HW 3

Property-Directed Reachability
or IC3

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 a state s $\in X$ and a state s$' \in Y$ such that $(s, s') \in R$ (there is an edge from s to s')

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

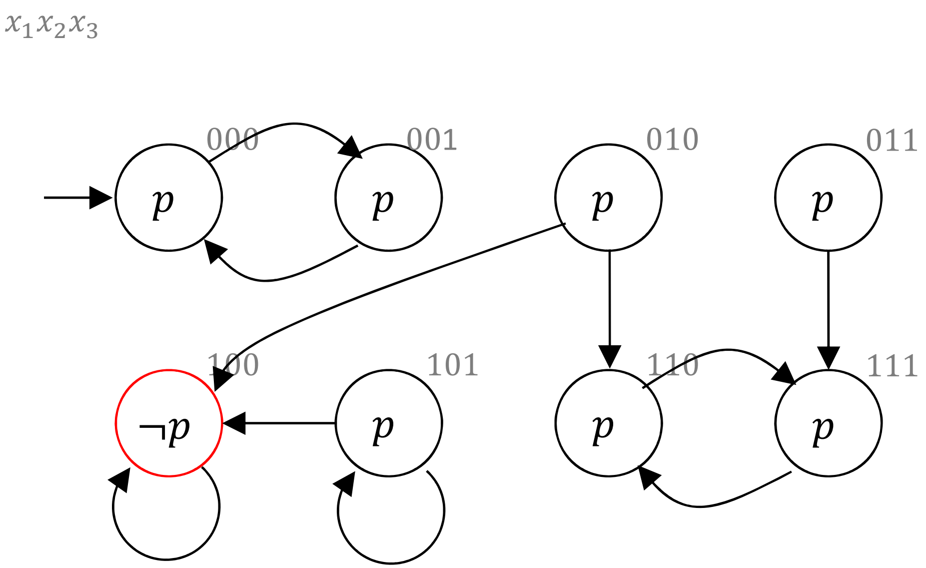
- $s := FALSE$ if ¬SAT($F_i \wedge R \wedge \neg P'$)
- s := a state satisfying in F_i with an edge to a state in $\neg P$ in otherwise

PDR: Notation

Definition

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

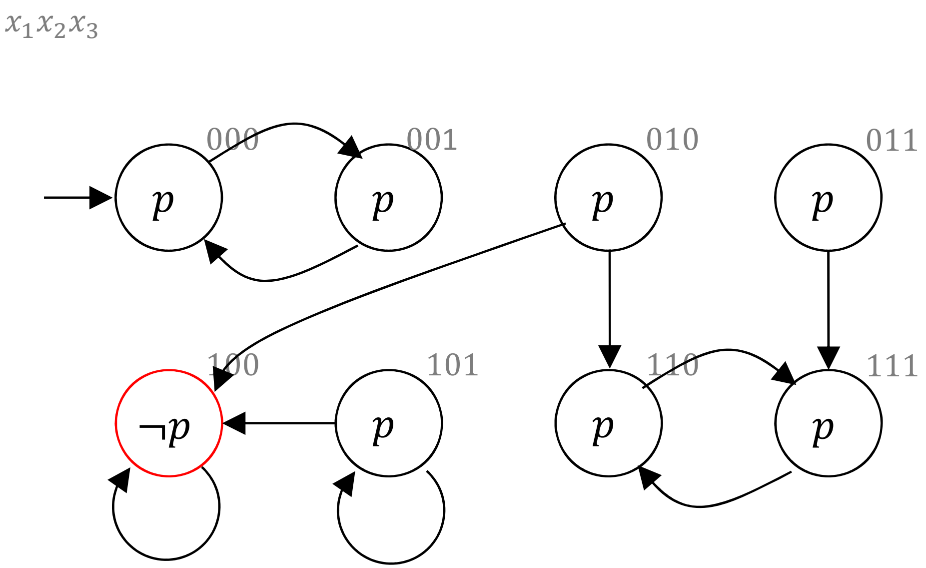
Relative Inductiveness



Inductive facts

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

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 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

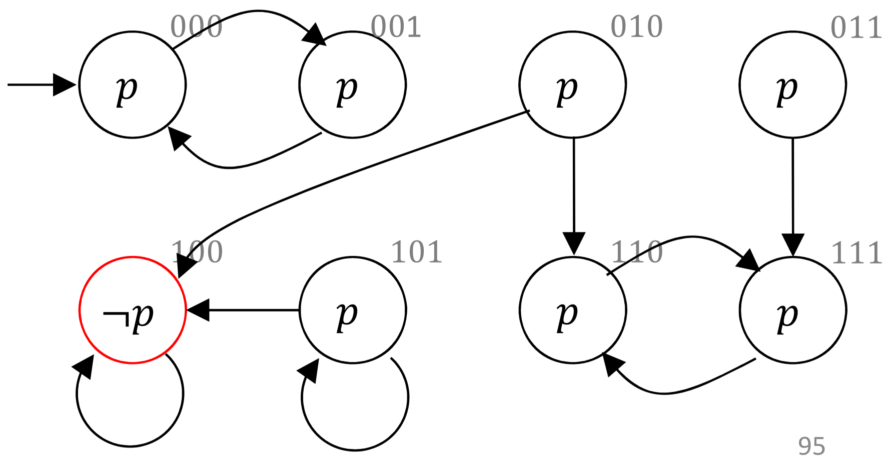
Data Structure

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

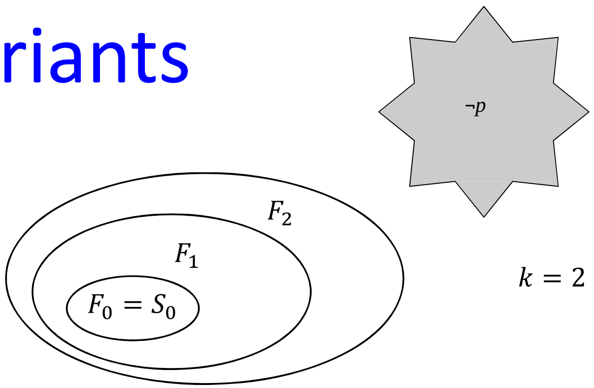
Clause and cubes signify a set of states. Longer clauses – more states. Longer cubes – fewer states

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

Example



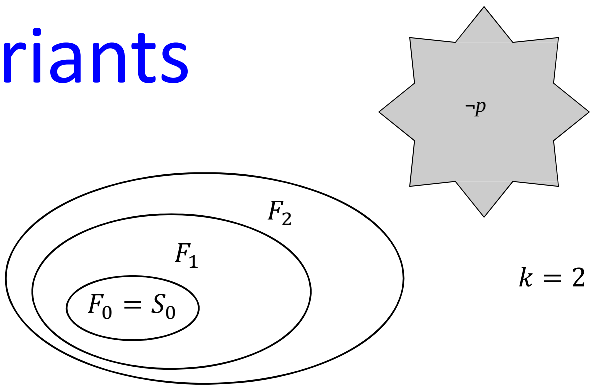
PDR: Data Structures & Invariants



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

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': $F_i = F_{i+1} \wedge c_{i1} \wedge \cdots \wedge c_{in}$
- I3: $F_i \rightarrow P = FALSE$ ($F_i \subseteq P$)
- 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

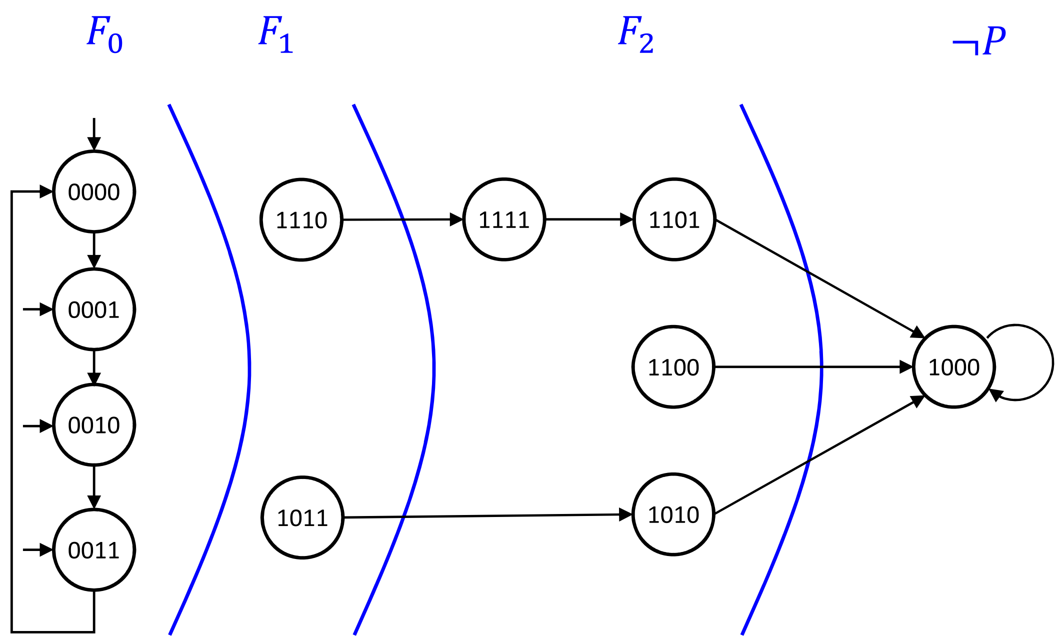
function PDR(Model M)
    if SAT($S_0 \wedge \neg P$) **or** SAT($S_0 \wedge R \wedge \neg P'$) **then FAIL**
    $F_0 := S_0$; $F_1 := P$; $k := 1$;
    while(true)
you are here ⇒   **while**(s := SAT($F_k \wedge R \wedge \neg P'$))
            removeBad(k, s)
        k++; $F_k := P$

        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($S_0 \wedge c$) **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

$k = 2$



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 := 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

you are here

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)

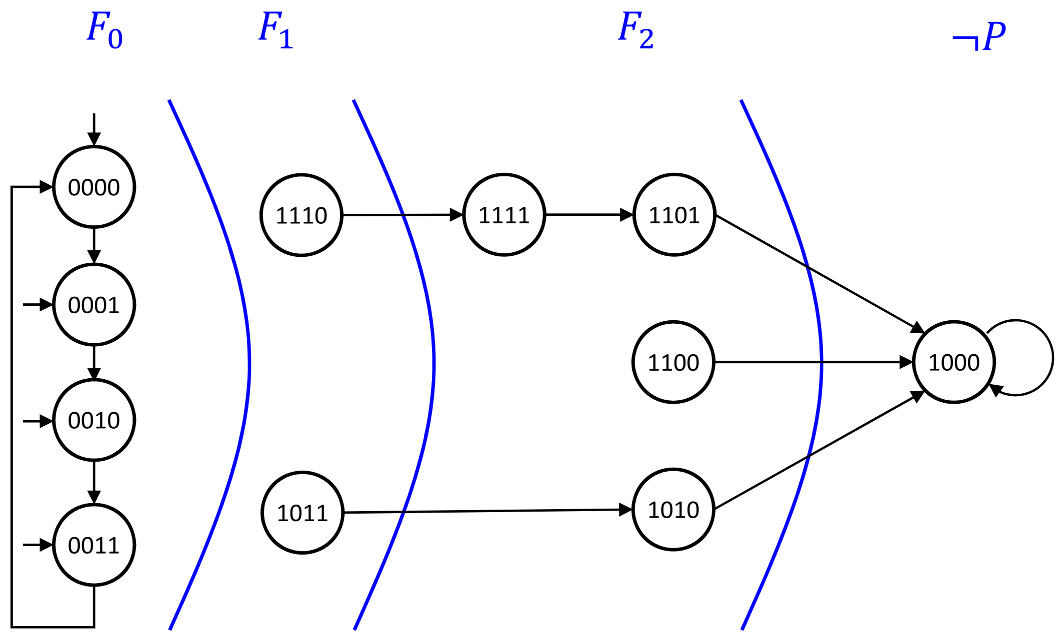
if SAT($S_0 \land c$) **then FAIL**

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

$k = 2$



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 := 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: $\neg SAT(F_i \wedge s)$

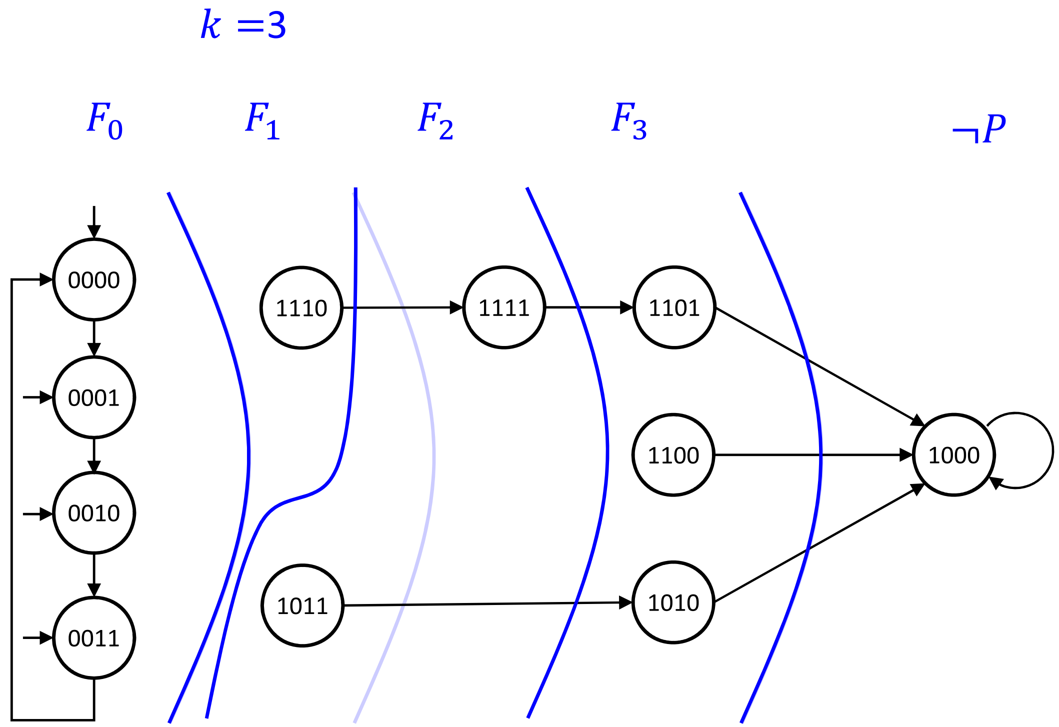
function removeBad($i \in N$, state s)

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

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

removeBad($i-1, t$)

$\forall 0 < j \leq i : F_j \coloneqq F_j \wedge \neg$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: $\neg SAT(F_i \land s)$

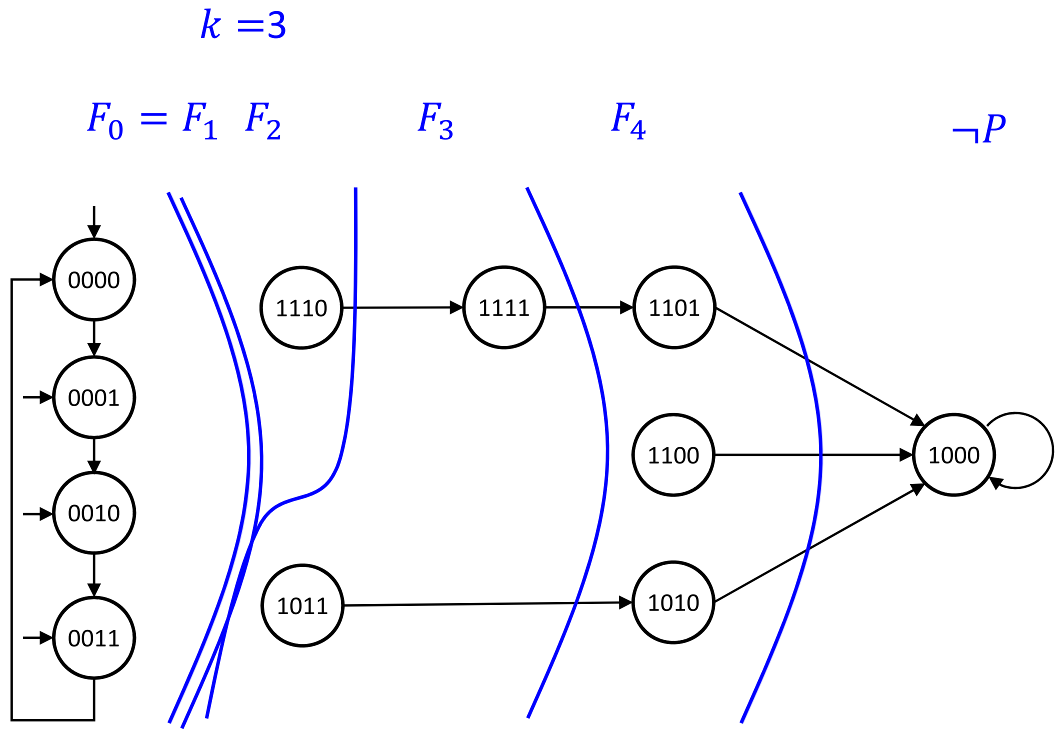
function removeBad($i \in N$, state s)

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

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

removeBad($i - 1, t$)

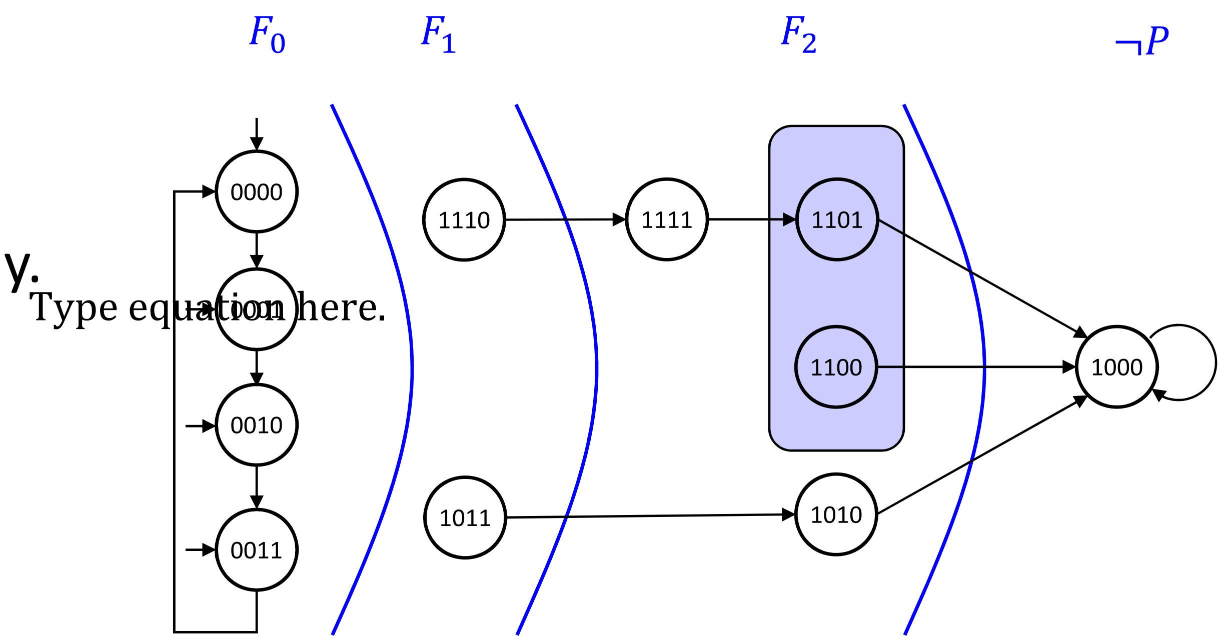
$\forall 0 < j \leq i : F_j \coloneqq F_j \land \neg$s



Drawback

$k = 2$

$x_1 x_2 x_3 x_4$



- The first version considers every state individually
- But similar states behave similarly. Example: 1100 and 1101. Generalize 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($S_0 \land \neg P$)) **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 $\exists 0 \le i < k-1 : F_i := F_{i+1}$ **then SUCCEED**

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

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

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

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

    removeBad($i-1, t$)

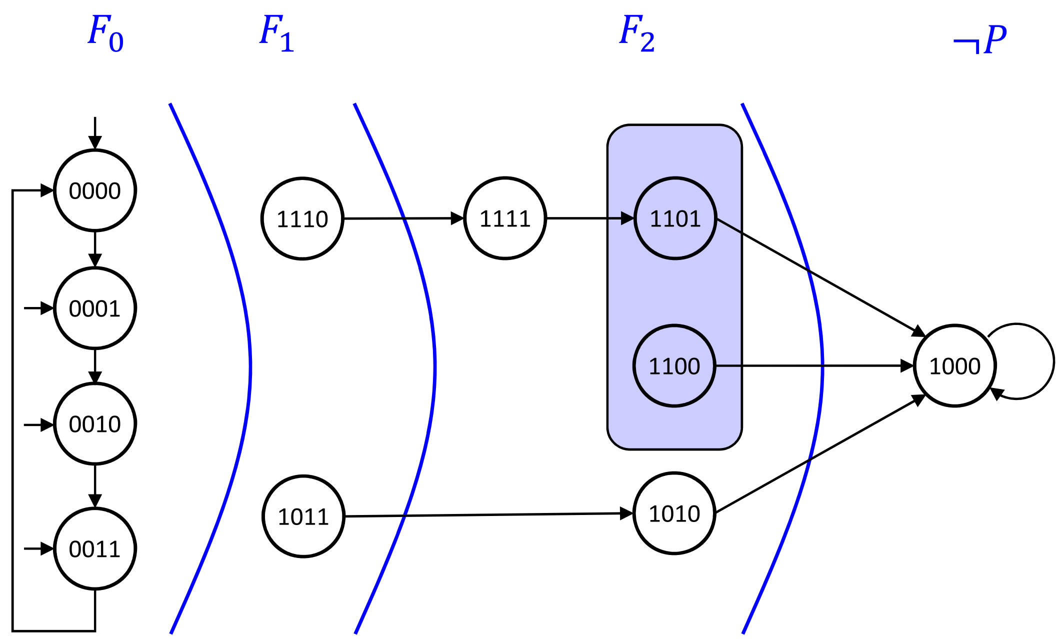
  g := generalizeNaive(i, s)

  $\forall 0 < j \le i : F_j \coloneqq F_j \land \neg g$

function generalizeNaive(i, state s)

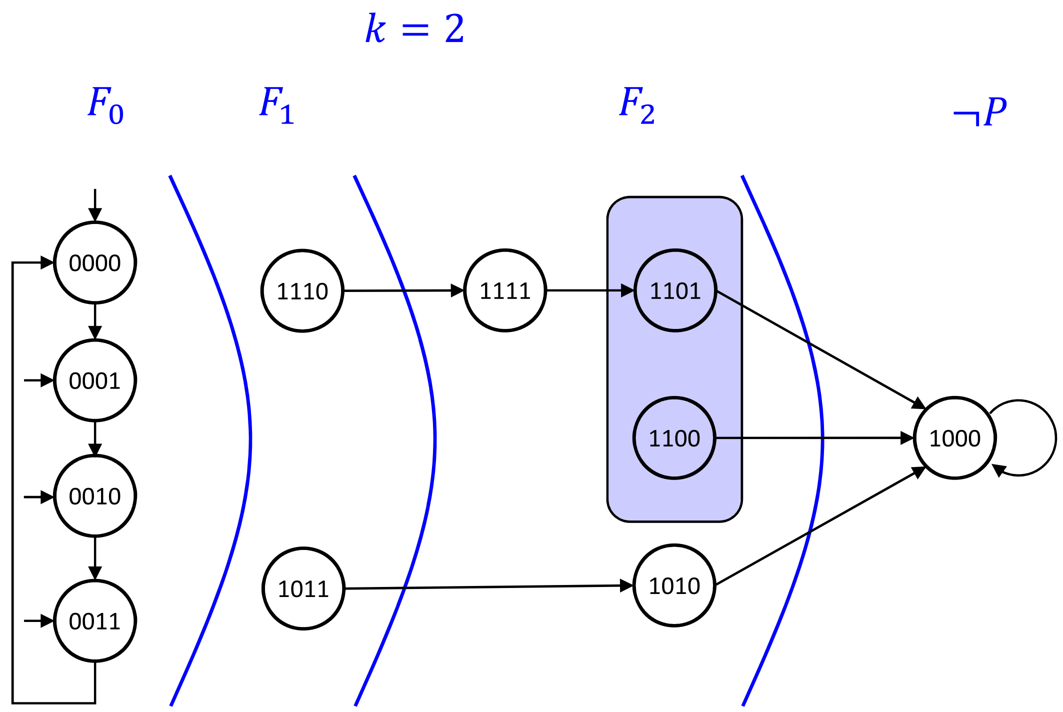
  return a shortest cube c such that

  - $c \leftarrow s$
  - $\neg \mathrm{SAT}(\mathrm{F_{i-1}} \land R \land c')$
  - $\neg \mathrm{SAT}(S_0 \land c)$



Generalize Further?

- We can generalize to 110-
- Can we generalize to 11--?
  - NO: $f = x_1 \wedge x_2$, we have SAT($F_1 \wedge R \wedge f$)
- State 1100 is the problem



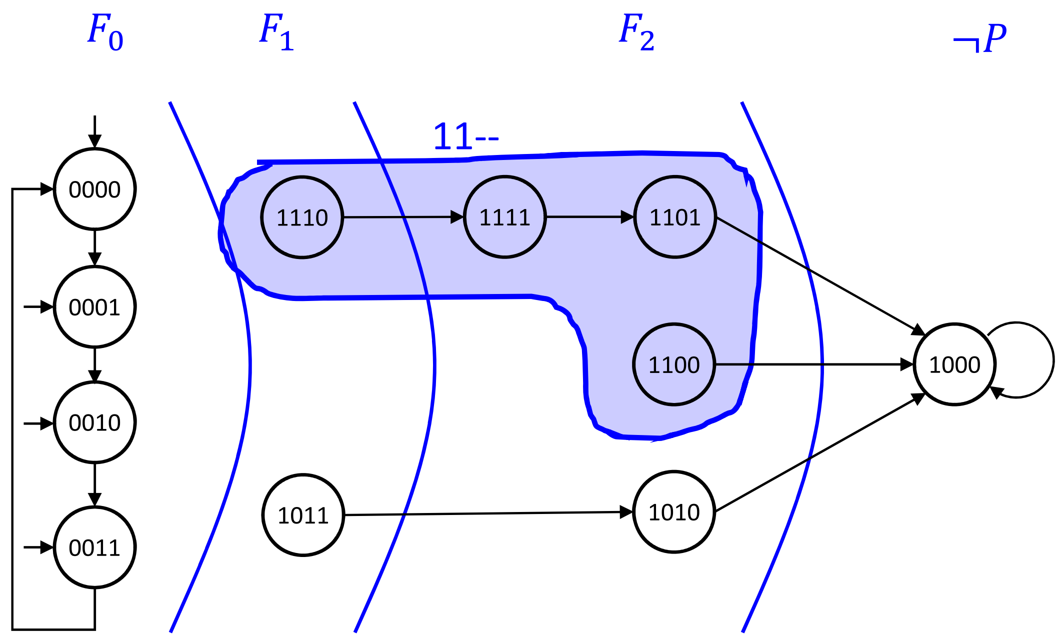
Relative Inductiveness

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_2

Why?



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

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

    removeBad($i-1, t$)

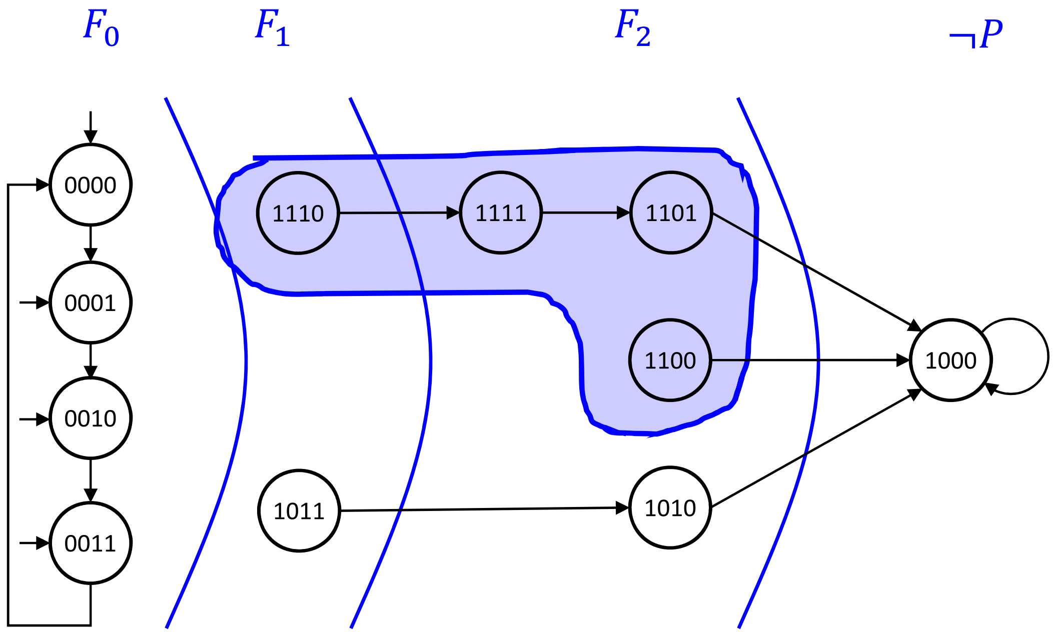
  g := generalize(i,s)

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

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

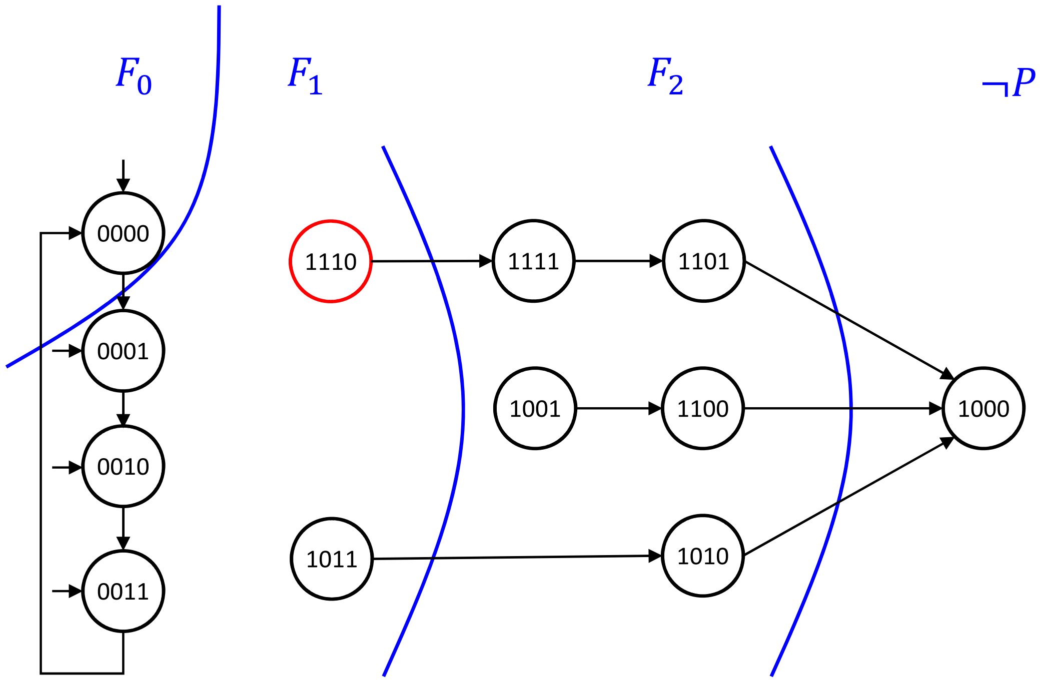
function generalize(i, state s)
$c := s$
while c changes
let $l_1, \ldots, l_n$ be the literals of c
for $i := 1$ **to** n
$c' = c$ with l_i removed
if relInd(c') **then** $c = c'$
return c

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

Propagate Clauses

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

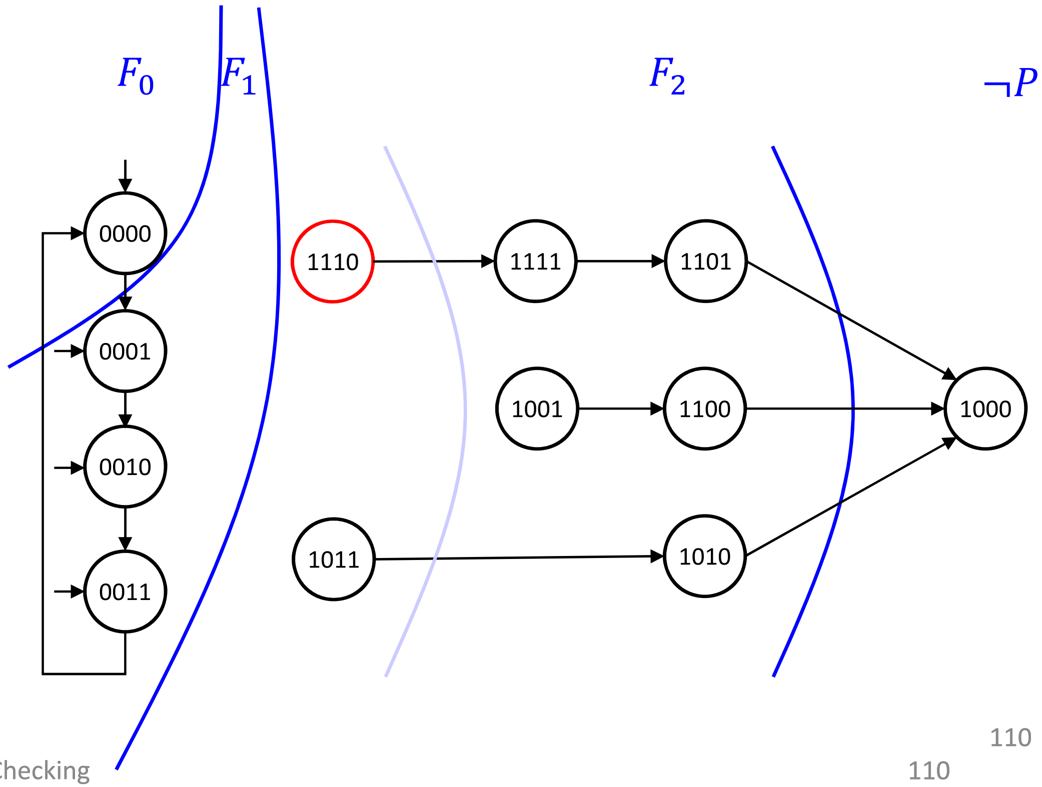
$$F_1 := F_1 \wedge \neg x_1$$



Propagate Clauses

$F_2 \wedge x_1 \notin postimg(F_1)$, 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 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

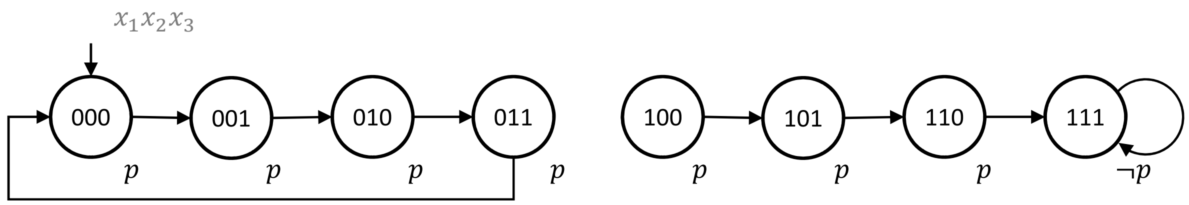
Hardware Model Checking Competition 2020

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-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

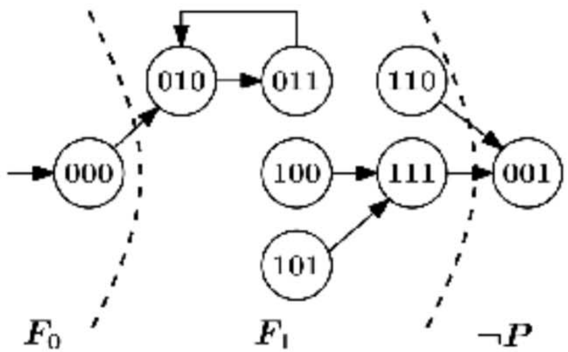


$x_1x_2x_3$
000
001
010
011
100
101
110
111
p
p
p
p
p
p
p
$\neg p$

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 $\neg x_1$.