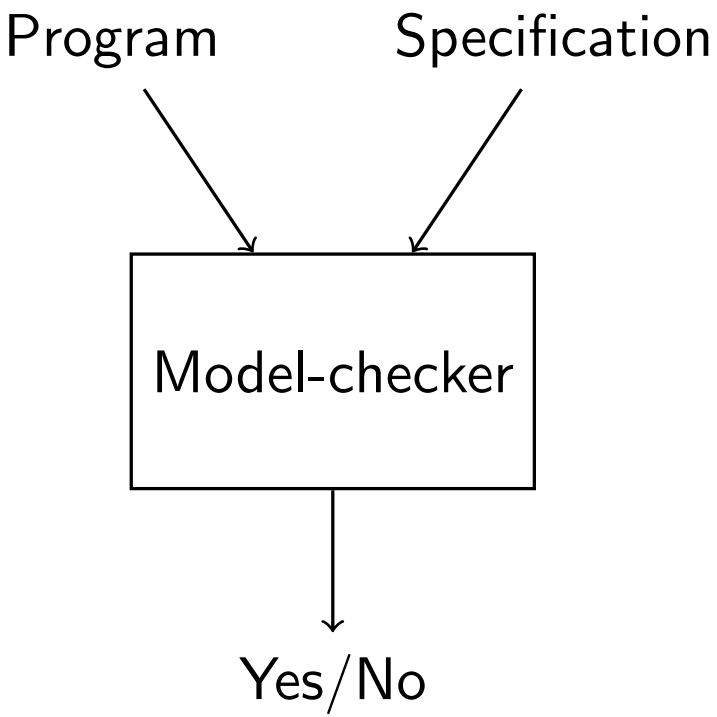
Software Model-Checking: an algorithmic approach to prove programs correct

Damien Zufferey

IST Austria

May 25, 2011

Push-button approach



Running example: Lamport's bakery algorithm

There is bakery with only one clerk. The clerk can serve only one customer at the time. To avoid conflict between customers, a numbering machine gives a ticket to each customer.

What customers do:

1. Customer enters the bakery and get a ticket;
2. If he is alone or has the ticket with lowest number then he orders;
3. When he is done, he leaves the bakery and throws away the ticket.

When the bakery is empty the numbering machine is reseted.

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)

```
1  while (true) {
2    if(pc1 == 0){
3      x1 = x2 + 1;
4      pc1 = 1;
5    }else if(pc1 == 1 &&
6                (x2 == 0 ||
7                 x1 < x2 )){
8      pc1 = 2;
9    }else if(pc1 == 2){
10     pc1 = 0;
11     x1 = 0;
12   }
13   if(pc1==2 && pc2==2){
14     ERROR;
15   }
16 }
```

```
1  while (true) {
2    if(pc2 == 0){
3      x2 = x1 + 1;
4      pc2 = 1;
5    }else if(pc2 == 1 &&
6                (x1 == 0 ||
7                 x2 < x1 )){
8      pc2 = 2;
9    }else if(pc2 == 2){
10     pc2 = 0;
11     x2 = 0;
12   }
13   if(pc1==2 && pc2==2){
14     ERROR;
15   }
16 }
```

Assumptions and model

unbounded integers: from $-\infty$ to ∞, no overflow.

atomicity: blocks of code are atomic in particular:

```
x1 = x2 + 1;
```

```
if(pc1 == 1 &&
   (x2 == 0 ||
    x1 < x2 )){
  pc1 = 2;
}
```

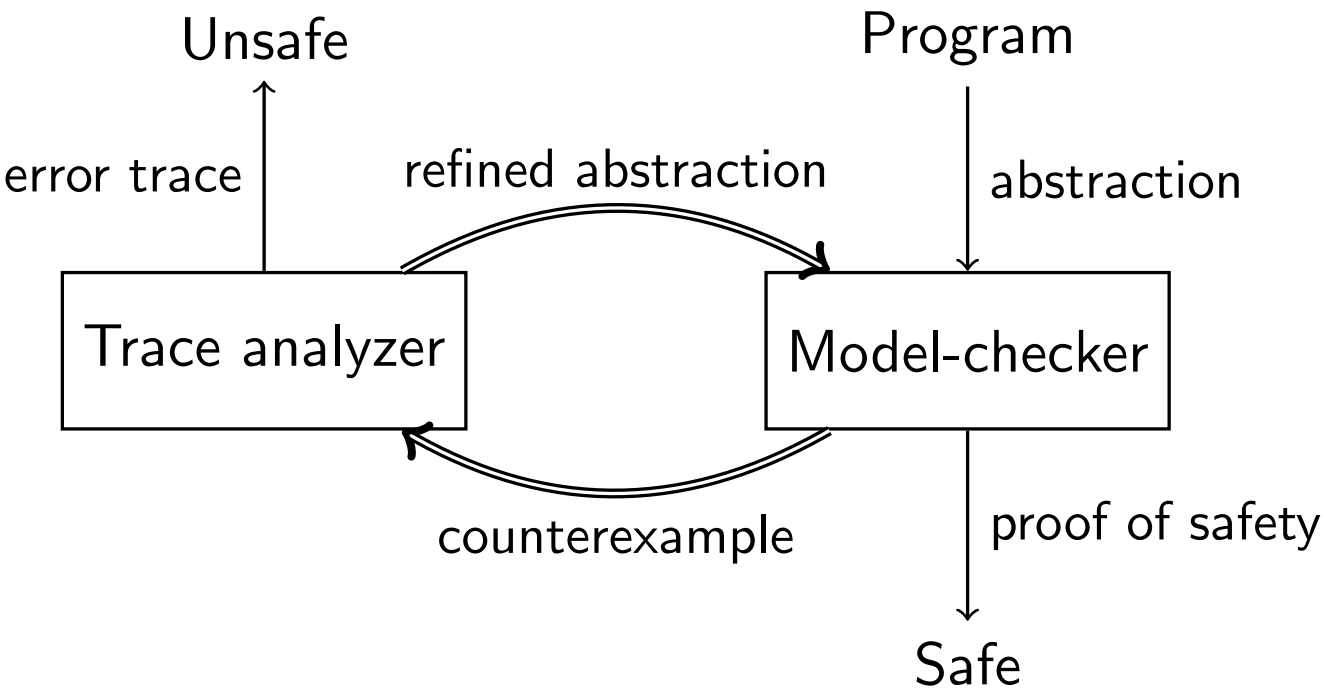
Why do we means by correct ?

reachability: (safety) no two customers fight.

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

For this talk we will care only about safety property.

General idea:



CEGAR: counterexample guided abstraction refinement

First iteration

predicates:

initial state:

```
1  while (true) {
2    if(          ){
3                   ;
4                ;
5    }else if(           &&
6                (          ||
7                            )){
8                ;
9    }else if(           ){
10               ;
11              ;
12   }
13   if(         &&          ){
14     ERROR;
15   }
16 }
```


```
1  while (true) {
2    if(          ){
3                   ;
4                ;
5    }else if(           &&
6                (          ||
7                            )){
8                ;
9    }else if(           ){
10               ;
11              ;
12   }
13   if(         &&          ){
14     ERROR;
15   }
16 }
```

First iteration

predicates:

initial state:

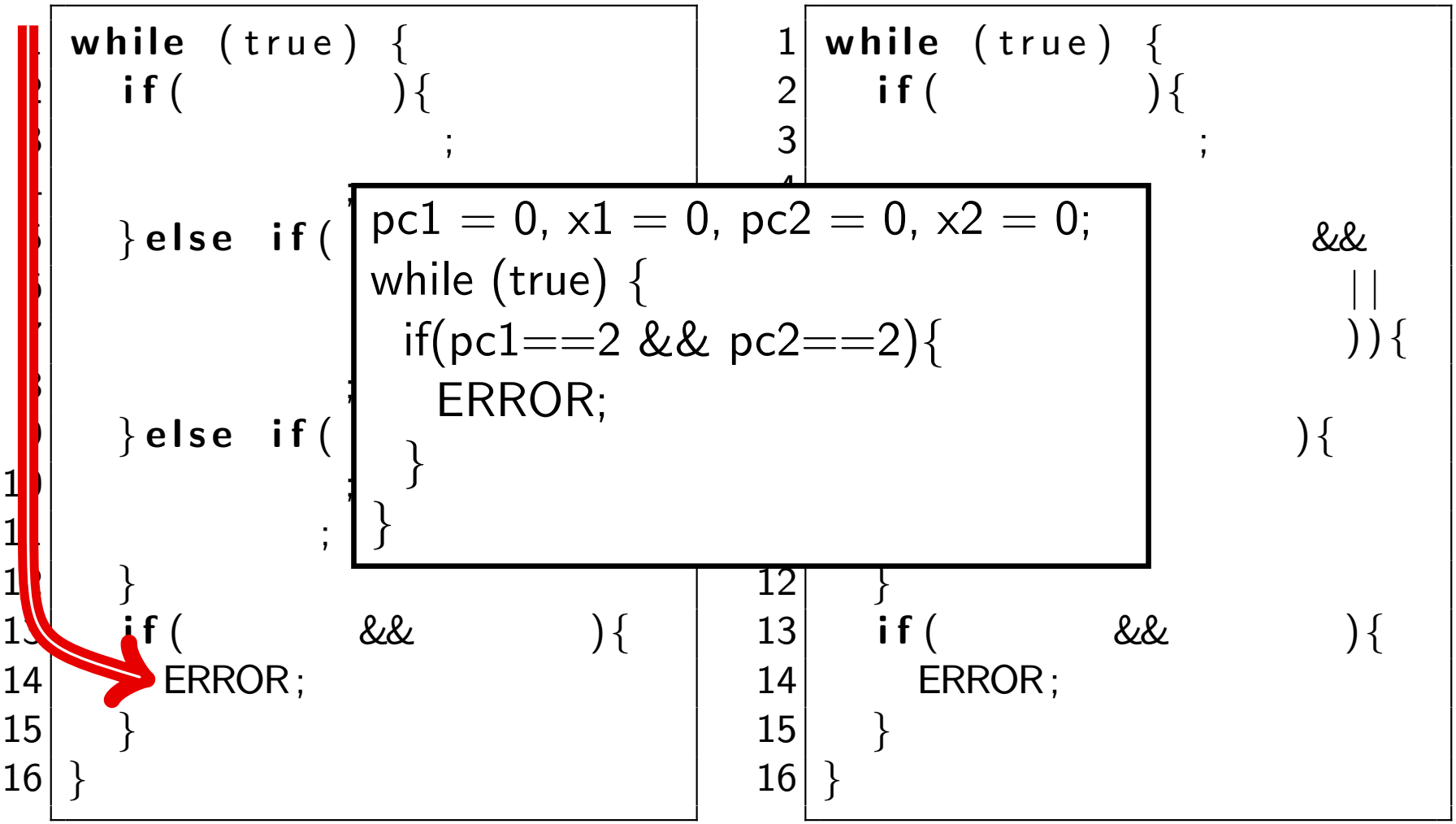
```
while (true) {
  if(          ){
                 ;
            ;
  }else if(           &&
             (          ||
                        )){
            ;
  }else if(          ){
             ;
           ;
  }
  if(        &&        ){
    ERROR;
  }
}
```

```
1  while (true) {
2    if(          ){
3                   ;
4              ;
5    }else if(           &&
6              (          ||
7                         )){
8              ;
9    }else if(          ){
10             ;
11           ;
12   }
13   if(        &&        ){
14     ERROR;
15   }
16 }
```

First iteration

predicates:

initial state:



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 $\Rightarrow$ 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: $pc1 = 0$

```
1  while (true) {
2    if(pc1 == 0){
3                    ;
4      pc1 = 1;
5    }else if(pc1 == 1 &&
6                (          ||
7                           )){
8      pc1 = 2;
9    }else if(pc1 == 2){
10     pc1 = 0;
11           ;
12   }
13   if(pc1==2 &&          ){
14     ERROR;
15   }
16 }
```

```
1  while (true) {
2    if(          ){
3                  ;
4              ;
5    }else if(          &&
6                (          ||
7                           )){
8              ;
9    }else if(          ){
10             ;
11           ;
12   }
13   if(pc1==2 &&          ){
14     ERROR;
15   }
16 }
```

Second iteration

predicates: pc1 = 0

initial state: pc1 = 0

```
1  while (true) {
2    if(pc1 == 0){
3                  ;
4      pc1 = 1;
5    }else if(pc1 == 1 &&
6               (          ||
7                          )){
8      pc1 = 2;
9    }else if(pc1 == 2){
10     pc1 = 0;
11           ;
12   }
13   if(pc1==2 &&          ){
14     ERROR;
15   }
16 }
```

```
1  while (true) {
2    if(          ){
3                  ;
4              ;
5    }else if(          &&
6               (          ||
7                          )){
8              ;
9    }else if(          ){
10             ;
11           ;
12   }
13   if(pc1==2 &&          ){
14     ERROR;
15   }
16 }
```

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:

$$pc_1 = 0 \wedge x_1 = 0 \wedge pc_2 = 0 \wedge x_2 = 0$$
$$pc_1 = 0$$
$$x_1' = x_2 + 1$$
$$pc_1' = 1$$
$$pc_1' = 2 \wedge pc_2 = 2$$

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:

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

$$pc_1 = 0$$

$$x_1' = x_2 + 1$$

$$pc_1' = 1$$

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

Formula is unsat $\Rightarrow$ 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.

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

$$pc_1 = 0$$

$$x_1' = x_2 + 1$$

$$pc_1' = 1$$

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

Finding out why the cex is spurious.

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

$$\top$$

$$pc_1 = 0$$

$$\top$$

$$x_1' = x_2 + 1$$

$$pc_1' = 1$$

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

Finding out why the cex is spurious.

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

$\top$

$$pc_1 = 0$$

$\top$

$$x_1' = x_2 + 1$$

$\top$

$$pc_1' = 1$$

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

Finding out why the cex is spurious.

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

$$\top$$

$$pc_1 = 0$$

$$\top$$

$$x_1' = x_2 + 1$$

$$\top$$

$$pc_1' = 1$$

$$pc_1' = 1$$

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

Third iteration

predicates: pc1 = 0, pc1 = 1

initial state: pc1 = 0

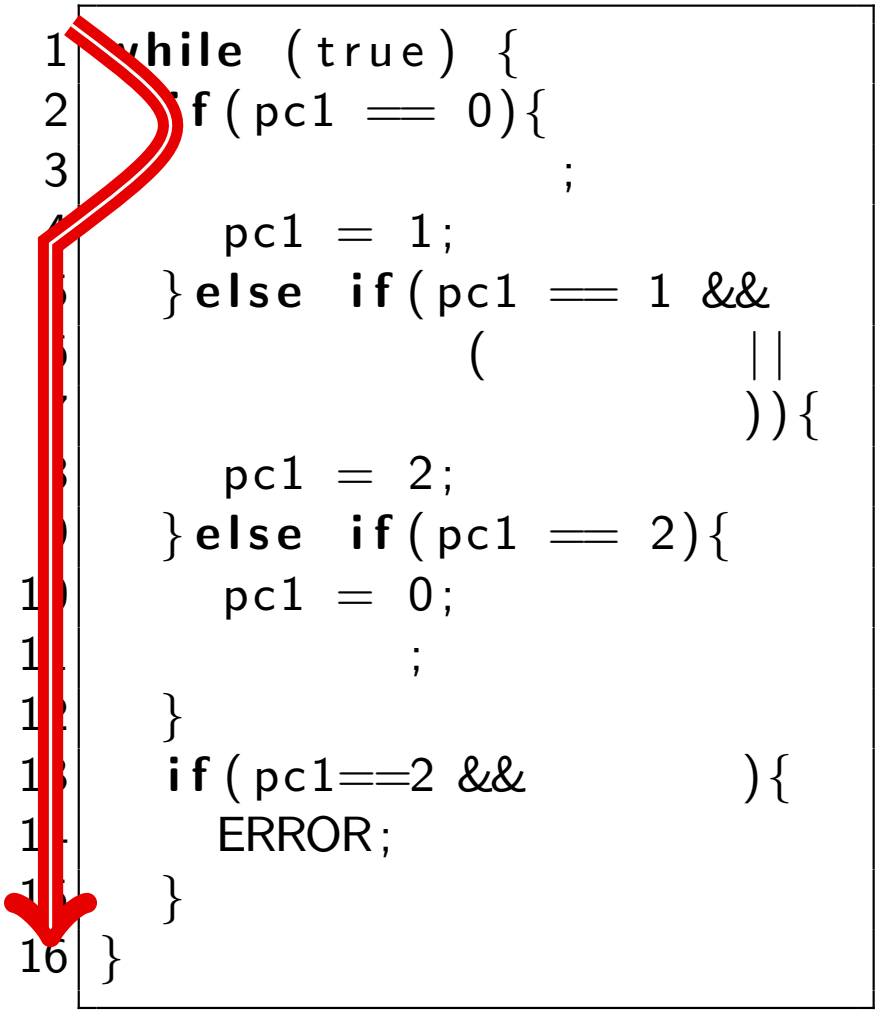
```
1  while (true) {
2    if(pc1 == 0){
3                    ;
4      pc1 = 1;
5    }else if(pc1 == 1 &&
6                (          ||
7                           )){
8      pc1 = 2;
9    }else if(pc1 == 2){
10     pc1 = 0;
11           ;
12   }
13   if(pc1==2 &&          ){
14     ERROR;
15   }
16 }
```

```
1  while (true) {
2    if(          ){
3                  ;
4              ;
5    }else if(          &&
6                (          ||
7                           )){
8              ;
9    }else if(          ){
10             ;
11           ;
12   }
13   if(pc1==2 &&          ){
14     ERROR;
15   }
16 }
```

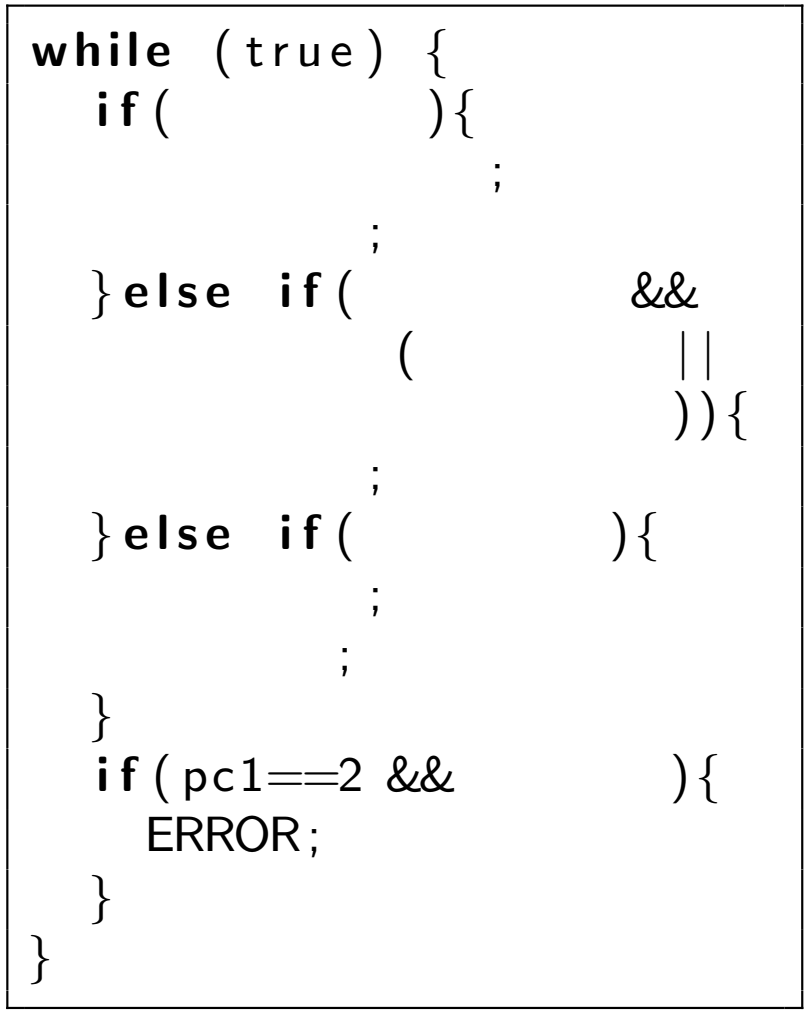
Third iteration

predicates: pc1 = 0, pc1 = 1

initial state: pc1 = 0



```
1  while (true) {
2    if(pc1 == 0){
3                  ;
4      pc1 = 1;
5    }else if(pc1 == 1 &&
6               (          ||
7                          )){
8      pc1 = 2;
9    }else if(pc1 == 2){
10     pc1 = 0;
11             ;
12   }
13   if(pc1==2 &&          ){
14     ERROR;
15   }
16 }
```

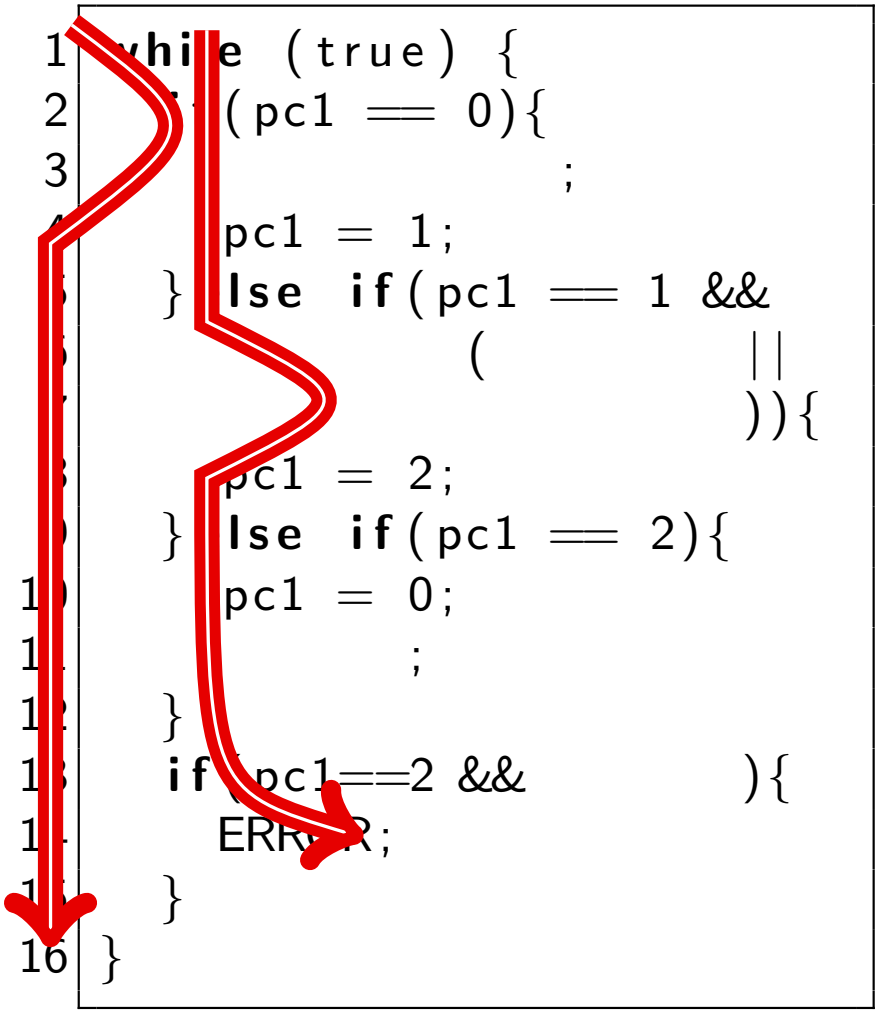


```
1  while (true) {
2    if(          ){
3                  ;
4              ;
5    }else if(          &&
6               (          ||
7                          )){
8              ;
9    }else if(          ){
10             ;
11           ;
12   }
13   if(pc1==2 &&          ){
14     ERROR;
15   }
16 }
```

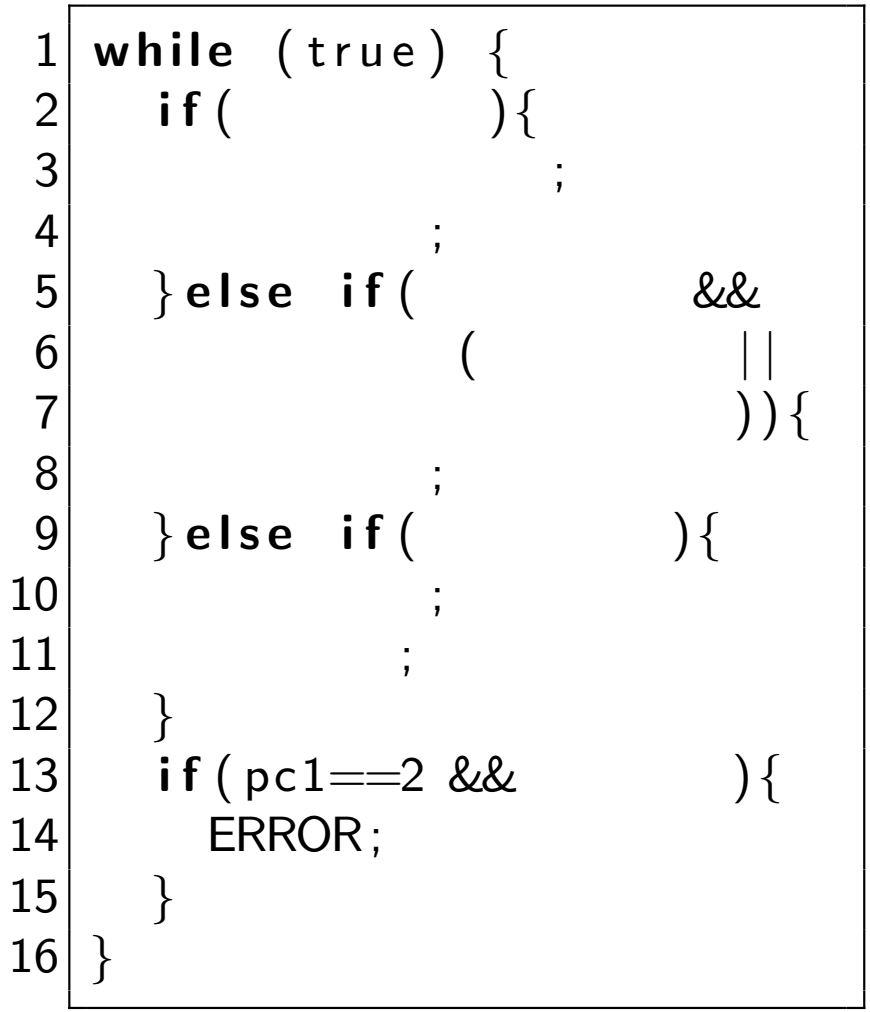
Third iteration

predicates: pc1 = 0, pc1 = 1

initial state: pc1 = 0



```
1  while (true) {
2    if(pc1 == 0){
3                  ;
4      pc1 = 1;
5    }else if(pc1 == 1 &&
6               (          ||
7                          )){
8      pc1 = 2;
9    }else if(pc1 == 2){
10     pc1 = 0;
11             ;
12   }
13   if(pc1==2 &&          ){
14     ERROR;
15   }
16 }
```



```
1  while (true) {
2    if(          ){
3                  ;
4              ;
5    }else if(          &&
6               (          ||
7                          )){
8              ;
9    }else if(          ){
10             ;
11           ;
12   }
13   if(pc1==2 &&          ){
14     ERROR;
15   }
16 }
```

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;
```

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

$$pc_1 = 0$$

$$x_1' = x_2 + 1$$

$$pc_1' = 1$$

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

$$pc_1'' = 2$$

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

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;
```

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

$$pc_1 = 0$$

$$x_1' = x_2 + 1$$

$$pc_1' = 1$$

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

$$pc_1'' = 2$$

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

A few iterations later ...

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

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

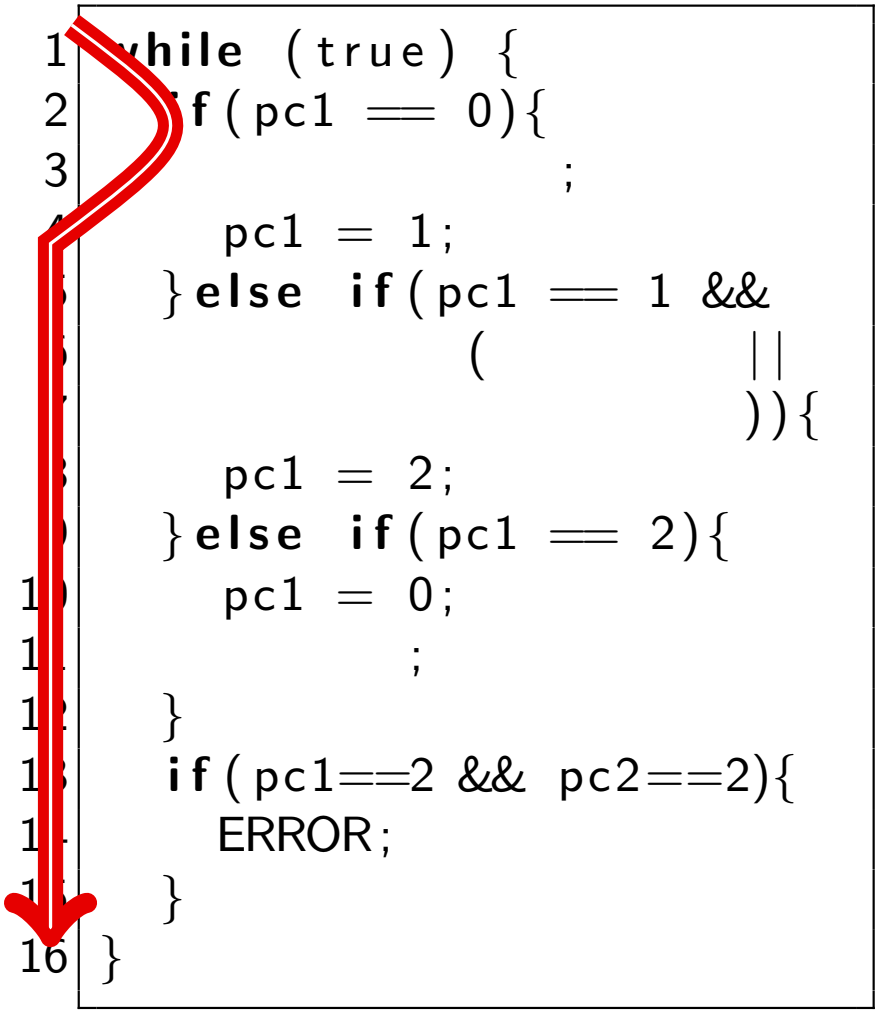
```
1  while (true) {
2    if(pc1 == 0){
3                    ;
4      pc1 = 1;
5    }else if(pc1 == 1 &&
6                (          ||
7                           )){
8      pc1 = 2;
9    }else if(pc1 == 2){
10     pc1 = 0;
11           ;
12   }
13   if(pc1==2 && pc2==2){
14     ERROR;
15   }
16 }
```

```
1  while (true) {
2    if(pc2 == 0){
3                    ;
4      pc2 = 1;
5    }else if(pc2 == 1 &&
6                (          ||
7                           )){
8      pc2 = 2;
9    }else if(pc2 == 2){
10     pc2 = 0;
11           ;
12   }
13   if(pc1==2 && pc2==2){
14     ERROR;
15   }
16 }
```

A few iterations later ...

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

initial state: pc1 = 0, pc2 = 0



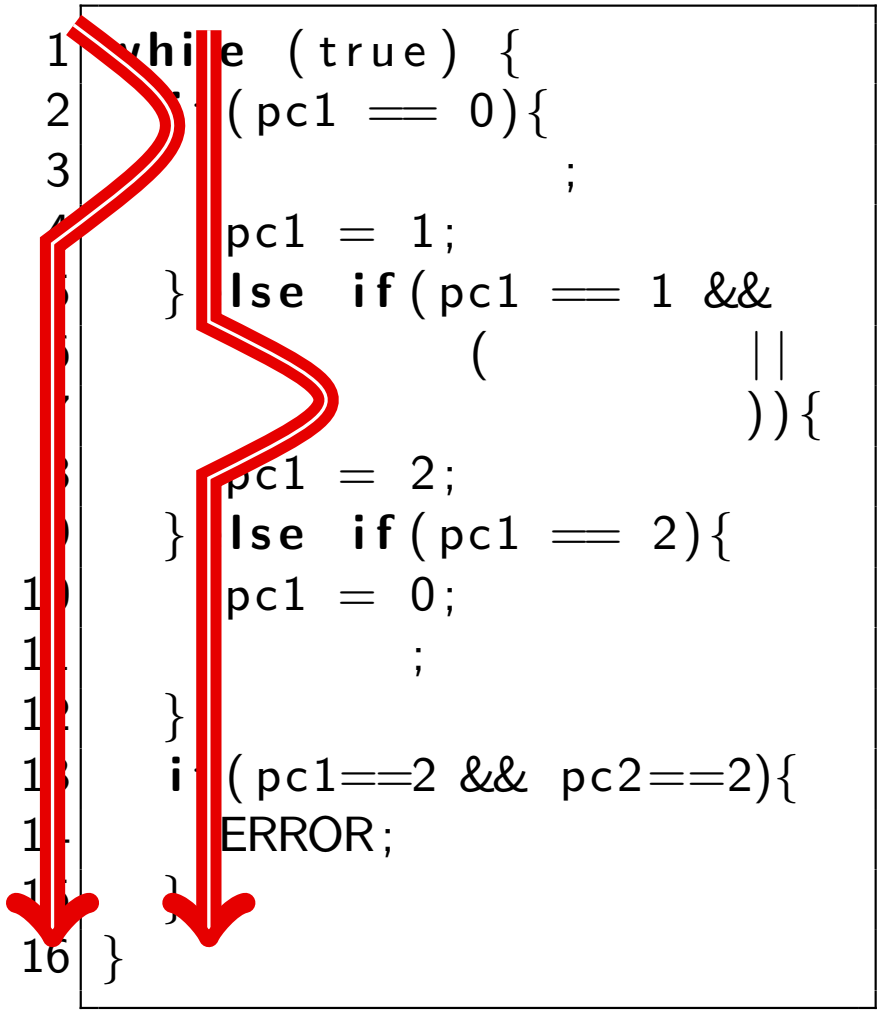
```
1  while (true) {
2    if(pc1 == 0){
3                  ;
4      pc1 = 1;
5    }else if(pc1 == 1 &&
6               (          ||
7                          )){
8      pc1 = 2;
9    }else if(pc1 == 2){
10     pc1 = 0;
11             ;
12   }
13   if(pc1==2 && pc2==2){
14     ERROR;
15   }
16 }
```

```
1  while (true) {
2    if(pc2 == 0){
3                  ;
4      pc2 = 1;
5    }else if(pc2 == 1 &&
6               (          ||
7                          )){
8      pc2 = 2;
9    }else if(pc2 == 2){
10     pc2 = 0;
11             ;
12   }
13   if(pc1==2 && pc2==2){
14     ERROR;
15   }
16 }
```

A few iterations later ...

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

initial state: pc1 = 0, pc2 = 0



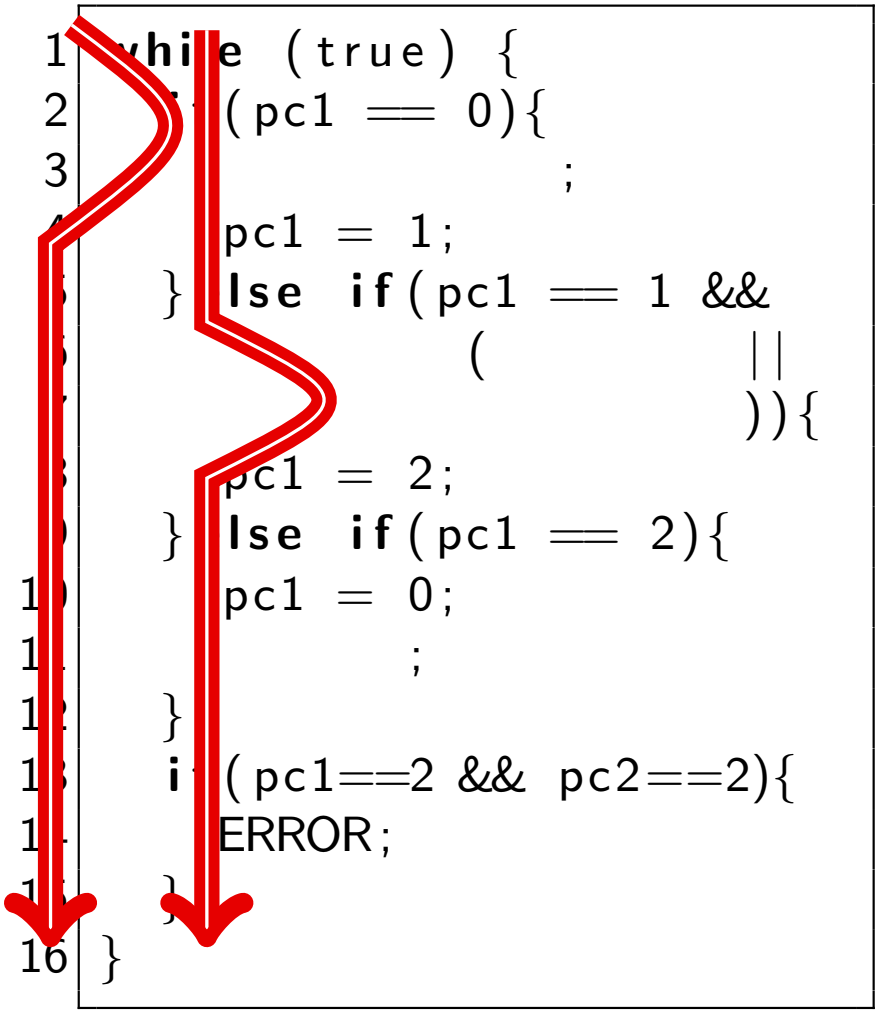
```
1  while (true) {
2    if(pc1 == 0){
3                  ;
4      pc1 = 1;
5    }else if(pc1 == 1 &&
6               (          ||
7                          )){
8      pc1 = 2;
9    }else if(pc1 == 2){
10     pc1 = 0;
11             ;
12   }
13   if(pc1==2 && pc2==2){
14     ERROR;
15   }
16 }
```

```
1  while (true) {
2    if(pc2 == 0){
3                  ;
4      pc2 = 1;
5    }else if(pc2 == 1 &&
6               (          ||
7                          )){
8      pc2 = 2;
9    }else if(pc2 == 2){
10     pc2 = 0;
11             ;
12   }
13   if(pc1==2 && pc2==2){
14     ERROR;
15   }
16 }
```

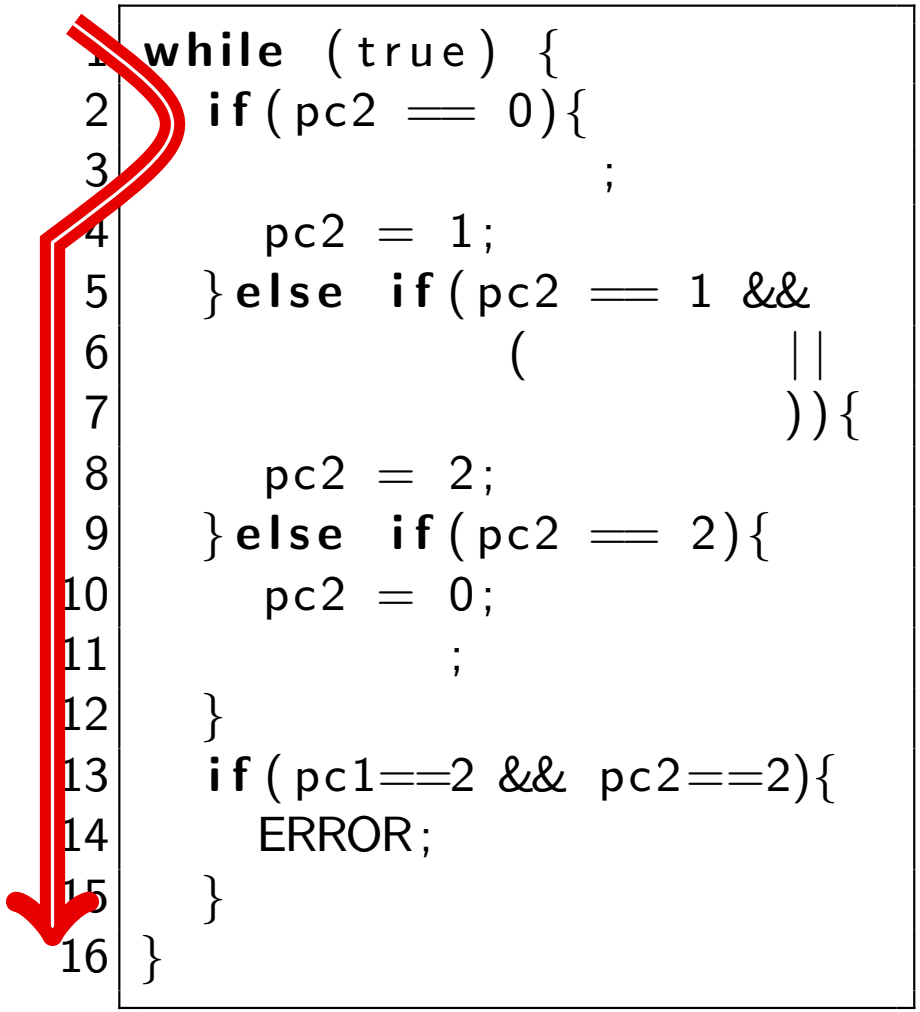
A few iterations later ...

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

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



```
1  while (true) {
2    if(pc1 == 0){
3                ;
4      pc1 = 1;
5    }else if(pc1 == 1 &&
6             (          ||
7                        )){
8      pc1 = 2;
9    }else if(pc1 == 2){
10     pc1 = 0;
11           ;
12   }
13   if(pc1==2 && pc2==2){
14     ERROR;
15   }
16 }
```

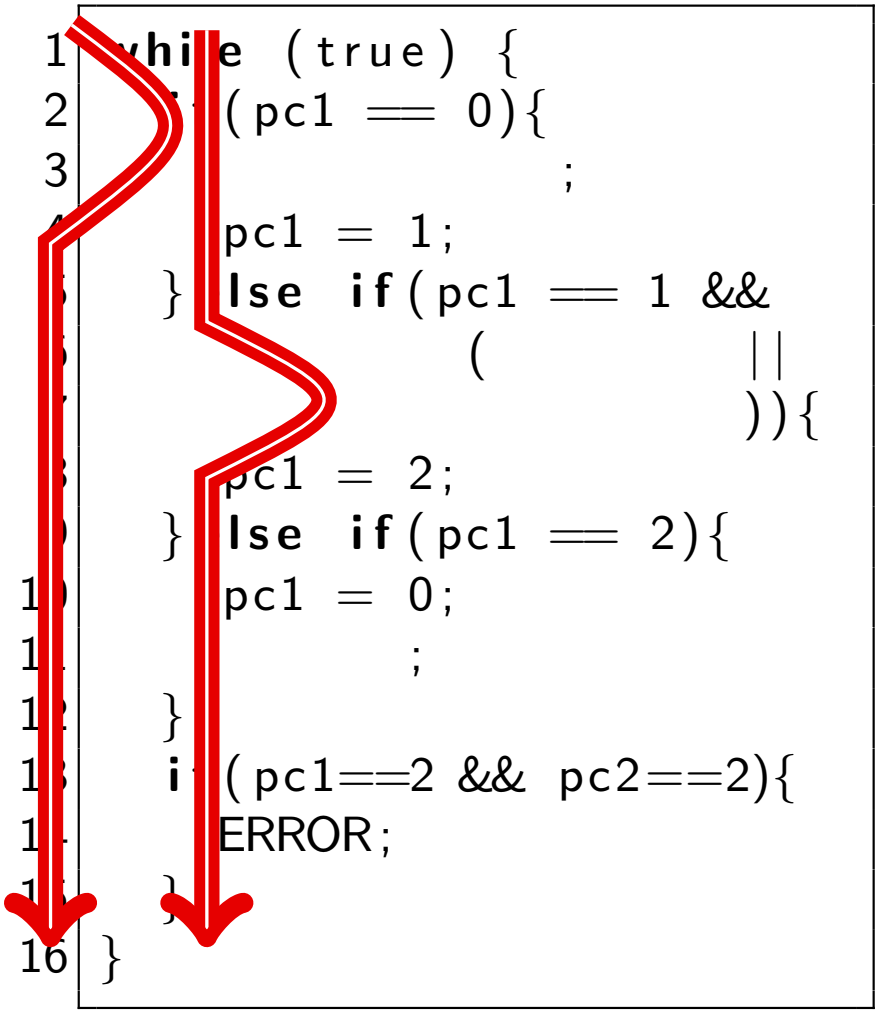


```
1  while (true) {
2    if(pc2 == 0){
3                ;
4      pc2 = 1;
5    }else if(pc2 == 1 &&
6             (          ||
7                        )){
8      pc2 = 2;
9    }else if(pc2 == 2){
10     pc2 = 0;
11           ;
12   }
13   if(pc1==2 && pc2==2){
14     ERROR;
15   }
16 }
```

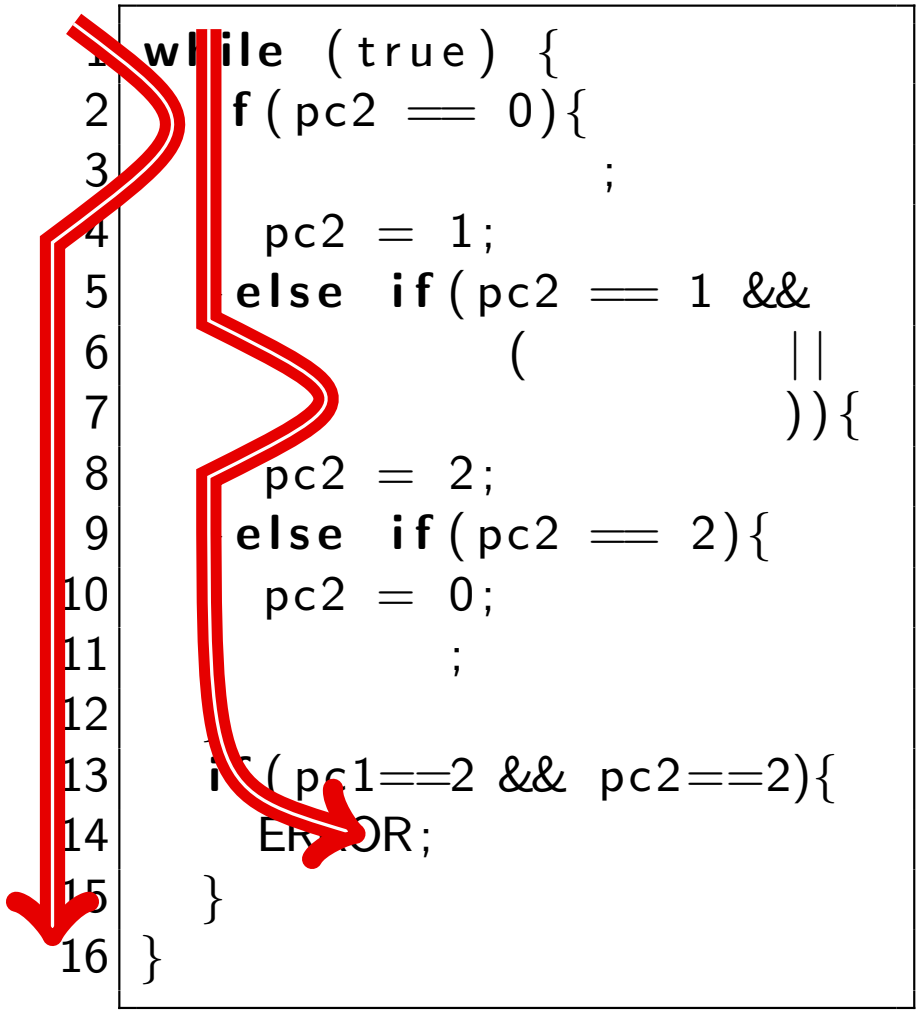
A few iterations later ...

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

initial state: pc1 = 0, pc2 = 0



```
1  while (true) {
2    if(pc1 == 0){
3                  ;
4      pc1 = 1;
5    } else if(pc1 == 1 &&
6                (          ||
7                           )){
8      pc1 = 2;
9    } else if(pc1 == 2){
10     pc1 = 0;
11               ;
12   }
13   if(pc1==2 && pc2==2){
14     ERROR;
15   }
16 }
```



```
1  while (true) {
2    if(pc2 == 0){
3                  ;
4      pc2 = 1;
5    else if(pc2 == 1 &&
6                (          ||
7                           )){
8      pc2 = 2;
9    else if(pc2 == 2){
10     pc2 = 0;
11               ;
12
13   if(pc1==2 && pc2==2){
14     ERROR;
15   }
16 }
```

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)

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

$$x_2 = 0$$

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

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

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

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

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

Finding out why the cex is spurious (first possibility)

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

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

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

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

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

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

$$x_2 = 0$$

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

Finding out why the cex is spurious (first possibility)

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

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

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

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

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

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

$$x_2 = 0$$

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

$$x_1' = 1$$

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)

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

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

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

$$pc_2 = 0 \wedge x_2' = x_1' + 1 \wedge pc_2' = 1 \qquad x_1' = 1 \wedge x_2' = 2$$

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

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

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

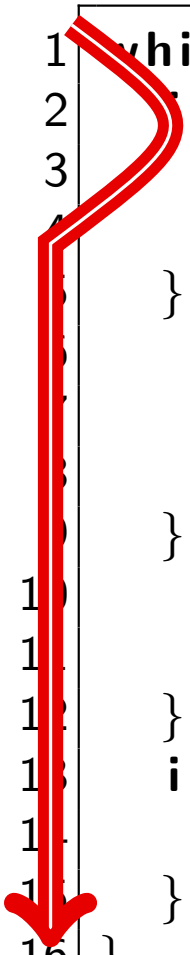
```
1  while (true) {
2    if(pc1 == 0){
3      x1 = x2 + 1;
4      pc1 = 1;
5    }else if(pc1 == 1 &&
6                (x2 == 0 ||
7                 x1 < x2 )){
8      pc1 = 2;
9    }else if(pc1 == 2){
10     pc1 = 0;
11     x1 = 0;
12   }
13   if(pc1==2 && pc2==2){
14     ERROR;
15   }
16 }
```

```
1  while (true) {
2    if(pc2 == 0){
3      x2 = x1 + 1;
4      pc2 = 1;
5    }else if(pc2 == 1 &&
6                (x1 == 0 ||
7                 x2 < x1 )){
8      pc2 = 2;
9    }else if(pc2 == 2){
10     pc2 = 0;
11     x2 = 0;
12   }
13   if(pc1==2 && pc2==2){
14     ERROR;
15   }
16 }
```

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



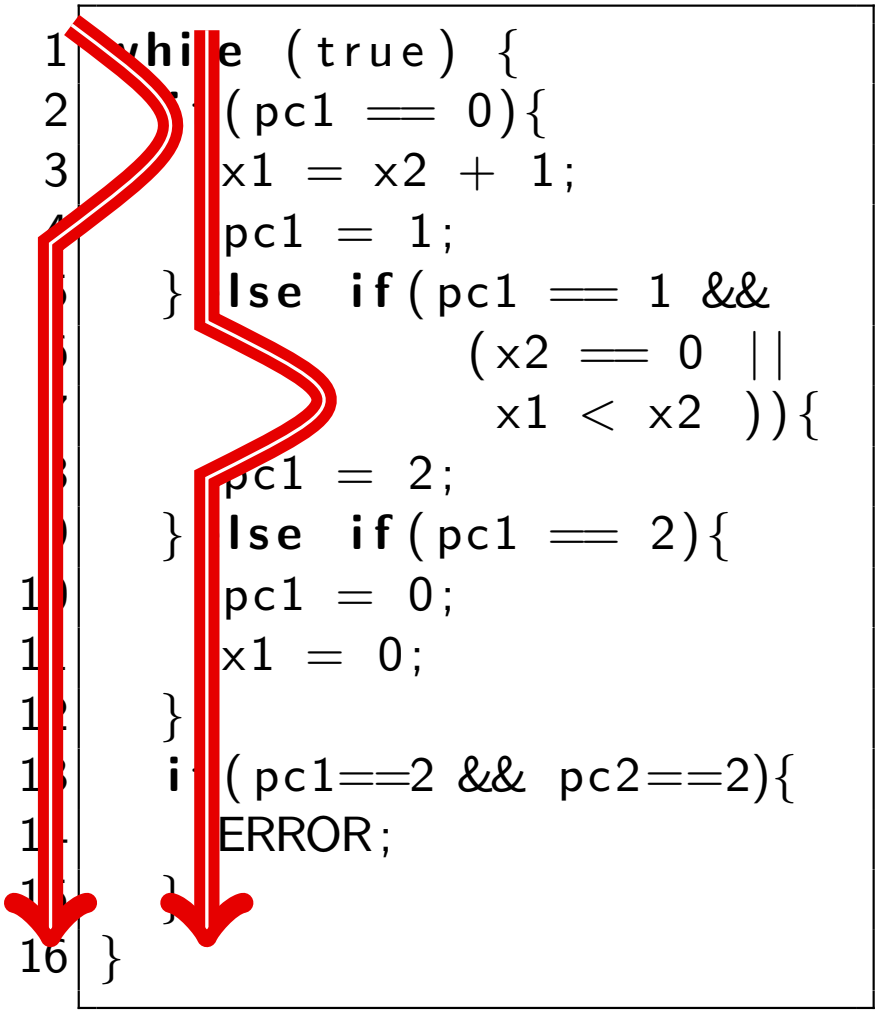
```
1  while (true) {
2    if(pc1 == 0){
3      x1 = x2 + 1;
4      pc1 = 1;
5    }else if(pc1 == 1 &&
6                (x2 == 0 ||
7                 x1 < x2 )){
8      pc1 = 2;
9    }else if(pc1 == 2){
10     pc1 = 0;
11     x1 = 0;
12   }
13   if(pc1==2 && pc2==2){
14     ERROR;
15   }
16 }
```

```
1  while (true) {
2    if(pc2 == 0){
3      x2 = x1 + 1;
4      pc2 = 1;
5    }else if(pc2 == 1 &&
6                (x1 == 0 ||
7                 x2 < x1 )){
8      pc2 = 2;
9    }else if(pc2 == 2){
10     pc2 = 0;
11     x2 = 0;
12   }
13   if(pc1==2 && pc2==2){
14     ERROR;
15   }
16 }
```

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



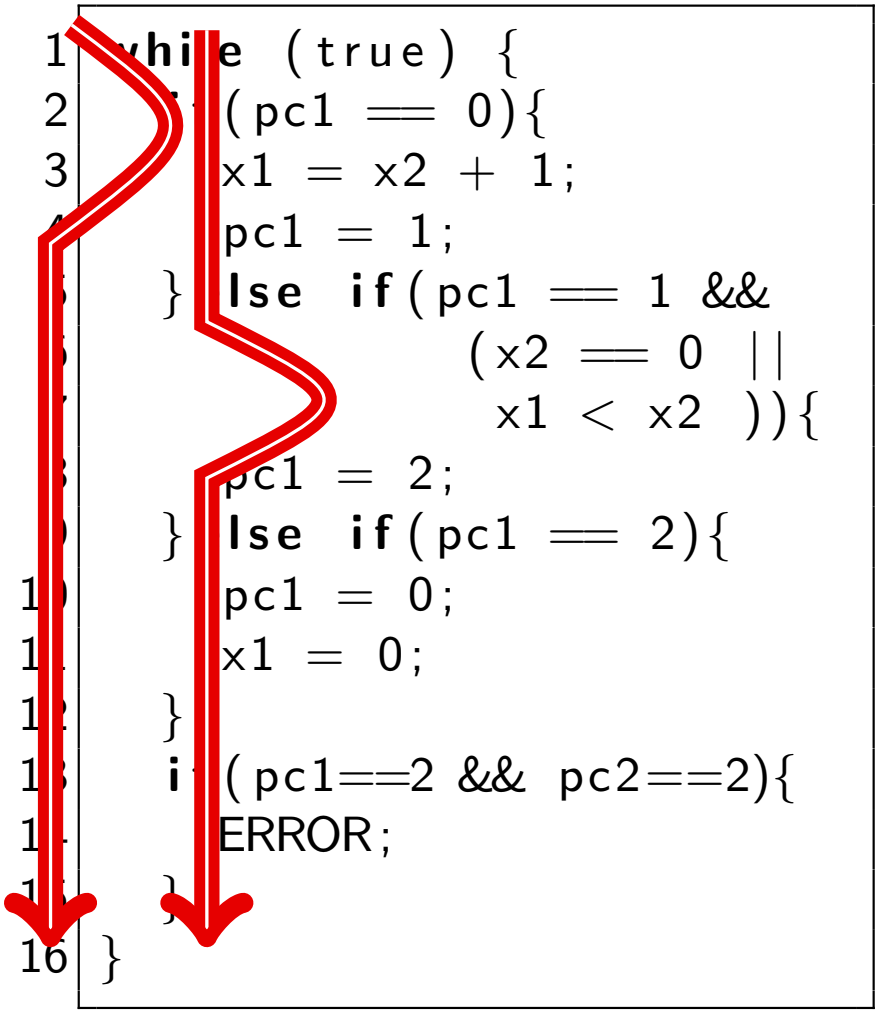
```
1  while (true) {
2    if(pc1 == 0){
3      x1 = x2 + 1;
4      pc1 = 1;
5    }else if(pc1 == 1 &&
6                (x2 == 0 ||
7                 x1 < x2 )){
8      pc1 = 2;
9    }else if(pc1 == 2){
10     pc1 = 0;
11     x1 = 0;
12   }
13   if(pc1==2 && pc2==2){
14     ERROR;
15   }
16 }
```

```
1  while (true) {
2    if(pc2 == 0){
3      x2 = x1 + 1;
4      pc2 = 1;
5    }else if(pc2 == 1 &&
6                (x1 == 0 ||
7                 x2 < x1 )){
8      pc2 = 2;
9    }else if(pc2 == 2){
10     pc2 = 0;
11     x2 = 0;
12   }
13   if(pc1==2 && pc2==2){
14     ERROR;
15   }
16 }
```

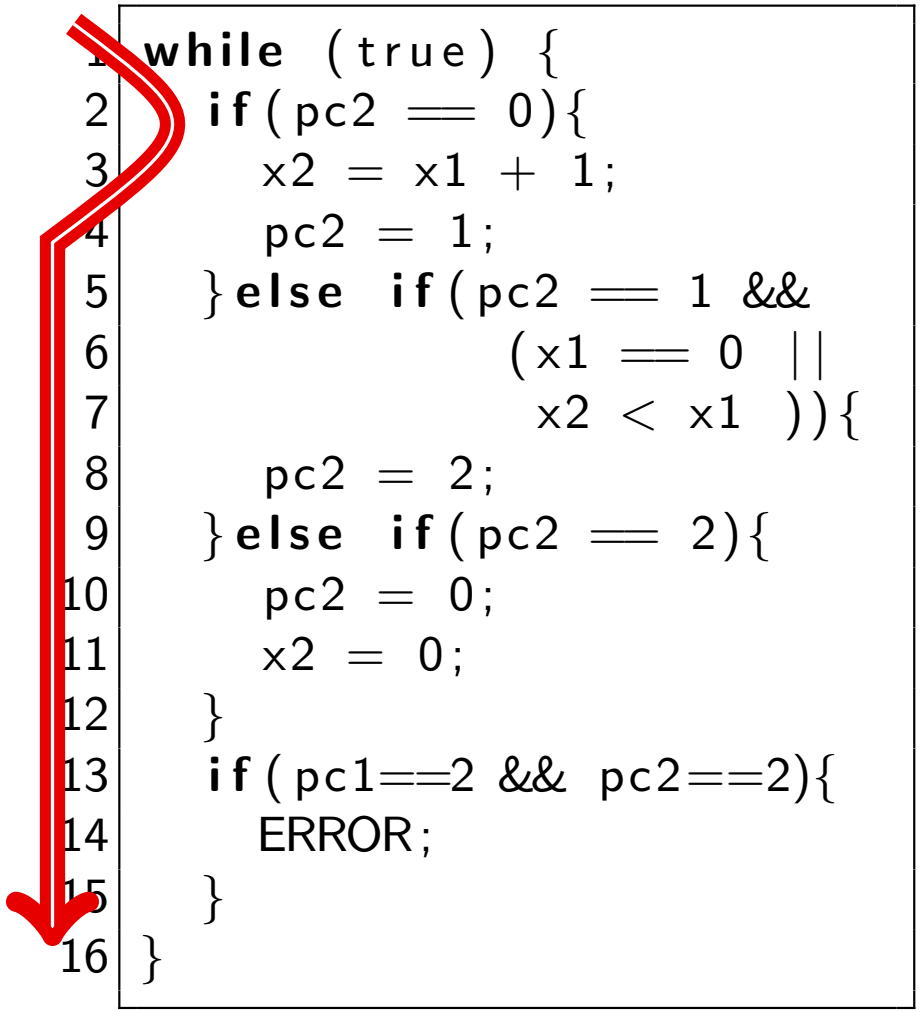
Does it work ?

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



```
1  while (true) {
2    if(pc1 == 0){
3      x1 = x2 + 1;
4      pc1 = 1;
5    }else if(pc1 == 1 &&
6                (x2 == 0 ||
7                 x1 < x2 )){
8      pc1 = 2;
9    }else if(pc1 == 2){
10     pc1 = 0;
11     x1 = 0;
12   }
13   if(pc1==2 && pc2==2){
14     ERROR;
15   }
16 }
```

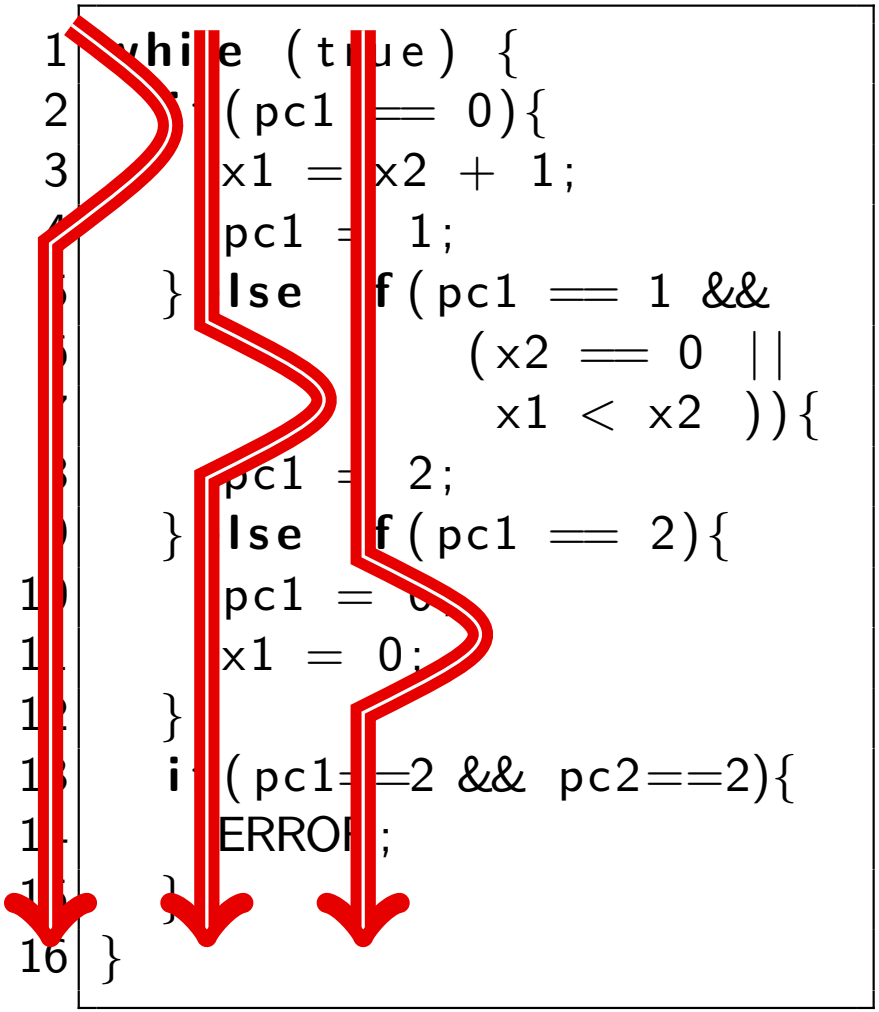


```
1  while (true) {
2    if(pc2 == 0){
3      x2 = x1 + 1;
4      pc2 = 1;
5    }else if(pc2 == 1 &&
6                (x1 == 0 ||
7                 x2 < x1 )){
8      pc2 = 2;
9    }else if(pc2 == 2){
10     pc2 = 0;
11     x2 = 0;
12   }
13   if(pc1==2 && pc2==2){
14     ERROR;
15   }
16 }
```

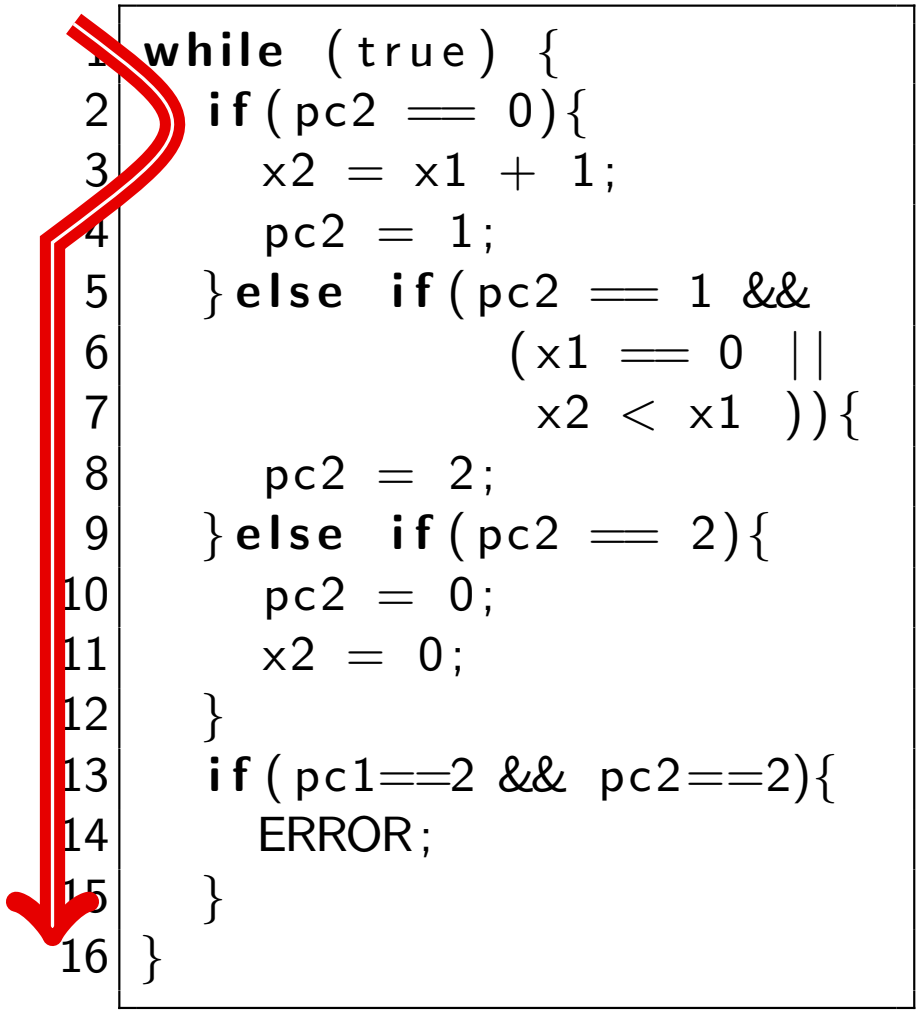
Does it work ?

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



```
1  while (true) {
2    if(pc1 == 0){
3      x1 = x2 + 1;
4      pc1 = 1;
5    }else if(pc1 == 1 &&
6             (x2 == 0 ||
7              x1 < x2 )){
8      pc1 = 2;
9    }else if(pc1 == 2){
10     pc1 = 0;
11     x1 = 0;
12   }
13   if(pc1==2 && pc2==2){
14     ERROR;
15   }
16 }
```

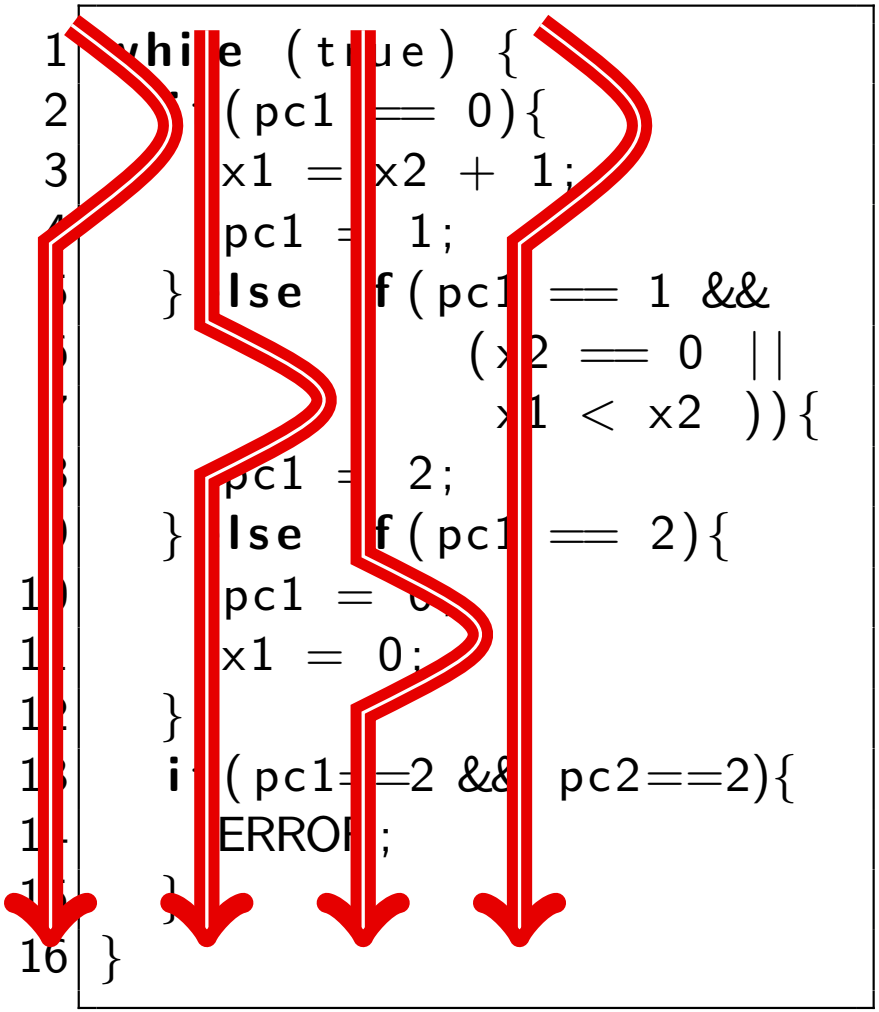


```
1  while (true) {
2    if(pc2 == 0){
3      x2 = x1 + 1;
4      pc2 = 1;
5    }else if(pc2 == 1 &&
6             (x1 == 0 ||
7              x2 < x1 )){
8      pc2 = 2;
9    }else if(pc2 == 2){
10     pc2 = 0;
11     x2 = 0;
12   }
13   if(pc1==2 && pc2==2){
14     ERROR;
15   }
16 }
```

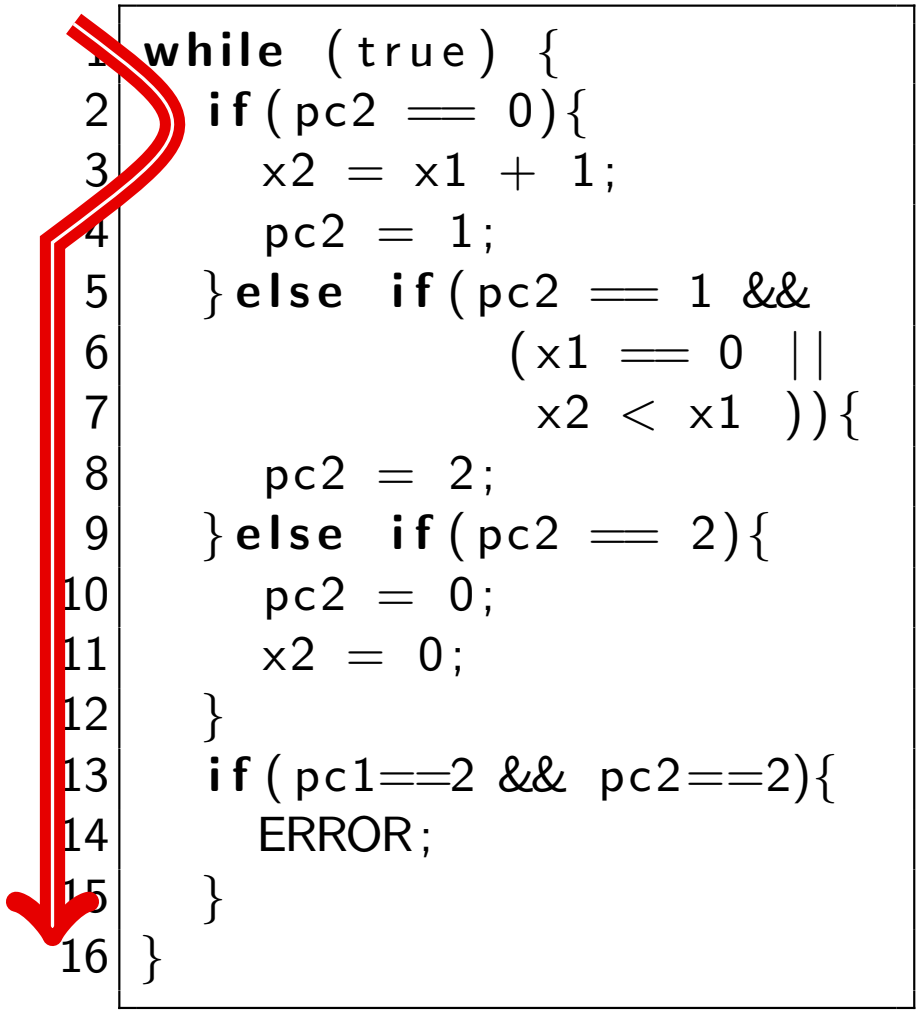
Does it work ?

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



```
1  while (true) {
2    if(pc1 == 0){
3      x1 = x2 + 1;
4      pc1 = 1;
5    }else if(pc1 == 1 &&
6             (x2 == 0 ||
7              x1 < x2 )){
8      pc1 = 2;
9    }else if(pc1 == 2){
10     pc1 = 0;
11     x1 = 0;
12   }
13   if(pc1==2 && pc2==2){
14     ERROR;
15   }
16 }
```

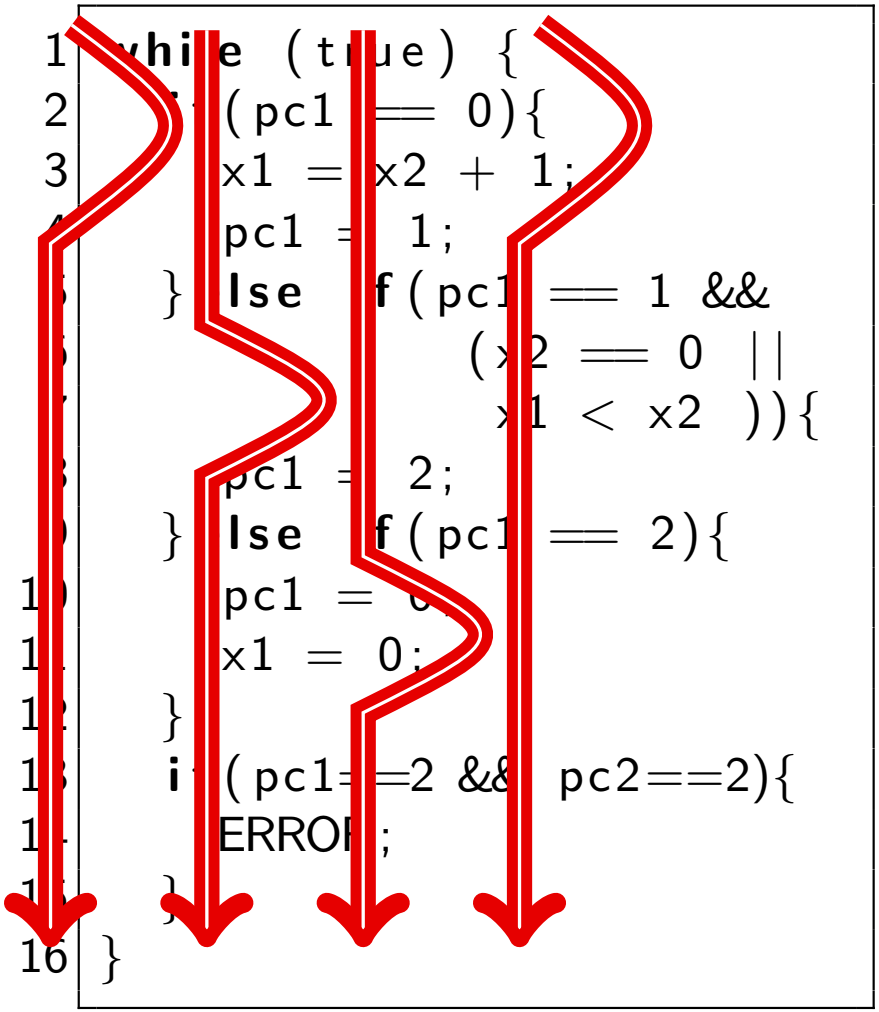


```
1  while (true) {
2    if(pc2 == 0){
3      x2 = x1 + 1;
4      pc2 = 1;
5    }else if(pc2 == 1 &&
6             (x1 == 0 ||
7              x2 < x1 )){
8      pc2 = 2;
9    }else if(pc2 == 2){
10     pc2 = 0;
11     x2 = 0;
12   }
13   if(pc1==2 && pc2==2){
14     ERROR;
15   }
16 }
```

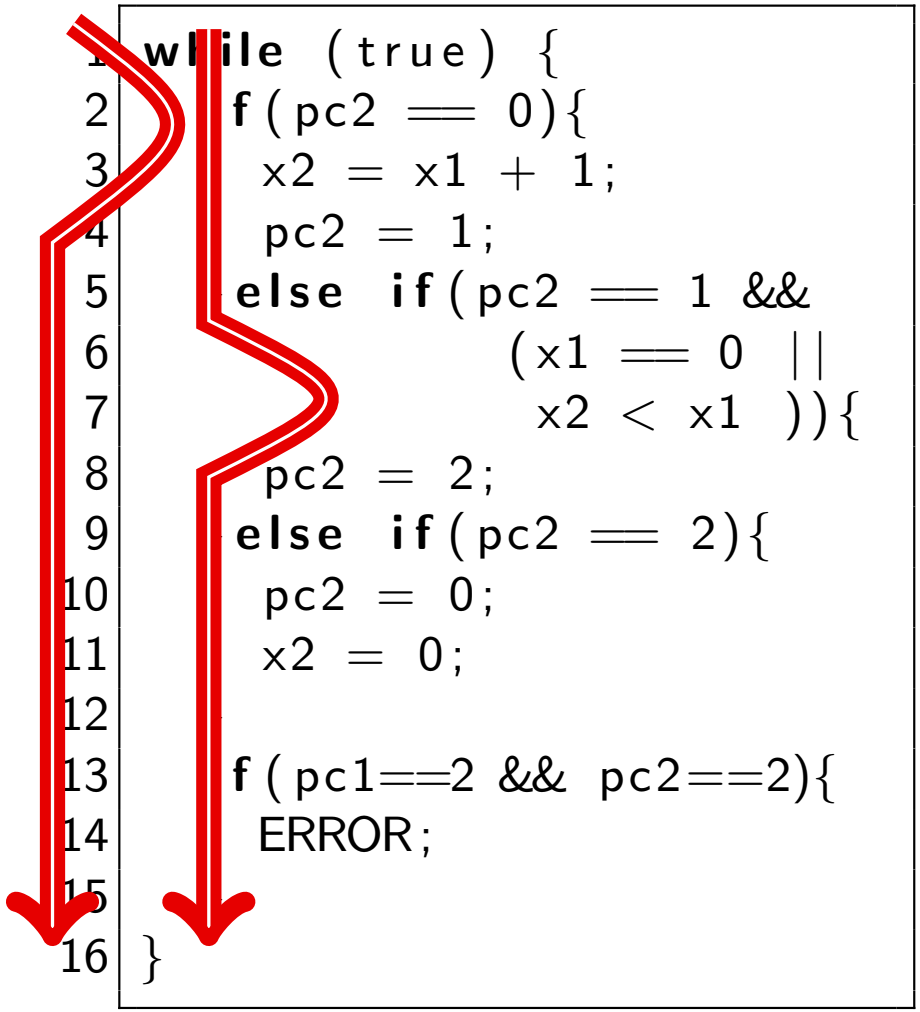
Does it work ?

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



```
1  while (true) {
2    if(pc1 == 0){
3      x1 = x2 + 1;
4      pc1 = 1;
5    } else if(pc1 == 1 &&
6              (x2 == 0 ||
7               x1 < x2 )){
8      pc1 = 2;
9    } else if(pc1 == 2){
10     pc1 = 0;
11     x1 = 0;
12   }
13   if(pc1==2 && pc2==2){
14     ERROR;
15   }
16 }
```

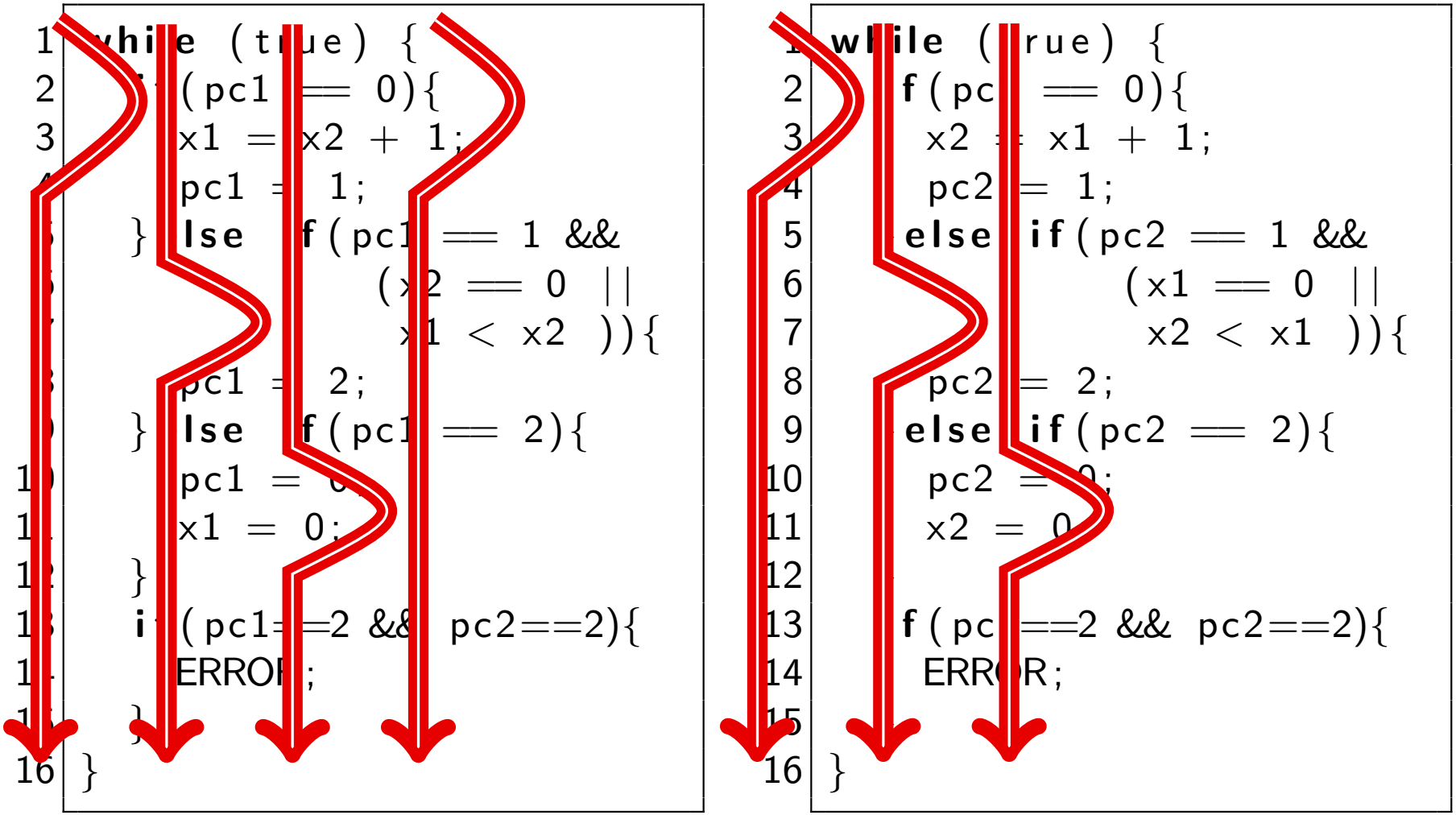


```
1  while (true) {
2    if(pc2 == 0){
3      x2 = x1 + 1;
4      pc2 = 1;
5    } else if(pc2 == 1 &&
6              (x1 == 0 ||
7               x2 < x1 )){
8      pc2 = 2;
9    } else if(pc2 == 2){
10     pc2 = 0;
11     x2 = 0;
12
13   if(pc1==2 && pc2==2){
14     ERROR;
15
16 }
```

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



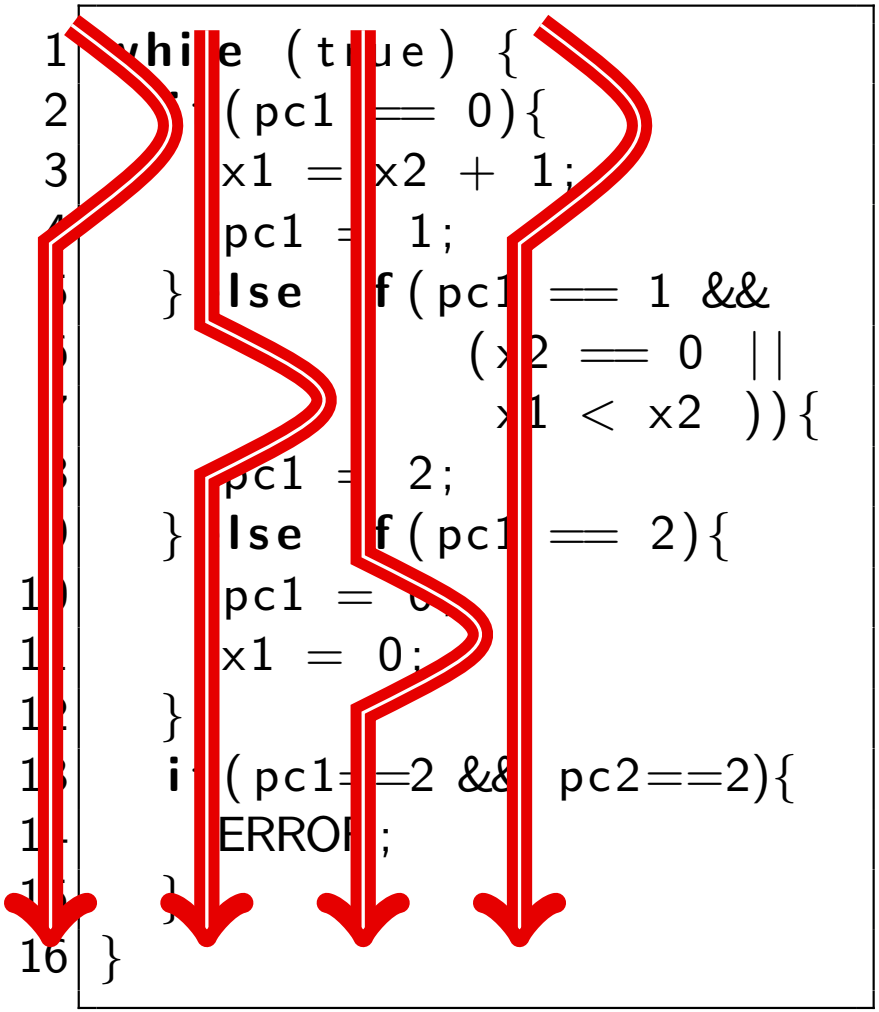
```
1  while (true) {
2    if(pc1 == 0){
3      x1 = x2 + 1;
4      pc1 = 1;
5    } else if(pc1 == 1 &&
6              (x2 == 0 ||
7               x1 < x2 )){
8      pc1 = 2;
9    } else if(pc1 == 2){
10     pc1 = 0;
11     x1 = 0;
12   }
13   if(pc1==2 && pc2==2){
14     ERROR;
15   }
16 }
```

```
1  while (true) {
2    if(pc2 == 0){
3      x2 = x1 + 1;
4      pc2 = 1;
5    } else if(pc2 == 1 &&
6              (x1 == 0 ||
7               x2 < x1 )){
8      pc2 = 2;
9    } else if(pc2 == 2){
10     pc2 = 0;
11     x2 = 0;
12   }
13   if(pc2==2 && pc2==2){
14     ERROR;
15   }
16 }
```

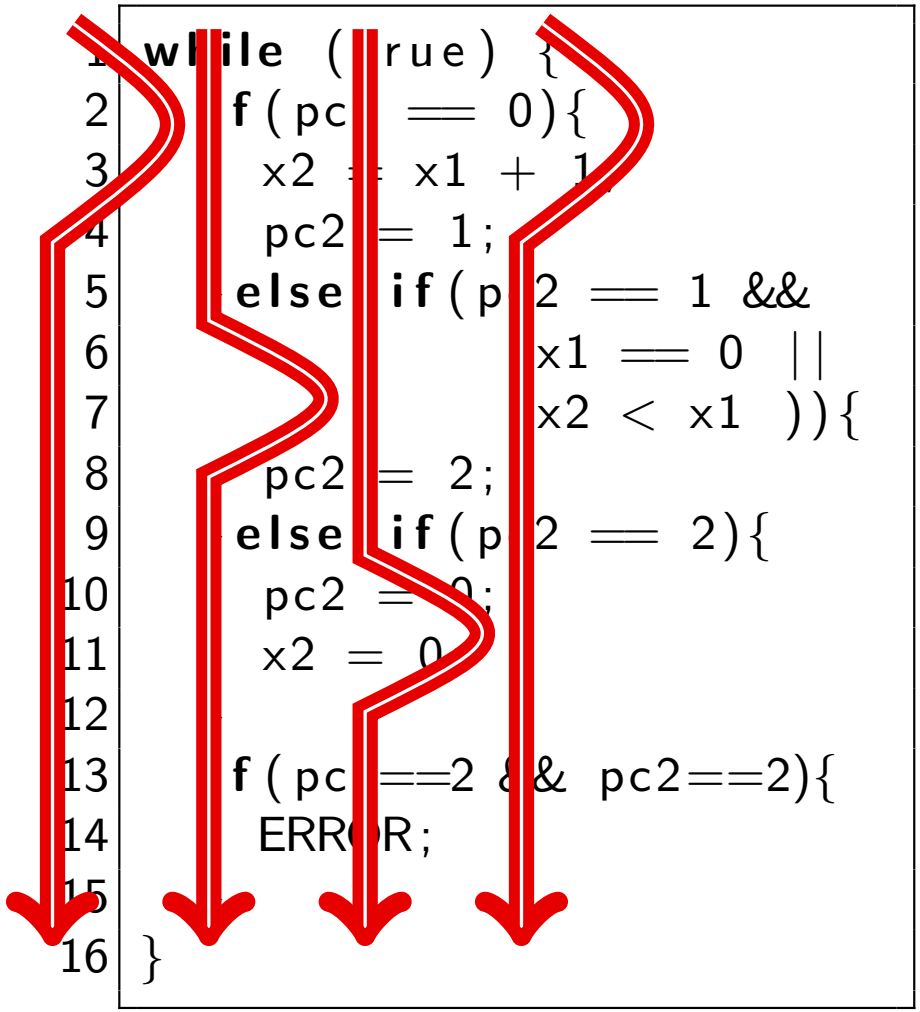
Does it work ?

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



```
1  while (true) {
2    if(pc1 == 0){
3      x1 = x2 + 1;
4      pc1 = 1;
5    } else if(pc1 == 1 &&
6              (x2 == 0 ||
7               x1 < x2 )){
8      pc1 = 2;
9    } else if(pc1 == 2){
10     pc1 = 0;
11     x1 = 0;
12   }
13   if(pc1==2 && pc2==2){
14     ERROR;
15   }
16 }
```

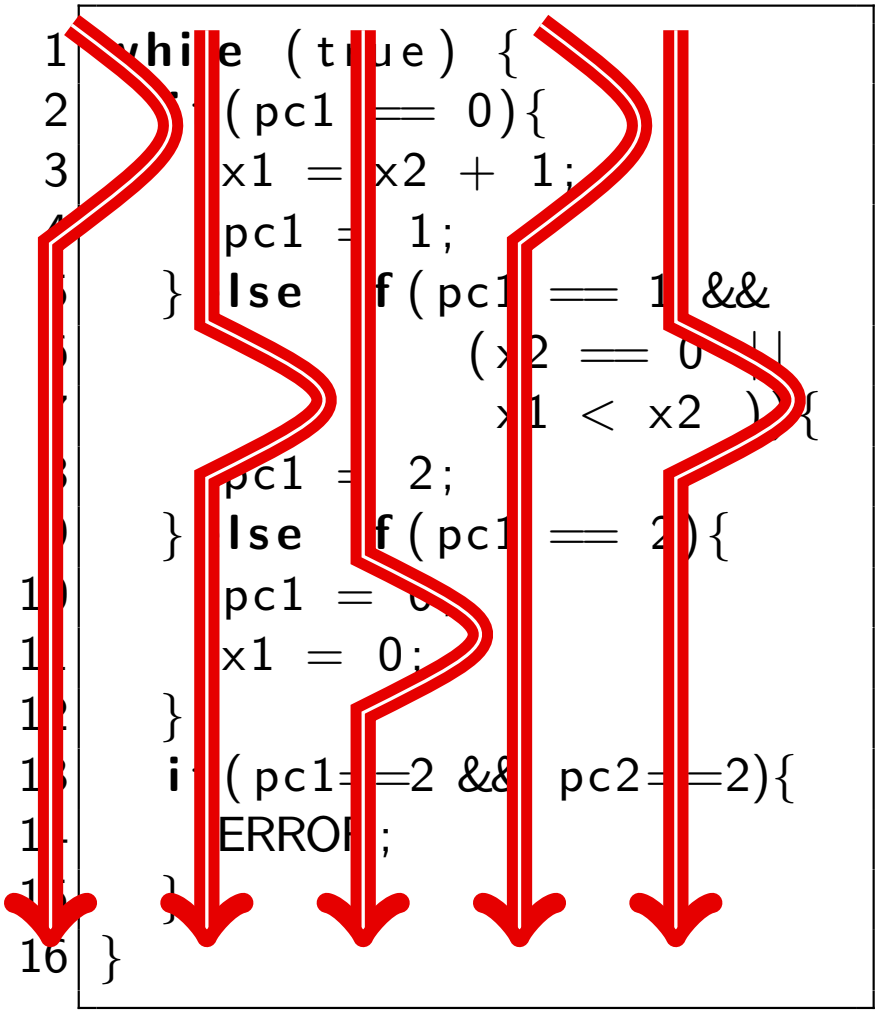


```
1  while (true) {
2    if(pc2 == 0){
3      x2 = x1 + 1;
4      pc2 = 1;
5    } else if(pc2 == 1 &&
6              (x1 == 0 ||
7               x2 < x1 )){
8      pc2 = 2;
9    } else if(pc2 == 2){
10     pc2 = 0;
11     x2 = 0;
12   }
13   if(pc2==2 && pc2==2){
14     ERROR;
15   }
16 }
```

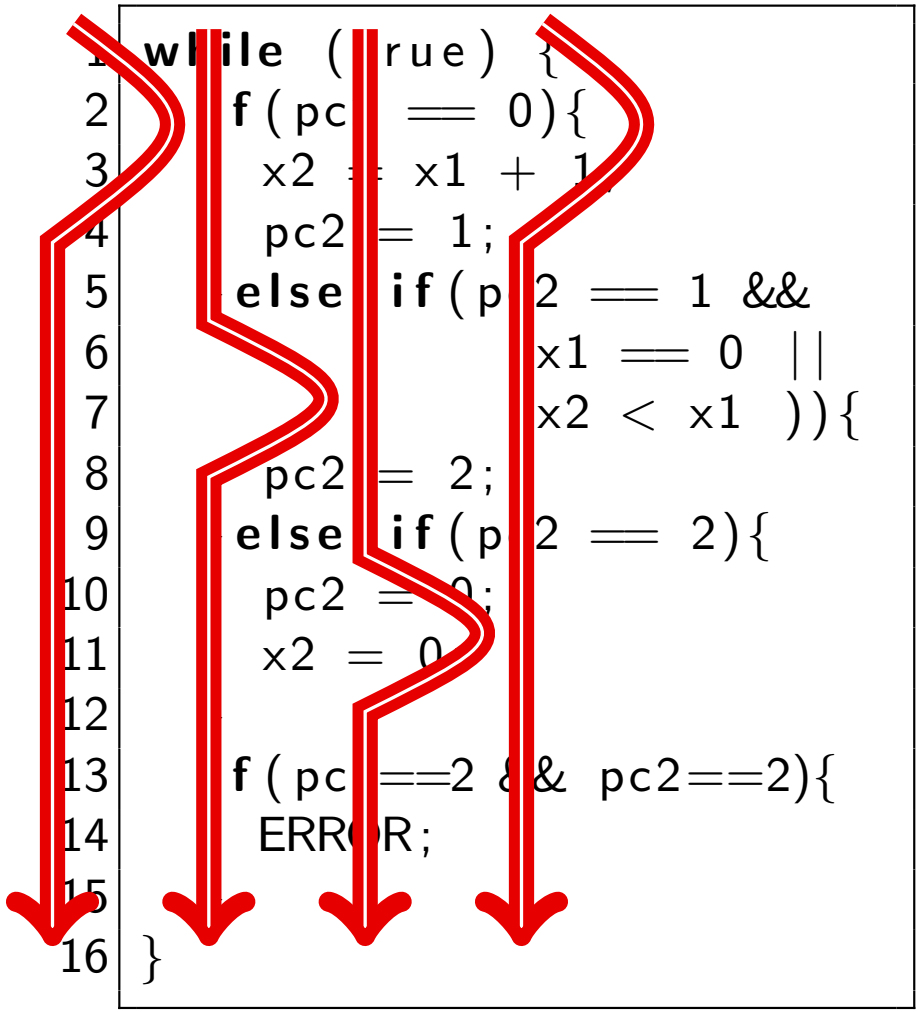
Does it work ?

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



```
1  while (true) {
2    if(pc1 == 0){
3      x1 = x2 + 1;
4      pc1 = 1;
5    } else if(pc1 == 1 &&
6              (x2 == 0 ||
7               x1 < x2 )){
8      pc1 = 2;
9    } else if(pc1 == 2){
10     pc1 = 0;
11     x1 = 0;
12   }
13   if(pc1==2 && pc2==2){
14     ERROR;
15   }
16 }
```

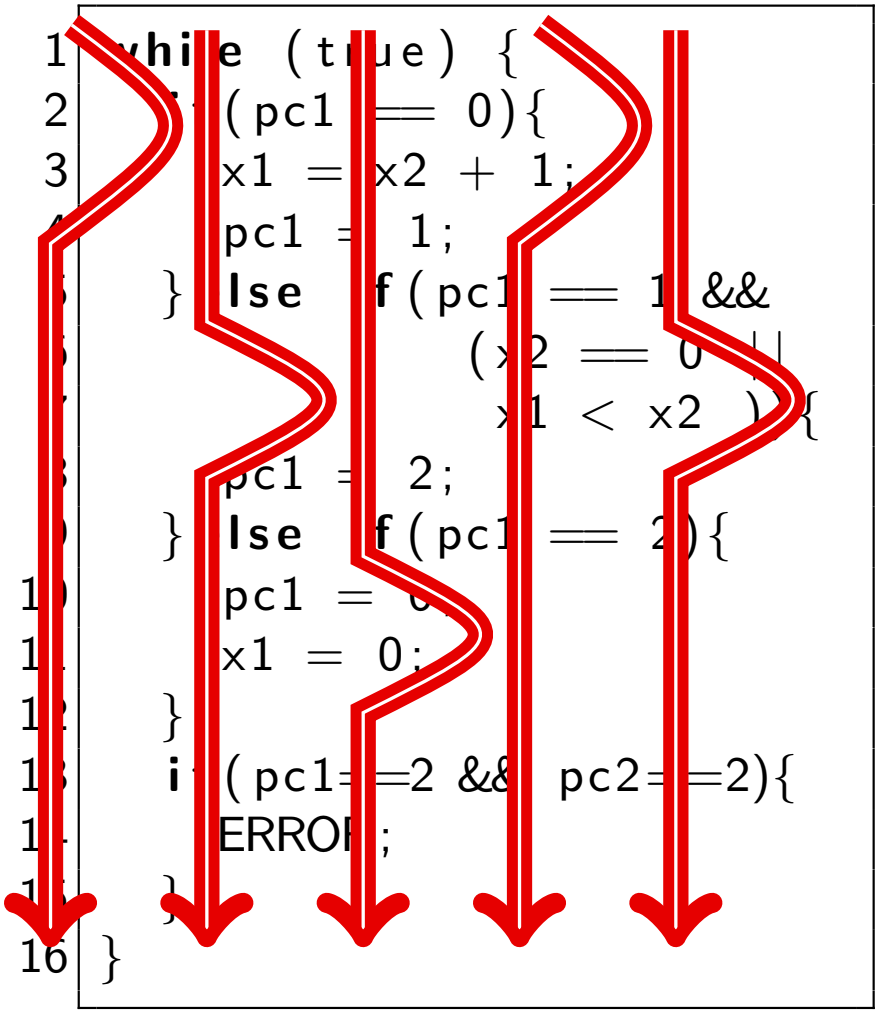


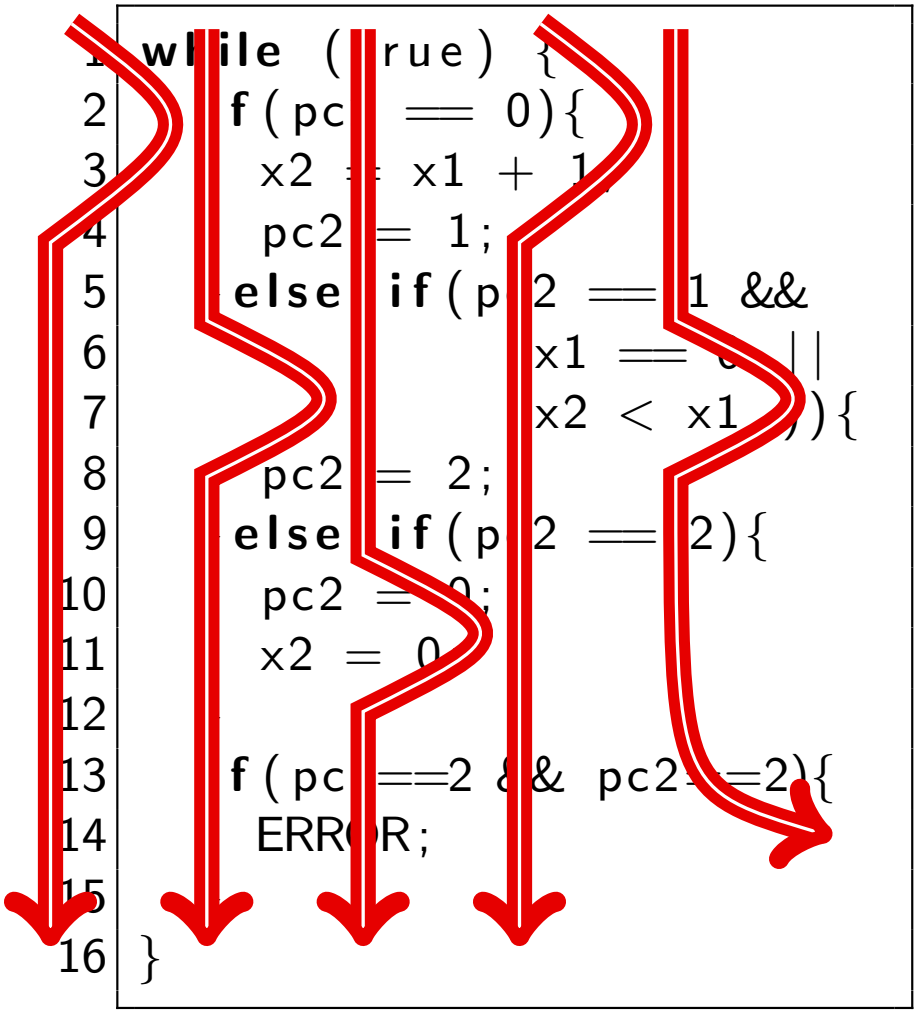
```
1  while (true) {
2    if(pc2 == 0){
3      x2 = x1 + 1;
4      pc2 = 1;
5    } else if(pc2 == 1 &&
6              (x1 == 0 ||
7               x2 < x1 )){
8      pc2 = 2;
9    } else if(pc2 == 2){
10     pc2 = 0;
11     x2 = 0;
12   }
13   if(pc1==2 && pc2==2){
14     ERROR;
15   }
16 }
```

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





Finding out why the cex is spurious (second possibility)

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

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

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

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

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

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

Finding out why the cex is spurious (second possibility)

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

$$x_2 = 0$$

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

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

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

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

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

Finding out why the cex is spurious (second possibility)

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

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

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

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

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

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

$$x_2 = 0$$

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

Finding out why the cex is spurious (second possibility)

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

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

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

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

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

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

$$x_2 = 0$$

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

$$x_1' > 0$$

Finding out why the cex is spurious (second possibility)

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

$$x_2 = 0$$

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

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

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

$$x_1' > 0$$

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

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

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

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

Finding out why the cex is spurious (second possibility)

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

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

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

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

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

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

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

```
1  while (true) {
2    if(pc1 == 0){
3      x1 = x2 + 1;
4      pc1 = 1;
5    }else if(pc1 == 1 &&
6                (x2 == 0 ||
7                 x1 < x2 )){
8      pc1 = 2;
9    }else if(pc1 == 2){
10     pc1 = 0;
11     x1 = 0;
12   }
13   if(pc1==2 && pc2==2){
14     ERROR;
15   }
16 }
```

```
1  while (true) {
2    if(pc2 == 0){
3      x2 = x1 + 1;
4      pc2 = 1;
5    }else if(pc2 == 1 &&
6                (x1 == 0 ||
7                 x2 < x1 )){
8      pc2 = 2;
9    }else if(pc2 == 2){
10     pc2 = 0;
11     x2 = 0;
12   }
13   if(pc1==2 && pc2==2){
14     ERROR;
15   }
16 }
```

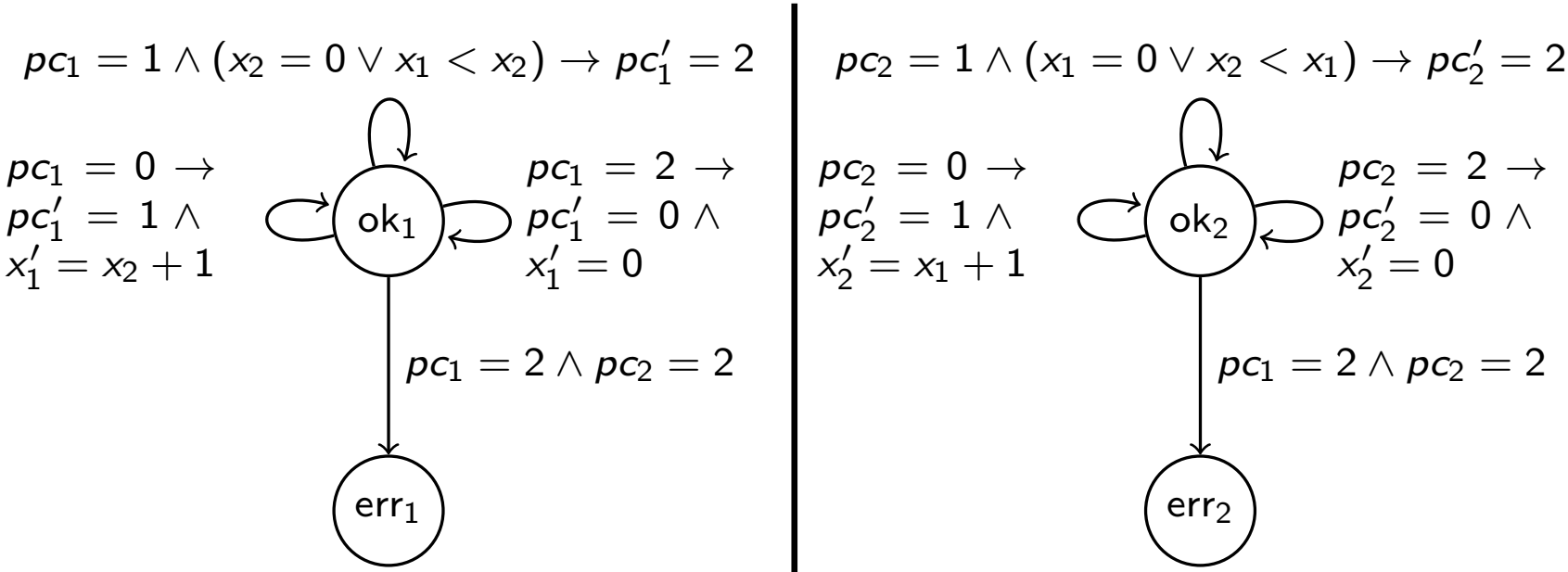
Building an actual proof

state-space: 2 * 16 loc and 8 predicates $\Rightarrow 16^2 * 2^8 = 65536$ states

Building an actual proof

state-space: 2 * 16 loc and 8 predicates $\Rightarrow 16^2 * 2^8 = 65536$ states

simpler version:

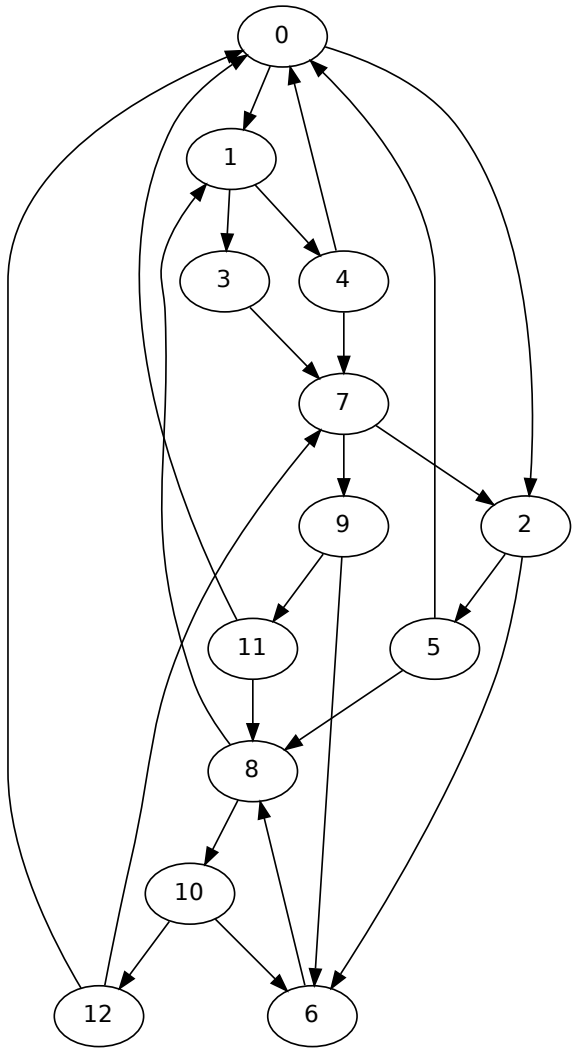


state-space: $2^2 * 2^8 = 1024$ states

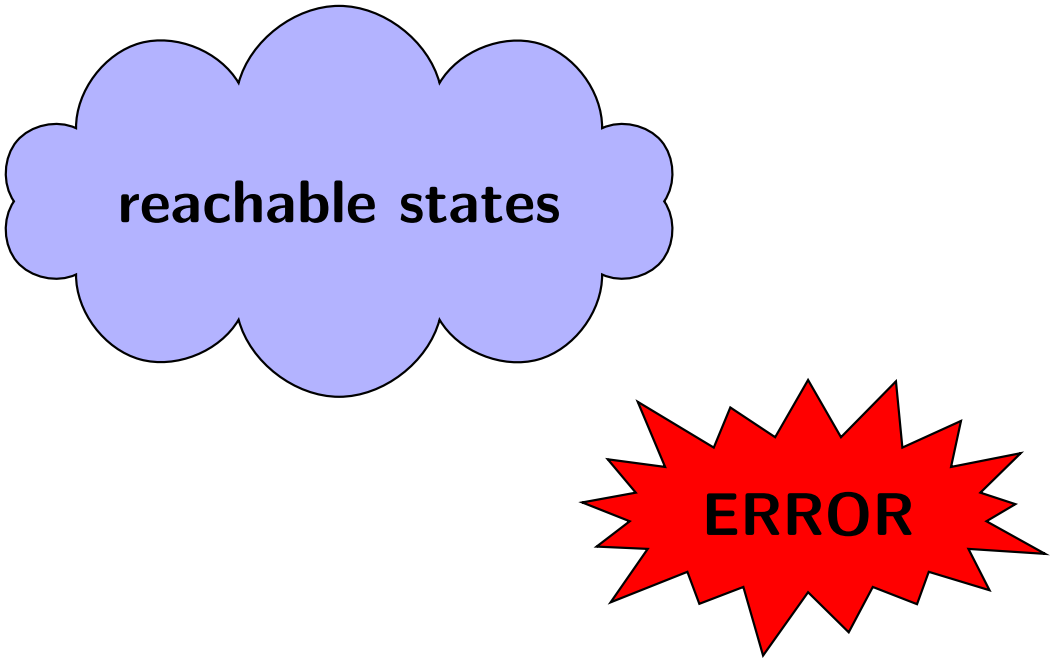
By being clever with the predicates we can go down to 432 states (12 are reachable).

Reachability graph

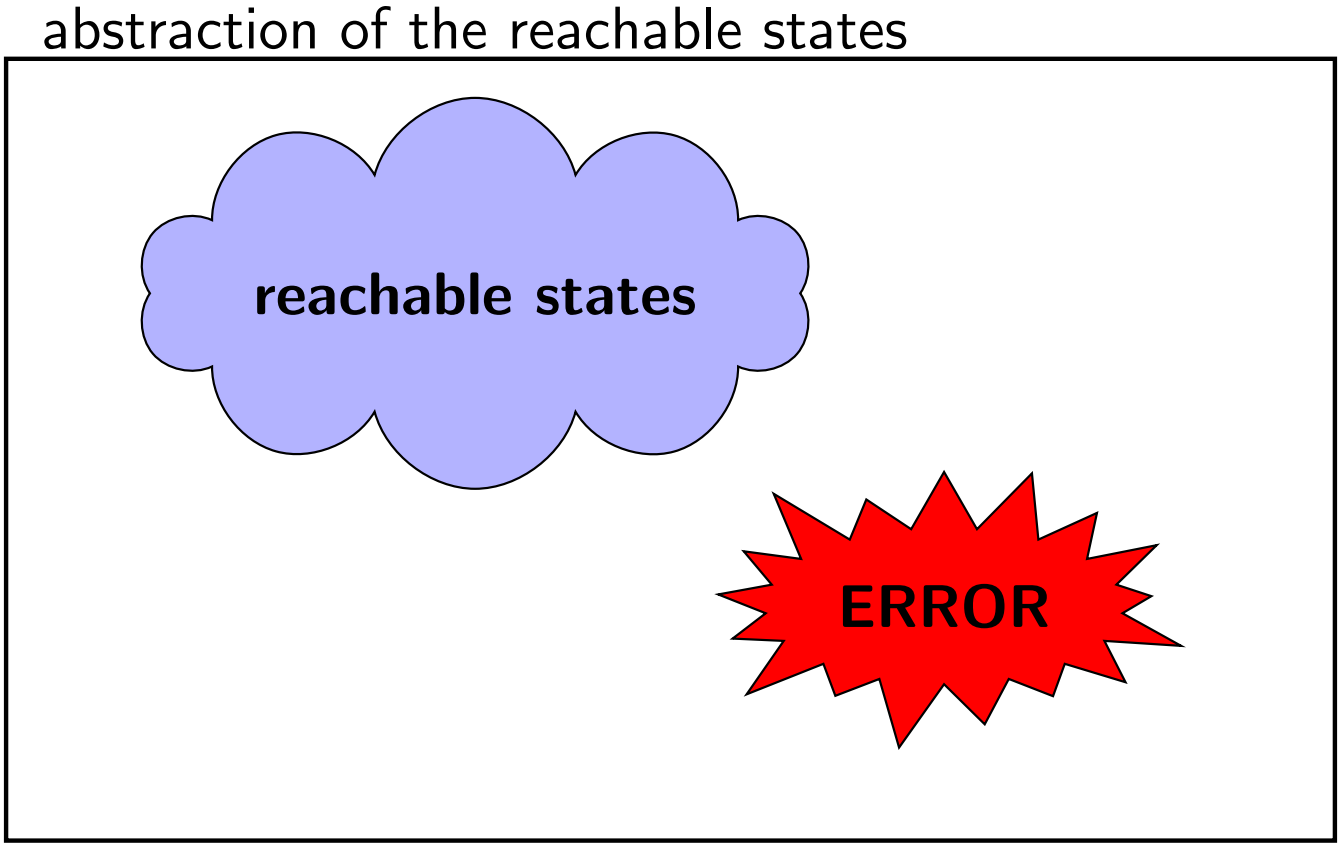
id	pc1	pc2	$x_1 = 0$	$x_2 = 0$	$x_1 ? x_2$
0	0	0	$\top$	$\top$	$x_1 = x_2$
1	1	0	$\bot$	$\top$	$x_1 > x_2$
2	0	1	$\top$	$\bot$	$x_1 < x_2$
3	1	1	$\bot$	$\bot$	$x_1 < x_2$
4	2	0	$\bot$	$\top$	$x_1 > x_2$
5	0	2	$\top$	$\bot$	$x_1 < x_2$
6	1	1	$\bot$	$\bot$	$x_1 > x_2$
7	2	1	$\bot$	$\bot$	$x_1 < x_2$
8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



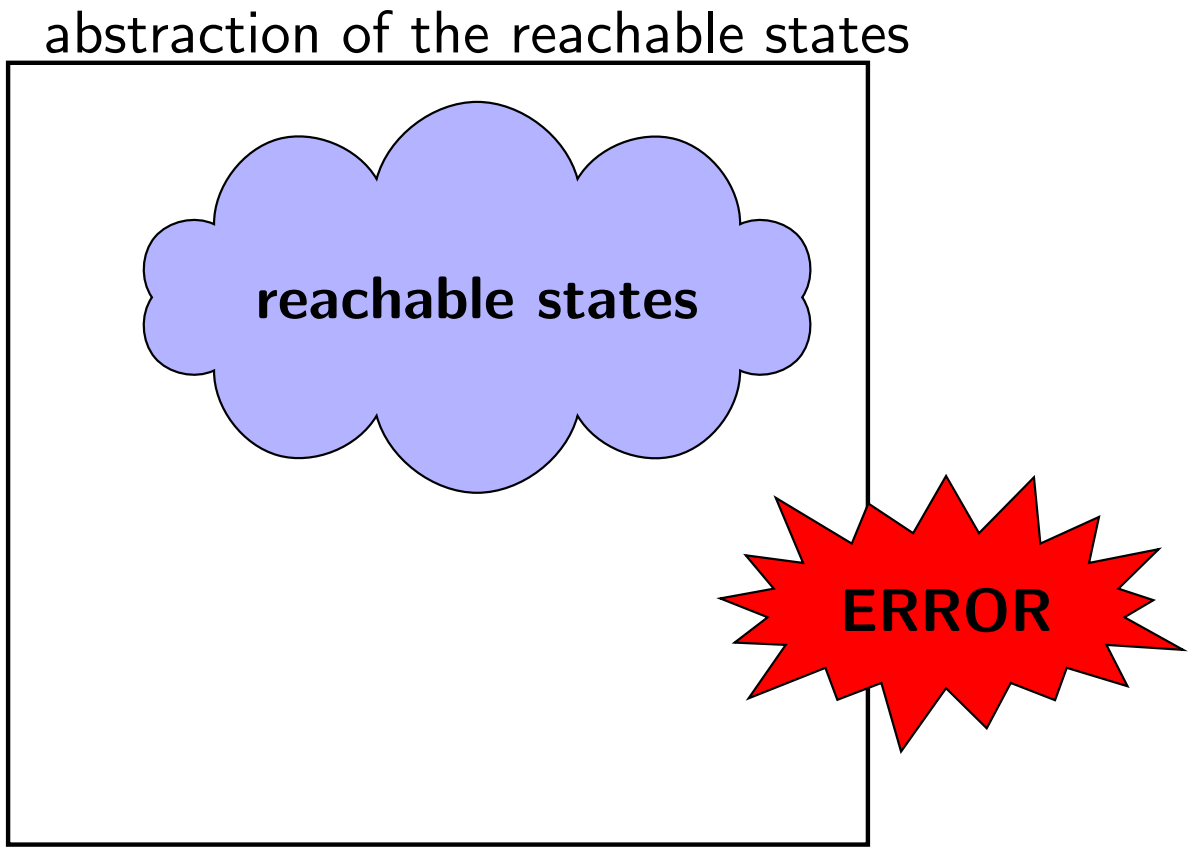
Why is it a proof ?



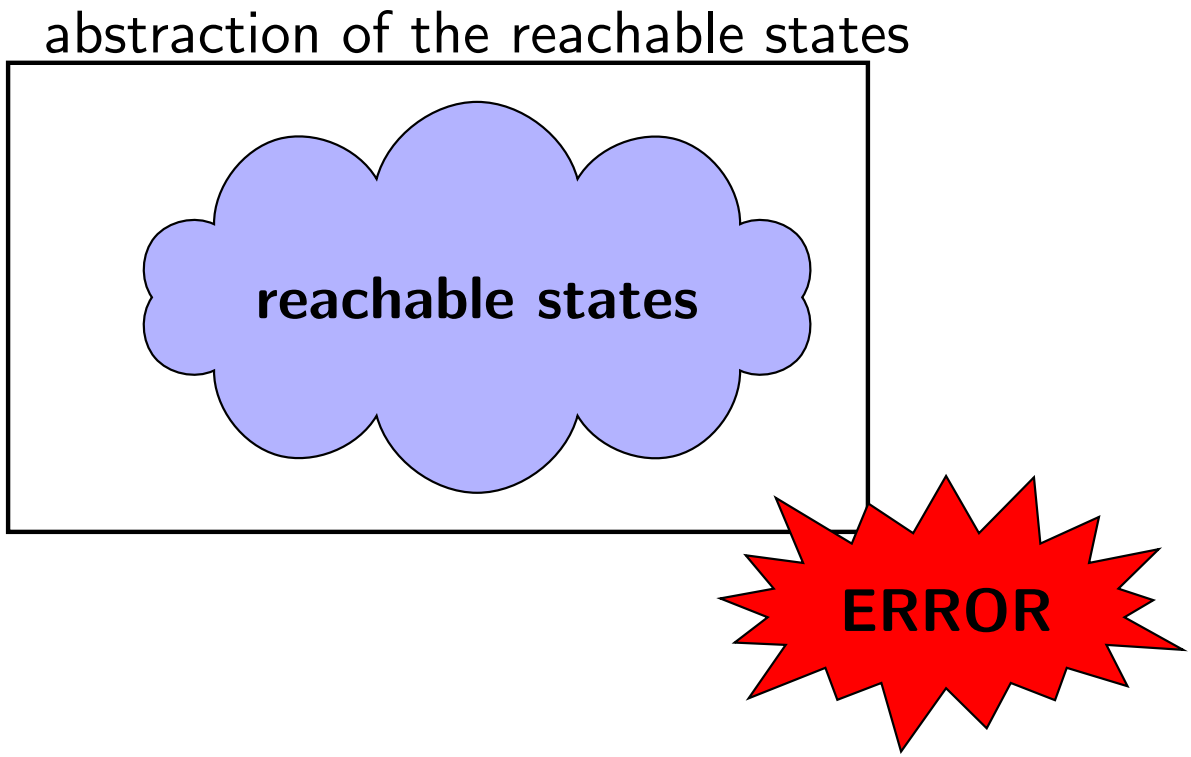
Why is it a proof ?



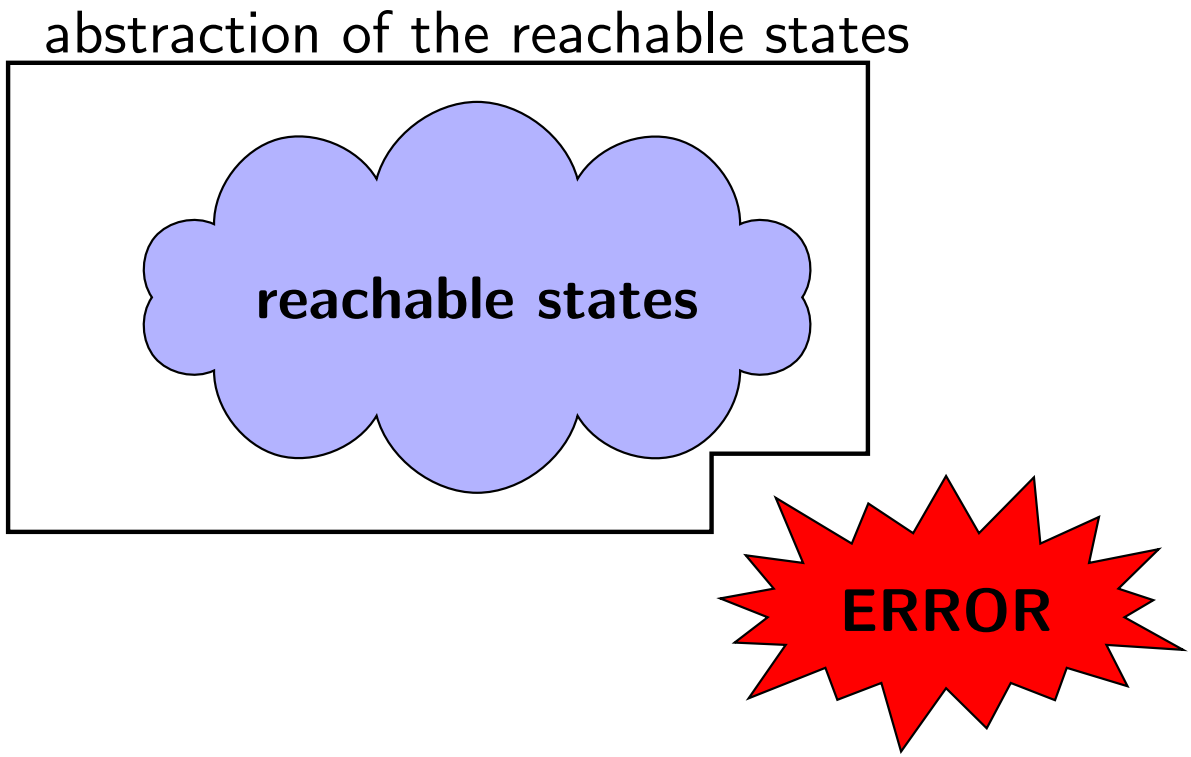
Why is it a proof ?



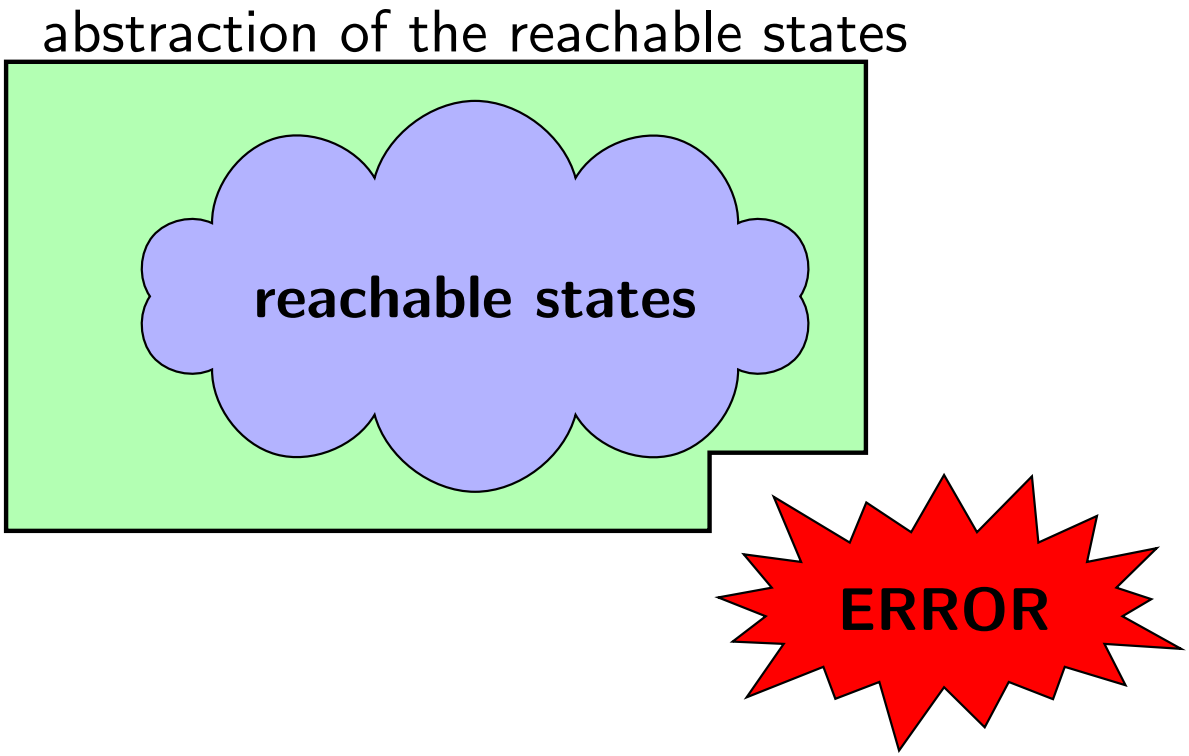
Why is it a proof ?



Why is it a proof ?



Why is it a proof ?



Questions ?