

Property-Directed Reachability *or IC3*



Aaron Bradley

- Next lecture: 27 April 2023 in

i 1 1

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 $\mathbf{X} \wedge \mathbf{R} \wedge \mathbf{Y}'$

Meaning: there is an edge from $s \in X$ to $s' \in Y$

$\mathbf{s} := \mathbf{SAT}(F_i \wedge \mathbf{R} \wedge \neg \mathbf{P}')$

- $s :=$ some state in F_i with an edge to a state in $\neg P$ **if** $\mathbf{SAT}(F_i \wedge \mathbf{R} \wedge \neg \mathbf{P}')$
- $s := \mathbf{FALSE}$ **if** $\neg \mathbf{SAT}(F_i \wedge \mathbf{R} \wedge \neg \mathbf{P}')$

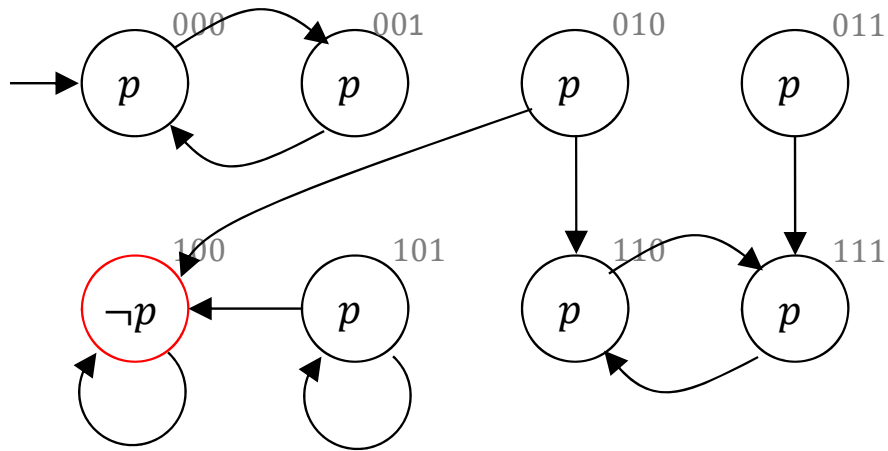
PDR: Notation

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

$x_1x_2x_3$

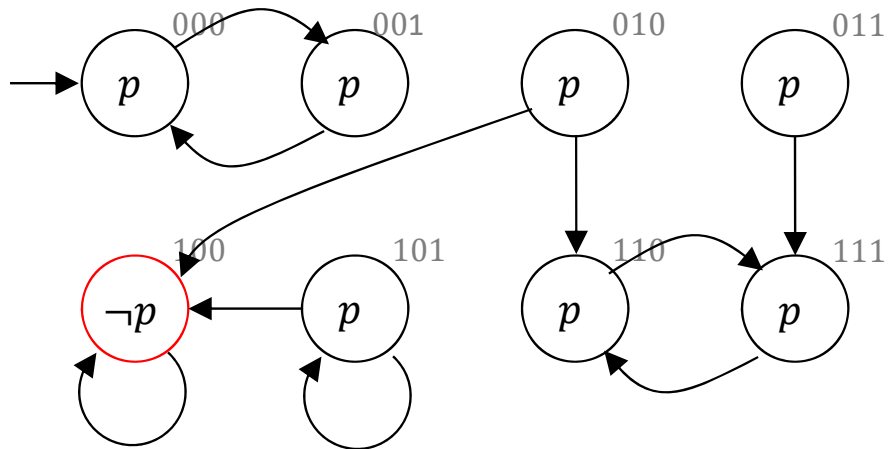


Inductive facts

- Is $\neg x_1$ inductive?

Relative Inductiveness

$x_1x_2x_3$

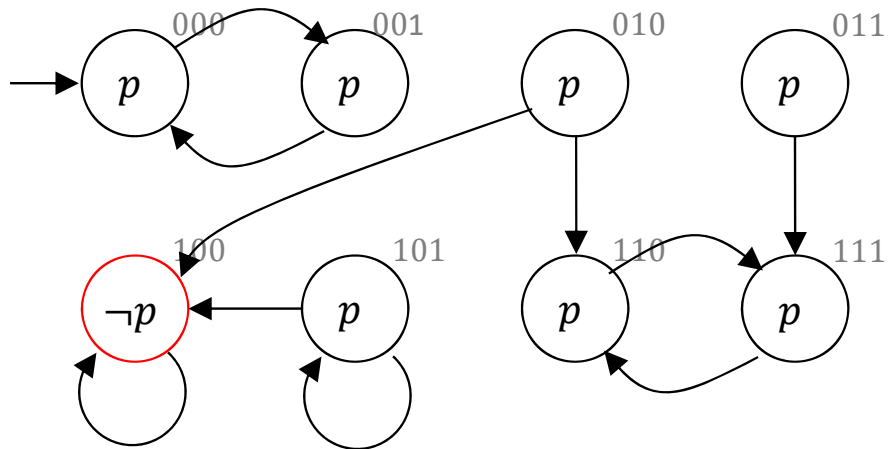


Inductive facts

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

Relative Inductiveness

$x_1x_2x_3$



Inductive facts

- Is $\neg x_1$ inductive?

- $S_0 \rightarrow \neg x_1$

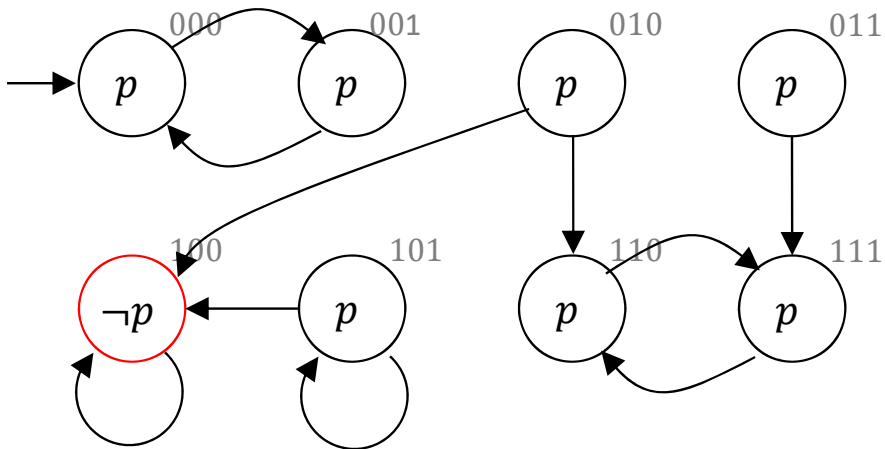
- $\neg x_1 \wedge R \rightarrow \neg x'_1$ is **false**

- **No!**

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

Relative Inductiveness

$x_1x_2x_3$



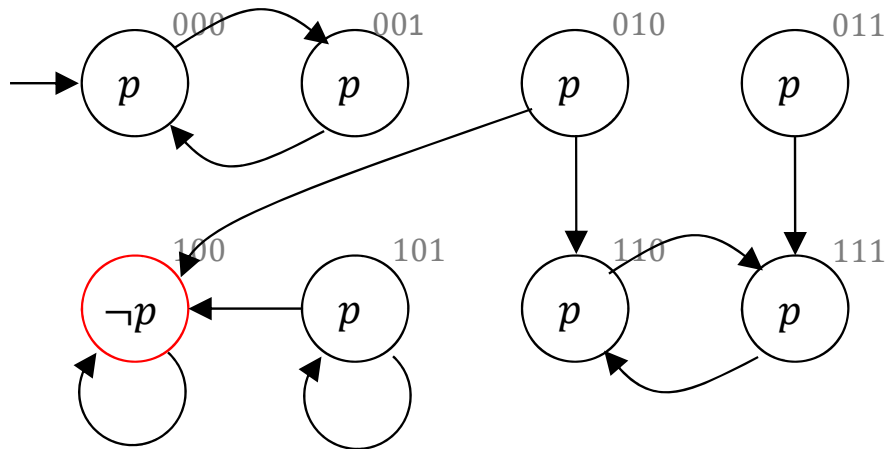
Inductive facts

- 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 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.

Relative Inductiveness

$x_1x_2x_3$



Inductive facts

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

PDR: Data Structures & Invariants

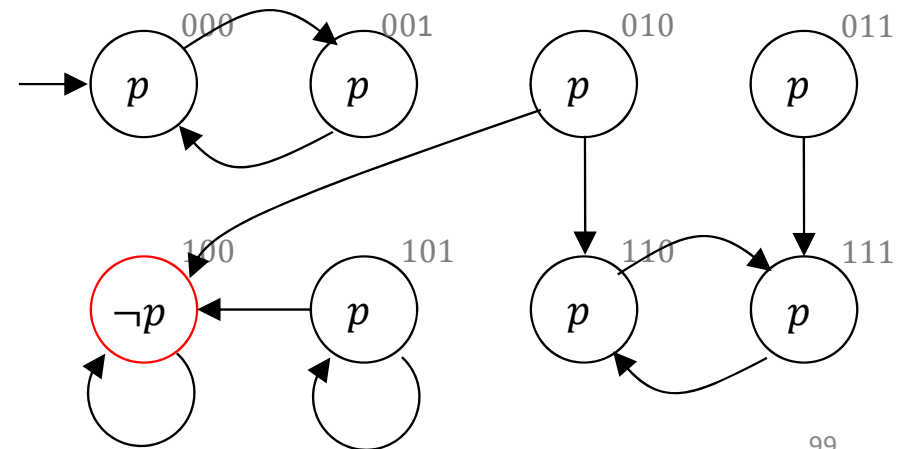
Data Structure

Clause: Disjunction of literals. **Cube:** Conjunction of literals.

Clause and cubes signify a set of states. Longer clauses – more states. Longer cubes – fewer states

Formulas F_0, \dots, F_k over V , stored as sets of clauses (Sets $F_0, \dots, F_k \subseteq S$)

Meaning: CNF

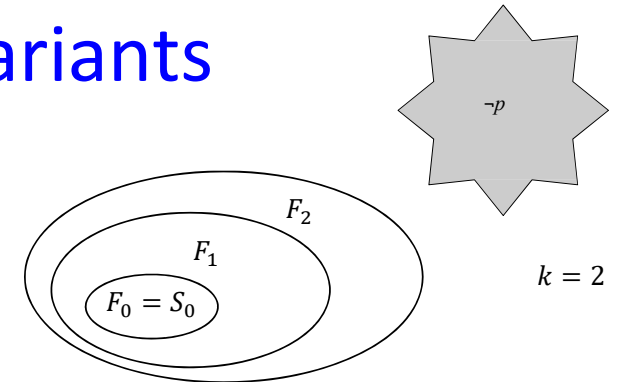


PDR: Data Structures & Invariants

We have **frames** F_0, F_1, \dots, F_k

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



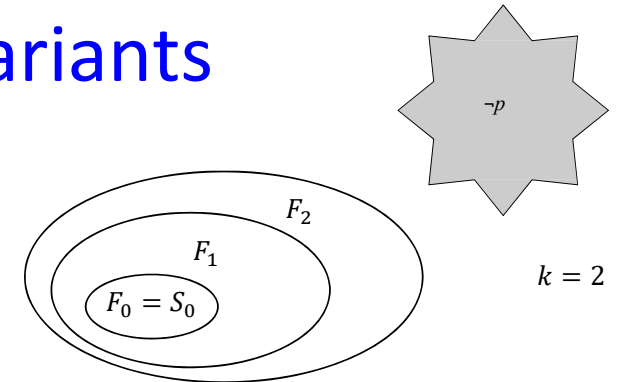
PDR: Data Structures & Invariants

Invariants

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

Facts

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. Because of I4:
 1. $postimg^i(S_0) \subseteq F_i$
 2. There is no trace from S_0 to F_i of length $< i$
 3. There is no counterexample $k + 1$ edges or less (with I3)
- If $F_i = F_{i+1}$ then system is correct. (By I3, I4, F_i is an inductive invariant)



PDR, First Version

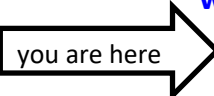
```

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( $s := \text{SAT}(F_k \wedge R \wedge \neg P')$ )
      removeBad( $k, s$ )
       $k++ ; F_k =: P$ 

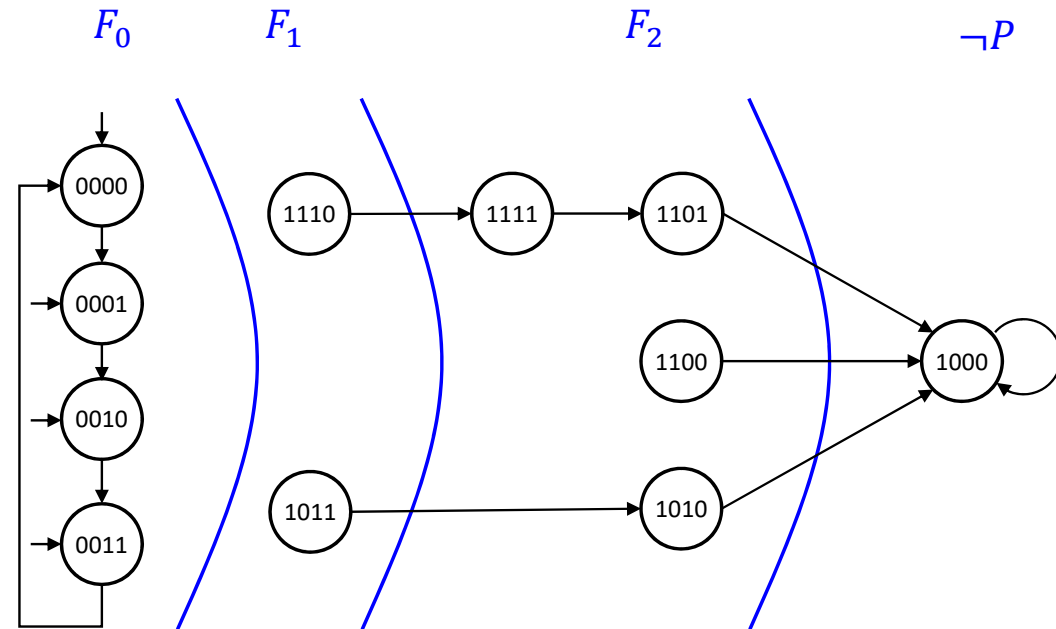
    if  $\exists 0 \leq i < k - 1 : F_i =: F_{i+1}$  then SUCCEED

  // post:  $\neg \text{SAT}(F_i \wedge s)$ 
  function removeBad( $i \in N, \text{state } s$ )
    if SAT( $S_0 \wedge s$ ) then FAIL
    while( $t := \text{SAT}(F_{i-1} \wedge R \wedge s')$ )
      removeBad( $i - 1, t$ )

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



$k = 2$



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)

while($s := \text{SAT}(F_k \wedge R \wedge \neg P')$)

 removeBad(k, s)

$k++; F_k := P$

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

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

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

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

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

 removeBad($i - 1, t$)

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

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

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

PDR, First Version

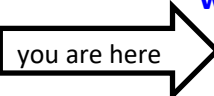
```

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)
    while( $s := \text{SAT}(F_k \wedge R \wedge \neg P')$ )
      removeBad( $k, s$ )
       $k++; F_k := P$ 

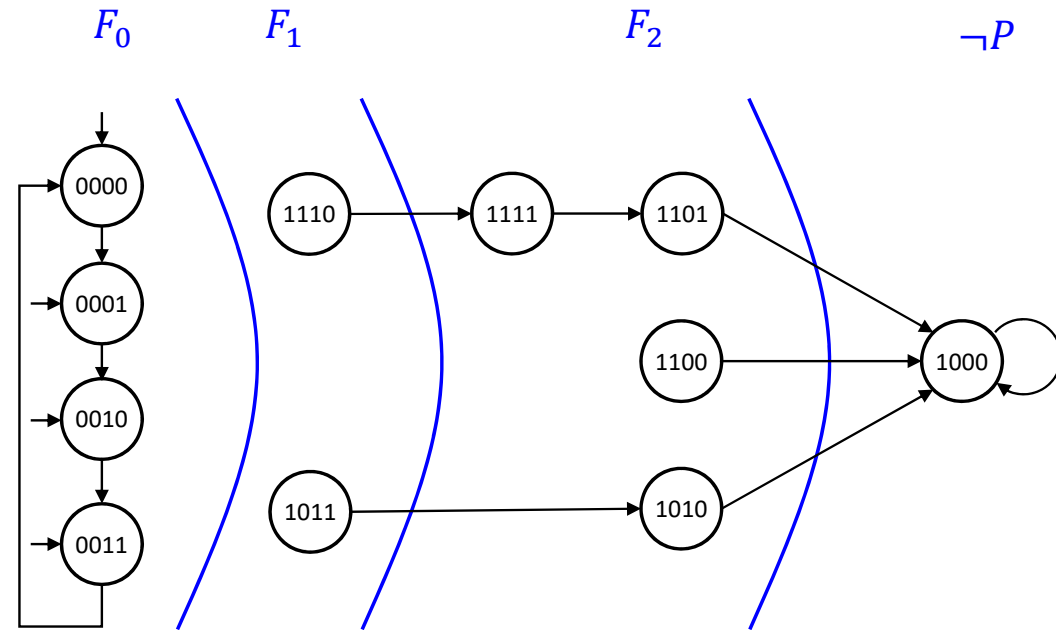
    if  $\exists 0 \leq i < k - 1: F_i := F_{i+1}$  then SUCCEED

  // post:  $\neg \text{SAT}(F_i \wedge s)$ 
  function removeBad( $i \in N, \text{state } s$ )
    if SAT( $S_0 \wedge c$ ) then FAIL
    while( $t := \text{SAT}(F_{i-1} \wedge R \wedge s')$ )
      removeBad( $i - 1, t$ )

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



$k = 2$



Do the invariants hold?

PDR, First Version

```

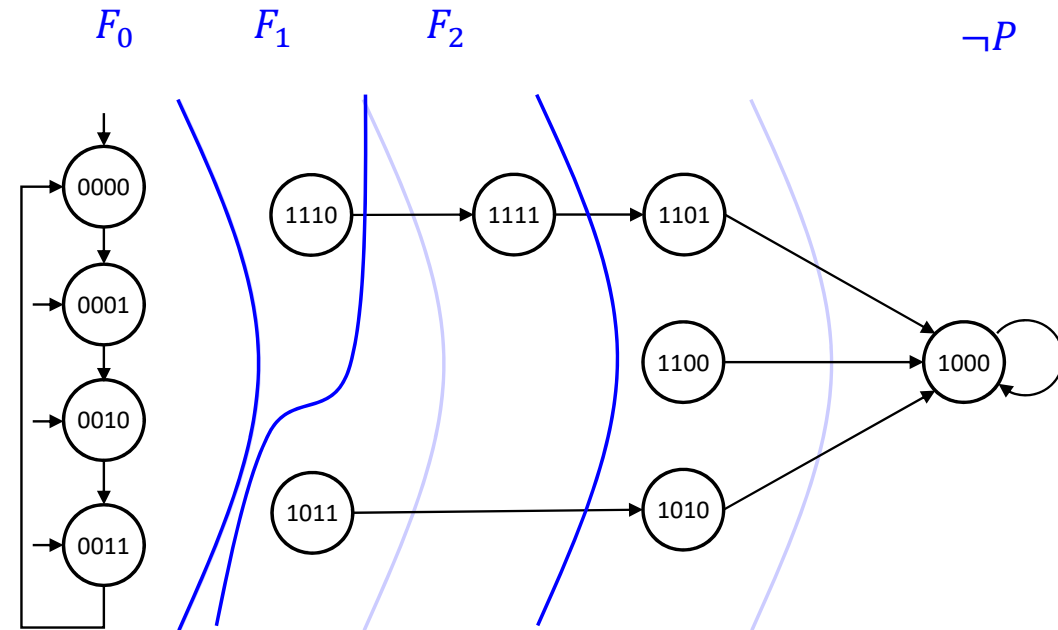
function PDR(Model  $M$ )
  if SAT( $S_0 \wedge \neg P$ ) or SAT( $S_0 \wedge R \wedge \neg P'$ ) then FAIL
   $F_0 := S_0; F_1 := P; k := 1;$ 
  while(true)
    while( $s := \text{SAT}(F_k \wedge R \wedge \neg P')$ )
      removeBad( $k, s$ )
       $k++; F_k := P$ 

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

  // post:  $\neg \text{SAT}(F_i \wedge s)$ 
  function removeBad( $i \in N, \text{state } s$ )
    if SAT( $S_0 \wedge c$ ) then FAIL
    while( $t := \text{SAT}(F_{i-1} \wedge R \wedge s')$ )
      removeBad( $i - 1, t$ )

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

$k = 3$



PDR, First Version

```

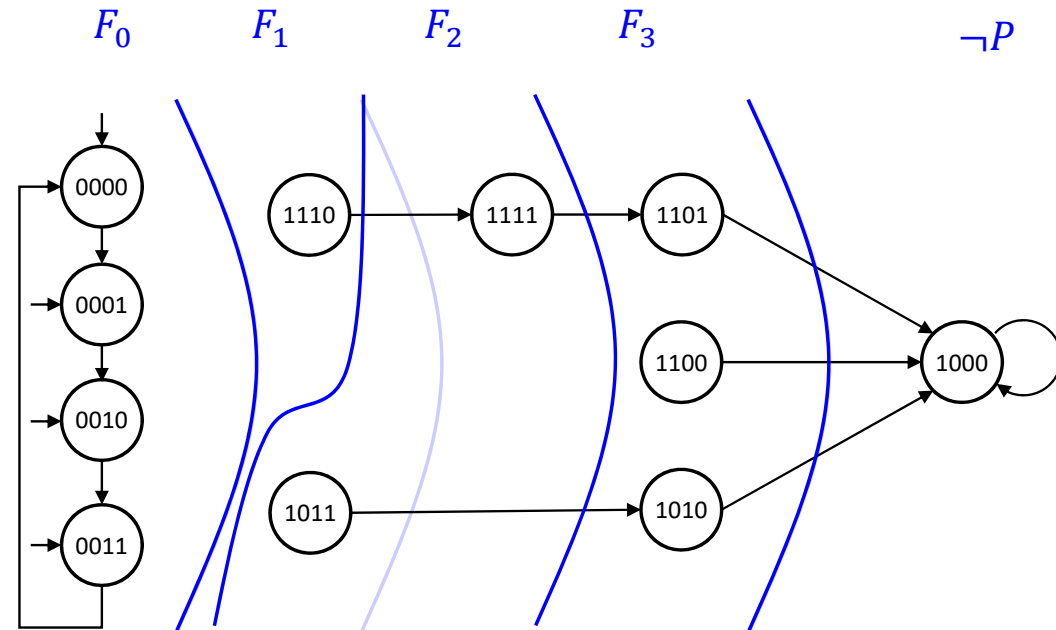
function PDR(Model  $M$ )
  if SAT( $S_0 \wedge \neg P$ ) or SAT( $S_0 \wedge R \wedge \neg P'$ ) then FAIL
   $F_0 := S_0; F_1 := P; k := 1;$ 
  while(true)
    while( $s := \text{SAT}(F_k \wedge R \wedge \neg P')$ )
      removeBad( $k, s$ )
     $k++; F_k := P$ 
    if  $\exists 0 \leq i < k - 1: F_i := F_{i+1}$  then SUCCEED

  // post:  $\neg \text{SAT}(F_i \wedge s)$ 
  function removeBad( $i \in N, \text{state } s$ )
    if SAT( $S_0 \wedge c$ ) then FAIL
    while( $t := \text{SAT}(F_{i-1} \wedge R \wedge s')$ )
      removeBad( $i - 1, t$ )

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

you are here

$k = 3$



PDR, First Version

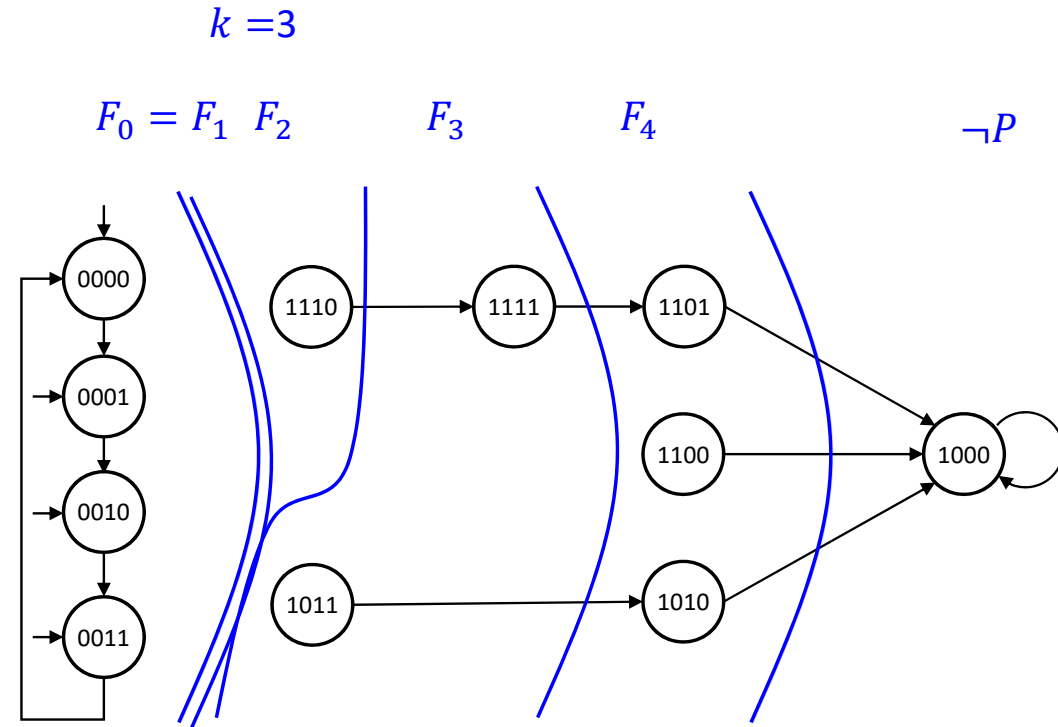
```

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)
    while( $s := \text{SAT}(F_k \wedge R \wedge \neg P')$ )
      removeBad( $k, s$ )
       $k++; F_k := P$ 

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

  // post:  $\neg \text{SAT}(F_i \wedge s)$ 
  function removeBad( $i \in N, \text{state } s$ )
    if SAT( $S_0 \wedge c$ ) then FAIL
    while( $t := \text{SAT}(F_{i-1} \wedge R \wedge s')$ )
      removeBad( $i - 1, t$ )

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

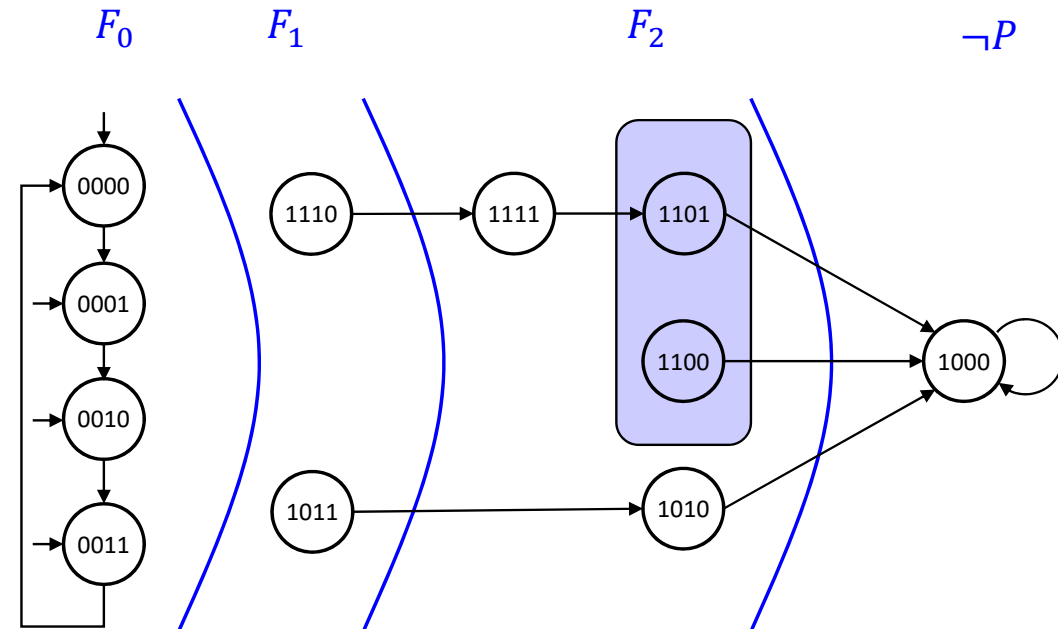


Drawback

$k = 2$

$x_1x_2x_3x_4$

- Version 1 considers each state individually
- Similar states behave similarly
- Example:
 - On call `removeBad(2, 1101)`
 - 1100 and 1101 cannot be reached from F_1 . Generalize 1101 to 110- = $x_1 \wedge x_2 \wedge \neg x_3$



Conditions

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

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( $s := \text{SAT}(F_k \wedge R \wedge \neg P')$ )
      removeBad( $k, s$ )
     $k++; F_k := P$ 

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

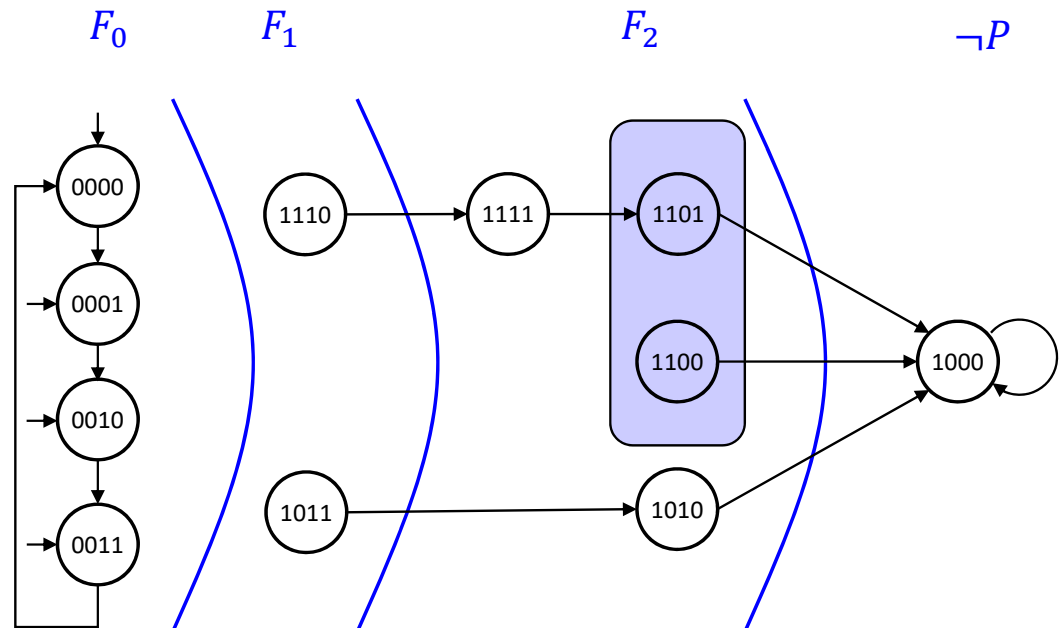
// post: $\neg \text{SAT}(F_i \wedge s)$

```

function removeBad( $i \in N, \text{state } s$ )
  if SAT( $S_0 \wedge c$ ) then FAIL
  while( $t := \text{SAT}(F_{i-1} \wedge R \wedge s')$ )
    removeBad( $i - 1, t$ )
   $g := \text{generalizeNaive}(i, s)$ 
   $\forall 0 < j \leq i: F_j := F_j \wedge \neg g$ 
  
```

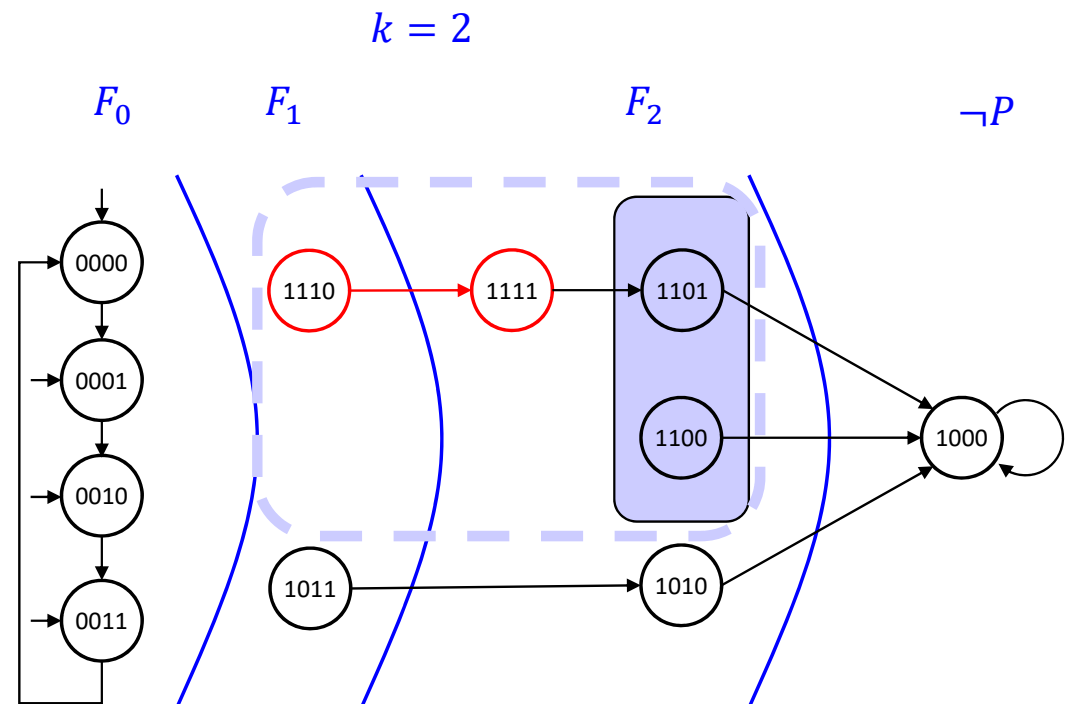
```

function generalizeNaive( $i, \text{state } s$ )
  return a shortest cube  $c$  such that
    -  $s \rightarrow c$ 
    -  $\neg \text{SAT}(F_{i-1} \wedge R \wedge c')$ 
    -  $\neg \text{SAT}(S_0 \wedge c)$ 
  
```



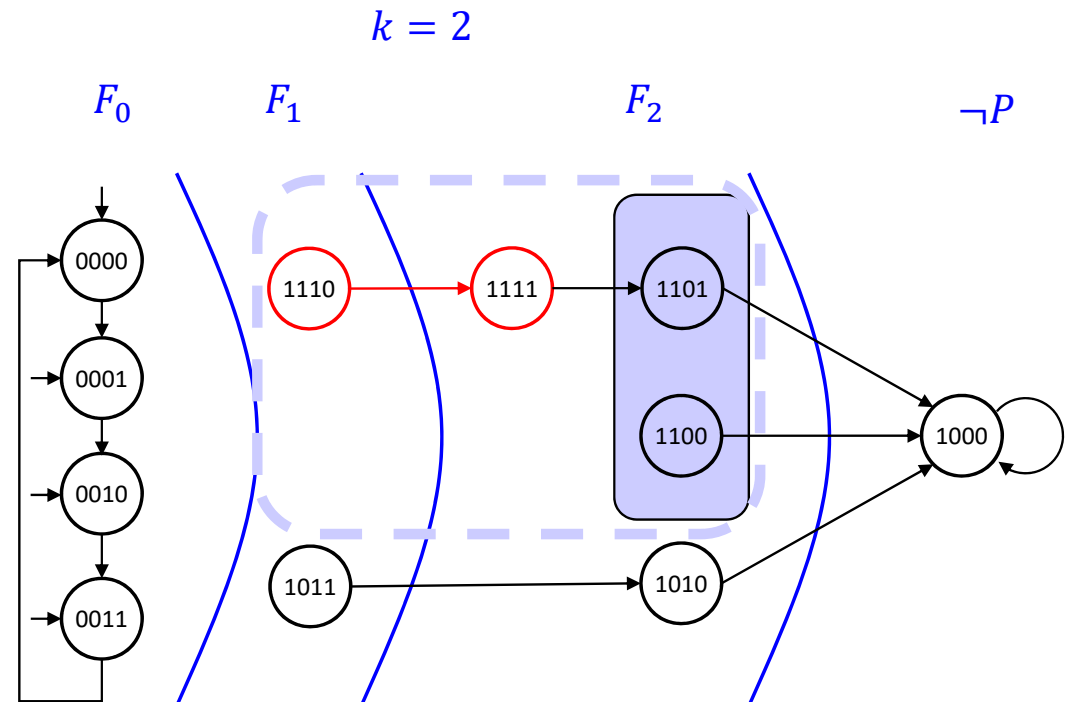
Generalize Further?

- We can generalize to 110-
- Can we generalize to 11--?
 - NO: $f = x_1 \wedge x_2$, we have $\text{SAT}(F_1 \wedge R \wedge f)$
- Transition $1110 \rightarrow 1111$ is the problem
- Is this a real problem?



Generalize Further?

- We can generalize to 110-
- Can we generalize to 11--?
 - NO: $f = x_1 \wedge x_2$, we have $\text{SAT}(F_1 \wedge R \wedge f)$
- Transition $1110 \rightarrow 1111$ is the problem
- Is this a real problem?
 - No: $x_1 \wedge x_2$ is inductive

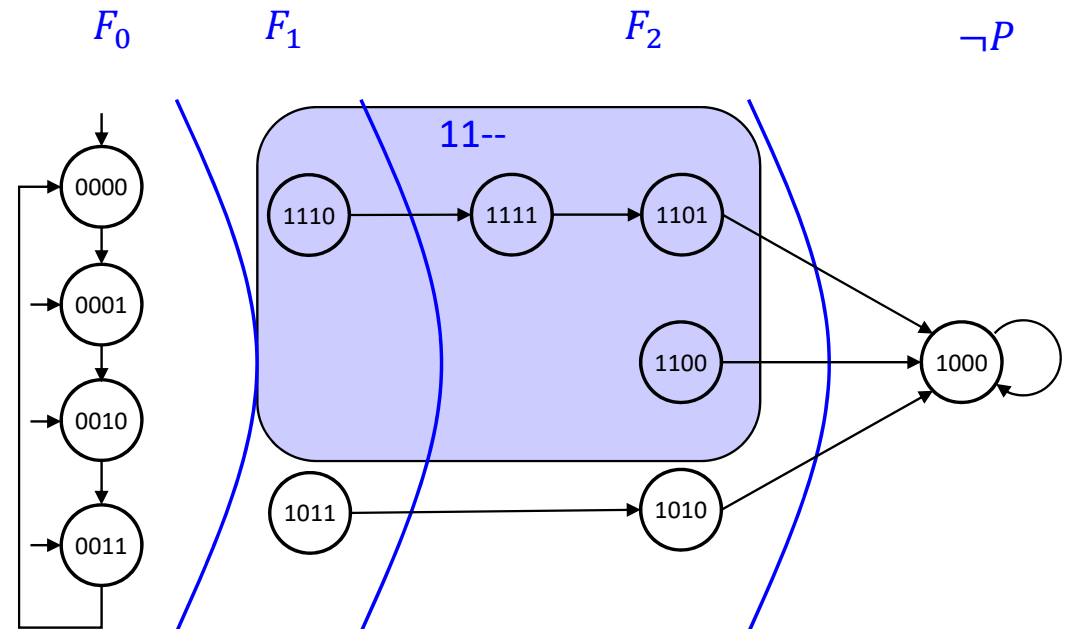


Relative Inductiveness

We can pick shortest cube c such that

- $S \rightarrow c$
- $\neg \text{SAT}(\neg c \wedge F_1 \wedge R \wedge c')$
- $\neg \text{SAT}(S_0 \wedge c)$

$\neg c$ is **relative inductive wrt F_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( $s := \text{SAT}(F_k \wedge R \wedge \neg P')$ )
      removeBad( $k, s$ )
     $k++; F_k := P$ 

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

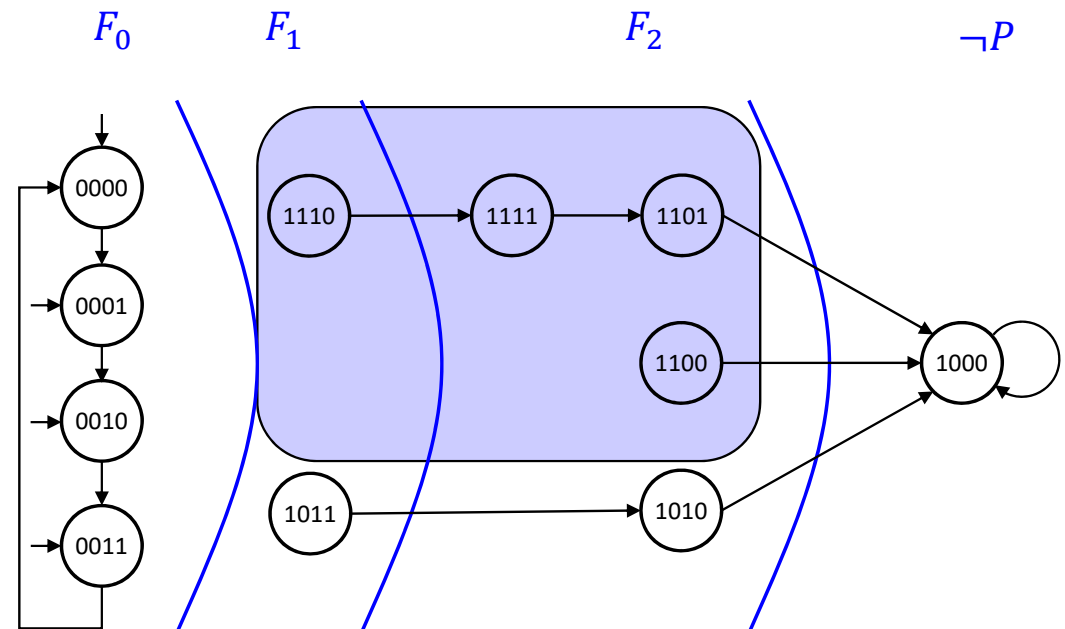
// post: $\neg \text{SAT}(F_i \wedge s)$

```

function removeBad( $i \in N, \text{state } s$ )
  if SAT( $S_0 \wedge c$ ) then FAIL
  while( $t := \text{SAT}(F_{i-1} \wedge R \wedge s')$ )
    removeBad( $i - 1, t$ )
   $g := \text{generalize}(i, s)$ 
   $\forall 0 < j \leq i: F_j := F_j \wedge \neg g$ 
  
```

```

function generalize( $i, \text{state } s$ )
  return a shortest cube  $c$  such that
  -  $c \leftarrow s$ 
  -  $\neg \text{SAT}(\neg c \wedge F_{i-1} \wedge R \wedge c')$ 
  -  $\neg \text{SAT}(S_0 \wedge c)$ 
  
```



Generalization

How do you find a shortest cube?

- See algorithm to the right
- Note
 - this gives you a cube from which you cannot remove more literals
 - There may be cubes that are even shorter

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

c := *s*

while *c* changes

let l_1, \dots, l_n be the literals of *c*

for *i* := 1 **to** *n*

c' = *c* with l_i removed

if relInd(*c'*) **then** *c* = *c'*

return *c*

function relInd(cube *c*)

return $\neg\text{SAT}(\neg c \wedge F_{i-1} \wedge R \wedge c)$
 $\wedge \neg\text{SAT}(S_0 \wedge c)$

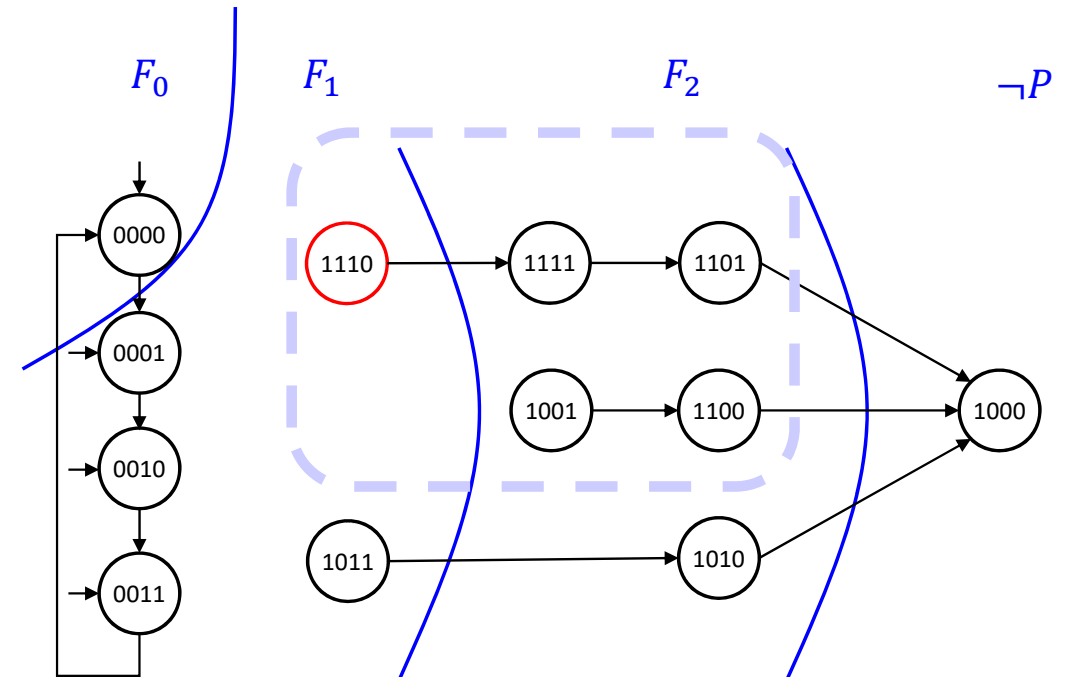
Propagate Clauses

Suppose we have situation on the right.

We are removing 1110 from F_1 .

We can generalize 1110 to 1---

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

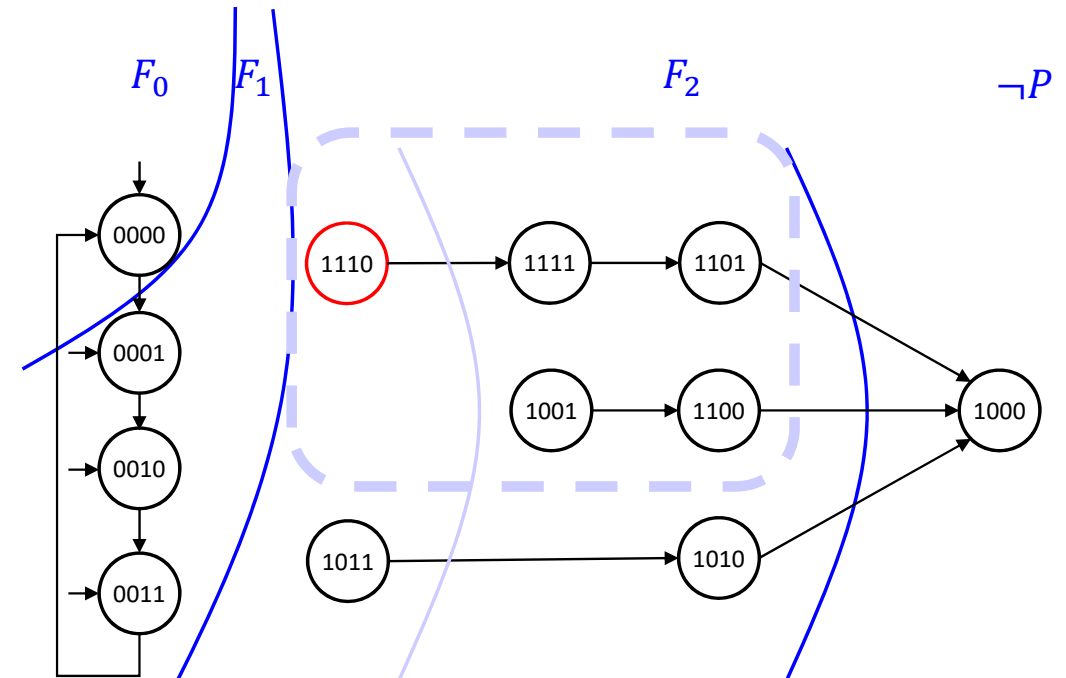


Propagate Clauses

After we update F_1

$F_2 \cap x_1 \cap \text{postimg}(F_1) = \emptyset$, so
we can add $\neg x_1$ to F_2

$\text{UNSAT}(F_1 \wedge R \wedge F'_2 \wedge x'_1)$



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( $s := \text{SAT}(F_k \wedge R \wedge \neg P')$ )  
      removeBad( $k, s$ )  
     $k++; F_k := P$   
    propagateClauses( $k$ )  
    if  $\exists 0 \leq i < k - 1: F_i = F_{i+1}$  then SUCCEED
```

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

```
function removeBad( $i \in N, \text{state } s$ )  
  if SAT( $S_0 \wedge s$ ) then FAIL  
  while( $t := \text{SAT}(F_{i-1} \wedge R \wedge s')$ )  
    removeBad( $i - 1, t$ )  
   $g := \text{generalize}(i, s)$   
   $\forall 0 < j \leq i: F_j := F_j \wedge \neg g$ 
```

```
function generalize( $i, \text{state } s$ )  
  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 somewhat simplified and doesn't find long counterexamples quickly
- Equivalence of frames = syntactic check
 - Use implication and subsumption to simplify clauses
 - Check Mischchenko paper to see if we add clauses when they are subsumed

Performance

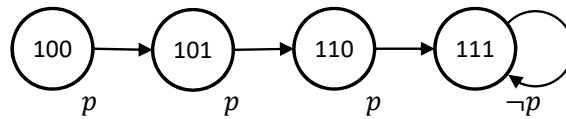
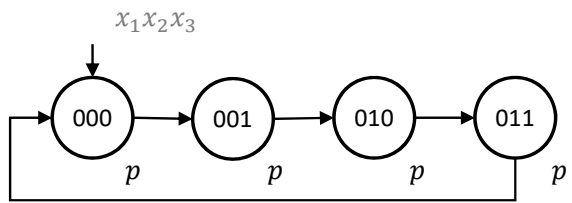
Hardware Model Checking Competition 2020

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-Based 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

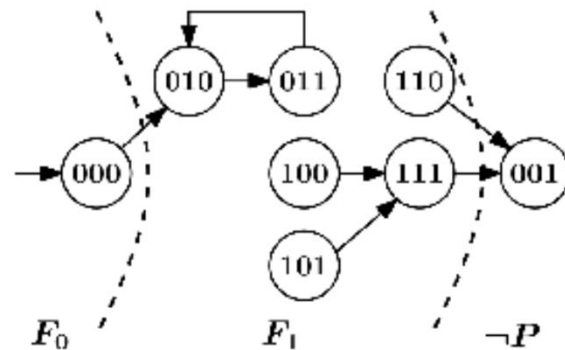


Model Checking Homework 4

Deadline: 7 April 4:00pm

Send solution to: modelchecking@iaik.tugraz.at

Consider the following synchronous Kripke structure K , with states of the form $x_1x_2x_3$. The only initial state is 000, and we are given a property P that holds everywhere but in 001.



We wish to prove that P is always true using PDR. We began the algorithm, obtaining the frames F_0 and F_1 as shown in the figure.

Task 4a [4 points]. Starting from the figure, carry out two iterations of the first variant of the PDR (from $k=1$, until $k=3$) shown in class. Clearly indicate the steps and the frames at the end of each iteration. Is the property P verified at the end? Why/Why not?

Task 4b [3 points]. As before, perform two iterations of PDR starting from the figure. This time use "naive generalization" during the removal of bad states, as shown in class. Clearly indicate the steps and the frames at the end of each iteration. Is the property P verified at the end? Why/Why not?

Task 4c [3 points]. Which of the following statements are false? Justify your answer.

- The set $\neg x_1$ is inductive.
- The set $\neg x_3$ is inductive.
- The set $\neg x_2$ is inductive relative to $\neg x_1$.
- The set $\neg x_3$ is inductive relative to $\neg x_1$.