neg P'$) **then FAIL**

  $F_0$ =: $S_0$ ;$F_1$ =: $P$ ;$k$;1=:

  **while**(true)

you are here →

    **while**($s$ := SAT($F_k \wedge\ R \wedge \neg P'$))

      removeBad(k, $s$)

    $k$++; $F_k$ =: $P$

    **if** $\exists 0 \leq i < k - 1 : F_i = F_{i+1}$ **then SUCCEED**

// post: $\neg SAT(F_i \wedge s)$

**function** removeBad($i \in N$, state s)

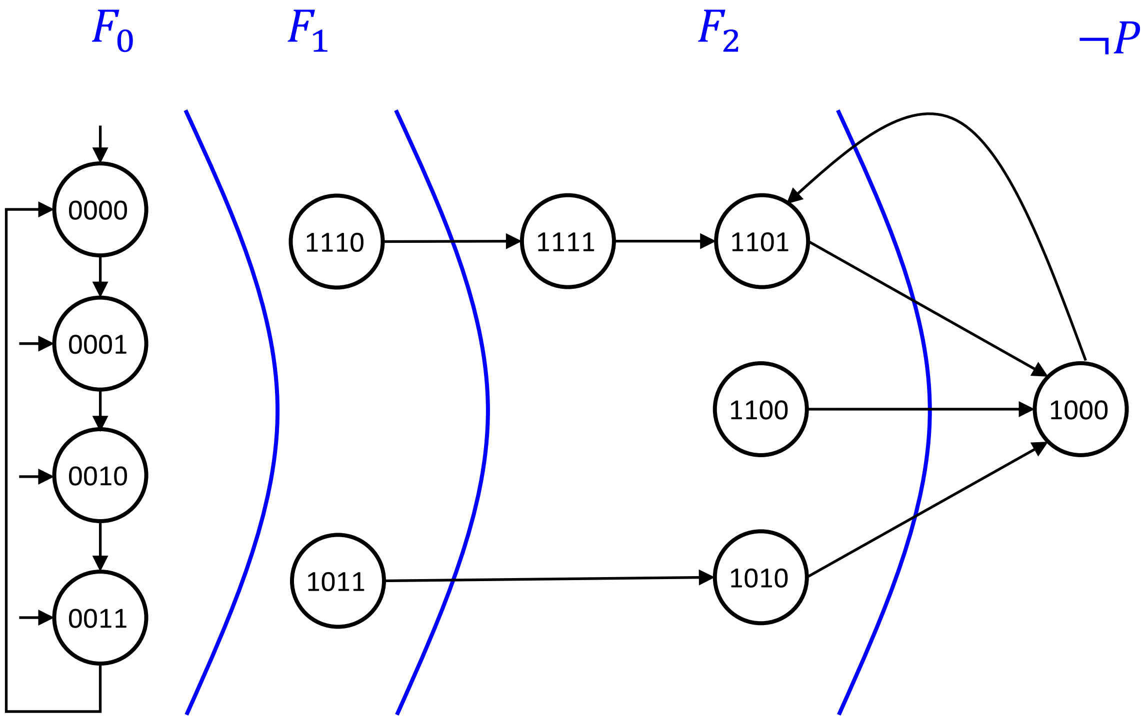
  **if** SAT($S_0 \wedge s$) **then FAIL**

  **while**($t$ := SAT($F_{i-1} \wedge R \wedge s'$))

    removeBad($i - 1, t$)

  $\forall 0 < j \leq i : F_j \coloneqq F_j \wedge \neg$s

$k = 2$




Goal: look for counterexamples of length 3.
If we find one – done
If we don't – We have an $F_3$

# PDR, First Version

**function** PDR(Model $M$)

  **if** SAT($S_0 \wedge \neg P$) **or** SAT($S_0 \wedge R \wedge \neg P'$) **then FAIL**

  $F_0$ =: $S_0$ ;$F_1$ =: $P$ ;$k$;1=:

  **while**(true)

you are here →  **while**($s$ := SAT($F_k \wedge\ R \wedge \neg P'$))

      removeBad(k, $s$)

    $k$++; $F_k$ =: $P$

    **if** $\exists 0 \leq i < k-1 : F_i := F_{i+1}$ **then SUCCEED**

// post: $\neg SAT(F_i \wedge s)$

**function** removeBad($i \in N$, state s)

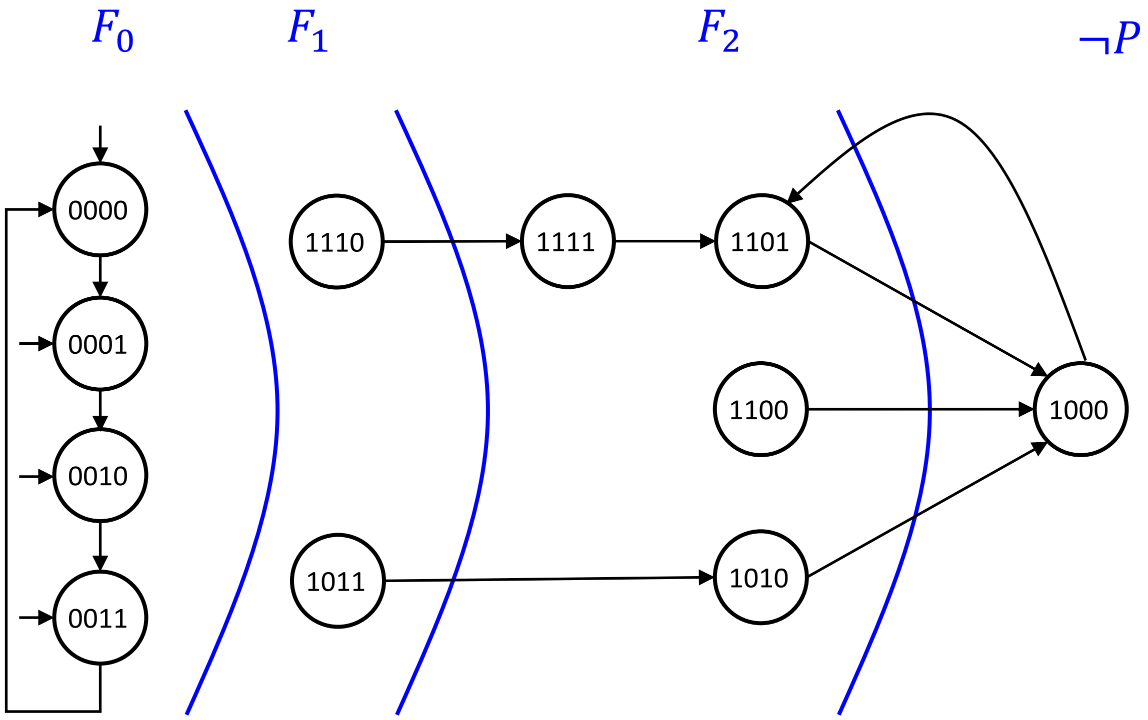
  **if** SAT($S_0 \wedge s$) **then FAIL**

  **while**($t$ := SAT($F_{i-1} \wedge R \wedge s'$((

    removeBad($i - 1, t$)

  $\forall 0 < j \leq i : F_j \coloneqq F_j \wedge \neg$s

$k = 2$




Do the invariants hold?

# PDR, First Version

**function** PDR(Model $M$)

**if** SAT($S_0 \wedge \neg P$) **or** SAT($S_0 \wedge R \wedge \neg P'$) **then FAIL**

$F_0$ =: $S_0$ ;$F_1$ =: $P$ ;$k$;1=:

**while**(true)

you are here

**while**($s$ := SAT($F_k \wedge \ R \wedge \neg P'$ ((

removeBad(k, $s$)

$k$++; $F_k$ =: $P$

**if** $\exists 0 \leq i < k-1 : F_i := F_{i+1}$ **then SUCCEED**

// post: $\neg SAT(F_i \wedge s)$

**function** removeBad($i \in N$, state s)

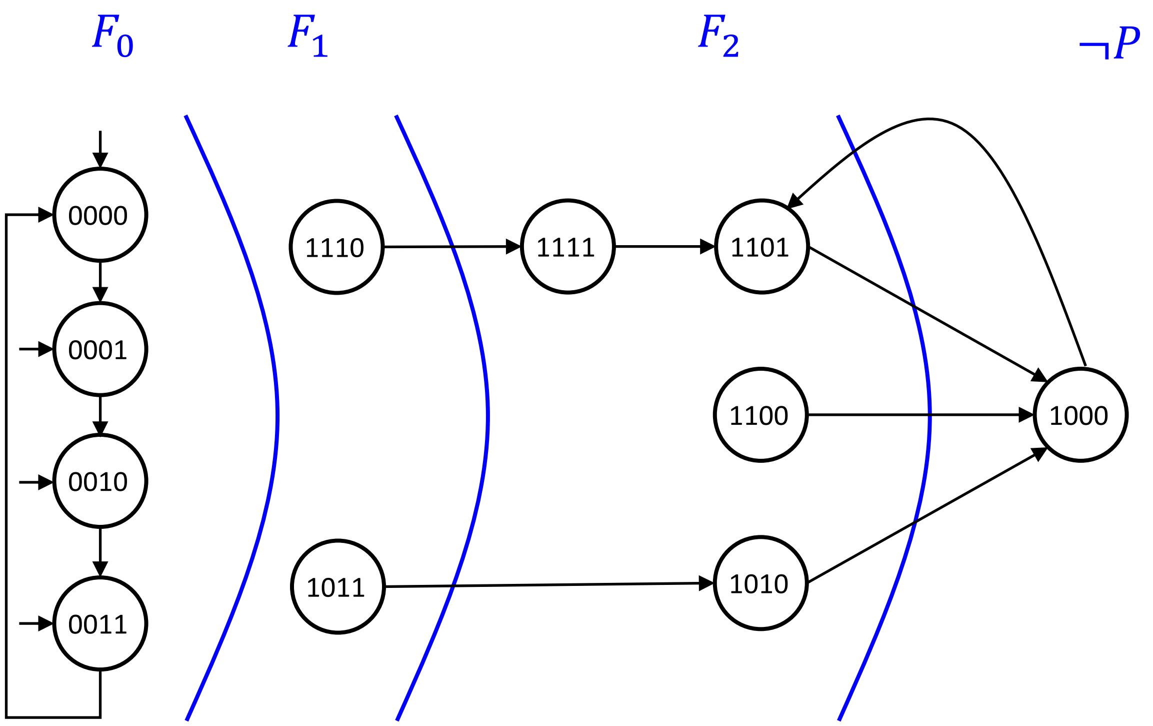
**if** SAT($S_0 \wedge s$) **then FAIL**

**while**($t$ := SAT($F_{i-1} \wedge R \wedge s'$))

removeBad($i-1, t$)

$\forall 0 < j \leq i : F_j \coloneqq F_j \wedge \neg$s

$k = 2$




Suppose $s$ := 1101

removeBad(2, 1101)

SAT($F_{i-1} \wedge R \wedge s'$) = FALSE

$F_2 \coloneqq F_2 \wedge \neg(x_1 \wedge x_2, \neg x_3 \wedge x_4)$, same for $F_1$

# PDR, First Version

**function** PDR(Model $M$)

  **if** SAT($S_0 \wedge \neg P$) **or** SAT($S_0 \wedge R \wedge \neg P'$) **then FAIL**

  $F_0$ =: $S_0$ ;$F_1$ =: $P$ ;$k$;1=:

  **while**(true)

you are here →   **while**($s$ := SAT($F_k \wedge\ R \wedge \neg P'$ ((

    removeBad(k, $s$)

   $k$++; $F_k$ =: $P$

   **if** $\exists 0 \leq i < k - 1: F_i := F_{i+1}$ **then SUCCEED**

// post: $\neg SAT(F_i \wedge s)$

**function** removeBad($i \in N$, state s)

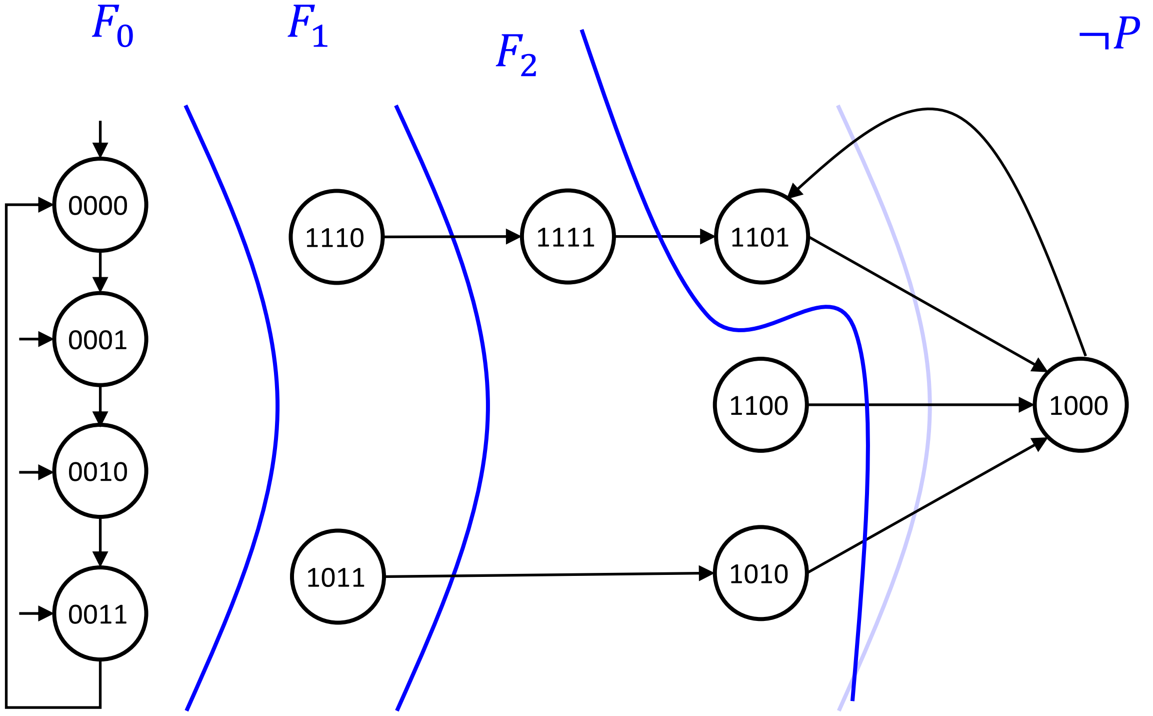
  **if** SAT($S_0 \wedge s$) **then FAIL**

  **while**($t$ := SAT($F_{i-1} \wedge R \wedge s'$((

   removeBad($i - 1, t$)

  $\forall 0 < j \leq i: F_j := F_j \wedge \neg$s

$k = 2$




Suppose $s$ := 1101

removeBad(2, 1101)

SAT($F_{i-1} \wedge R \wedge s'$) = FALSE

$F_2 := F_2 \wedge \neg(x_1 \wedge x_2, \neg x_3 \wedge x_4)$, same for $F_1$

# PDR, First Version

**function** PDR(Model $M$)

**if** SAT($S_0 \wedge \neg P$) **or** SAT($S_0 \wedge R \wedge \neg P'$) **then FAIL**

$F_0$ =: $S_0$ ;$F_1$ =: $P$ ;$k$;1=:

**while**(true)

you are here

**while**($s$ := SAT($F_k \wedge\ R \wedge \neg P'$ ((

removeBad(k, $s$)

$k$++; $F_k$ =: $P$

**if** $\exists 0 \leq i < k-1 : F_i := F_{i+1}$ **then SUCCEED**

// post: $\neg SAT(F_i \wedge s)$

**function** removeBad($i \in N$, state s)

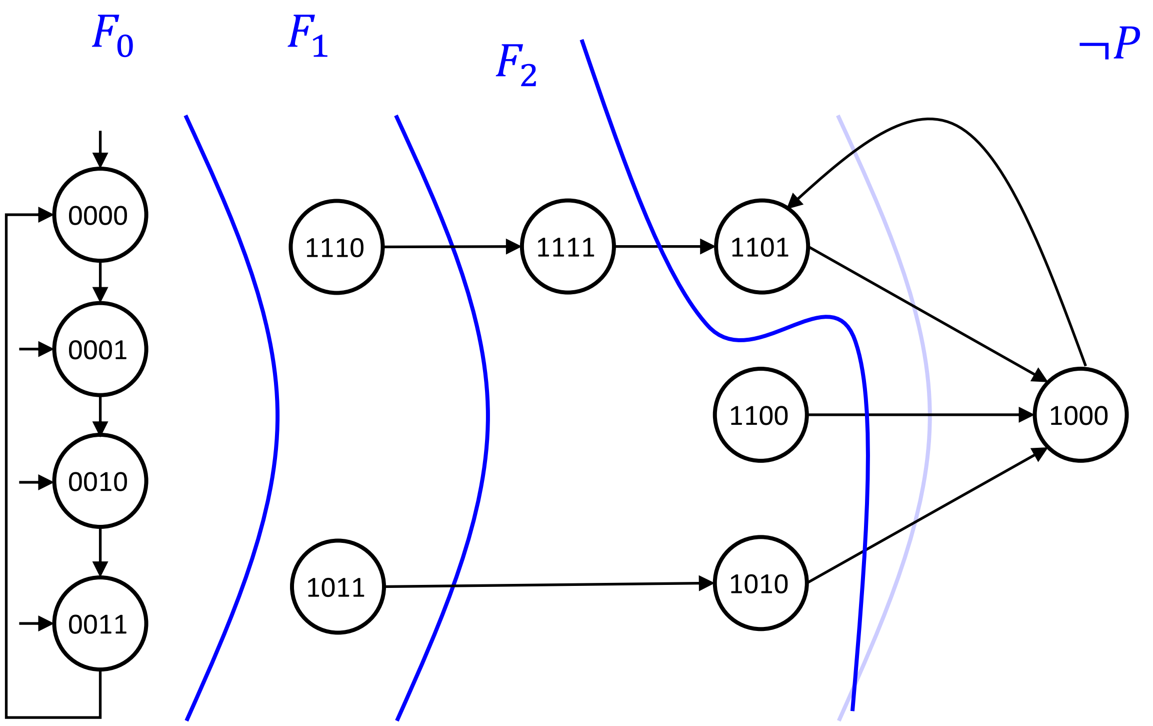
**if** SAT($S_0 \wedge s$) **then FAIL**

**while**($t$ := SAT($F_{i-1} \wedge R \wedge s'$((

removeBad($i-1, t$)

$\forall 0 < j \leq i : F_j \coloneqq F_j \wedge \neg$s

$k = 2$




Suppose s= 1100, this state is removed from $F_2$

Suppose $s$ = 1010

removeBad(2, 1010), leads to removal of 1011 from $F_1$

# PDR, First Version

**function** PDR(Model $M$)

  **if** SAT($S_0 \wedge \neg P$) **or** SAT($S_0 \wedge R \wedge \neg P'$) **then FAIL**

  $F_0$ := $S_0$; $F_1$ := $P$; $k$ :=1;

  **while**(true)

    **while**($s$ := SAT($F_k \wedge R \wedge \neg P'$))

      removeBad(k, $s$)

    $k$++; $F_k$ := $P$

    **if** $\exists 0 \leq i < k-1: F_i := F_{i+1}$ **then SUCCEED**

// post: $\neg SAT(F_i \wedge s)$

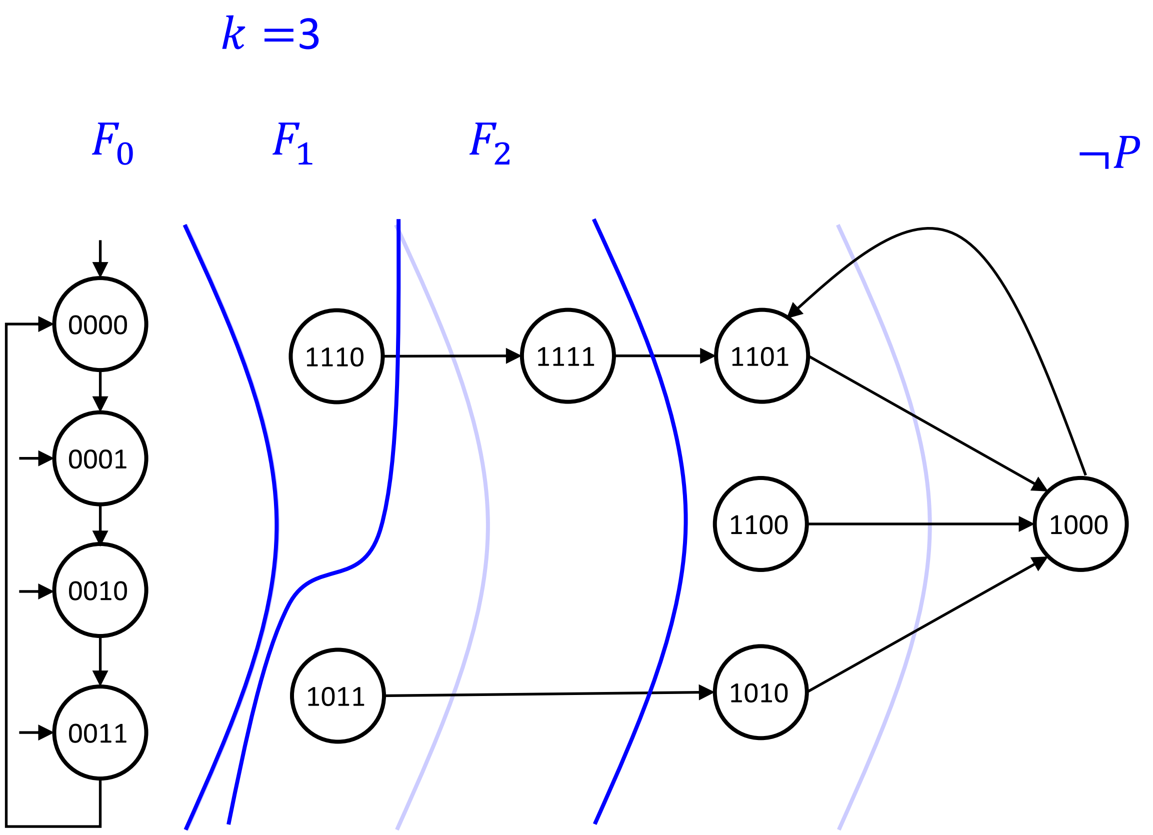
**function** removeBad($i \in N$, state s)

  **if** SAT($S_0 \wedge c$) **then FAIL**

  **while**($t$ := SAT($F_{i-1} \wedge R \wedge s'$))

    removeBad($i-1, t$)

  $\forall 0 < j \leq i: F_j := F_j \wedge \neg$s



# PDR, First Version

**function** PDR(Model $M$)

**if** SAT($S_0 \wedge \neg P$) **or** SAT($S_0 \wedge R \wedge \neg P'$) **then FAIL**

$F_0$ := $S_0$ ;$F_1$ := $P$; k := 1

**while**(true)

**while**($s$ := SAT($F_k \wedge\ R \wedge \neg P'$))

removeBad(k, $s$)

$k$++; $F_k$ =: $P$

**if** $\exists 0 \leq i < k-1: F_i := F_{i+1}$ **then SUCCEED**

// post: $\neg SAT(F_i \wedge s)$

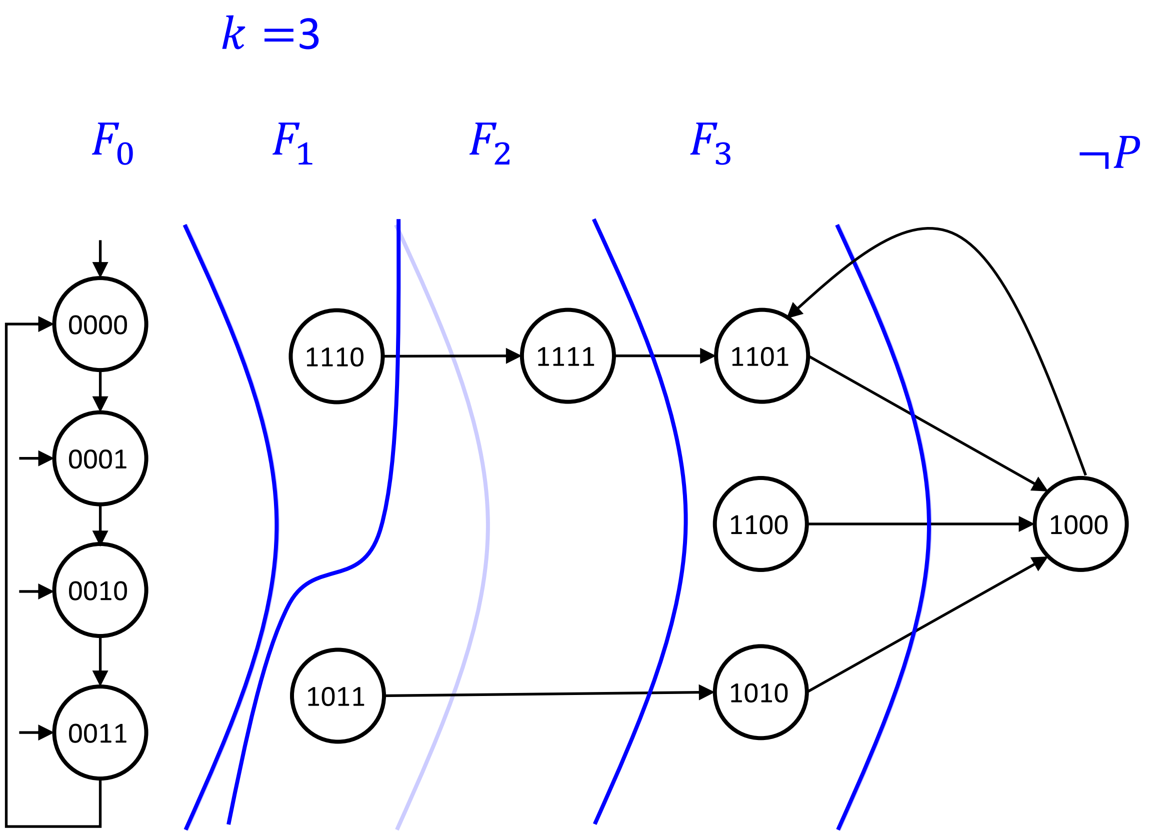
**function** removeBad($i \in N$, state s)

**if** SAT($S_0 \wedge s$) **then FAIL**

**while**($t$ := SAT($F_{i-1} \wedge R \wedge s'$))

removeBad($i-1, t$)

$\forall 0 < j \leq i: F_j \coloneqq F_j \wedge \neg$s



# PDR, First Version

**function** PDR(Model $M$)

**if** SAT($S_0 \land \neg P$) **or** SAT($S_0 \land R \land \neg P'$) **then FAIL**

$F_0$ := $S_0$ ;$F_1$ := $P$; k := 1

**while**(true)

**while**($s$ := SAT($F_k \land R \land \neg P'$))

removeBad(k, $s$)

$k$++; $F_k$ =: $P$

**if** $\exists 0 \leq i < k - 1 : F_i := F_{i+1}$ **then SUCCEED**

// post: $\neg SAT(F_i \land s)$

**function** removeBad($i \in N$, state s)

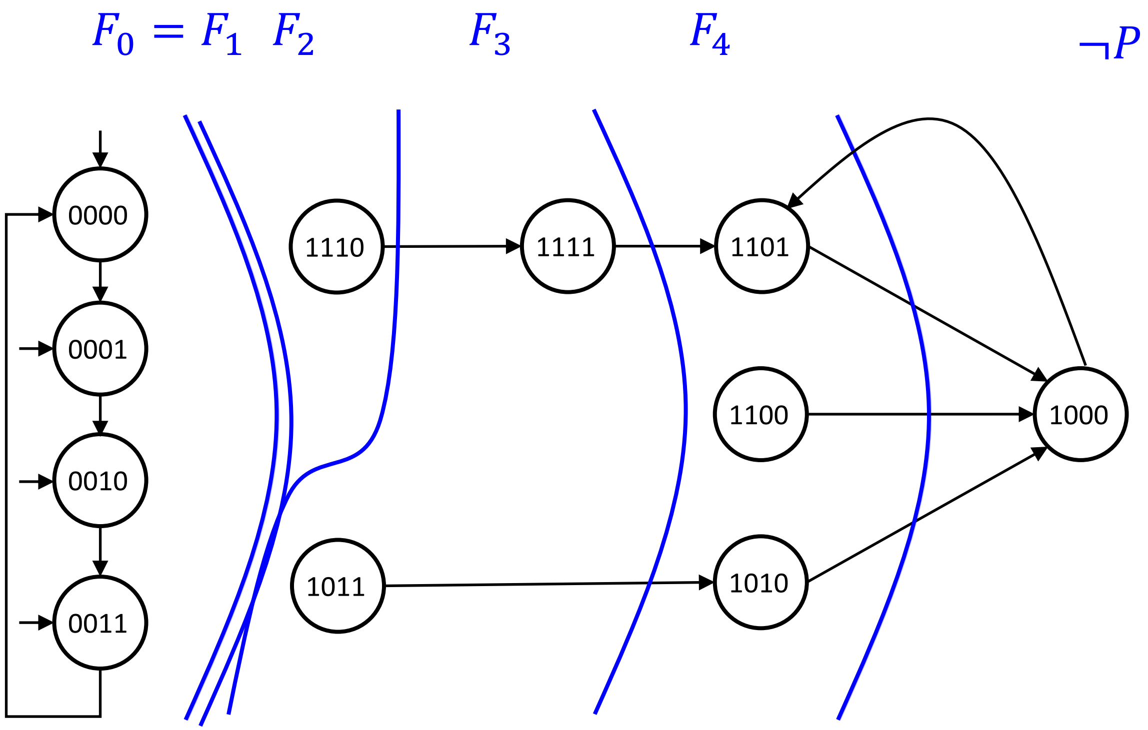
**if** SAT($S_0 \land s$) **then FAIL**

**while**($t$ := SAT($F_{i-1} \land R \land s'$))

removeBad($i - 1, t$)

$\forall 0 < j \leq i : F_j \coloneqq F_j \land \neg$s

$k$ =3

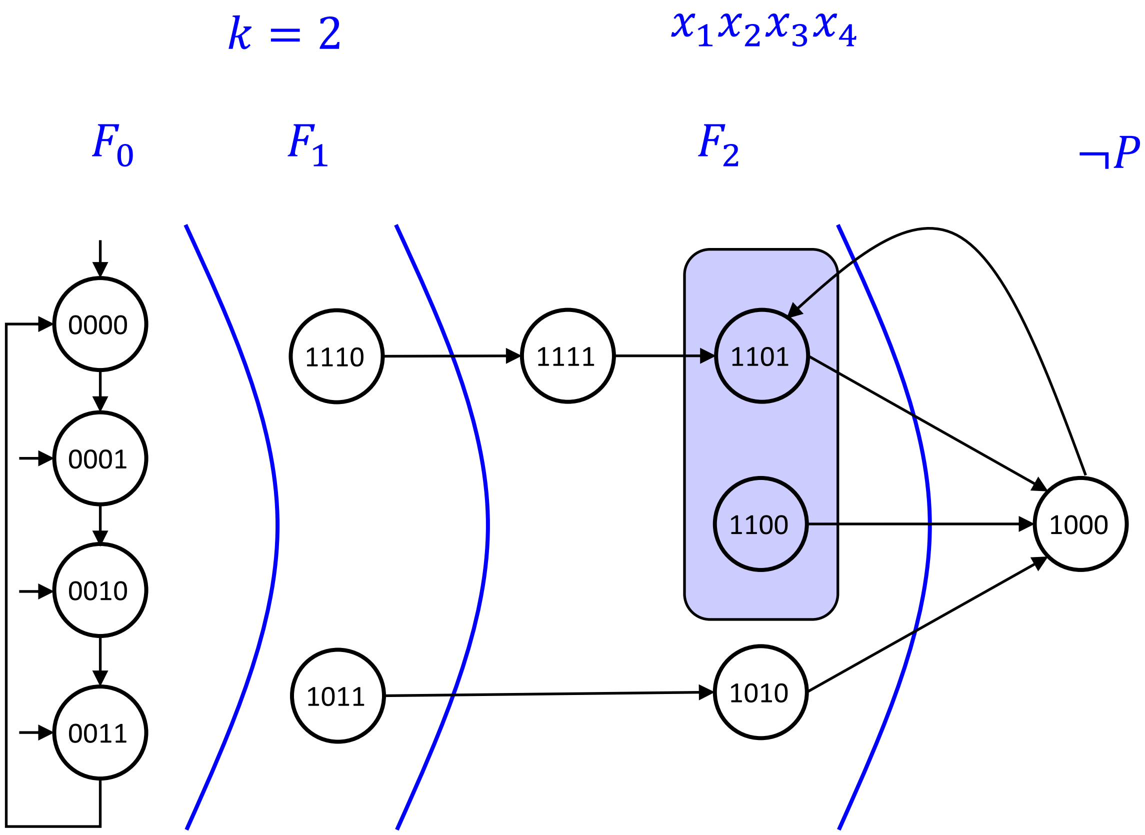


# Drawback

- First version considers every state individually
- Similar states behave similarly! Example: 1100 and 1101. Generalize

  1101 to 110- = $x_1 \land x_2 \land \neg x_3$!

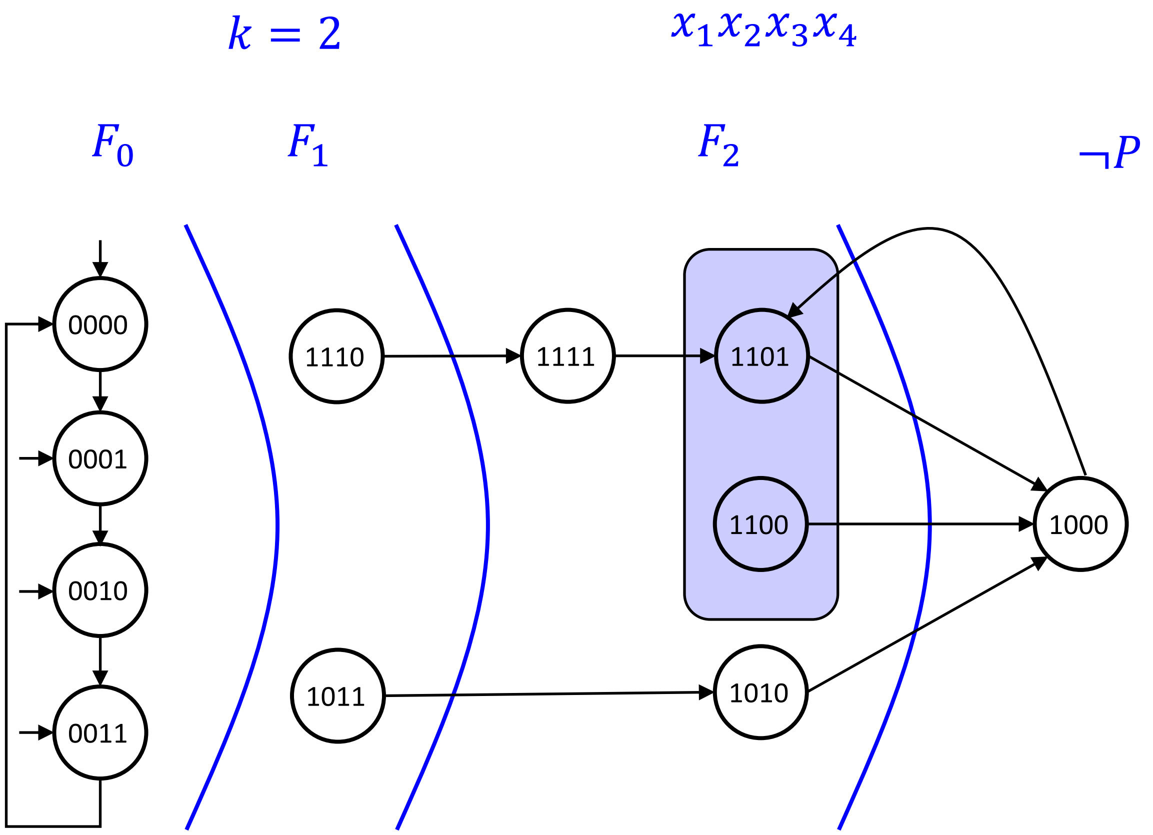
**Conditions**



# Drawback

- First version considers every state individually
- Similar states behave similarly! Example: 1100 and 1101. Generalize

  1101 to 110- = $x_1 \wedge x_2 \wedge \neg x_3$!




**Conditions**

- UNSAT$(F_1 \wedge R \wedge x'_1 \wedge x'_2 \wedge \neg x'_3)$
- UNSAT$(S_0 \wedge x_1 \wedge x_2 \wedge \neg x_3)$

# PDR: Naive Generalization

**function** PDR(Model $M$)

  **if** SAT($S_0 \wedge \neg P$)) **or** $SAT(S\_0 \wedge R \wedge \neg P')$ **then FAIL**

  $F_0$ =: $S_0$ ;$F_1$ =: $P$ ;$k$;1=:

  **while**(true)

    **while**($s$ := SAT($F_k \wedge\ R \wedge \neg P'$))

      removeBad(k, $s$)

    $k$++; $F_k$ =: $P$

    **if** $\exists 0 \leq i < k-1: F_i := F_{i+1}$ **then SUCCEED**

// post: $\neg SAT(F_i \wedge s)$

**function** removeBad($i \in N$, state s)

  **if** SAT($S_0 \wedge s$) **then FAIL**

  **while**($t$ := SAT($F_{i-1} \wedge R \wedge s'$))

    removeBad($i-1, t$)

  $g$ := generalizeNaive($i, s$)

  $\forall 0 < j \leq i: F_j \coloneqq F_j \wedge \neg g$

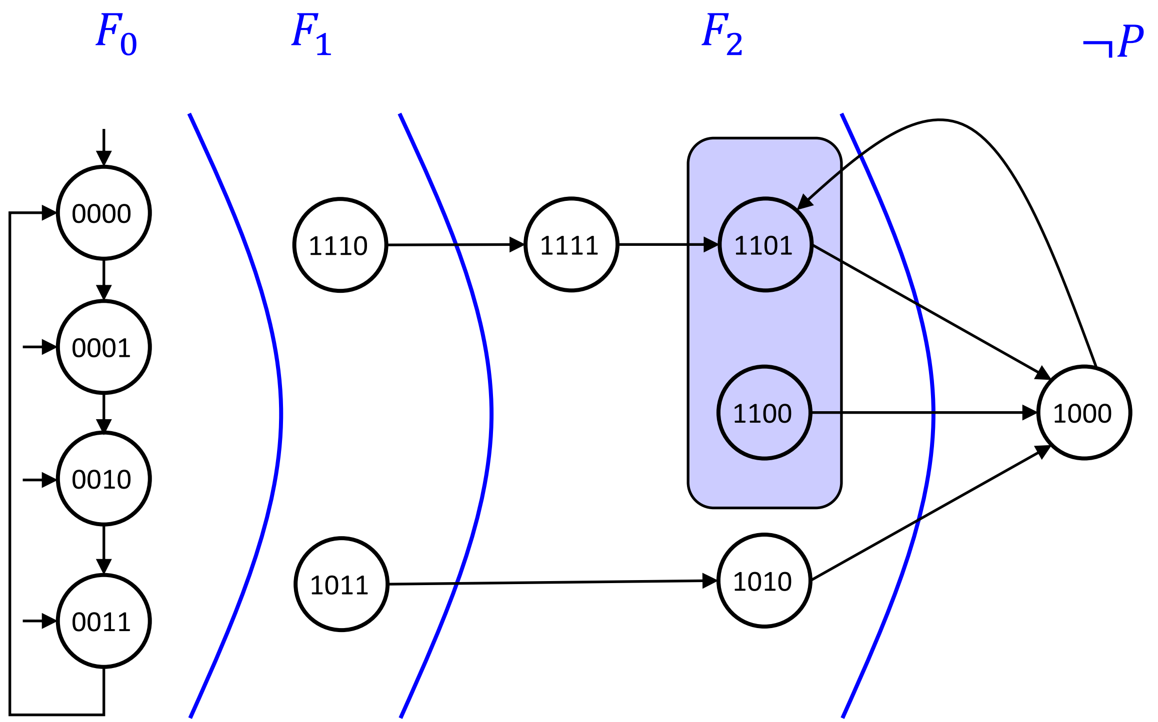
**function** generalizeNaive(i, state $s$)

  return a shortest cube $c$ such that

    - $c \leftarrow s$

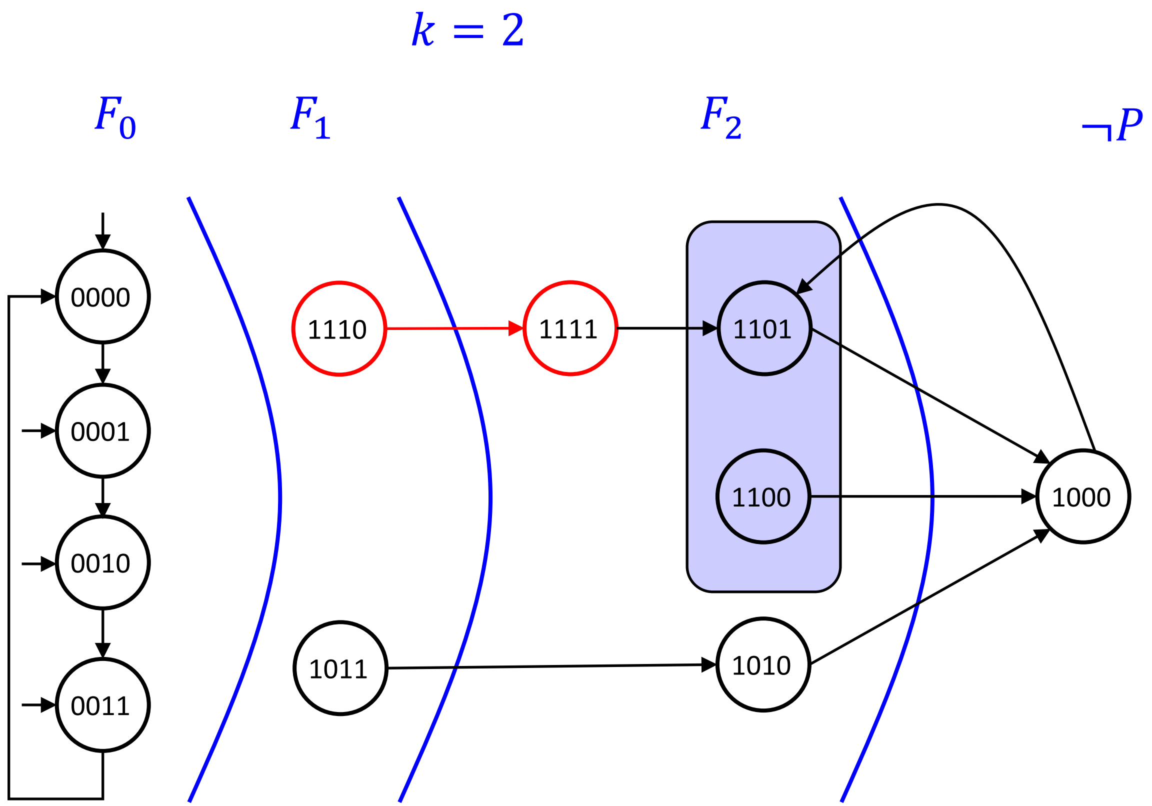
    - $\neg\mathrm{SAT}(\mathrm{F}_{\mathrm{i}-1} \wedge R \wedge c')$

    - $\neg\mathrm{SAT}(S_0 \wedge c)$



# Generalize Further?

- We can generalize to 110-
- Can we generalize to 11--?
  - NO: for $c = x_1 \land x_2$, we have $\text{SAT}(F_1 \land R \land c')$
- Transition 1110 $\rightarrow$ 1111 is the problem



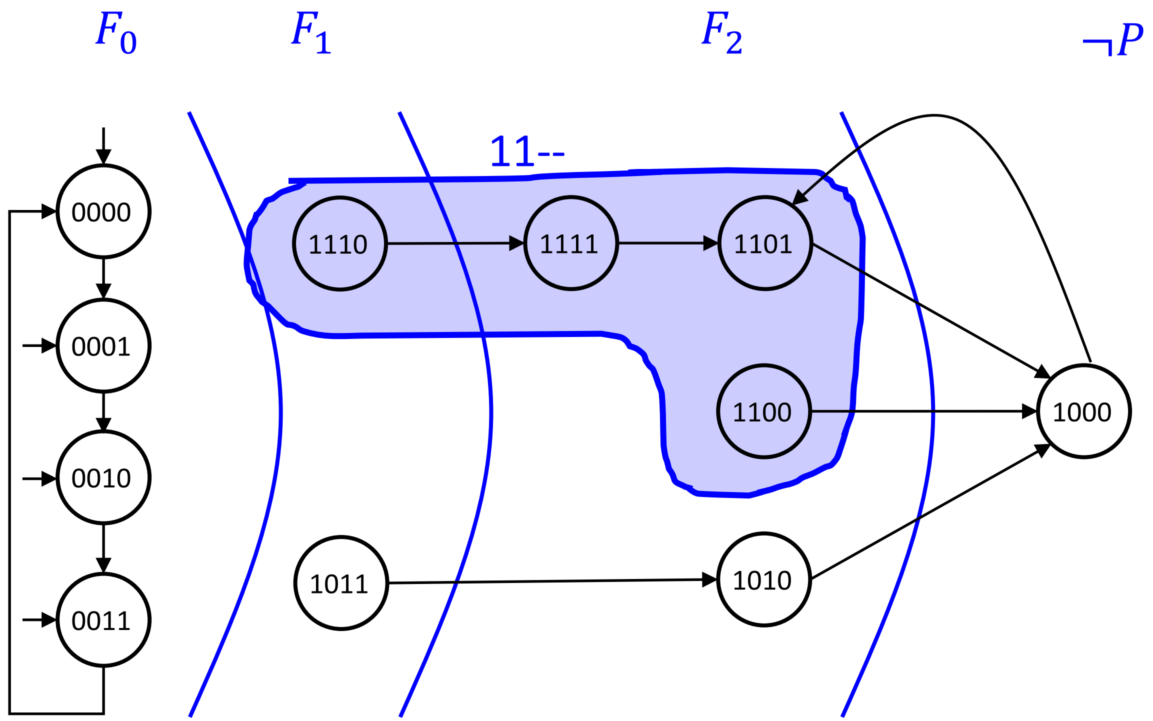
# Relative Inductiveness

shortest cube $c$ such that

- $c \leftarrow s$
- $\neg\text{SAT}(\neg c \wedge \text{F}_1 \wedge R \wedge c')$
- $\neg\text{SAT}(S_0 \wedge c)$

$\neg c$ is relative inductive wrt $F_1$

Why?




28

# Relative Inductiveness

shortest cube $c$ such that

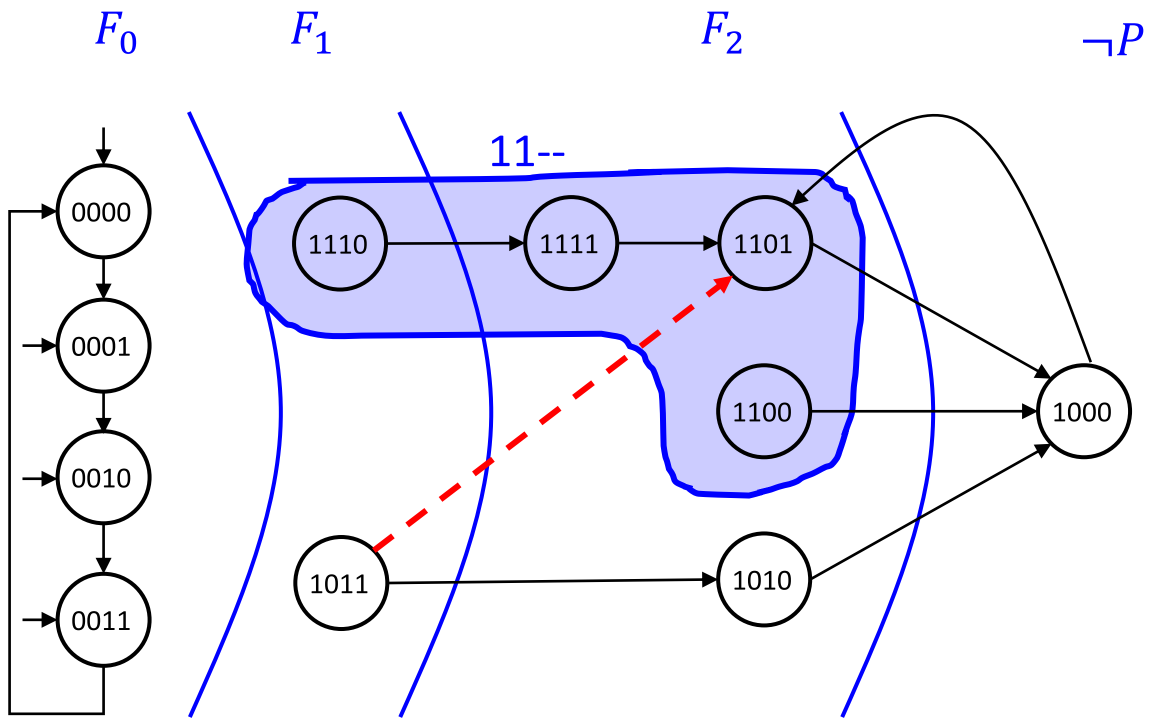
- $c \leftarrow s$
- $\neg\mathrm{SAT}(\neg c \wedge \mathrm{F}_1 \wedge R \wedge c')$
- $\neg\mathrm{SAT}(S_0 \wedge c)$

$\neg c$ is relative inductive wrt $F_1$

Why?

Because we need to maintain

I4: $F_i \wedge R \rightarrow F'_{i+1}$



# PDR: Relative Inductiveness

**function** PDR(Model $M$)

**if** SAT($S_0 \wedge \neg P$) **or** $SAT(S_0 \wedge R \wedge \neg P')$ **then FAIL**

$F_0$ =: $S_0$ ;$F_1$ =: $P$ ;$k$;1=:

**while**(true)

**while**($s$ := SAT($F_k \wedge\ R \wedge \neg P'$))

removeBad(k, $s$)

$k$++; $F_k$ =: $P$

**if** $\exists 0 \leq i < k - 1 : F_i = F_{i+1}$ **then SUCCEED**

// post: $\neg SAT(F_i \wedge s)$

**function** removeBad($i \in N$, state s)

**if** SAT($S_0 \wedge s$) **then FAIL**

**while**($t$ := SAT($F_{i-1} \wedge R \wedge s'$))

removeBad($i - 1, t$)

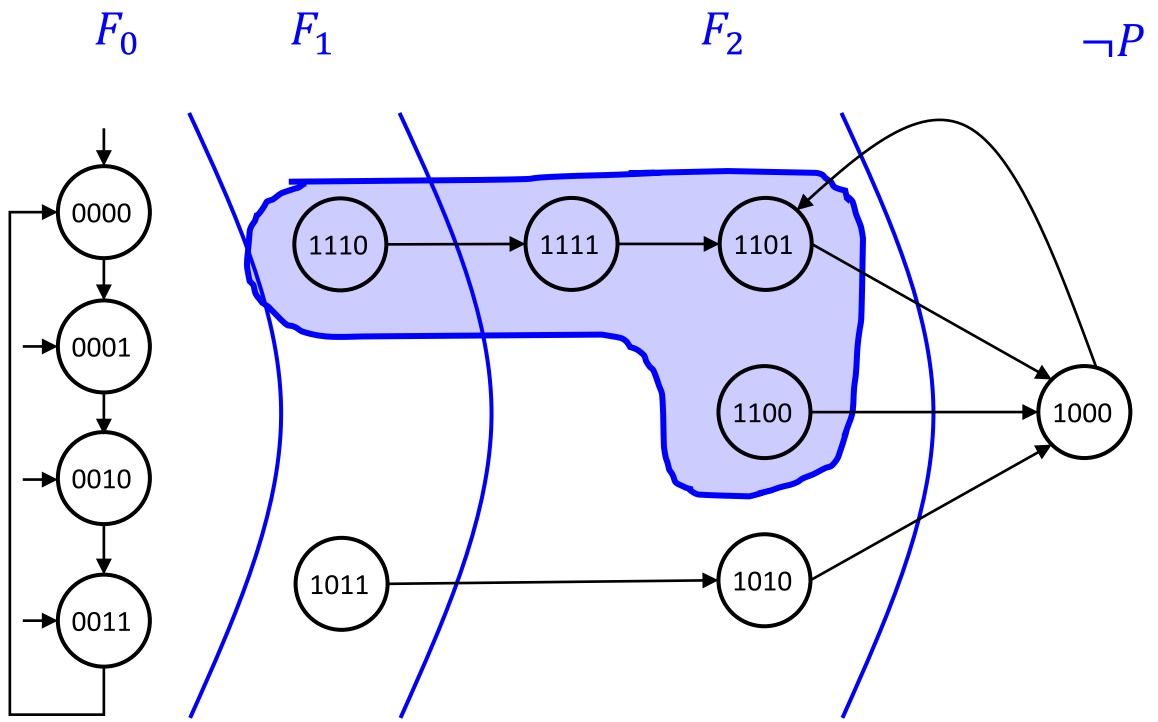
$g$ := generalize($i$,s)

$\forall 0 < j \leq i : F_j \coloneqq F_j \wedge \neg g$

**function** generalize(i, state $s$)

return a shortest cube $c$ such that

- $c \leftarrow s$
- $\neg\mathrm{SAT}(\neg c \wedge \mathrm{F}_{i-1} \wedge R \wedge c')$
- $\neg\mathrm{SAT}(S_0 \wedge c)$



# Generalization

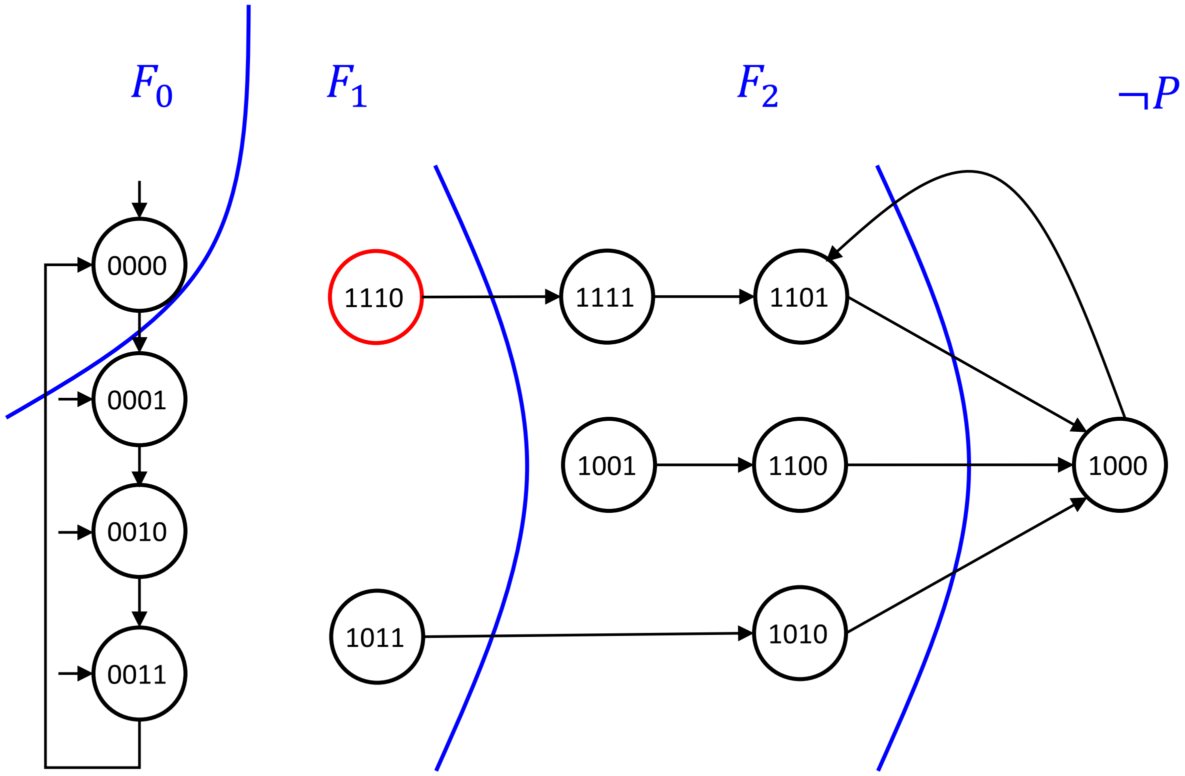
**function** generalize(i, state $s$)
  $c \coloneqq s$
  **while** $c$ changes
    let $l_1, \dots, l_n$ be the literals of $c$
    **for** $i \coloneqq 1$ **to** $n$
      $c' = c$ with $l_i$ removed
      **if** relInd(c’) **then** $c = c'$
  **return** c

**function** relInd(cube $c$)
  **return** $\neg\mathrm{SAT}(\neg c \wedge \mathrm{F_{i-1}} \wedge R \wedge c)$
$\wedge \neg\mathrm{SAT}(S_0 \wedge c)$

# Propagate Clauses

Suppose you are removing 1110.
You can generalize to 1---

$$F_1 \coloneqq F_1 \land \neg x_1$$




32

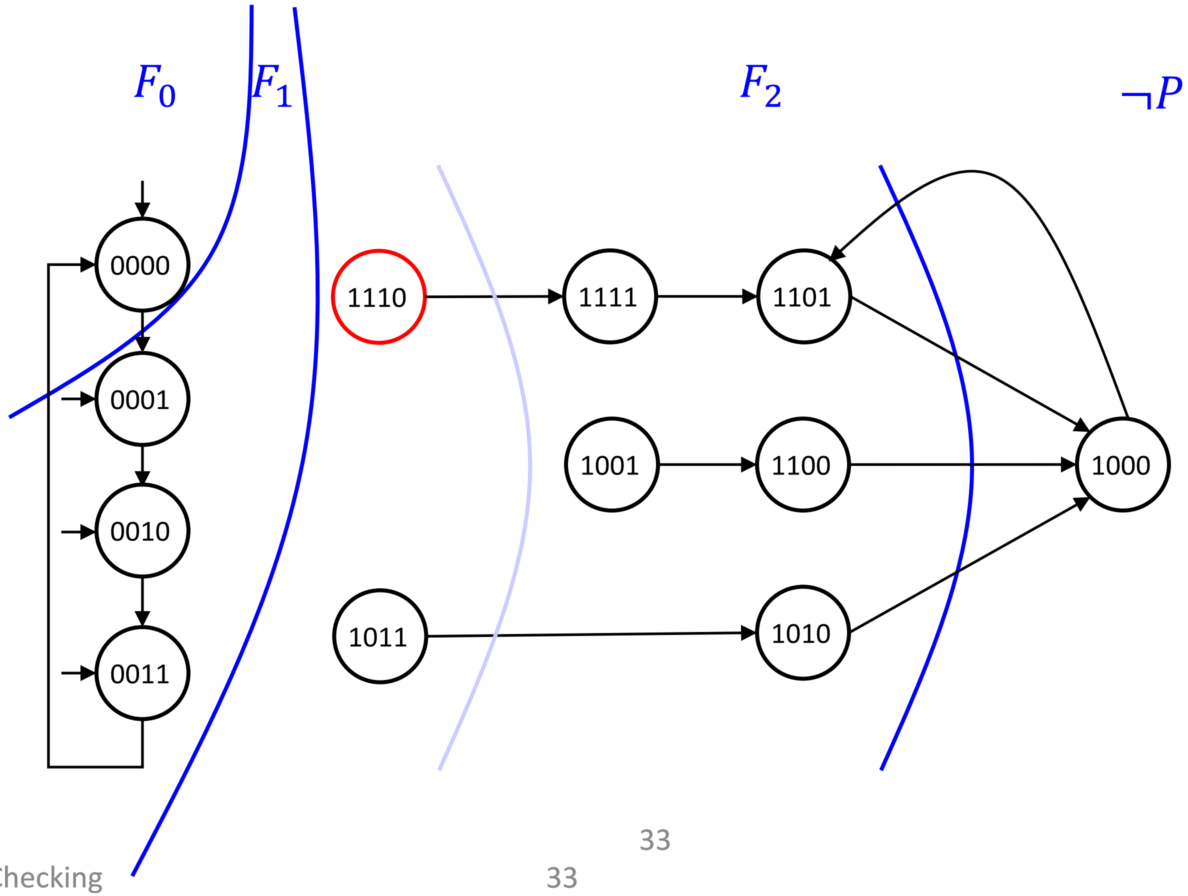
# Propagate Clauses

Suppose you are removing 1110.
You can generalize to 1---

$$F_1 := F_1 \wedge \neg x_1$$

$F_2 \wedge x_1 \notin postimg(F_1)$, so we can add $\neg x_1$ to $F_2$

UNSAT$(F_1 \wedge R \wedge F_2' \wedge x'_1)$



# PDR, Final: Propagate Clauses

**function** PDR(Model $M$)

  **if** SAT($S_0 \wedge \neg P$) **or** $SAT(S_0 \wedge R \wedge \neg P')$ **then FAIL**

  $F_0$ := $S_0$; $F_1$ := $P$; $k$ :=1;

  **while**(true)

    **while**($s$ := SAT($F_k \wedge\ R \wedge \neg P'$))

      removeBad(k, $s$)

    $k$++; $F_k$ := $P$

    propagateClauses(k)

    **if** $\exists 0 \leq i < k - 1 : F_i = F_{i+1}$ **then SUCCEED**

// post: $\neg SAT(F_i \wedge s)$

**function** removeBad($i \in N$, state s)

  **if** SAT($S_0 \wedge s$) **then FAIL**

  **while**($t$ := SAT($F_{i-1} \wedge R \wedge s'$))

    removeBad($i - 1, t$)

  $g$ := generalize($i$,s)

  $\forall 0 < j \leq i : F_j \coloneqq F_j \wedge \neg g$

**function** gesneralize(i, state $s$)

  return a shortest cube $c$ such that

    - $c \leftarrow s$

    - $\neg c$ inductive relative to $F_{i-1}$

**function** propagateClauses(k)

  **for** i := 1 **to** $k$ – 1

    **for** every clause $c \in F_i$

      **if** $\neg \text{SAT}(F_i \wedge R \wedge \neg c')$

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

# Further Ideas

- This version is simplified, doesn't find long counterexamples quickly
- Equivalence of frames = syntactic check
  - Use implication and subsumption to simplify clauses

# Performance

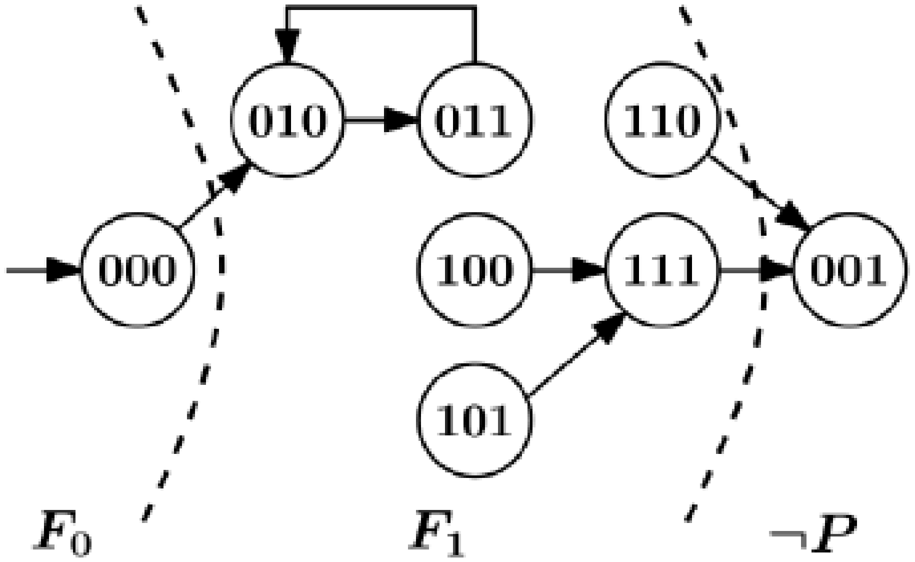
Hardware Model Checking Competition 2020

1. AVR 11 variants of IC3+[abstraction], 2x BMC, 3x k-induction
2. AVY interpolation + PDR
3. nuXmv “portfolio”, including IC3
4. Pono: protfolio, including BMC, k-induction, interpolation, IC3

# Literature

## Literature


- A. R. Bradley, SAT-Bassed Model Checking without Unrolling, VMCAI 2011. http://ecee.colorado.edu/~bradleya/ic3/ic3_bradley.pdf
- N. Een, A. Mishchenko, R. Brayton, Efficient Implementation of Property Directed Reachability, FMCAD 2011. https://people.eecs.berkeley.edu/~alanmi/publications/2011/fmcad11_pdr.pdf
- F. Somenzi, Aaron R. Bradley: IC3: where monolithic and incremental meet. FMCAD 2011: 3-8. http://theory.stanford.edu/~arbrad/papers/ic3_tut.pdf
- A. R. Bradley: Understanding IC3. SAT 2012: 1-14. https://theory.stanford.edu/~arbrad/papers/Understanding_IC3.pdf




**Task 1. [ 40 points ]** Use the "first version" of PDR to prove that $P$ is always true, but stop after two iterations, when you have frames $F_0, \cdot, F_3$. Clearly indicate the steps and the frames at the end of each iteration.
Is the property verified at the end? Why (not)?

**Task 2. [ 40 points ]** Perform the same task using "naive generalization" during the removal of bad states, as shown in class. Again, stop after two iterations and indicate the steps and the frames at the end of each iteration.
Is the property verified at the end? Why (not)?

**Task 2. [ 20 points ]** Are the following statements true? Justify your answer.

2.1 The set $\neg x_1$ is inductive. **[ 10 points ]**

2.2 The set $\neg x_3$ is inductive. **[ 10 points ]**

2.3 The set $\neg x_2$ is inductive relative to $\neg x_1$. **[ 10 points ]**

2.4 The set $\neg x_3$ is inductive relative to $\neg x_1$. **[ 10 points ]**

Note: Tasks 2.x are 5 points each, See PDF