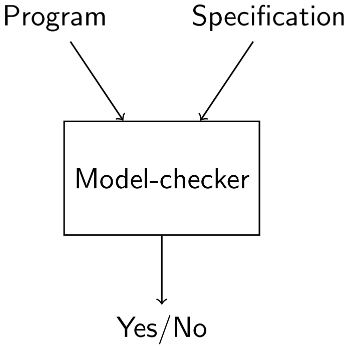
Software Model-Checking: an algorithmic approach to prove programs correct

Damien Zufferey

IST Austria

May 27, 2011

Push-button approach



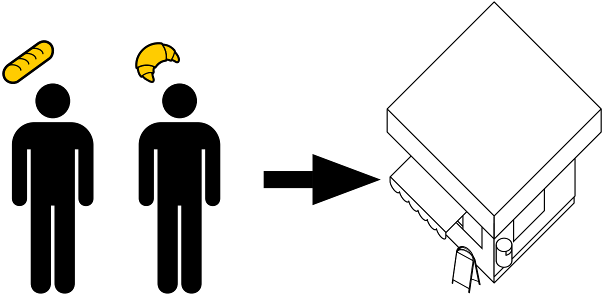
Program

Specification

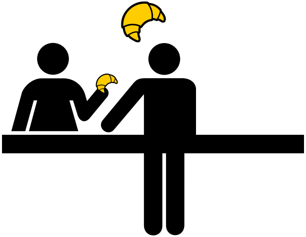
Model-checker

Yes/No

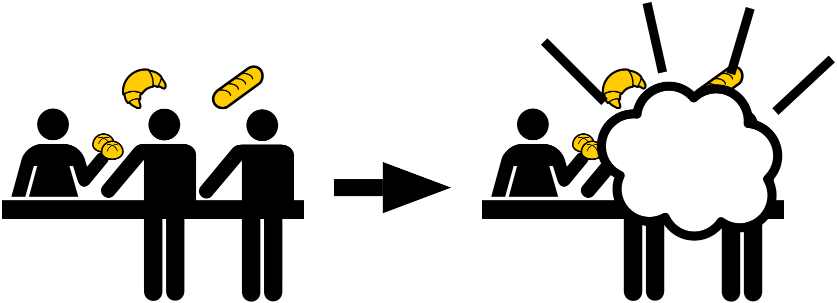
Running example: Lamport's bakery algorithm



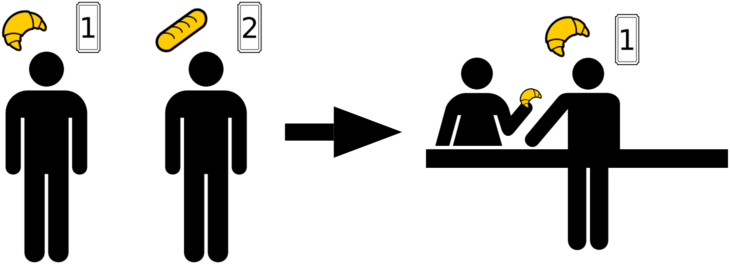
Running example: Lamport's bakery algorithm



Running example: Lamport's bakery algorithm



Running example: Lamport's bakery algorithm



Implementation of the algorithm for 2 customers.

initial state: pc1 = 0, x1 = 0, pc2 = 0, x2 = 0

pc values: (0 → outside), (1 → waiting), (2 → ordering)

```
while (true) {
  if(pc1 == 0){
    x1 = x2 + 1;
    pc1 = 1;
  }else if(pc1 == 1 &&
             (x2 == 0 ||
              x1 < x2 )){
    pc1 = 2;
  }else if(pc1 == 2){
    pc1 = 0;
    x1 = 0;
  }
  if(pc1==2 && pc2==2){
    ERROR;
  }
}
```

```
while (true) {
  if(pc2 == 0){
    x2 = x1 + 1;
    pc2 = 1;
  }else if(pc2 == 1 &&
             (x1 == 0 ||
              x2 < x1 )){
    pc2 = 2;
  }else if(pc2 == 2){
    pc2 = 0;
    x2 = 0;
  }
  if(pc1==2 && pc2==2){
    ERROR;
  }
}
```

Why do we means by correct ?

safety: no two customers fight.

liveness: every customers eventually get served (starvation-free).

For this talk we will care only about safety property.

The program as a dynamic system:

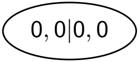
Initial state: $pc_1 = 0, x_1 = 0, pc_2 = 0, x_2 = 0$

Transitions:

$$\begin{array}{lcl}
pc_1 = 0 & \rightarrow & pc_1' = 1,\ x_1' = x_2 + 1 \\
pc_1 = 1 \wedge (x_2 = 0 \vee x_1 < x_2) & \rightarrow & pc_1' = 2 \\
pc_1 = 2 & \rightarrow & pc_1' = 0,\ x_1' = 0 \\
 & & \\
pc_2 = 0 & \rightarrow & pc_2' = 1,\ x_2' = x_1 + 1 \\
pc_2 = 1 \wedge (x_1 = 0 \vee x_2 < x_1) & \rightarrow & pc_2' = 2 \\
pc_2 = 2 & \rightarrow & pc_2' = 0,\ x_2' = 0 \\
 & & \\
pc_1 = 2 \wedge pc_2 = 2 & \rightarrow & \text{ERROR}
\end{array}$$

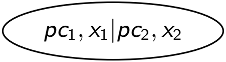
Reachability graph:

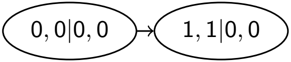
State: $pc_1, x_1 | pc_2, x_2$



$0, 0 | 0, 0$

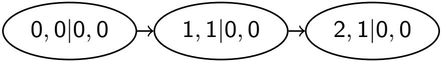
Reachability graph:

State: 



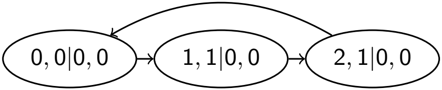
Reachability graph:

State: $pc_1, x_1 | pc_2, x_2$



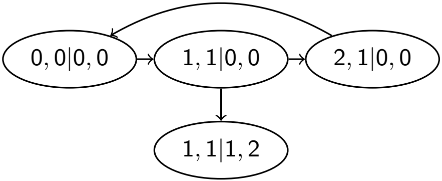
Reachability graph:

State: $pc_1, x_1 | pc_2, x_2$



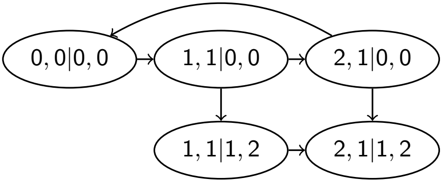
Reachability graph:

State: $pc_1, x_1 | pc_2, x_2$



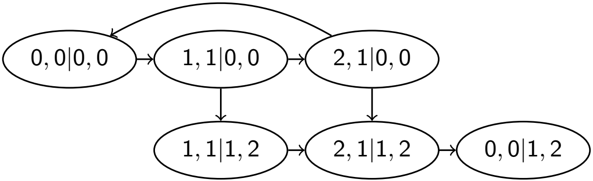
Reachability graph:

State: $pc_1, x_1 | pc_2, x_2$



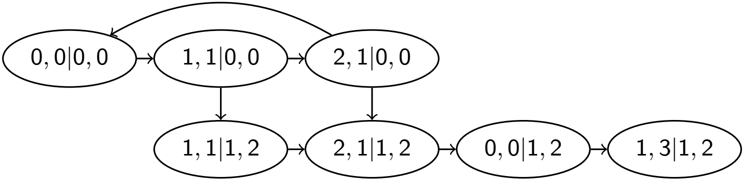
Reachability graph:

State: $pc_1, x_1 | pc_2, x_2$



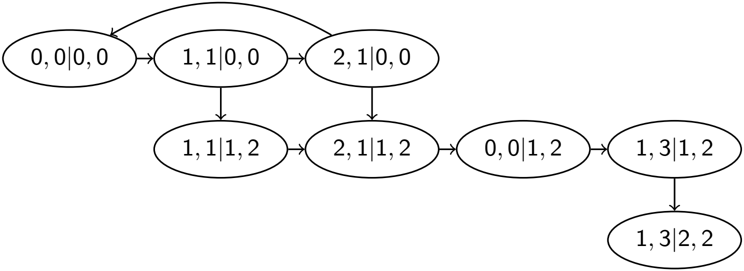
Reachability graph:

State: $pc_1, x_1 | pc_2, x_2$



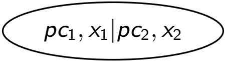
Reachability graph:

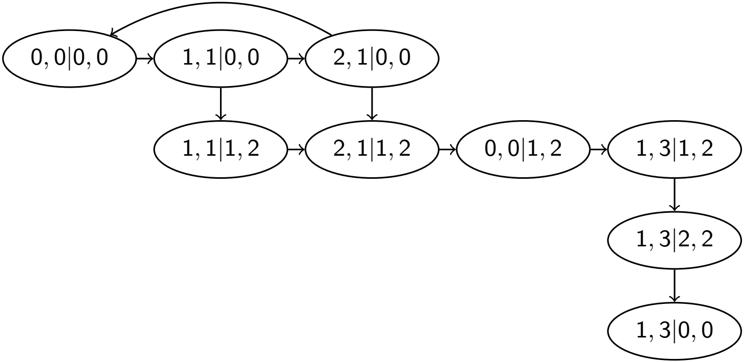
State: $pc_1, x_1 | pc_2, x_2$



Reachability graph:

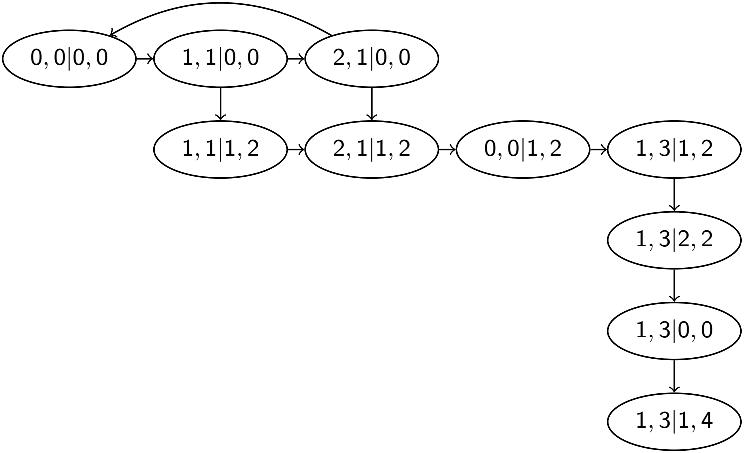
State:





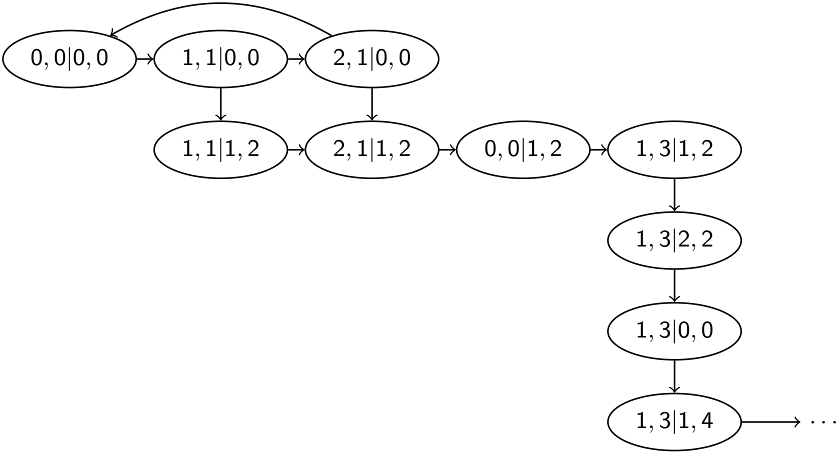
Reachability graph:

State: $pc_1, x_1 | pc_2, x_2$



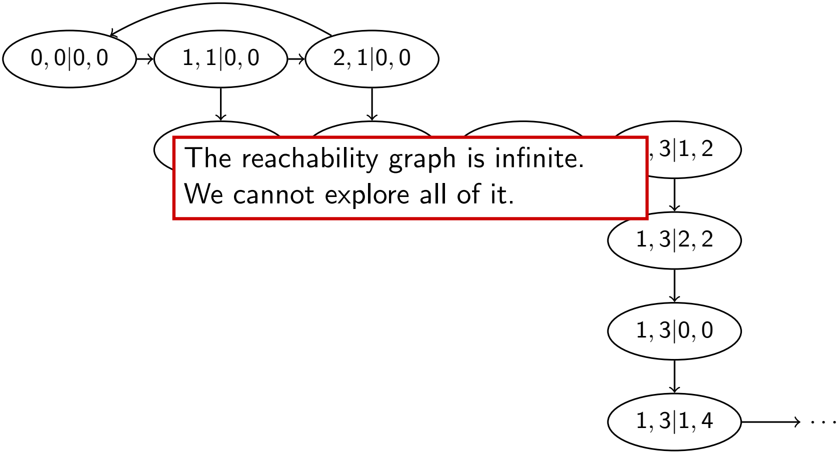
Reachability graph:

State: $pc_1, x_1 | pc_2, x_2$



Reachability graph:

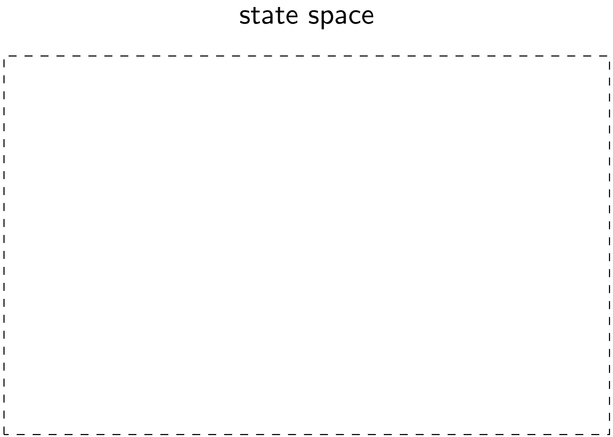
State: $pc_1, x_1 | pc_2, x_2$



The reachability graph is infinite.
We cannot explore all of it.

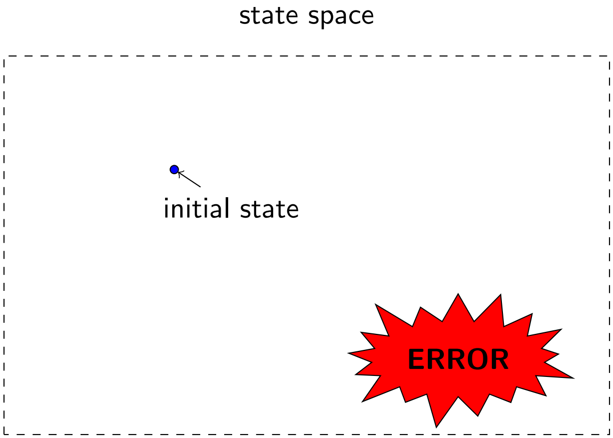
The picture of abstraction

The picture of abstraction



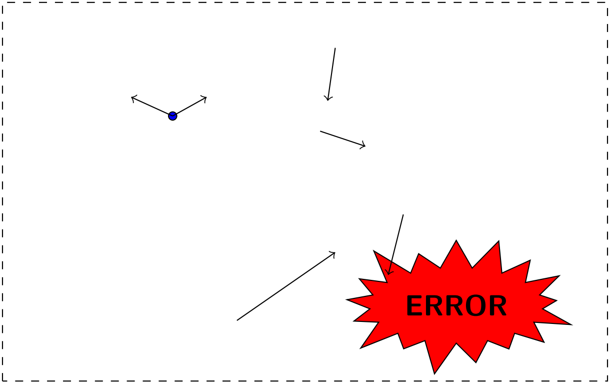
state space

The picture of abstraction

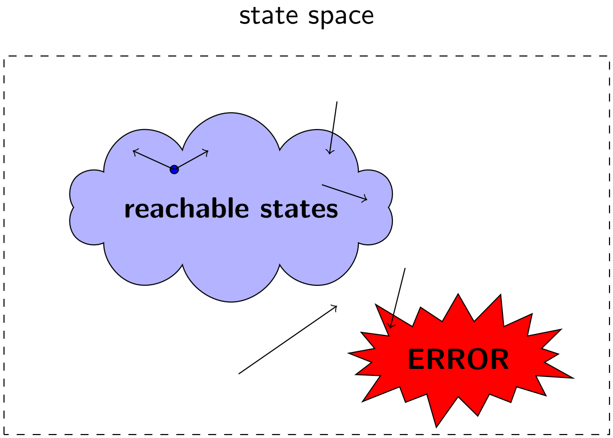


The picture of abstraction

state space

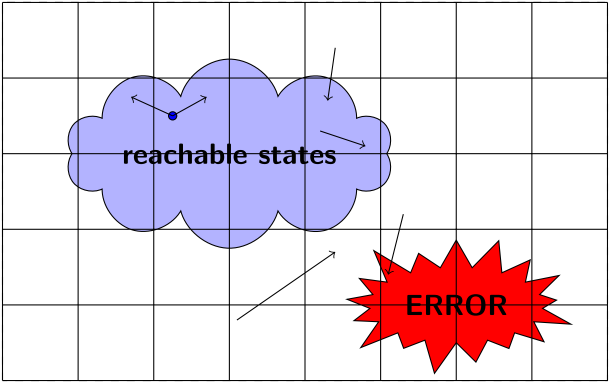


The picture of abstraction



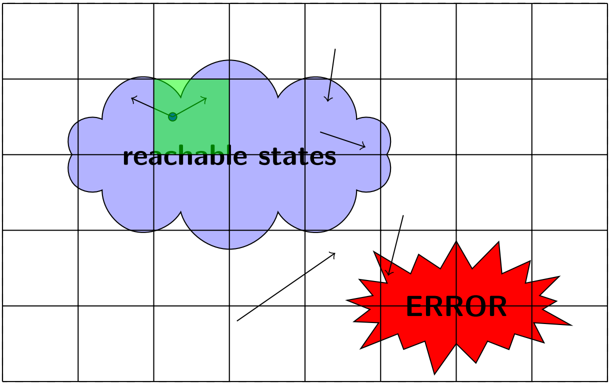
The picture of abstraction

finite abstraction of the state space



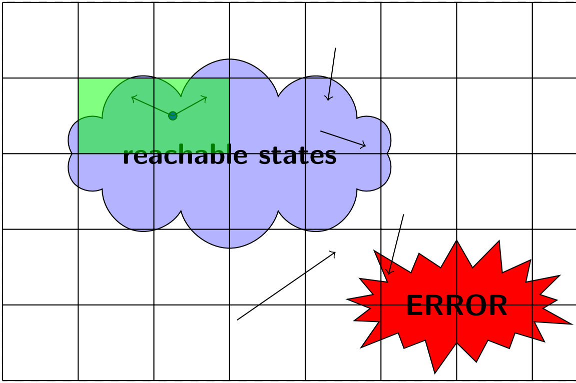
The picture of abstraction

finite abstraction of the state space



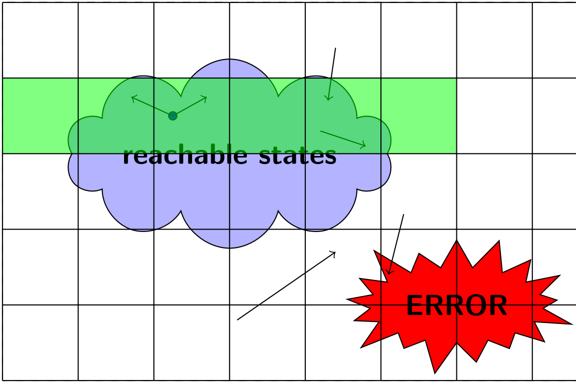
The picture of abstraction

finite abstraction of the state space



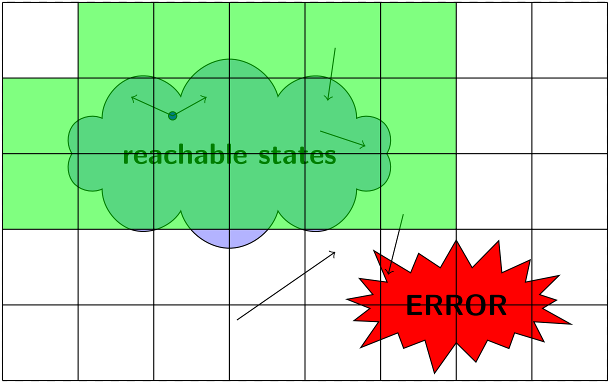
The picture of abstraction

finite abstraction of the state space



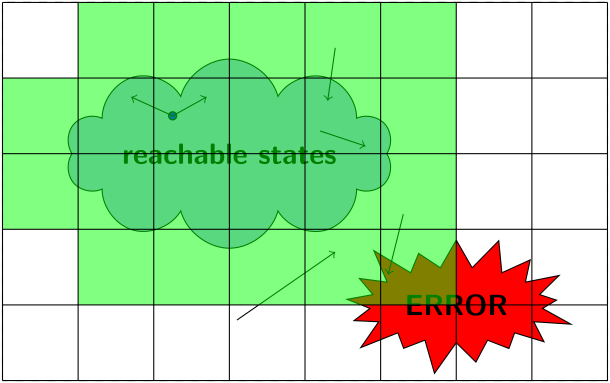
The picture of abstraction

finite abstraction of the state space



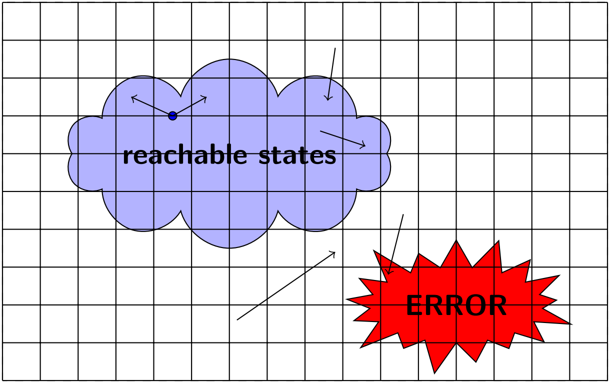
The picture of abstraction

finite abstraction of the state space



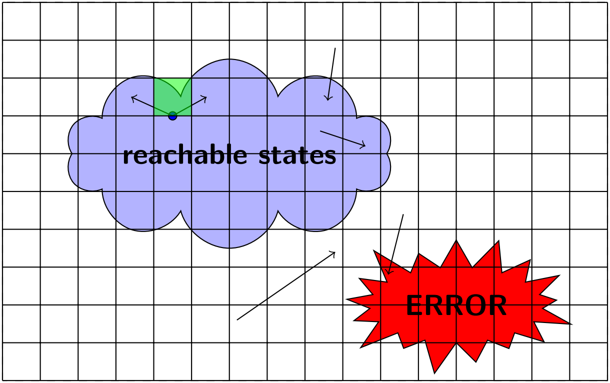
The picture of abstraction

finer abstraction of the state space



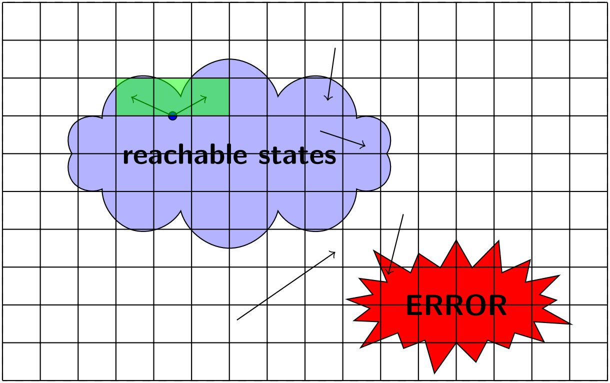
The picture of abstraction

finer abstraction of the state space



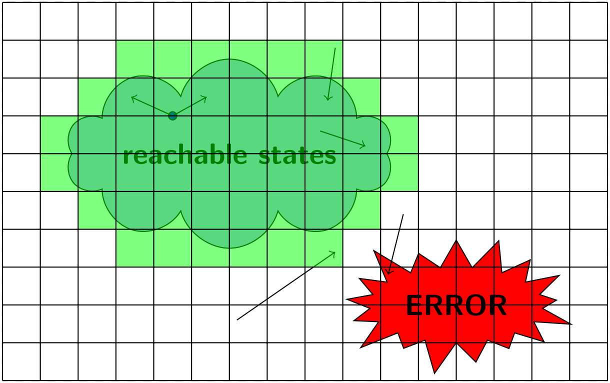
The picture of abstraction

finer abstraction of the state space



The picture of abstraction

finer abstraction of the state space



Abstraction: preserving only some facts

problem: the range of x_1 and x_2 is infinite.

Maybe they are not important.

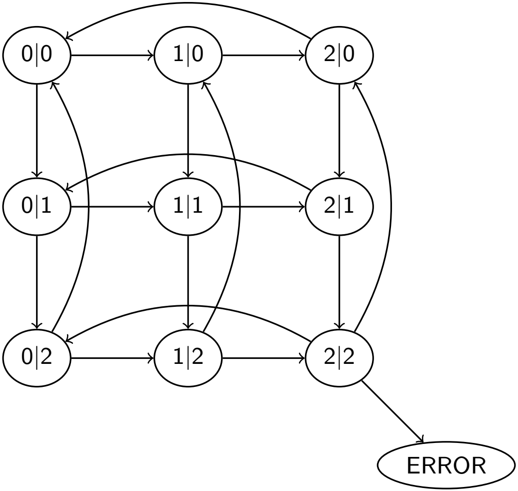
Initial state: $pc_1 = 0, pc_2 = 0$

Transitions:

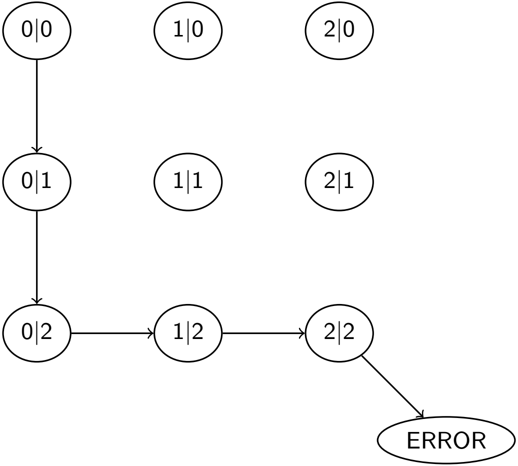
$$
\begin{array}{lcl}
pc_1 = 0 & \rightarrow & pc_1' = 1 \\
pc_1 = 1 & \rightarrow & pc_1' = 2 \\
pc_1 = 2 & \rightarrow & pc_1' = 0 \\
pc_2 = 0 & \rightarrow & pc_2' = 1 \\
pc_2 = 1 & \rightarrow & pc_2' = 2 \\
pc_2 = 2 & \rightarrow & pc_2' = 0 \\
pc_1 = 2 \land pc_2 = 2 & \rightarrow & \text{ERROR}
\end{array}
$$

The abstract system preserves traces, but adds new ones.

Abstract reachability graph



Abstract reachability graph



Analyzing the counterexample

Abstract trace:

Concrete trace:

$(pc_1 = 0 | pc_2 = 0)$

$pc_2 = 0 \rightarrow pc'_2 = 1$

$(pc_1 = 0 | pc_2 = 1)$

$pc_2 = 1 \rightarrow pc'_2 = 2$

$(pc_1 = 0 | pc_2 = 2)$

$pc_1 = 0 \rightarrow pc'_1 = 1$

$(pc_1 = 1 | pc_2 = 2)$

$pc_1 = 1 \rightarrow pc'_1 = 2$

$(pc_1 = 2 | pc_2 = 2)$

$pc_1 = 2 \wedge pc_2 = 2 \rightarrow \mathrm{ERROR}$

ERROR

Analyzing the counterexample

Abstract trace:

$(pc_1 = 0 | pc_2 = 0)$

$pc_2 = 0 \rightarrow pc_2' = 1$

$(pc_1 = 0 | pc_2 = 1)$

$pc_2 = 1 \rightarrow pc_2' = 2$

$(pc_1 = 0 | pc_2 = 2)$

$pc_1 = 0 \rightarrow pc_1' = 1$

$(pc_1 = 1 | pc_2 = 2)$

$pc_1 = 1 \rightarrow pc_1' = 2$

$(pc_1 = 2 | pc_2 = 2)$

$pc_1 = 2 \land pc_2 = 2 \rightarrow \text{ERROR}$

ERROR

Concrete trace:

$(pc_1 = 0, x_1 = 0 | pc_2 = 0, x_2 = 0)$

Analyzing the counterexample

Abstract trace:

$(pc_1 = 0 | pc_2 = 0)$

$pc_2 = 0 \rightarrow pc_2' = 1$

$(pc_1 = 0 | pc_2 = 1)$

$pc_2 = 1 \rightarrow pc_2' = 2$

$(pc_1 = 0 | pc_2 = 2)$

$pc_1 = 0 \rightarrow pc_1' = 1$

$(pc_1 = 1 | pc_2 = 2)$

$pc_1 = 1 \rightarrow pc_1' = 2$

$(pc_1 = 2 | pc_2 = 2)$

$pc_1 = 2 \wedge pc_2 = 2 \rightarrow \text{ERROR}$

ERROR

Concrete trace:

$(pc_1 = 0, x_1 = 0 | pc_2 = 0, x_2 = 0)$

$pc_2 = 0 \rightarrow pc_2' = 1,\ x_2' = x_1 + 1$

$(pc_1 = 0, x_1 = 0 | pc_2 = 1, x_2 = 1)$

Analyzing the counterexample

Abstract trace:

$(pc_1 = 0 | pc_2 = 0)$

$pc_2 = 0 \rightarrow pc_2' = 1$

$(pc_1 = 0 | pc_2 = 1)$

$pc_2 = 1 \rightarrow pc_2' = 2$

$(pc_1 = 0 | pc_2 = 2)$

$pc_1 = 0 \rightarrow pc_1' = 1$

$(pc_1 = 1 | pc_2 = 2)$

$pc_1 = 1 \rightarrow pc_1' = 2$

$(pc_1 = 2 | pc_2 = 2)$

$pc_1 = 2 \wedge pc_2 = 2 \rightarrow \text{ERROR}$

ERROR

Concrete trace:

$(pc_1 = 0, x_1 = 0 | pc_2 = 0, x_2 = 0)$

$pc_2 = 0 \rightarrow pc_2' = 1,\ x_2' = x_1 + 1$

$(pc_1 = 0, x_1 = 0 | pc_2 = 1, x_2 = 1)$

$pc_2 = 1 \wedge (x_1 = 0 \vee x_2 < x_1) \rightarrow pc_2' = 2$

$(pc_1 = 0, x_1 = 0 | pc_2 = 2, x_2 = 1)$

Analyzing the counterexample

Abstract trace:

$(pc_1 = 0 | pc_2 = 0)$

$pc_2 = 0 \rightarrow pc_2' = 1$

$(pc_1 = 0 | pc_2 = 1)$

$pc_2 = 1 \rightarrow pc_2' = 2$

$(pc_1 = 0 | pc_2 = 2)$

$pc_1 = 0 \rightarrow pc_1' = 1$

$(pc_1 = 1 | pc_2 = 2)$

$pc_1 = 1 \rightarrow pc_1' = 2$

$(pc_1 = 2 | pc_2 = 2)$

$pc_1 = 2 \wedge pc_2 = 2 \rightarrow \text{ERROR}$

ERROR

Concrete trace:

$(pc_1 = 0, x_1 = 0 | pc_2 = 0, x_2 = 0)$

$pc_2 = 0 \rightarrow pc_2' = 1,\ x_2' = x_1 + 1$

$(pc_1 = 0, x_1 = 0 | pc_2 = 1, x_2 = 1)$

$pc_2 = 1 \wedge (x_1 = 0 \vee x_2 < x_1) \rightarrow pc_2' = 2$

$(pc_1 = 0, x_1 = 0 | pc_2 = 2, x_2 = 1)$

$pc_1 = 0 \rightarrow pc_1' = 1,\ x_1' = x_2 + 1$

$(pc_1 = 1, x_1 = 2 | pc_2 = 2, x_2 = 1)$

Analyzing the counterexample

Abstract trace:

$(pc_1 = 0 | pc_2 = 0)$

$pc_2 = 0 \to pc_2' = 1$

$(pc_1 = 0 | pc_2 = 1)$

$pc_2 = 1 \to pc_2' = 2$

$(pc_1 = 0 | pc_2 = 2)$

$pc_1 = 0 \to pc_1' = 1$

$(pc_1 = 1 | pc_2 = 2)$

$pc_1 = 1 \to pc_1' = 2$

$(pc_1 = 2 | pc_2 = 2)$

$pc_1 = 2 \wedge pc_2 = 2 \to \text{ERROR}$

ERROR

Concrete trace:

$(pc_1 = 0, x_1 = 0 | pc_2 = 0, x_2 = 0)$

$pc_2 = 0 \to pc_2' = 1,\ x_2' = x_1 + 1$

$(pc_1 = 0, x_1 = 0 | pc_2 = 1, x_2 = 1)$

$pc_2 = 1 \wedge (x_1 = 0 \vee x_2 < x_1) \to pc_2' = 2$

$(pc_1 = 0, x_1 = 0 | pc_2 = 2, x_2 = 1)$

$pc_1 = 0 \to pc_1' = 1,\ x_1' = x_2 + 1$

$(pc_1 = 1, x_1 = 2 | pc_2 = 2, x_2 = 1)$

$pc_1 = 1 \wedge (x_2 = 0 \vee x_1 < x_2) \to pc_1' = 2$

Analyzing the counterexample

Abstract trace:

$(pc_1 = 0 | pc_2 = 0)$

$pc_2 = 0 \rightarrow pc_2' = 1$

$(pc_1 = 0 | pc_2 = 1)$

$pc_2 = 1 \rightarrow pc_2' =$

$(pc_1 = 0 | pc_2 = 2$

$pc_1 = 0 \rightarrow pc_1' =$

$(pc_1 = 1 | pc_2 = 2)$

$pc_1 = 1 \rightarrow pc_1' = 2$

$(pc_1 = 2 | pc_2 = 2)$

$pc_1 = 2 \wedge pc_2 = 2 \rightarrow \text{ERROR}$

ERROR

Concrete trace:

$(pc_1 = 0, x_1 = 0 | pc_2 = 0, x_2 = 0)$

$pc_2 = 0 \rightarrow pc_2' = 1,\ x_2' = x_1 + 1$

$(pc_1 = 0, x_1 = 0 | pc_2 = 1, x_2 = 1)$

$_1) \rightarrow pc_2' = 2$

$_2 = 1)$

$x_2 + 1$

$(pc_1 = 1, x_1 = 2 | pc_2 = 2, x_2 = 1)$

$pc_1 = 1 \wedge (x_2 = 0 \vee x_1 < x_2) \rightarrow pc_1' = 2$

The counterexample is spurious.
The abstraction is too coarse.
We need to add new facts.

Refinement: adding facts

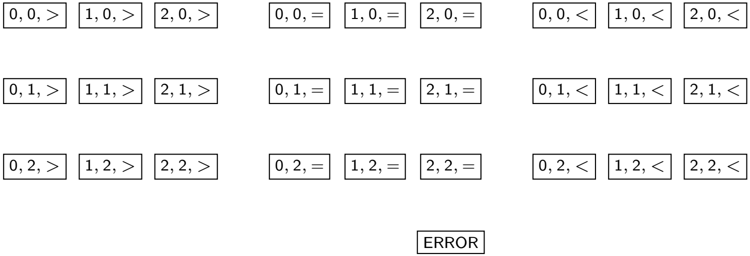
We cannot track the exact ticket value. But we can remember who has the smallest number $(<, =, >)$.

Initial state: $pc_1 = 0, pc_2 = 0, x_1 = x_2$

Transitions:

$$\begin{array}{lcl}
pc_1 = 0 & \rightarrow & pc_1' = 1,\ x_1' > x_2' \\
pc_1 = 1 \wedge (? \vee x_1 < x_2) & \rightarrow & pc_1' = 2 \\
pc_1 = 2 & \rightarrow & pc_1' = 0,\ x_1' \ ?\ x_2' \\
pc_2 = 0 & \rightarrow & pc_2' = 1,\ x_1' < x_2' \\
pc_2 = 1 \wedge (? \vee x_2 < x_1) & \rightarrow & pc_2' = 2 \\
pc_2 = 2 & \rightarrow & pc_2' = 0,\ x_1' \ ?\ x_2' \\
pc_1 = 2 \wedge pc_2 = 2 & \rightarrow & \text{ERROR}
\end{array}$$

Abstract reachability graph



$0, 0, >$ $1, 0, >$ $2, 0, >$ $0, 0, =$ $1, 0, =$ $2, 0, =$ $0, 0, <$ $1, 0, <$ $2, 0, <$

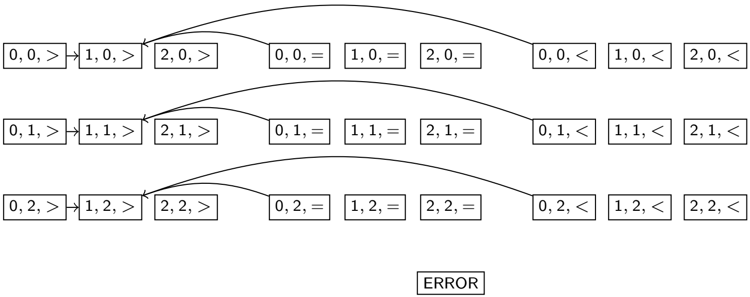
$0, 1, >$ $1, 1, >$ $2, 1, >$ $0, 1, =$ $1, 1, =$ $2, 1, =$ $0, 1, <$ $1, 1, <$ $2, 1, <$

$0, 2, >$ $1, 2, >$ $2, 2, >$ $0, 2, =$ $1, 2, =$ $2, 2, =$ $0, 2, <$ $1, 2, <$ $2, 2, <$

ERROR

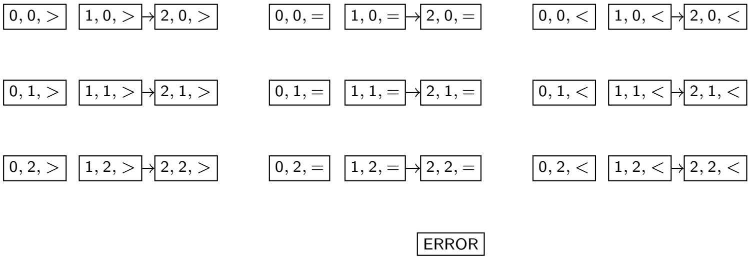
Abstract reachability graph

$$pc_1 = 0 \quad \rightarrow \quad pc_1' = 1,\ x_1' > x_2'$$



Abstract reachability graph

$$pc_1 = 1 \wedge (? \vee x_1 < x_2) \quad \rightarrow \quad pc_1' = 2$$



$0, 0, >$ $1, 0, > \rightarrow 2, 0, >$ $0, 0, =$ $1, 0, = \rightarrow 2, 0, =$ $0, 0, <$ $1, 0, < \rightarrow 2, 0, <$

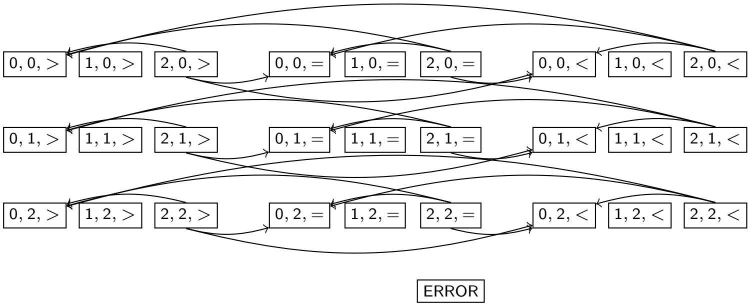
$0, 1, >$ $1, 1, > \rightarrow 2, 1, >$ $0, 1, =$ $1, 1, = \rightarrow 2, 1, =$ $0, 1, <$ $1, 1, < \rightarrow 2, 1, <$

$0, 2, >$ $1, 2, > \rightarrow 2, 2, >$ $0, 2, =$ $1, 2, = \rightarrow 2, 2, =$ $0, 2, <$ $1, 2, < \rightarrow 2, 2, <$

ERROR

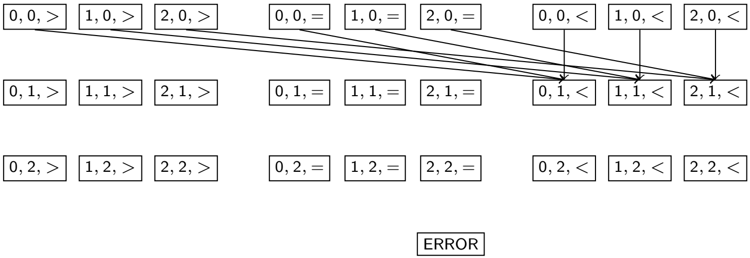
Abstract reachability graph

$$pc_1 = 2 \quad \rightarrow \quad pc_1' = 0,\ x_1' \ ?\ x_2'$$



Abstract reachability graph

$$pc_2 = 0 \quad \rightarrow \quad pc_2' = 1,\ x_1' < x_2'$$



$0, 0, >$ $1, 0, >$ $2, 0, >$ $0, 0, =$ $1, 0, =$ $2, 0, =$ $0, 0, <$ $1, 0, <$ $2, 0, <$

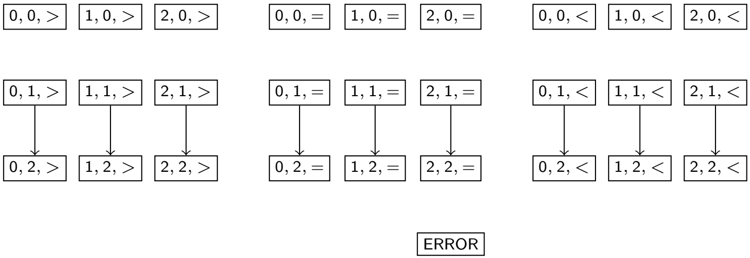
$0, 1, >$ $1, 1, >$ $2, 1, >$ $0, 1, =$ $1, 1, =$ $2, 1, =$ $0, 1, <$ $1, 1, <$ $2, 1, <$

$0, 2, >$ $1, 2, >$ $2, 2, >$ $0, 2, =$ $1, 2, =$ $2, 2, =$ $0, 2, <$ $1, 2, <$ $2, 2, <$

ERROR

Abstract reachability graph

$$pc_2 = 1 \land (? \lor x_2 < x_1) \quad \rightarrow \quad pc_2' = 2$$



$0, 0, >$ $1, 0, >$ $2, 0, >$ $0, 0, =$ $1, 0, =$ $2, 0, =$ $0, 0, <$ $1, 0, <$ $2, 0, <$

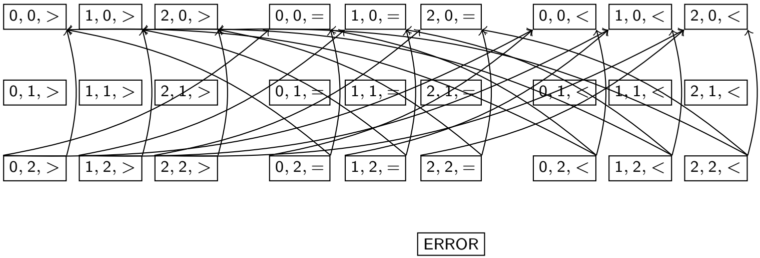
$0, 1, >$ $1, 1, >$ $2, 1, >$ $0, 1, =$ $1, 1, =$ $2, 1, =$ $0, 1, <$ $1, 1, <$ $2, 1, <$

$0, 2, >$ $1, 2, >$ $2, 2, >$ $0, 2, =$ $1, 2, =$ $2, 2, =$ $0, 2, <$ $1, 2, <$ $2, 2, <$

ERROR

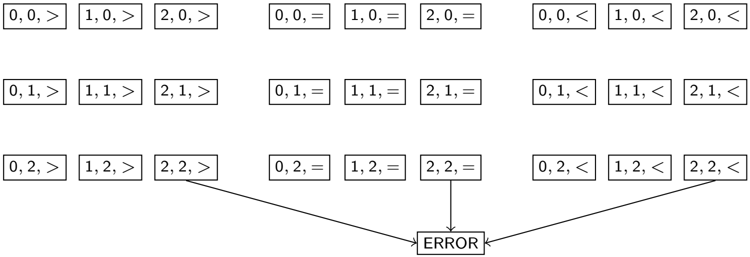
Abstract reachability graph

$$pc_1 = 2 \quad \rightarrow \quad pc_1' = 0,\ x_1' \ ?\ x_2'$$



Abstract reachability graph

$$pc_1 = 2 \land pc_2 = 2 \quad \rightarrow \quad \text{ERROR}$$



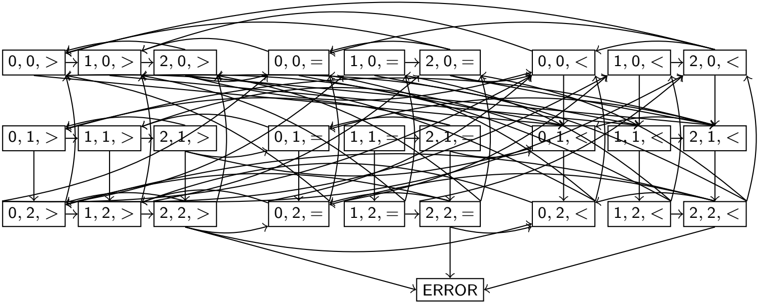
$0, 0, >$ $1, 0, >$ $2, 0, >$ $0, 0, =$ $1, 0, =$ $2, 0, =$ $0, 0, <$ $1, 0, <$ $2, 0, <$

$0, 1, >$ $1, 1, >$ $2, 1, >$ $0, 1, =$ $1, 1, =$ $2, 1, =$ $0, 1, <$ $1, 1, <$ $2, 1, <$

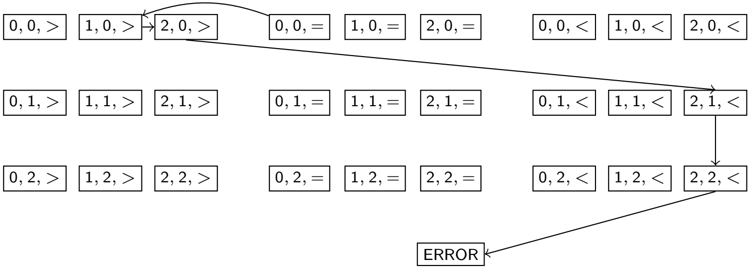
$0, 2, >$ $1, 2, >$ $2, 2, >$ $0, 2, =$ $1, 2, =$ $2, 2, =$ $0, 2, <$ $1, 2, <$ $2, 2, <$

ERROR

Abstract reachability graph



Abstract reachability graph



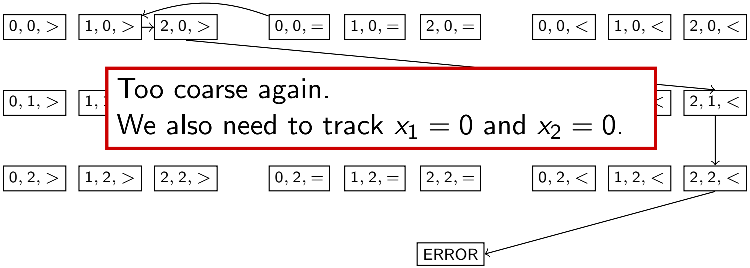
$0, 0, >$ $1, 0, >$ $2, 0, >$ $0, 0, =$ $1, 0, =$ $2, 0, =$ $0, 0, <$ $1, 0, <$ $2, 0, <$

$0, 1, >$ $1, 1, >$ $2, 1, >$ $0, 1, =$ $1, 1, =$ $2, 1, =$ $0, 1, <$ $1, 1, <$ $2, 1, <$

$0, 2, >$ $1, 2, >$ $2, 2, >$ $0, 2, =$ $1, 2, =$ $2, 2, =$ $0, 2, <$ $1, 2, <$ $2, 2, <$

ERROR

Abstract reachability graph

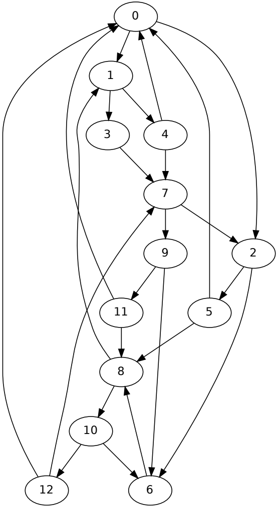


Too coarse again.

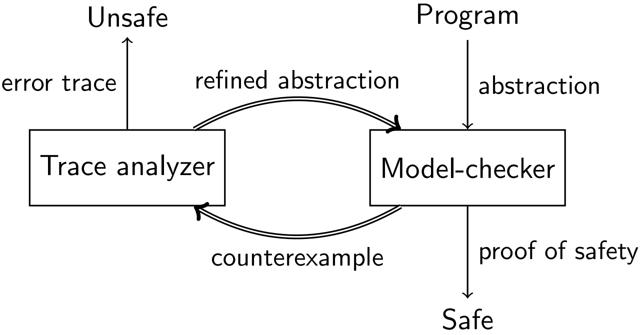
We also need to track $x_1 = 0$ and $x_2 = 0$.

Safe abstract reachability graph

id	pc1	pc2	$x_1 = 0$	$x_2 = 0$	$x_1 ? x_2$
0	0	0	$\top$	$\top$	$x_1 = x_2$
1	1	0	$\bot$	$\top$	$x_1 > x_2$
2	0	1	$\top$	$\bot$	$x_1 < x_2$
3	1	1	$\bot$	$\bot$	$x_1 < x_2$
4	2	0	$\bot$	$\top$	$x_1 > x_2$
5	0	2	$\top$	$\bot$	$x_1 < x_2$
6	1	1	$\bot$	$\bot$	$x_1 > x_2$
7	2	1	$\bot$	$\bot$	$x_1 < x_2$
8	1	2	$\bot$	$\bot$	$x_1 > x_2$
9	0	1	$\top$	$\bot$	$x_1 > x_2$
10	1	0	$\bot$	$\top$	$x_1 < x_2$
11	0	2	$\top$	$\bot$	$x_1 > x_2$
12	2	0	$\bot$	$\top$	$x_1 < x_2$



General idea:



CEGAR: counterexample guided abstraction refinement

Questions ?

Next part: The magic behind “*finding new facts*”

(hard hat required)

First iteration

predicates:

initial state:

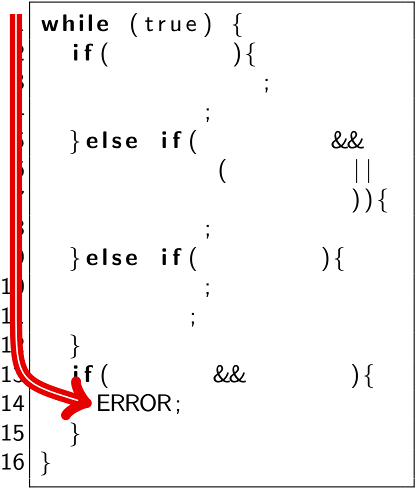
```
while (true) {
    if(              ){
                              ;
                        ;
    }else if(                &&
                   (               ||
                                     )){
                        ;
    }else if(              ){
                        ;
                      ;
    }
    if(            &&             ){
        ERROR;
    }
}
```

```
while (true) {
    if(              ){
                              ;
                        ;
    }else if(                &&
                   (               ||
                                     )){
                        ;
    }else if(              ){
                        ;
                      ;
    }
    if(            &&             ){
        ERROR;
    }
}
```

First iteration

predicates:

initial state:



```
while (true) {
  if (            ){
                      ;
                ;
  }else if (              &&
             (            ||
                          )){
                ;
  }else if (            ){
                ;
              ;
  }
  if (        &&          ){
    ERROR;
  }
}
```

```
while (true) {
  if (            ){
                      ;
                ;
  }else if (              &&
             (            ||
                          )){
                ;
  }else if (            ){
                ;
              ;
  }
  if (        &&          ){
    ERROR;
  }
}
```

First iteration

predicates:

initial state:

```
while (true) {
  if(            ){
                 ;
                ;
  }else if(

                ;
  }else if(
                ;
             ;
  }
  if(        &&        ){
    ERROR;
  }
}
```

```
pc1 = 0, x1 = 0, pc2 = 0, x2 = 0;
while (true) {
  if(pc1==2 && pc2==2){
    ERROR;
  }
}
```

```
while (true) {
  if(            ){
                 ;

                                      &&
                                        ||
                                        )){

                                    ){


  }
  if(        &&        ){
    ERROR;
  }
}
```

First counterexample

```
pc1=0, x1=0, pc2=0, x2=0;
assume(pc1==2 && pc2==2);
ERROR;
```

SSA formula:

$$pc_1 = 0 \wedge x_1 = 0 \wedge pc_2 = 0 \wedge x_2 = 0$$
$$pc_1 = 2 \wedge pc_2 = 2$$

First counterexample

```
 pc1=0, x1=0, pc2=0, x2=0;
 assume(pc1==2 && pc2==2);
ERROR;
```

SSA formula:

$$pc_1 = 0 \wedge x_1 = 0 \wedge pc_2 = 0 \wedge x_2 = 0$$
$$pc_1 = 2 \wedge pc_2 = 2$$

Formula is unsat ⇒ spurious counterexample

Finding out why the cex is spurious.

Let A and B be two formulas such that $A \wedge B$ unsat.
A [Craig] interpolant I has the following properties:

- I contains only AB-common symbols.
- A implies I
- $I \wedge B$ unsat.

$$pc_1 = 0 \wedge x_1 = 0 \wedge pc_2 = 0 \wedge x_2 = 0$$

$$pc_1 = 2 \wedge pc_2 = 2$$

Finding out why the cex is spurious.

Let A and B be two formulas such that $A \wedge B$ unsat.
A [Craig] interpolant I has the following properties:

- I contains only AB-common symbols.
- A implies I
- $I \wedge B$ unsat.

$$pc_1 = 0 \wedge x_1 = 0 \wedge pc_2 = 0 \wedge x_2 = 0$$

$$pc_1 = 0$$

$$pc_1 = 2 \wedge pc_2 = 2$$

Second iteration

predicates: $pc1 = 0$

initial state: $pc1 = 0$

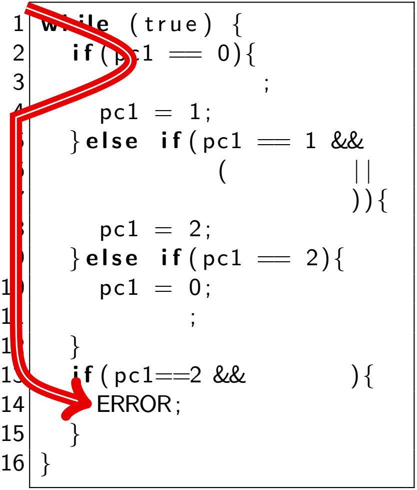
```
while (true) {
  if(pc1 == 0){
               ;
    pc1 = 1;
  }else if(pc1 == 1 &&
            (         ||
                     )){
    pc1 = 2;
  }else if(pc1 == 2){
    pc1 = 0;
          ;
  }
  if(pc1==2 &&         ){
    ERROR;
  }
}
```

```
while (true) {
  if(          ){
               ;
           ;
  }else if(           &&
            (         ||
                     )){
           ;
  }else if(         ){
           ;
          ;
  }
  if(pc1==2 &&         ){
    ERROR;
  }
}
```

Second iteration

predicates: pc1 $= 0$

initial state: pc1 $= 0$



```
while (true) {
  if(pc1 == 0){
                         ;
    pc1 = 1;
  }else if(pc1 == 1 &&
                (            ||
                              )){
    pc1 = 2;
  }else if(pc1 == 2){
    pc1 = 0;
              ;
  }
  if(pc1==2 &&               ){
    ERROR;
  }
}
```

```
while (true) {
  if(            ){
                         ;
                  ;
  }else if(              &&
                (            ||
                              )){
                  ;
  }else if(            ){
                  ;
                ;
  }
  if(pc1==2 &&               ){
    ERROR;
  }
}
```

Second counterexample

```
pc1=0, x1=0, pc2=0, x2=0;
assume(pc1 == 0);
x1 = x2 + 1;
pc1 = 1;
assume(pc1==2 && pc2==2);
ERROR;
```

SSA formula:

$$
\begin{aligned}
&pc_1 = 0 \wedge x_1 = 0 \wedge pc_2 = 0 \wedge x_2 = 0\\
&pc_1 = 0\\
&x_1' = x_2 + 1\\
&pc_1' = 1\\
&pc_1' = 2 \wedge pc_2 = 2
\end{aligned}
$$

Second counterexample

```
pc1=0, x1=0, pc2=0, x2=0;
assume(pc1 == 0);
x1 = x2 + 1;
pc1 = 1;
assume(pc1==2 && pc2==2);
ERROR;
```

SSA formula:

$$
\begin{aligned}
&pc_1 = 0 \land x_1 = 0 \land pc_2 = 0 \land x_2 = 0\\
&pc_1 = 0\\
&x_1' = x_2 + 1\\
&pc_1' = 1\\
&pc_1' = 2 \land pc_2 = 2
\end{aligned}
$$

Formula is unsat ⇒ spurious counterexample

Finding out why the cex is spurious.

$$pc_1 = 0 \wedge x_1 = 0 \wedge pc_2 = 0 \wedge x_2 = 0$$

$$pc_1 = 0$$

$$x_1' = x_2 + 1$$

$$pc_1' = 1$$

$$pc_1' = 2 \wedge pc_2 = 2$$

Finding out why the cex is spurious.

$$pc_1 = 0 \wedge x_1 = 0 \wedge pc_2 = 0 \wedge x_2 = 0$$

$$\top$$

$$pc_1 = 0$$

$$x_1' = x_2 + 1$$

$$pc_1' = 1$$

$$pc_1' = 2 \wedge pc_2 = 2$$

Finding out why the cex is spurious.

$$pc_1 = 0 \land x_1 = 0 \land pc_2 = 0 \land x_2 = 0$$

$$\top$$

$$pc_1 = 0$$

$$\top$$

$$x_1' = x_2 + 1$$

$$pc_1' = 1$$

$$pc_1' = 2 \land pc_2 = 2$$

Finding out why the cex is spurious.

$$pc_1 = 0 \land x_1 = 0 \land pc_2 = 0 \land x_2 = 0$$

$$\top$$

$$pc_1 = 0$$

$$\top$$

$$x_1' = x_2 + 1$$

$$\top$$

$$pc_1' = 1$$

$$pc_1' = 2 \land pc_2 = 2$$

Finding out why the cex is spurious.

$pc_1 = 0 \wedge x_1 = 0 \wedge pc_2 = 0 \wedge x_2 = 0$

$\top$

$pc_1 = 0$

$\top$

$x_1' = x_2 + 1$

$\top$

$pc_1' = 1$

$pc_1' = 1$

$pc_1' = 2 \wedge pc_2 = 2$

Third iteration

predicates: pc1 = 0, pc1 = 1

initial state: pc1 = 0

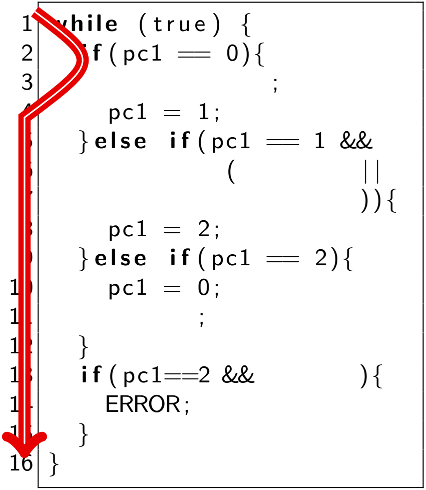
```
while (true) {
  if(pc1 == 0){
                  ;
    pc1 = 1;
  }else if(pc1 == 1 &&
              (          ||
                         )){
    pc1 = 2;
  }else if(pc1 == 2){
    pc1 = 0;
            ;
  }
  if(pc1==2 &&           ){
    ERROR;
  }
}
```

```
while (true) {
  if(           ){
                  ;
             ;
  }else if(             &&
              (          ||
                         )){
             ;
  }else if(            ){
             ;
            ;
  }
  if(pc1==2 &&           ){
    ERROR;
  }
}
```

Third iteration

predicates: pc1 $=$ 0, pc1 $=$ 1

initial state: pc1 $=$ 0



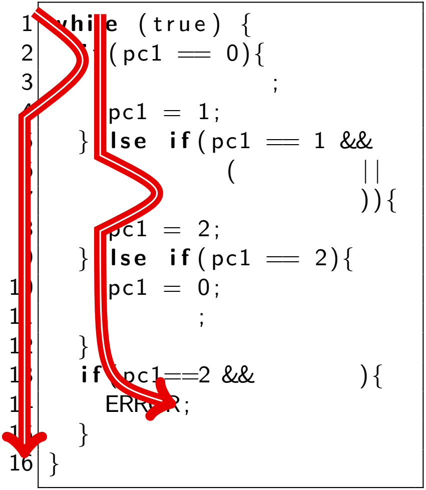
```
while (true) {
  if(pc1 == 0){
                    ;
    pc1 = 1;
  }else if(pc1 == 1 &&
             (         ||
                        )){
    pc1 = 2;
  }else if(pc1 == 2){
    pc1 = 0;
          ;
  }
  if(pc1==2 &&         ){
    ERROR;
  }
}
```

```
while (true) {
  if(           ){
                    ;
                ;
  }else if(            &&
             (         ||
                        )){
                ;
  }else if(          ){
                ;
              ;
  }
  if(pc1==2 &&         ){
    ERROR;
  }
}
```

Third iteration

predicates: $pc1 = 0$, $pc1 = 1$

initial state: $pc1 = 0$



```
while (true) {
  if (pc1 == 0){
                        ;
    pc1 = 1;
  } else if (pc1 == 1 &&
              (          ||
                          )){
    pc1 = 2;
  } else if (pc1 == 2){
    pc1 = 0;
              ;
  }
  if (pc1==2 &&          ){
    ERROR;
  }
}
```

```
while (true) {
  if (           ){
                        ;
                    ;
  }else if (           &&
              (          ||
                          )){
                    ;
  }else if (         ){
                    ;
                  ;
  }
  if (pc1==2 &&          ){
    ERROR;
  }
}
```

Third counterexample

```
pc1=0, x1=0, pc2=0, x2=0;
assume(pc1 == 0);
x1 = x2 + 1;
pc1 = 1;
assume(pc1==1 && (x1==0 || x2<x1));
pc1 = 2;
assume(pc1==2 && pc2==2);
ERROR;
```

$$pc_1 = 0 \land x_1 = 0 \land pc_2 = 0 \land x_2 = 0$$

$$pc_1 = 0$$

$$x_1' = x_2 + 1$$

$$pc_1' = 1$$

$$pc_1' = 1 \land (x_2 = 0 \lor x_1' < x_2)$$

$$pc_1'' = 2$$

$$pc_1'' = 2 \land pc_2 = 2$$

Third counterexample

```
pc1=0, x1=0, pc2=0, x2=0;
assume(pc1 == 0);
x1 = x2 + 1;
pc1 = 1;
assume(pc1==1 && (x1==0 || x2<x1));
pc1 = 2;
assume(pc1==2 && pc2==2);
ERROR;
```

$$
\begin{aligned}
& pc_1 = 0 \wedge x_1 = 0 \wedge pc_2 = 0 \wedge x_2 = 0 \\
& pc_1 = 0 \\
& x_1' = x_2 + 1 \\
& pc_1' = 1 \\
& pc_1' = 1 \wedge (x_2 = 0 \vee x_1' < x_2) \\
& pc_1'' = 2 \\
& pc_1'' = 2 \wedge pc_2 = 2
\end{aligned}
$$

A few iterations later ...

predicates: pc1 = 0, pc1 = 1, pc2 = 0, pc2 = 1

initial state: pc1 = 0, pc2 = 0

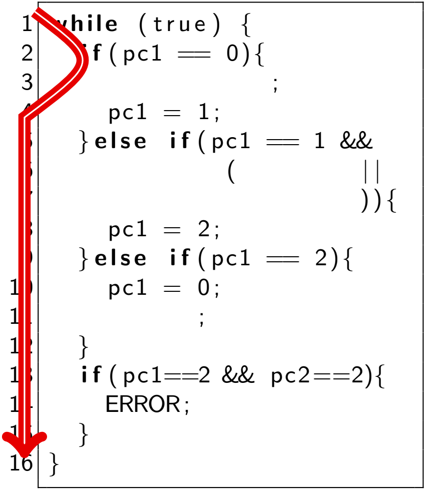
```
while (true) {
  if(pc1 == 0){
                    ;
    pc1 = 1;
  }else if(pc1 == 1 &&
              (           ||
                          )){
    pc1 = 2;
  }else if(pc1 == 2){
    pc1 = 0;
          ;
  }
  if(pc1==2 && pc2==2){
    ERROR;
  }
}
```

```
while (true) {
  if(pc2 == 0){
                    ;
    pc2 = 1;
  }else if(pc2 == 1 &&
              (           ||
                          )){
    pc2 = 2;
  }else if(pc2 == 2){
    pc2 = 0;
          ;
  }
  if(pc1==2 && pc2==2){
    ERROR;
  }
}
```

A few iterations later ...

predicates: pc1 $= 0$, pc1 $= 1$, pc2 $= 0$, pc2 $= 1$

initial state: pc1 $= 0$, pc2 $= 0$



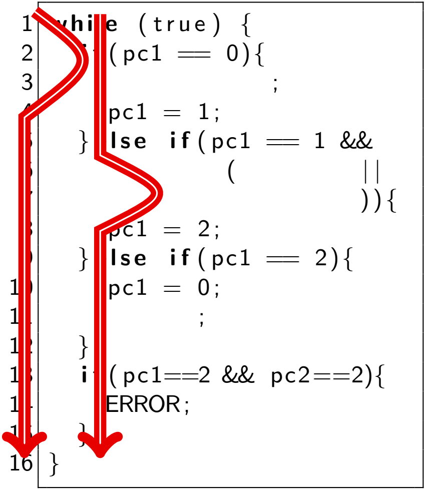
```
while (true) {
  if(pc1 == 0){
                    ;
    pc1 = 1;
  }else if(pc1 == 1 &&
           (          ||
                      )){
    pc1 = 2;
  }else if(pc1 == 2){
    pc1 = 0;
        ;
  }
  if(pc1==2 && pc2==2){
    ERROR;
  }
}
```

```
while (true) {
  if(pc2 == 0){
                    ;
    pc2 = 1;
  }else if(pc2 == 1 &&
           (          ||
                      )){
    pc2 = 2;
  }else if(pc2 == 2){
    pc2 = 0;
        ;
  }
  if(pc1==2 && pc2==2){
    ERROR;
  }
}
```

A few iterations later ...

predicates: pc1 = 0, pc1 = 1, pc2 = 0, pc2 = 1

initial state: pc1 = 0, pc2 = 0



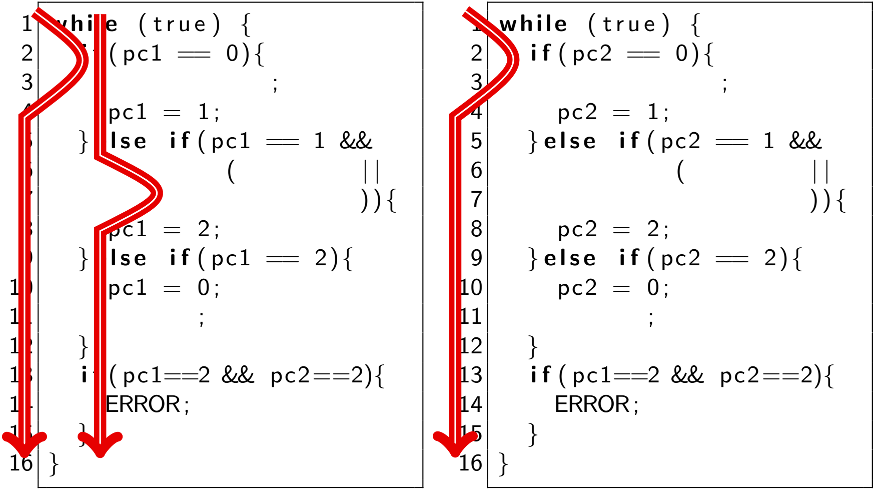
```
while (true) {
  if(pc1 == 0){
                      ;
    pc1 = 1;
  }else if(pc1 == 1 &&
           (            ||
                          )){
    pc1 = 2;
  }else if(pc1 == 2){
    pc1 = 0;
             ;
  }
  if(pc1==2 && pc2==2){
    ERROR;
  }
}
```

```
while (true) {
  if(pc2 == 0){
                      ;
    pc2 = 1;
  }else if(pc2 == 1 &&
           (            ||
                          )){
    pc2 = 2;
  }else if(pc2 == 2){
    pc2 = 0;
             ;
  }
  if(pc1==2 && pc2==2){
    ERROR;
  }
}
```

A few iterations later ...

predicates: $pc1 = 0$, $pc1 = 1$, $pc2 = 0$, $pc2 = 1$

initial state: $pc1 = 0$, $pc2 = 0$



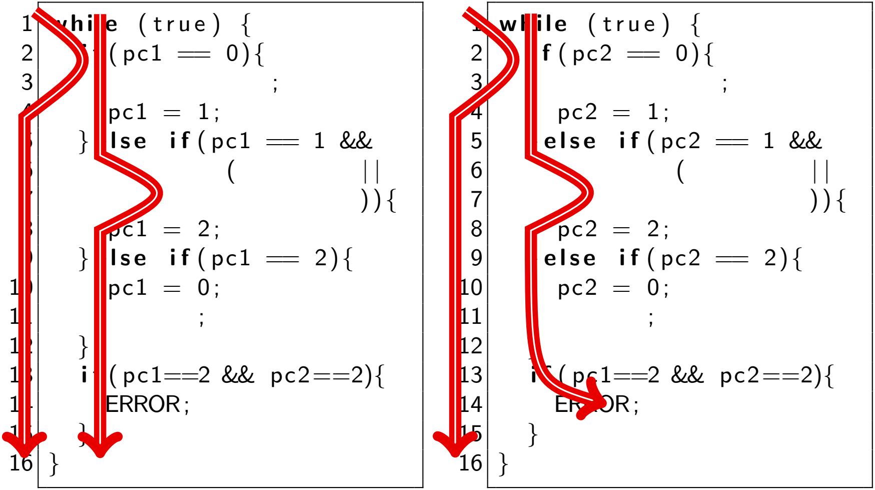
```
while (true) {
  if(pc1 == 0){
                          ;
    pc1 = 1;
  }else if(pc1 == 1 &&
              (                ||
                                )){
    pc1 = 2;
  }else if(pc1 == 2){
    pc1 = 0;
          ;
  }
  if(pc1==2 && pc2==2){
    ERROR;
  }
}
```

```
while (true) {
  if(pc2 == 0){
                          ;
    pc2 = 1;
  }else if(pc2 == 1 &&
              (                ||
                                )){
    pc2 = 2;
  }else if(pc2 == 2){
    pc2 = 0;
          ;
  }
  if(pc1==2 && pc2==2){
    ERROR;
  }
}
```

A few iterations later ...

predicates: $pc1 = 0$, $pc1 = 1$, $pc2 = 0$, $pc2 = 1$

initial state: $pc1 = 0$, $pc2 = 0$



```
while (true) {
  if(pc1 == 0){
                        ;
    pc1 = 1;
  } else if(pc1 == 1 &&
            (              ||
                            )){
    pc1 = 2;
  } else if(pc1 == 2){
    pc1 = 0;
          ;
  }
  if(pc1==2 && pc2==2){
    ERROR;
  }
}
```

```
while (true) {
  if(pc2 == 0){
                        ;
    pc2 = 1;
  } else if(pc2 == 1 &&
            (              ||
                            )){
    pc2 = 2;
  } else if(pc2 == 2){
    pc2 = 0;
          ;
  }
  if(pc1==2 && pc2==2){
    ERROR;
  }
}
```

Finding out why the cex is spurious (first possibility)

$$pc_1 = 0 \wedge x_1 = 0 \wedge pc_2 = 0 \wedge x_2 = 0$$

$$pc_1 = 0 \wedge x_1' = x_2 + 1 \wedge pc_1' = 1$$

$$pc_1' = 1 \wedge (x_2 = 0 \vee x_1' < x_2) \wedge pc_1'' = 2$$

$$pc_2 = 0 \wedge x_2' = x_1' + 1 \wedge pc_2' = 1$$

$$pc_2' = 1 \wedge (x_1' = 0 \vee x_2' < x_1') \wedge pc_2'' = 2$$

$$pc_1'' = 2 \wedge pc_2'' = 2$$

Finding out why the cex is spurious (first possibility)

$$pc_1 = 0 \land x_1 = 0 \land pc_2 = 0 \land x_2 = 0$$

$$x_2 = 0$$

$$pc_1 = 0 \land x_1' = x_2 + 1 \land pc_1' = 1$$

$$pc_1' = 1 \land (x_2 = 0 \lor x_1' < x_2) \land pc_1'' = 2$$

$$pc_2 = 0 \land x_2' = x_1' + 1 \land pc_2' = 1$$

$$pc_2' = 1 \land (x_1' = 0 \lor x_2' < x_1') \land pc_2'' = 2$$

$$pc_1'' = 2 \land pc_2'' = 2$$

Finding out why the cex is spurious (first possibility)

$$pc_1 = 0 \land x_1 = 0 \land pc_2 = 0 \land x_2 = 0$$

$$x_2 = 0$$

$$pc_1 = 0 \land x_1' = x_2 + 1 \land pc_1' = 1$$

$$x_1' = 1 \land x_2 = 0$$

$$pc_1' = 1 \land (x_2 = 0 \lor x_1' < x_2) \land pc_1'' = 2$$

$$pc_2 = 0 \land x_2' = x_1' + 1 \land pc_2' = 1$$

$$pc_2' = 1 \land (x_1' = 0 \lor x_2' < x_1') \land pc_2'' = 2$$

$$pc_1'' = 2 \land pc_2'' = 2$$

Finding out why the cex is spurious (first possibility)

$$pc_1 = 0 \wedge x_1 = 0 \wedge pc_2 = 0 \wedge x_2 = 0$$

$$x_2 = 0$$

$$pc_1 = 0 \wedge x_1' = x_2 + 1 \wedge pc_1' = 1$$

$$x_1' = 1 \wedge x_2 = 0$$

$$pc_1' = 1 \wedge (x_2 = 0 \vee x_1' < x_2) \wedge pc_1'' = 2$$

$$x_1' = 1$$

$$pc_2 = 0 \wedge x_2' = x_1' + 1 \wedge pc_2' = 1$$

$$pc_2' = 1 \wedge (x_1' = 0 \vee x_2' < x_1') \wedge pc_2'' = 2$$

$$pc_1'' = 2 \wedge pc_2'' = 2$$

Finding out why the cex is spurious (first possibility)

$$pc_1 = 0 \wedge x_1 = 0 \wedge pc_2 = 0 \wedge x_2 = 0$$

$$pc_1 = 0 \wedge x_1' = x_2 + 1 \wedge pc_1' = 1$$

$$pc_1' = 1 \wedge (x_2 = 0 \vee x_1' < x_2) \wedge pc_1'' = 2$$

$$pc_2 = 0 \wedge x_2' = x_1' + 1 \wedge pc_2' = 1$$

$$pc_2' = 1 \wedge (x_1' = 0 \vee x_2' < x_1') \wedge pc_2'' = 2$$

$$pc_1'' = 2 \wedge pc_2'' = 2$$

$$x_2 = 0$$

$$x_1' = 1 \wedge x_2 = 0$$

$$x_1' = 1$$

$$x_1' = 1 \wedge x_2' = 2$$

Finding out why the cex is spurious (first possibility)

$$pc_1 = 0 \land x_1 = 0 \land pc_2 = 0 \land x_2 = 0$$

$$pc_1 = 0 \land x'_1 = x_2 + 1 \land pc'_1 = 1$$

$$pc'_1 = 1 \land (x_2 = 0 \lor x'_1 < x_2) \land pc''_1 = 2$$

$$pc_2 = 0 \land x'_2 = x'_1 + 1 \land pc'_2 = 1$$

$$pc'_2 = 1 \land (x'_1 = 0 \lor x'_2 < x'_1) \land pc''_2 = 2$$

$$pc''_1 = 2 \land pc''_2 = 2$$

$$x_2 = 0$$

$$x'_1 = 1 \land x_2 = 0$$

$$x'_1 = 1$$

$$x'_1 = 1 \land x'_2 = 2$$

$$\bot$$

Does it work ?

predicates: pc1 = 0, pc1 = 1, pc2 = 0, pc2 = 1, x1=1, x2=0, x2=2

initial state: pc1 = 0, x1 = 0, pc2 = 0, x2 = 0

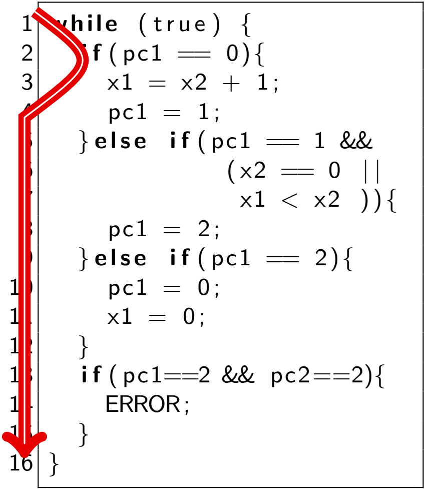
```
while (true) {
  if(pc1 == 0){
    x1 = x2 + 1;
    pc1 = 1;
  }else if(pc1 == 1 &&
               (x2 == 0 ||
                x1 < x2 )){
    pc1 = 2;
  }else if(pc1 == 2){
    pc1 = 0;
    x1 = 0;
  }
  if(pc1==2 && pc2==2){
    ERROR;
  }
}
```

```
while (true) {
  if(pc2 == 0){
    x2 = x1 + 1;
    pc2 = 1;
  }else if(pc2 == 1 &&
               (x1 == 0 ||
                x2 < x1 )){
    pc2 = 2;
  }else if(pc2 == 2){
    pc2 = 0;
    x2 = 0;
  }
  if(pc1==2 && pc2==2){
    ERROR;
  }
}
```

Does it work ?

predicates: pc1 = 0, pc1 = 1, pc2 = 0, pc2 = 1, x1=1, x2=0, x2=2

initial state: pc1 = 0, x1 = 0, pc2 = 0, x2 = 0



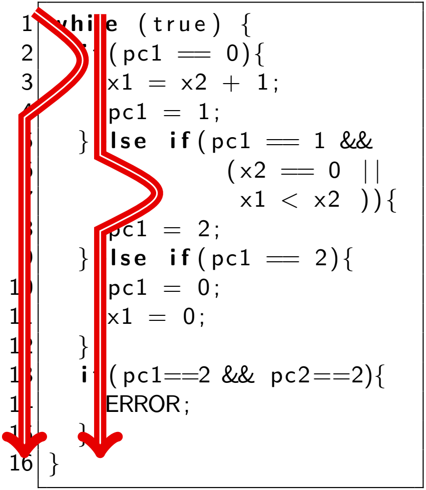
```
while (true) {
  if(pc1 == 0){
    x1 = x2 + 1;
    pc1 = 1;
  }else if(pc1 == 1 &&
             (x2 == 0 ||
              x1 < x2 )){
    pc1 = 2;
  }else if(pc1 == 2){
    pc1 = 0;
    x1 = 0;
  }
  if(pc1==2 && pc2==2){
    ERROR;
  }
}
```

```
while (true) {
  if(pc2 == 0){
    x2 = x1 + 1;
    pc2 = 1;
  }else if(pc2 == 1 &&
             (x1 == 0 ||
              x2 < x1 )){
    pc2 = 2;
  }else if(pc2 == 2){
    pc2 = 0;
    x2 = 0;
  }
  if(pc1==2 && pc2==2){
    ERROR;
  }
}
```

Does it work ?

predicates: pc1 = 0, pc1 = 1, pc2 = 0, pc2 = 1, x1=1, x2=0, x2=2

initial state: pc1 = 0, x1 = 0, pc2 = 0, x2 = 0



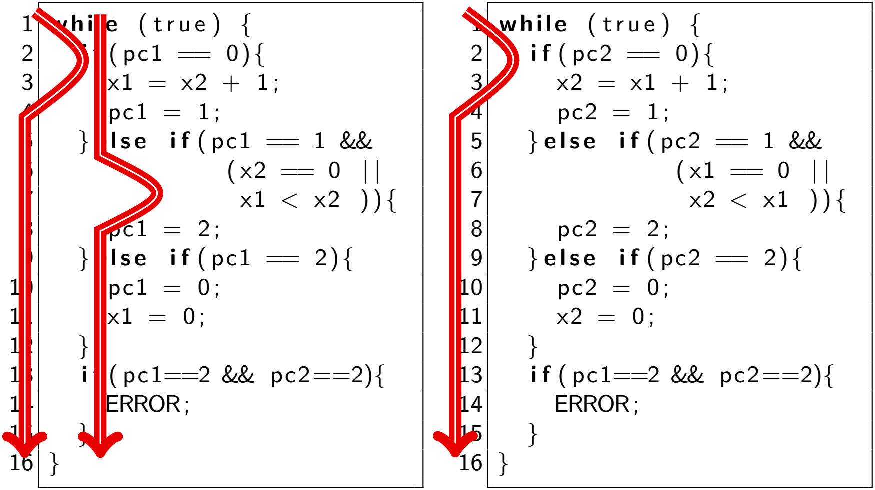
```
while (true) {
  if(pc1 == 0){
    x1 = x2 + 1;
    pc1 = 1;
  }else if(pc1 == 1 &&
           (x2 == 0 ||
            x1 < x2 )){
    pc1 = 2;
  }else if(pc1 == 2){
    pc1 = 0;
    x1 = 0;
  }
  if(pc1==2 && pc2==2){
    ERROR;
  }
}
```

```
while (true) {
  if(pc2 == 0){
    x2 = x1 + 1;
    pc2 = 1;
  }else if(pc2 == 1 &&
           (x1 == 0 ||
            x2 < x1 )){
    pc2 = 2;
  }else if(pc2 == 2){
    pc2 = 0;
    x2 = 0;
  }
  if(pc1==2 && pc2==2){
    ERROR;
  }
}
```

Does it work ?

predicates: pc1 = 0, pc1 = 1, pc2 = 0, pc2 = 1, x1=1, x2=0, x2=2

initial state: pc1 = 0, x1 = 0, pc2 = 0, x2 = 0



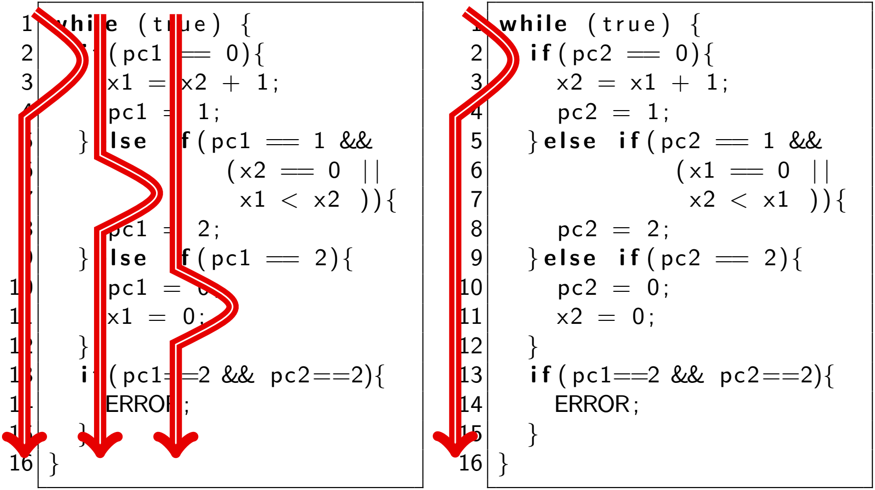
```
while (true) {
  if(pc1 == 0){
    x1 = x2 + 1;
    pc1 = 1;
  }else if(pc1 == 1 &&
           (x2 == 0 ||
            x1 < x2 )){
    pc1 = 2;
  }else if(pc1 == 2){
    pc1 = 0;
    x1 = 0;
  }
  if(pc1==2 && pc2==2){
    ERROR;
  }
}
```

```
while (true) {
  if(pc2 == 0){
    x2 = x1 + 1;
    pc2 = 1;
  }else if(pc2 == 1 &&
           (x1 == 0 ||
            x2 < x1 )){
    pc2 = 2;
  }else if(pc2 == 2){
    pc2 = 0;
    x2 = 0;
  }
  if(pc1==2 && pc2==2){
    ERROR;
  }
}
```

Does it work ?

predicates: pc1 = 0, pc1 = 1, pc2 = 0, pc2 = 1, x1=1, x2=0, x2=2

initial state: pc1 = 0, x1 = 0, pc2 = 0, x2 = 0



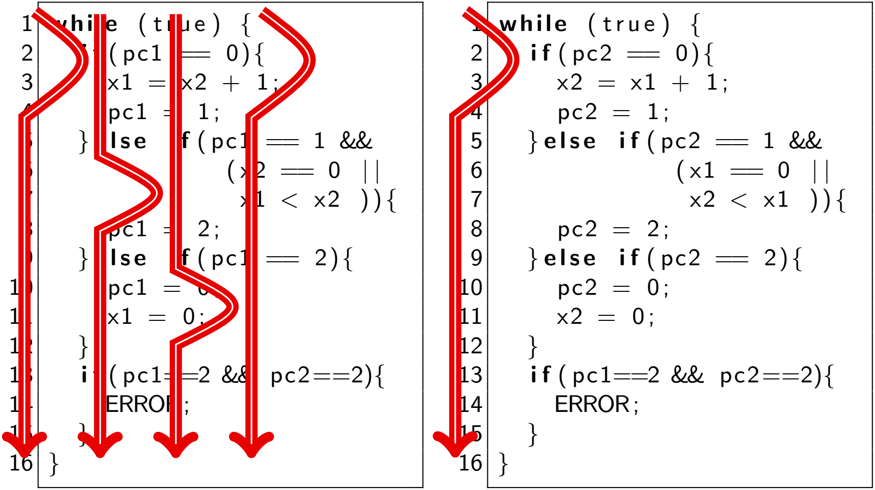
```
while (true) {
  if(pc1 == 0){
    x1 = x2 + 1;
    pc1 = 1;
  }else if(pc1 == 1 &&
          (x2 == 0 ||
           x1 < x2 )){
    pc1 = 2;
  }else if(pc1 == 2){
    pc1 = 0;
    x1 = 0;
  }
  if(pc1==2 && pc2==2){
    ERROR;
  }
}
```

```
while (true) {
  if(pc2 == 0){
    x2 = x1 + 1;
    pc2 = 1;
  }else if(pc2 == 1 &&
          (x1 == 0 ||
           x2 < x1 )){
    pc2 = 2;
  }else if(pc2 == 2){
    pc2 = 0;
    x2 = 0;
  }
  if(pc1==2 && pc2==2){
    ERROR;
  }
}
```

Does it work ?

predicates: pc1 = 0, pc1 = 1, pc2 = 0, pc2 = 1, x1=1, x2=0, x2=2

initial state: pc1 = 0, x1 = 0, pc2 = 0, x2 = 0



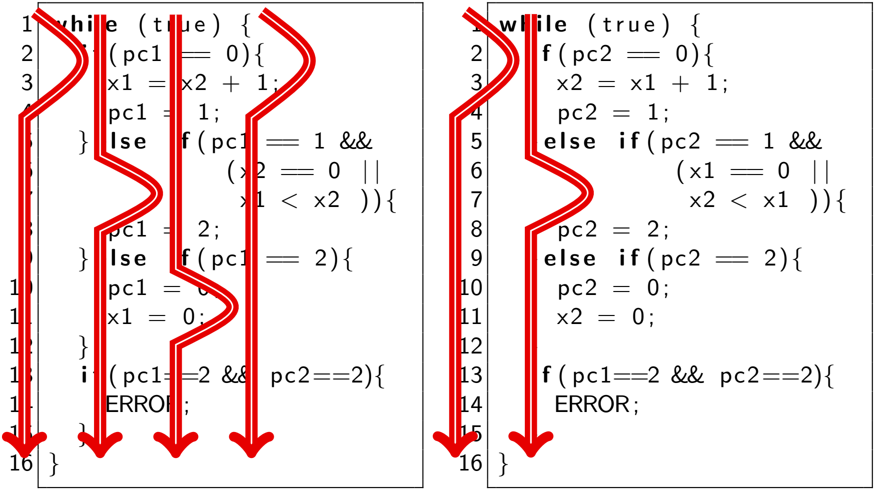
```
while (true) {
  if(pc1 == 0){
    x1 = x2 + 1;
    pc1 = 1;
  }else if(pc1 == 1 &&
           (x2 == 0 ||
            x1 < x2 )){
    pc1 = 2;
  }else if(pc1 == 2){
    pc1 = 0;
    x1 = 0;
  }
  if(pc1==2 && pc2==2){
    ERROR;
  }
}
```

```
while (true) {
  if(pc2 == 0){
    x2 = x1 + 1;
    pc2 = 1;
  }else if(pc2 == 1 &&
           (x1 == 0 ||
            x2 < x1 )){
    pc2 = 2;
  }else if(pc2 == 2){
    pc2 = 0;
    x2 = 0;
  }
  if(pc1==2 && pc2==2){
    ERROR;
  }
}
```

Does it work ?

predicates: pc1 = 0, pc1 = 1, pc2 = 0, pc2 = 1, x1=1, x2=0, x2=2

initial state: pc1 = 0, x1 = 0, pc2 = 0, x2 = 0



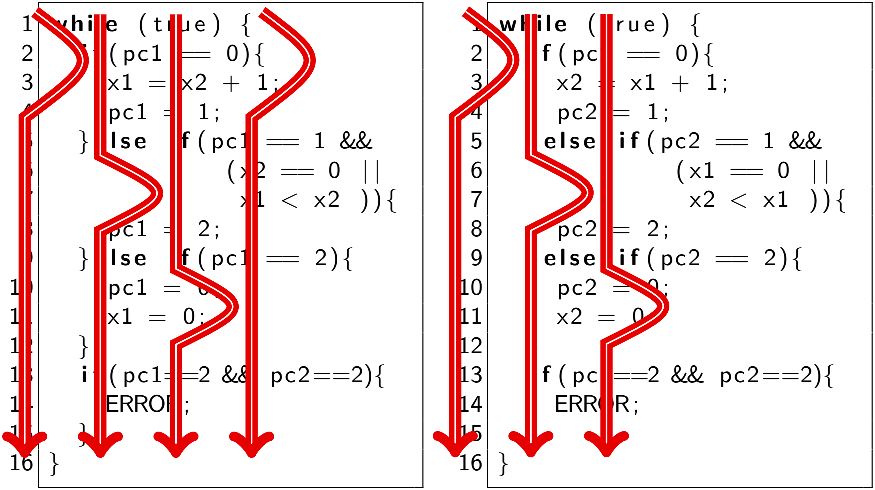
```
while (true) {
  if (pc1 == 0){
    x1 = x2 + 1;
    pc1 = 1;
  } else if (pc1 == 1 &&
             (x2 == 0 ||
              x1 < x2 )){
    pc1 = 2;
  } else if (pc1 == 2){
    pc1 = 0;
    x1 = 0;
  }
  if (pc1==2 && pc2==2){
    ERROR;
  }
}
```

```
while (true) {
  if (pc2 == 0){
    x2 = x1 + 1;
    pc2 = 1;
  } else if (pc2 == 1 &&
             (x1 == 0 ||
              x2 < x1 )){
    pc2 = 2;
  } else if (pc2 == 2){
    pc2 = 0;
    x2 = 0;
  }
  if (pc1==2 && pc2==2){
    ERROR;
  }
}
```

Does it work ?

predicates: pc1 = 0, pc1 = 1, pc2 = 0, pc2 = 1, x1=1, x2=0, x2=2

initial state: pc1 = 0, x1 = 0, pc2 = 0, x2 = 0



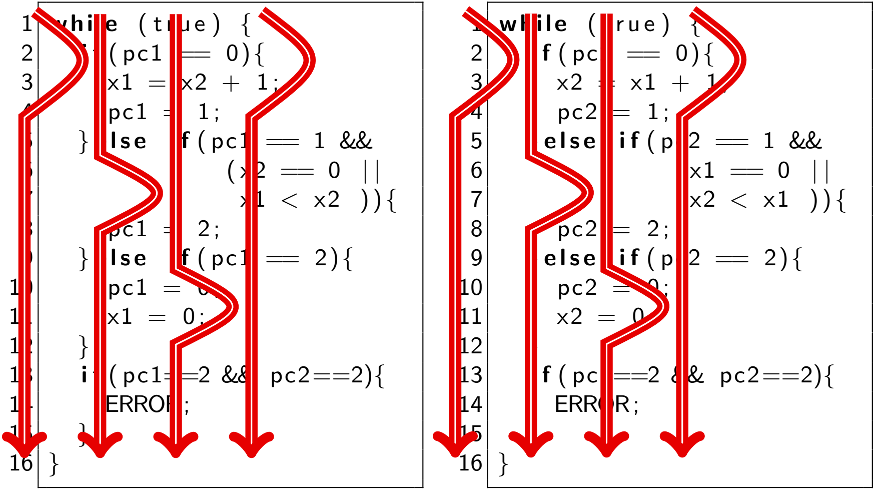
```
while (true) {
  if(pc1 == 0){
    x1 = x2 + 1;
    pc1 = 1;
  } else if(pc1 == 1 &&
           (x2 == 0 ||
            x1 < x2 )){
    pc1 = 2;
  } else if(pc1 == 2){
    pc1 = 0;
    x1 = 0;
  }
  if(pc1==2 && pc2==2){
    ERROR;
  }
}
```

```
while (true) {
  if(pc2 == 0){
    x2 = x1 + 1;
    pc2 = 1;
  } else if(pc2 == 1 &&
           (x1 == 0 ||
            x2 < x1 )){
    pc2 = 2;
  } else if(pc2 == 2){
    pc2 = 0;
    x2 = 0;
  }
  if(pc1==2 && pc2==2){
    ERROR;
  }
}
```

Does it work ?

predicates: pc1 = 0, pc1 = 1, pc2 = 0, pc2 = 1, x1=1, x2=0, x2=2

initial state: pc1 = 0, x1 = 0, pc2 = 0, x2 = 0



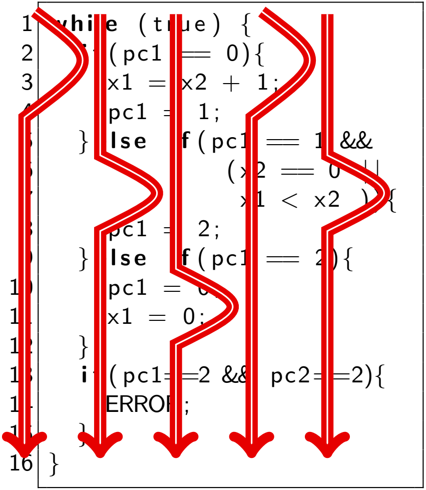
```
while (true) {
  if(pc1 == 0){
    x1 = x2 + 1;
    pc1 = 1;
  } else if(pc1 == 1 &&
            (x2 == 0 ||
             x1 < x2 )){
    pc1 = 2;
  } else if(pc1 == 2){
    pc1 = 0;
    x1 = 0;
  }
  if(pc1==2 && pc2==2){
    ERROR;
  }
}
```

```
while (true) {
  if(pc2 == 0){
    x2 = x1 + 1;
    pc2 = 1;
  } else if(pc2 == 1 &&
            (x1 == 0 ||
             x2 < x1 )){
    pc2 = 2;
  } else if(pc2 == 2){
    pc2 = 0;
    x2 = 0;
  }
  if(pc1==2 && pc2==2){
    ERROR;
  }
}
```

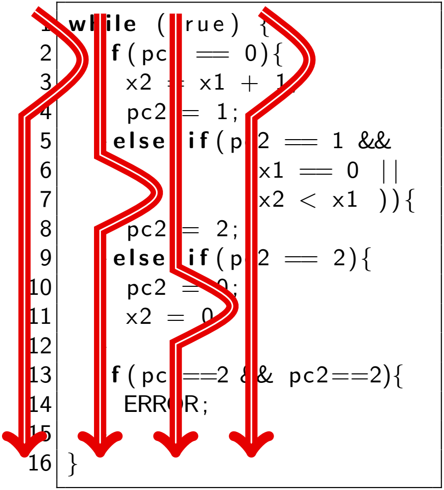
Does it work ?

predicates: pc1 = 0, pc1 = 1, pc2 = 0, pc2 = 1, x1=1, x2=0, x2=2

initial state: pc1 = 0, x1 = 0, pc2 = 0, x2 = 0



```
while (true) {
  if (pc1 == 0){
    x1 = x2 + 1;
    pc1 = 1;
  } else if (pc1 == 1 &&
             (x2 == 0 ||
              x1 < x2 )){
    pc1 = 2;
  } else if (pc1 == 2){
    pc1 = 0;
    x1 = 0;
  }
  if (pc1==2 && pc2==2){
    ERROR;
  }
}
```

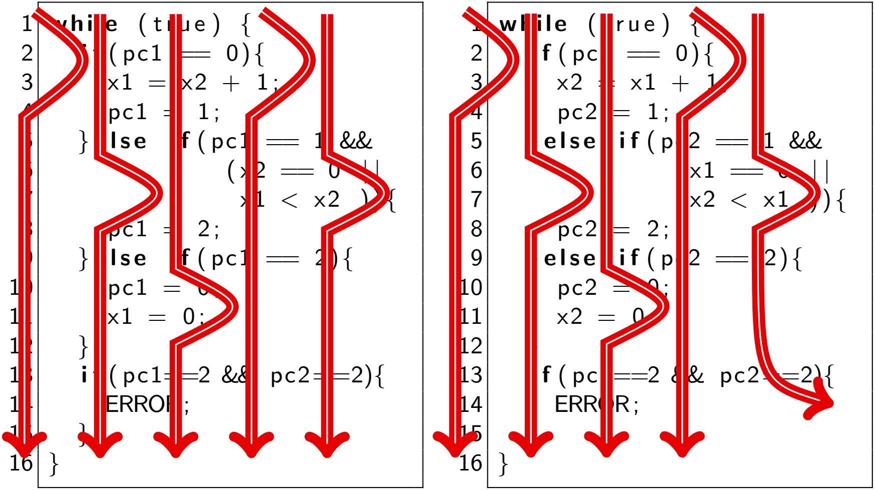


```
while (true) {
  if (pc2 == 0){
    x2 = x1 + 1;
    pc2 = 1;
  } else if (pc2 == 1 &&
             (x1 == 0 ||
              x2 < x1 )){
    pc2 = 2;
  } else if (pc2 == 2){
    pc2 = 0;
    x2 = 0;
  }
  if (pc1==2 && pc2==2){
    ERROR;
  }
}
```

Does it work ?

predicates: pc1 = 0, pc1 = 1, pc2 = 0, pc2 = 1, x1=1, x2=0, x2=2

initial state: pc1 = 0, x1 = 0, pc2 = 0, x2 = 0



```
while (true) {
  if(pc1 == 0){
    x1 = x2 + 1;
    pc1 = 1;
  } else if(pc1 == 1 &&
           (x2 == 0 ||
             x1 < x2 )){
    pc1 = 2;
  } else if(pc1 == 2){
    pc1 = 0;
    x1 = 0;
  }
  if(pc1==2 && pc2==2){
    ERROR;
  }
}
```

```
while (true) {
  if(pc2 == 0){
    x2 = x1 + 1;
    pc2 = 1;
  } else if(pc2 == 1 &&
           (x1 == 0 ||
             x2 < x1 )){
    pc2 = 2;
  } else if(pc2 == 2){
    pc2 = 0;
    x2 = 0;
  }
  if(pc1==2 && pc2==2){
    ERROR;
  }
}
```

Finding out why the cex is spurious (second possibility)

$$pc_1 = 0 \land x_1 = 0 \land pc_2 = 0 \land x_2 = 0$$

$$pc_1 = 0 \land x_1' = x_2 + 1 \land pc_1' = 1$$

$$pc_1' = 1 \land (x_2 = 0 \lor x_1' < x_2) \land pc_1'' = 2$$

$$pc_2 = 0 \land x_2' = x_1' + 1 \land pc_2' = 1$$

$$pc_2' = 1 \land (x_1' = 0 \lor x_2' < x_1') \land pc_2'' = 2$$

$$pc_1'' = 2 \land pc_2'' = 2$$

Finding out why the cex is spurious (second possibility)

$$pc_1 = 0 \land x_1 = 0 \land pc_2 = 0 \land x_2 = 0$$

$$x_2 = 0$$

$$pc_1 = 0 \land x_1' = x_2 + 1 \land pc_1' = 1$$

$$pc_1' = 1 \land (x_2 = 0 \lor x_1' < x_2) \land pc_1'' = 2$$

$$pc_2 = 0 \land x_2' = x_1' + 1 \land pc_2' = 1$$

$$pc_2' = 1 \land (x_1' = 0 \lor x_2' < x_1') \land pc_2'' = 2$$

$$pc_1'' = 2 \land pc_2'' = 2$$

Finding out why the cex is spurious (second possibility)

$$pc_1 = 0 \land x_1 = 0 \land pc_2 = 0 \land x_2 = 0$$

$$x_2 = 0$$

$$pc_1 = 0 \land x_1' = x_2 + 1 \land pc_1' = 1$$

$$x_1' > x_2 \land x_2 = 0$$

$$pc_1' = 1 \land (x_2 = 0 \lor x_1' < x_2) \land pc_1'' = 2$$

$$pc_2 = 0 \land x_2' = x_1' + 1 \land pc_2' = 1$$

$$pc_2' = 1 \land (x_1' = 0 \lor x_2' < x_1') \land pc_2'' = 2$$

$$pc_1'' = 2 \land pc_2'' = 2$$

Finding out why the cex is spurious (second possibility)

$$pc_1 = 0 \land x_1 = 0 \land pc_2 = 0 \land x_2 = 0$$

$$x_2 = 0$$

$$pc_1 = 0 \land x_1' = x_2 + 1 \land pc_1' = 1$$

$$x_1' > x_2 \land x_2 = 0$$

$$pc_1' = 1 \land (x_2 = 0 \lor x_1' < x_2) \land pc_1'' = 2$$

$$x_1' > 0$$

$$pc_2 = 0 \land x_2' = x_1' + 1 \land pc_2' = 1$$

$$pc_2' = 1 \land (x_1' = 0 \lor x_2' < x_1') \land pc_2'' = 2$$

$$pc_1'' = 2 \land pc_2'' = 2$$

Finding out why the cex is spurious (second possibility)

$$pc_1 = 0 \land x_1 = 0 \land pc_2 = 0 \land x_2 = 0$$

$$pc_1 = 0 \land x_1' = x_2 + 1 \land pc_1' = 1$$

$$pc_1' = 1 \land (x_2 = 0 \lor x_1' < x_2) \land pc_1'' = 2$$

$$pc_2 = 0 \land x_2' = x_1' + 1 \land pc_2' = 1$$

$$pc_2' = 1 \land (x_1' = 0 \lor x_2' < x_1') \land pc_2'' = 2$$

$$pc_1'' = 2 \land pc_2'' = 2$$

$$x_2 = 0$$

$$x_1' > x_2 \land x_2 = 0$$

$$x_1' > 0$$

$$x_1' > 0 \land x_2' > x_1'$$

Finding out why the cex is spurious (second possibility)

$$pc_1 = 0 \wedge x_1 = 0 \wedge pc_2 = 0 \wedge x_2 = 0$$

$$pc_1 = 0 \wedge x_1' = x_2 + 1 \wedge pc_1' = 1$$

$$pc_1' = 1 \wedge (x_2 = 0 \vee x_1' < x_2) \wedge pc_1'' = 2$$

$$pc_2 = 0 \wedge x_2' = x_1' + 1 \wedge pc_2' = 1$$

$$pc_2' = 1 \wedge (x_1' = 0 \vee x_2' < x_1') \wedge pc_2'' = 2$$

$$pc_1'' = 2 \wedge pc_2'' = 2$$

$$x_2 = 0$$

$$x_1' > x_2 \wedge x_2 = 0$$

$$x_1' > 0$$

$$x_1' > 0 \wedge x_2' > x_1'$$

$$\perp$$

Final version

predicates: pc1=0, pc1=1, pc2=0, pc2=1, x1=0, x2=0, x1<x2, x1>x2

initial state: pc1 = 0, x1 = 0, pc2 = 0, x2 = 0

```
while (true) {
  if(pc1 == 0){
    x1 = x2 + 1;
    pc1 = 1;
  }else if(pc1 == 1 &&
             (x2 == 0 ||
              x1 < x2 )){
    pc1 = 2;
  }else if(pc1 == 2){
    pc1 = 0;
    x1 = 0;
  }
  if(pc1==2 && pc2==2){
    ERROR;
  }
}
```

```
while (true) {
  if(pc2 == 0){
    x2 = x1 + 1;
    pc2 = 1;
  }else if(pc2 == 1 &&
             (x1 == 0 ||
              x2 < x1 )){
    pc2 = 2;
  }else if(pc2 == 2){
    pc2 = 0;
    x2 = 0;
  }
  if(pc1==2 && pc2==2){
    ERROR;
  }
}
```

Questions ?