Static analysis
Finding bugs to building proofs

Franck Cassez

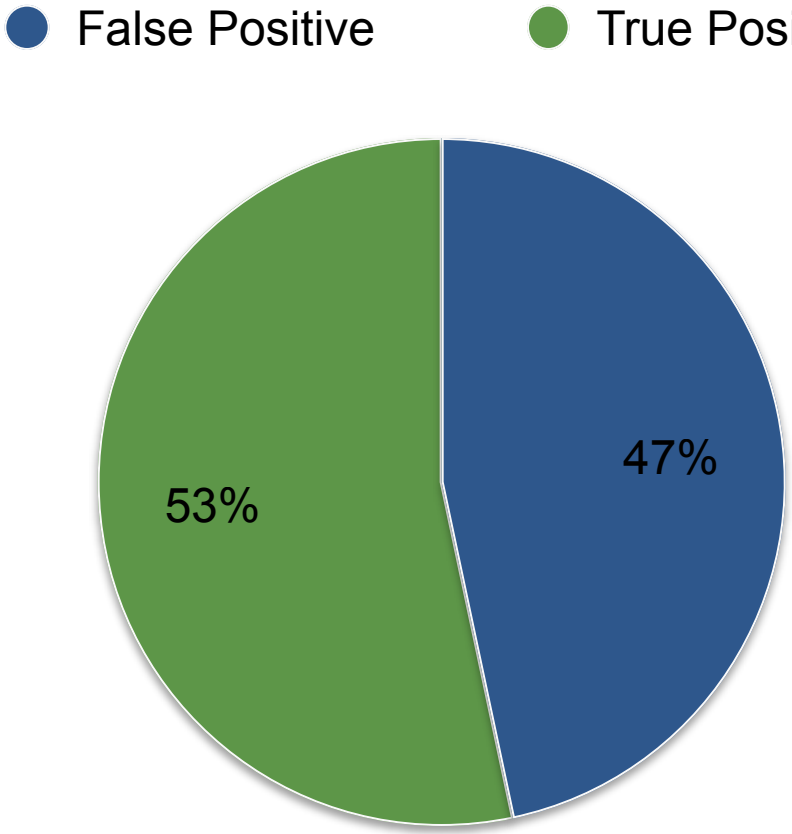
Programming Languages and Verification Research Group

Department of Computing

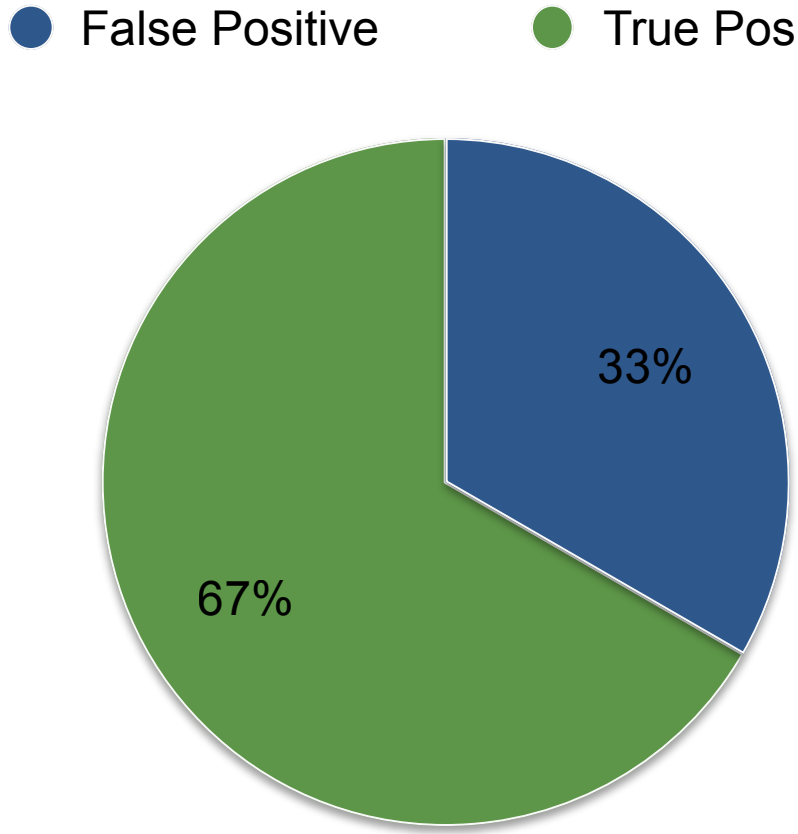
MACQUARIE University
SYDNEY·AUSTRALIA

Commercial Static Analysers

Asterisk: 30 selected

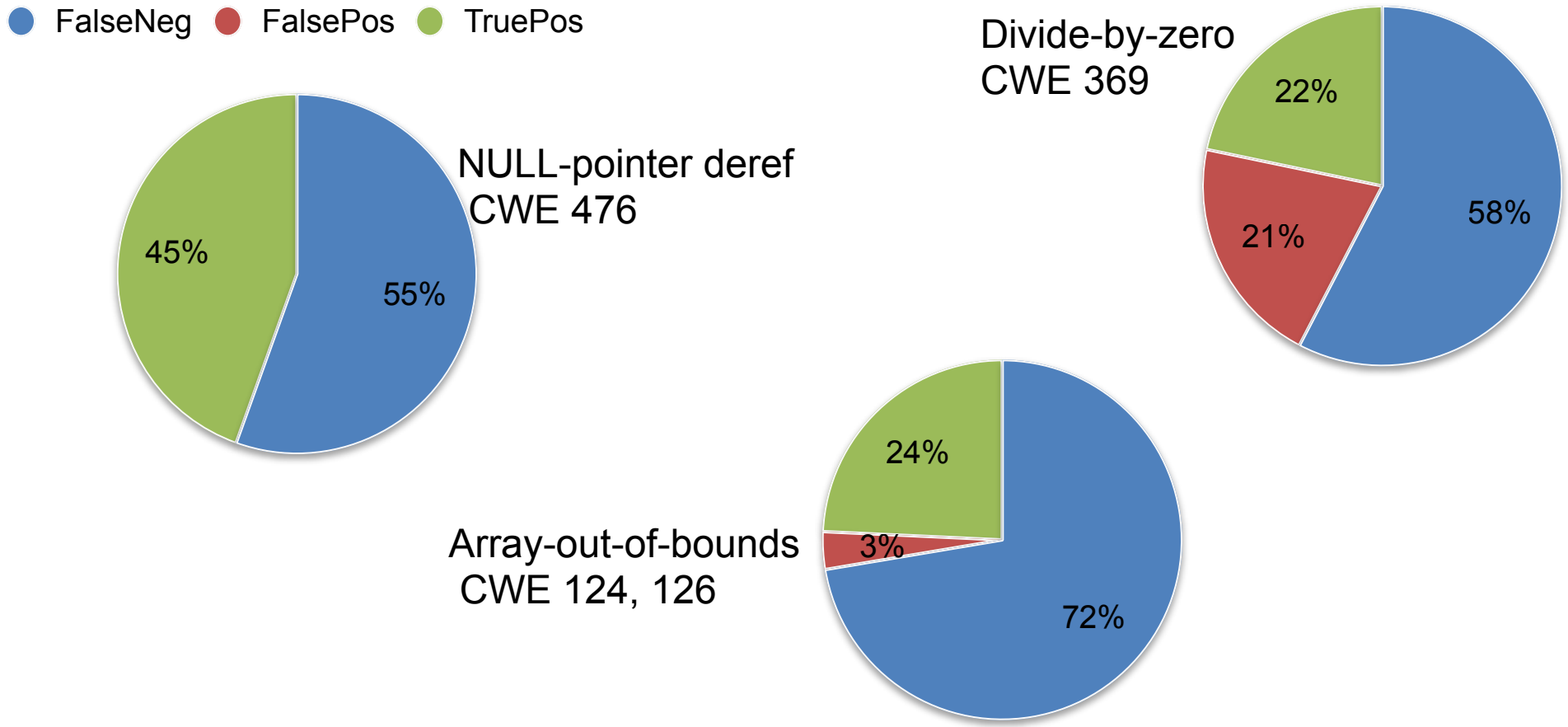


Wireshark: 30 selected



NIST SATE V Workshop, 2014

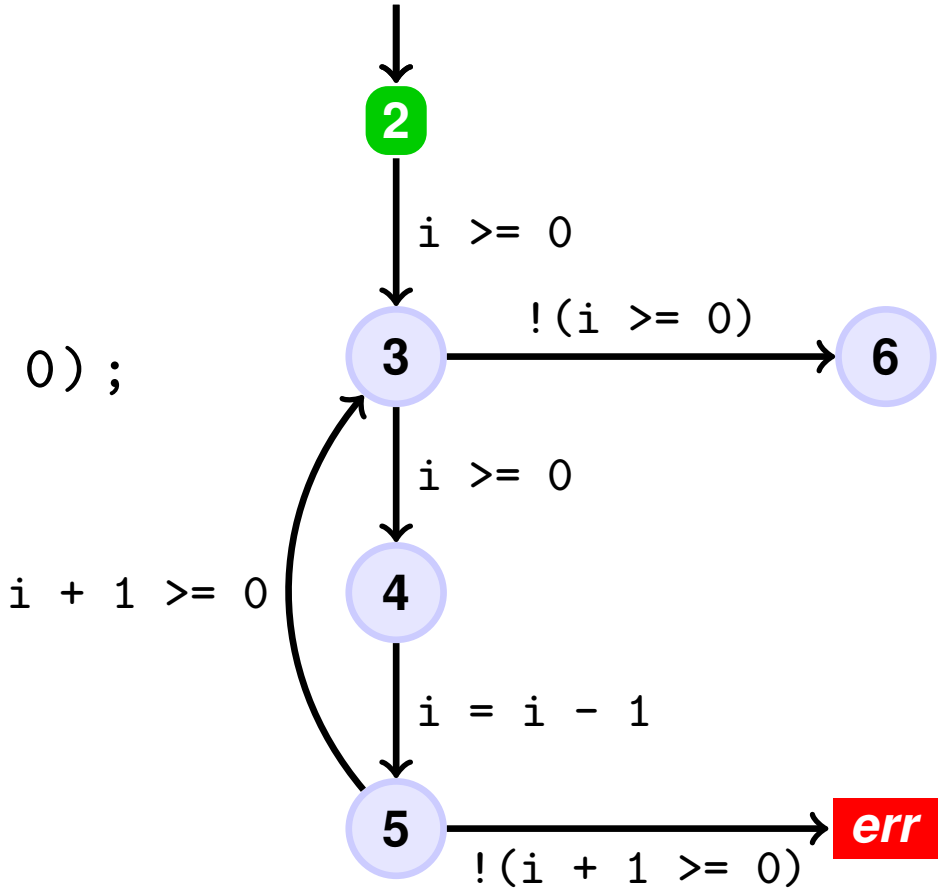
Results Juliet Test Suite (NIST)



NIST SATE V Workshop, 2014

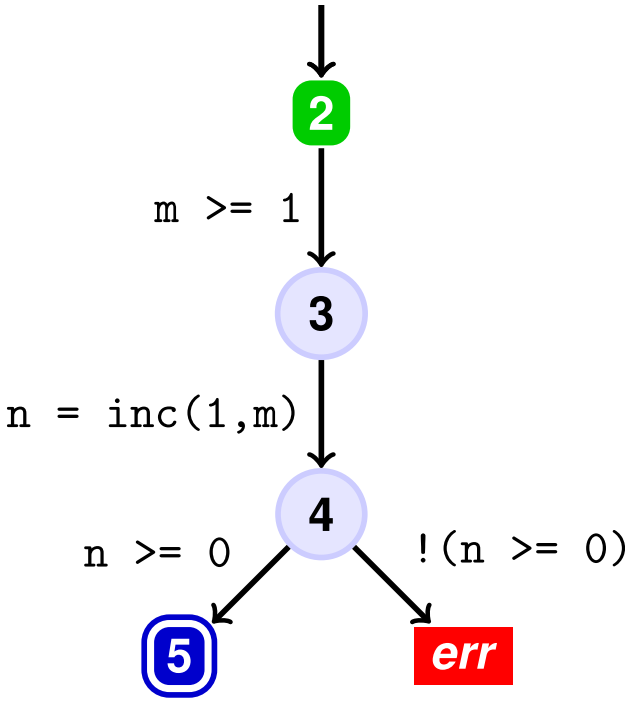
Intra-procedural

```
1   var i:int;
2   assume i >= 0;
3   while (i >= 0) do
4     i = i - 1;
5     assert (i + 1 >= 0);
6   done;
```

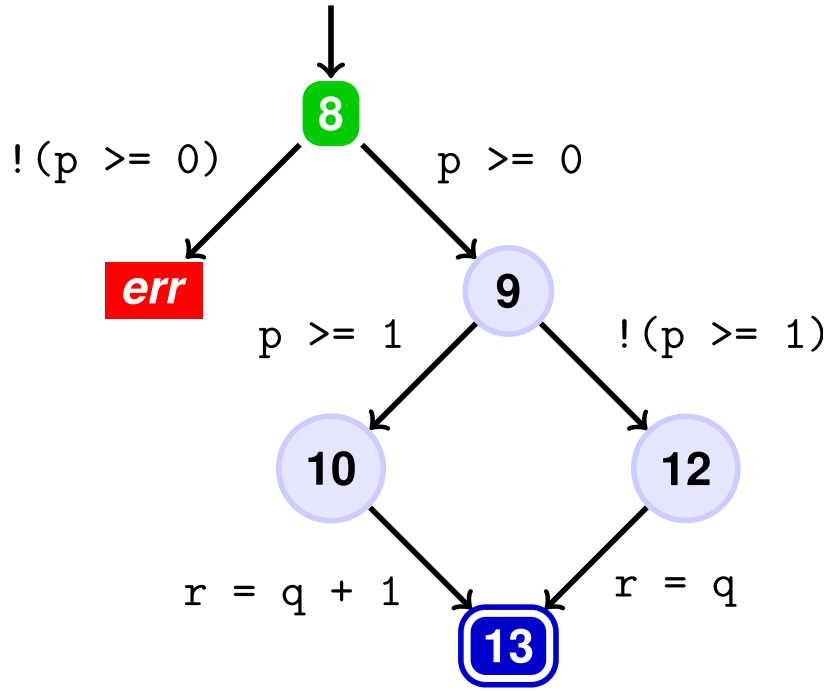


Inter-procedural

```
1   proc main() : (n) {
2     assume(m >= 1);
3     n = inc(1, m);
4     assert(n >= 0);
5   }
```

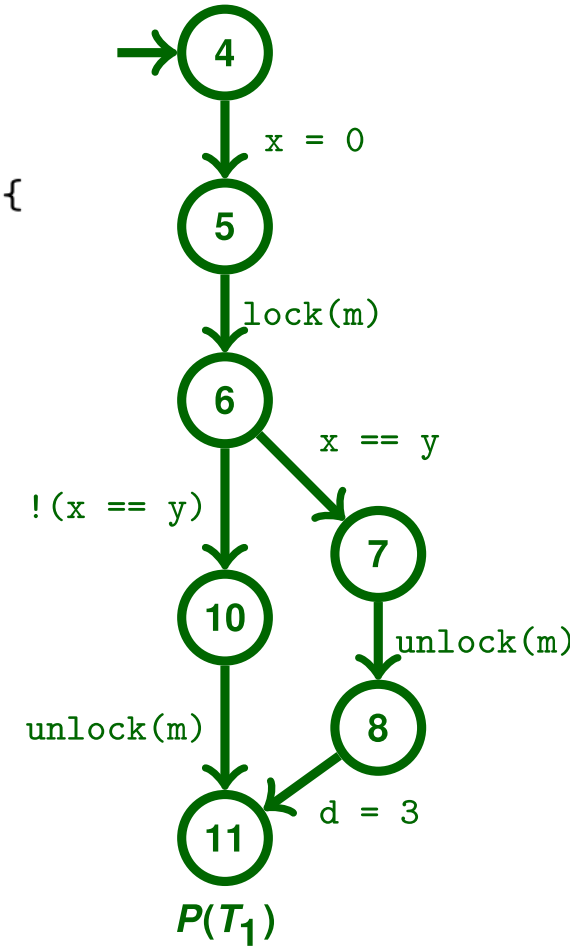


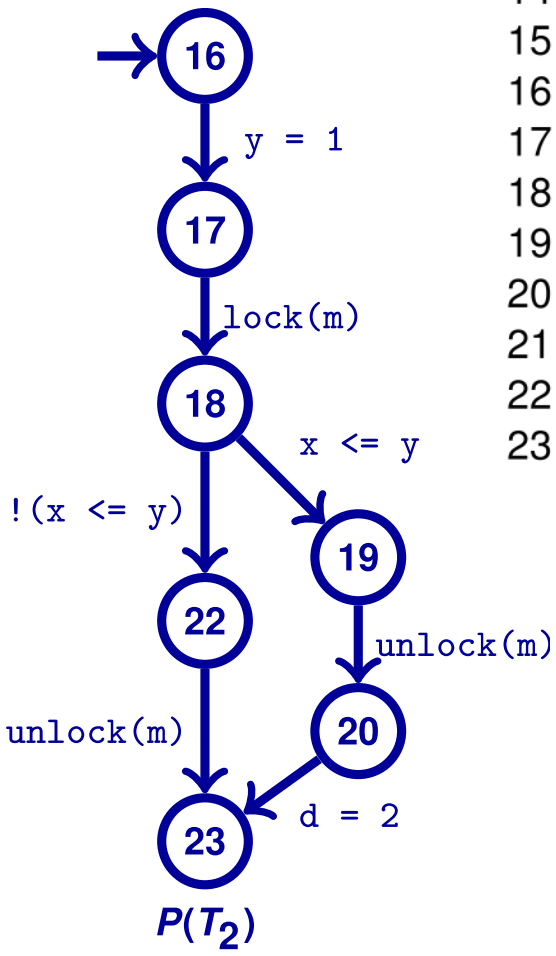

```
 7  proc inc(p,q) : (r) {
 8    assert(p >= 0);
 9    if (p >= 1)
10      r = q + 1;
11    else
12      r = q;
13    endif;
14  }
```



Multiple threads

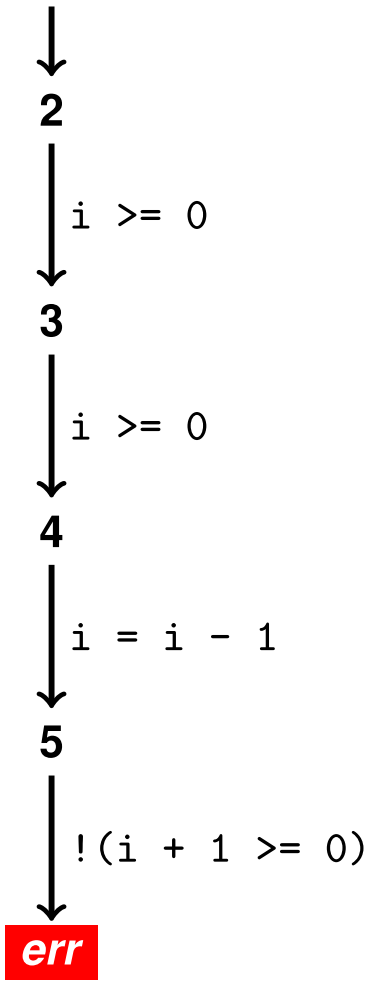
```
2   // thread T1
3   thread T1
4   x = 0;
5   lock(m);
6   if (x == y) {
7     unlock(m);
8     d = 3;
9   } else {
10    unlock(m);
11  }
12  /* end */
```

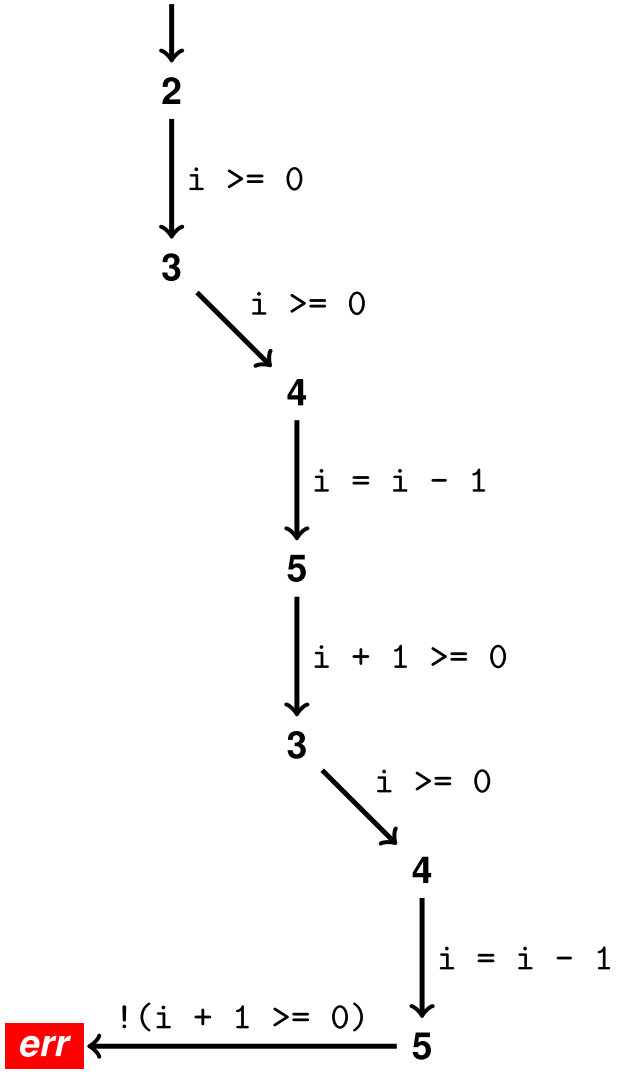


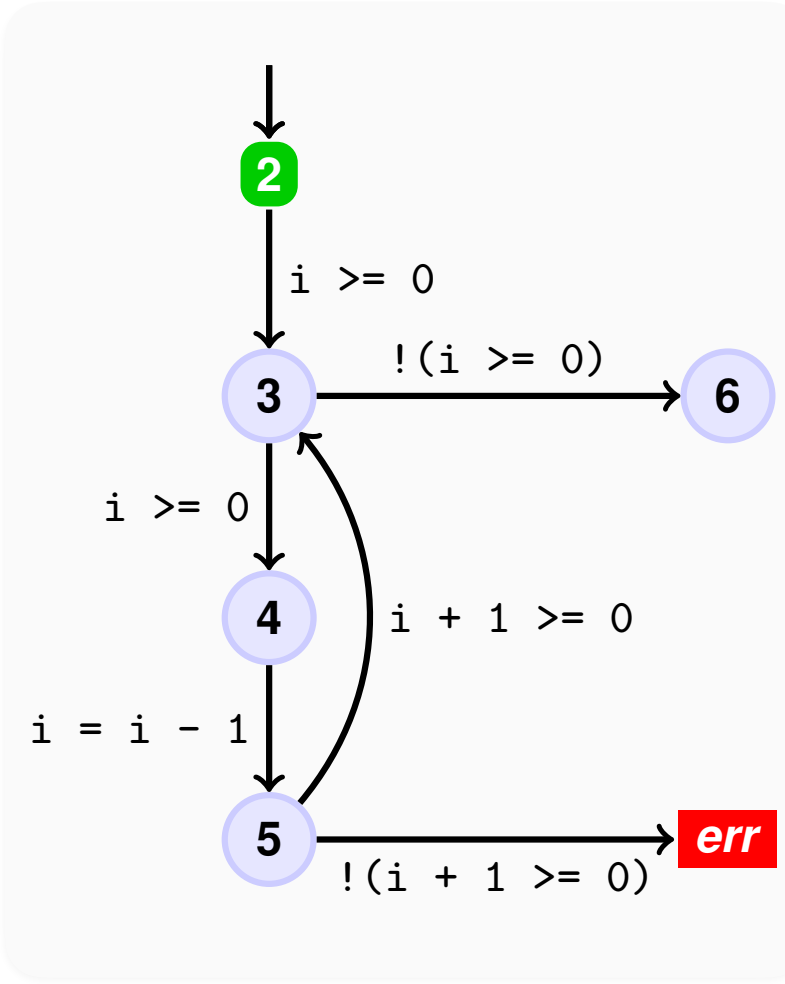



```
14  // Thread T2
15  thread T2
16  y = 1;
17  lock(m);
18  if (x <= y) {
19    unlock(m);
20    d = 2;
21  } else {
22    unlock(m);
23  }
```

Trace Abstraction Refinement

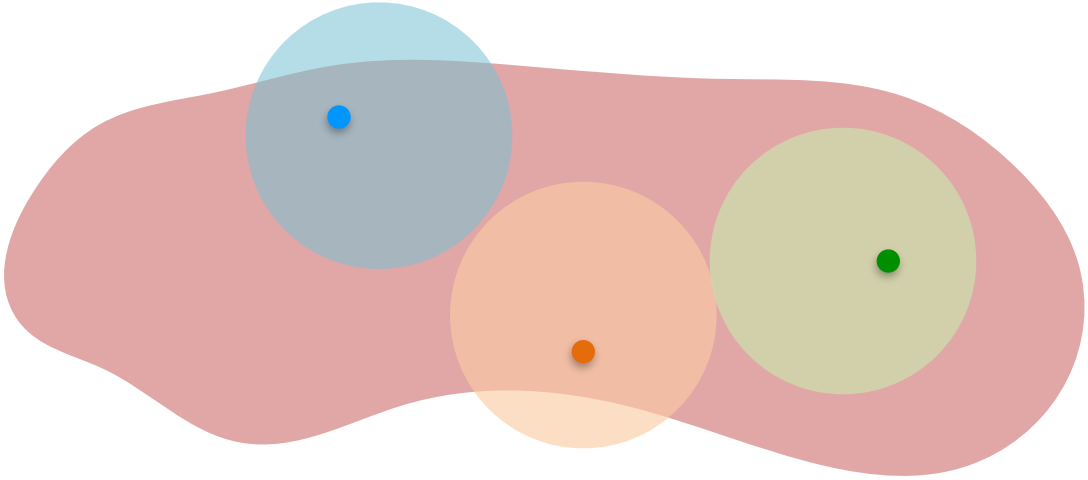


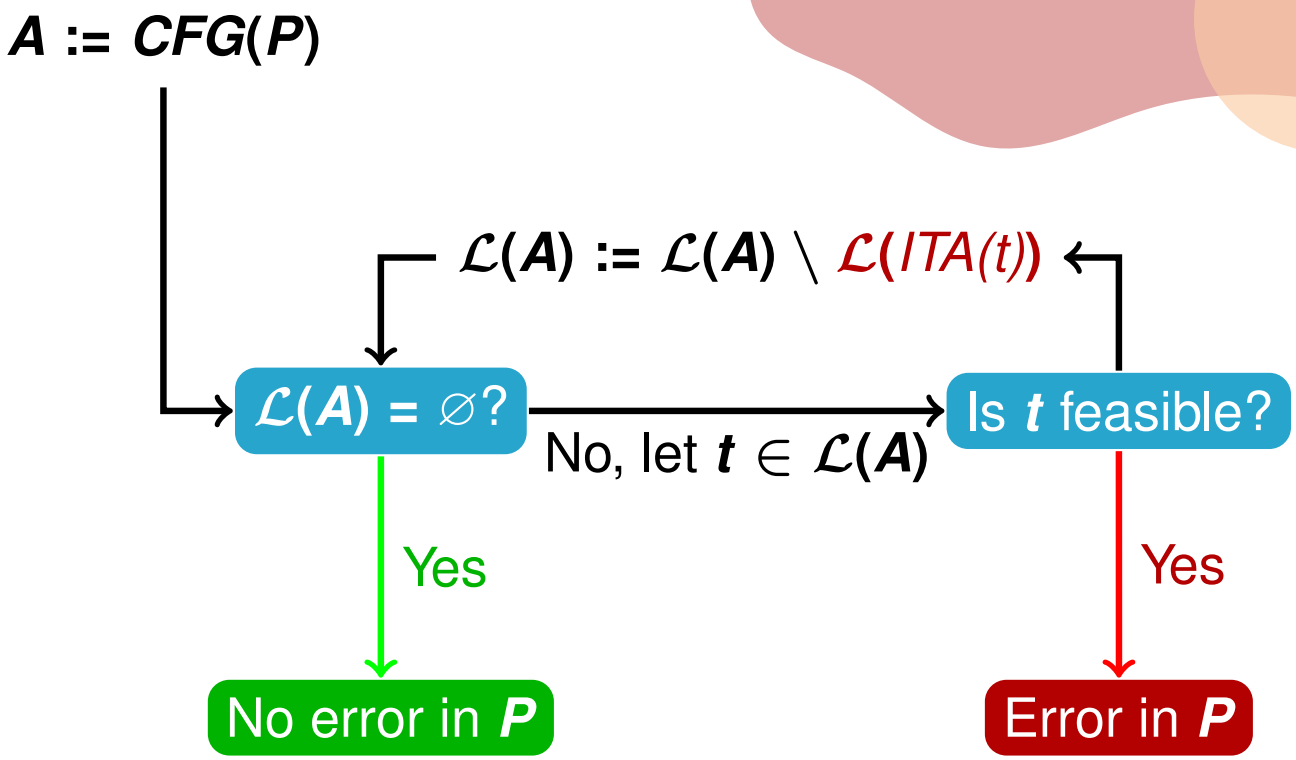




Trace Abstraction Refinement (cont'd)

Heizmann, M., Hoenicke, J., Podelski, A.
Refinement of trace abstraction.
Static Analysis Symposium, 2009.

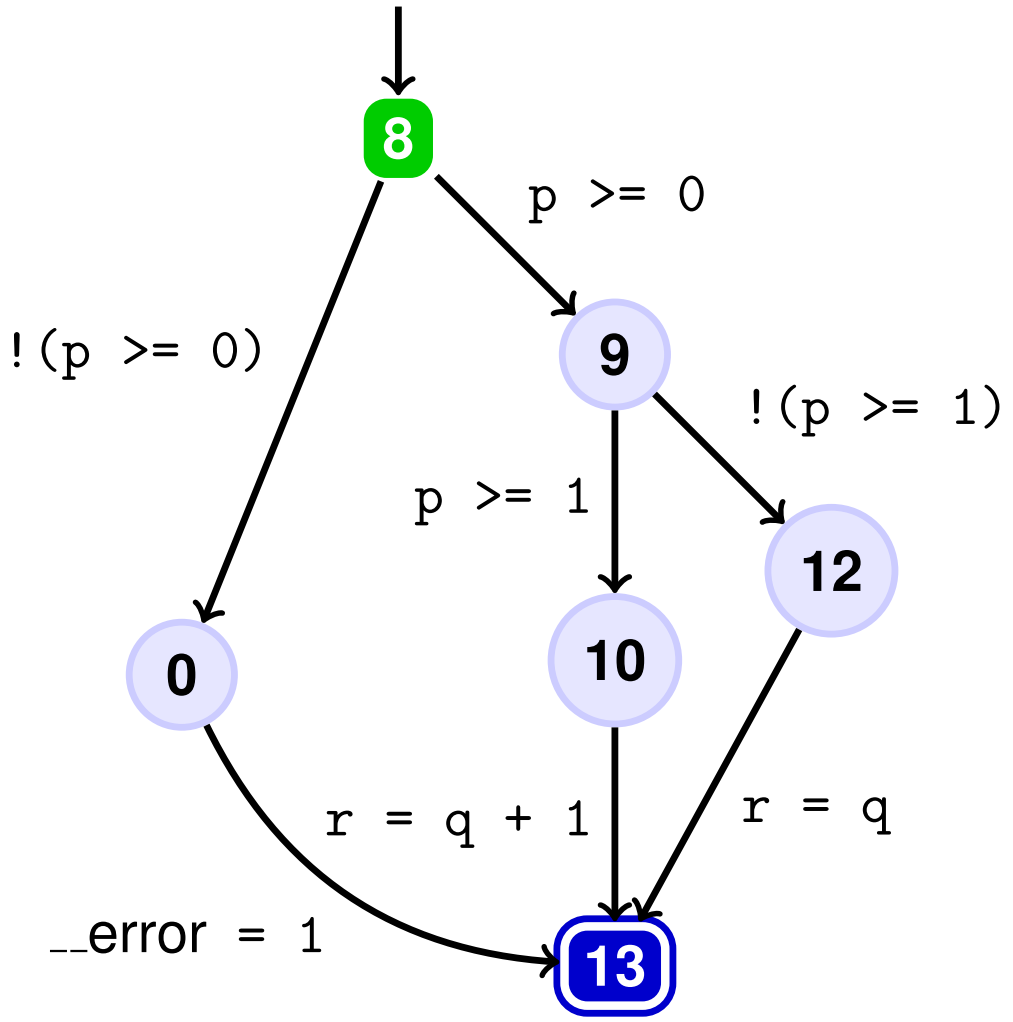




Partial Correctness

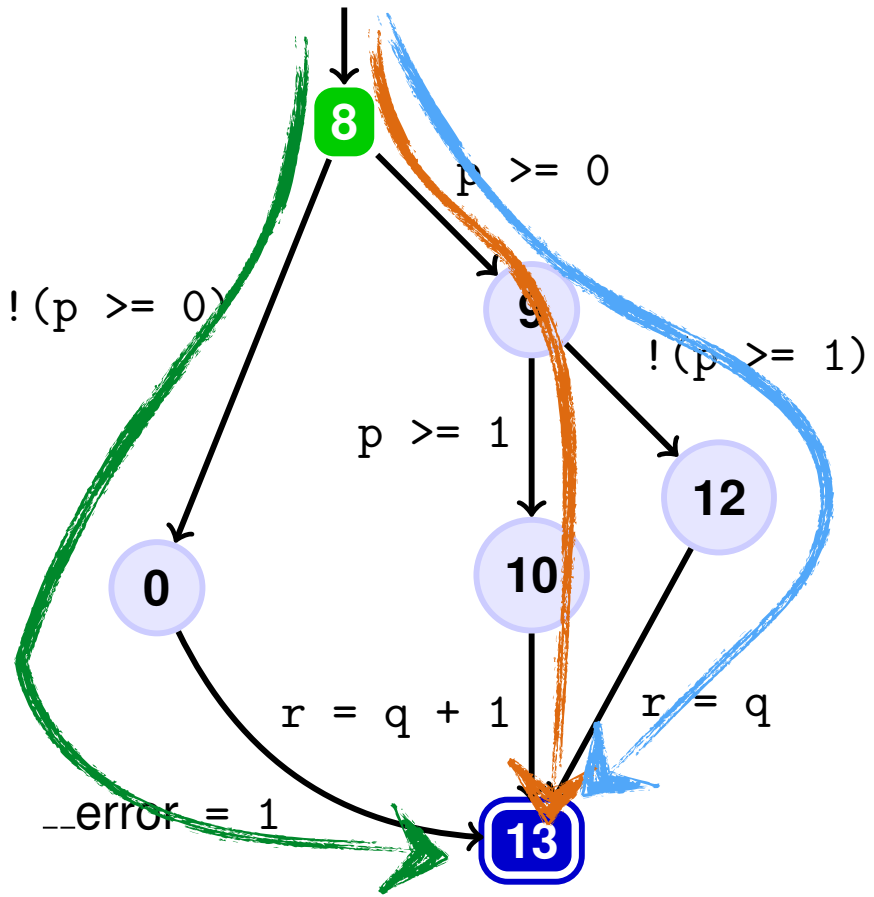
```
 7  proc inc(p,q) : (r) {
 8    assert(p >= 0);
 9    if (p >= 1)
10      r = q + 1;
11    else
12      r = q;
13    endif;
14  }
```

{¬__error} inc {¬__error}



Building Hoare Triples

$\{p = 1 \wedge \neg\_\_error\}\ \boldsymbol{A}_{inc}\ \{\neg\_\_error\}$



$\{p \geq \mathbf{0}\} \rightarrow \{\neg\_\_error\}$

$\{\neg\_\_error\} \rightarrow \{\neg\_\_error\}$

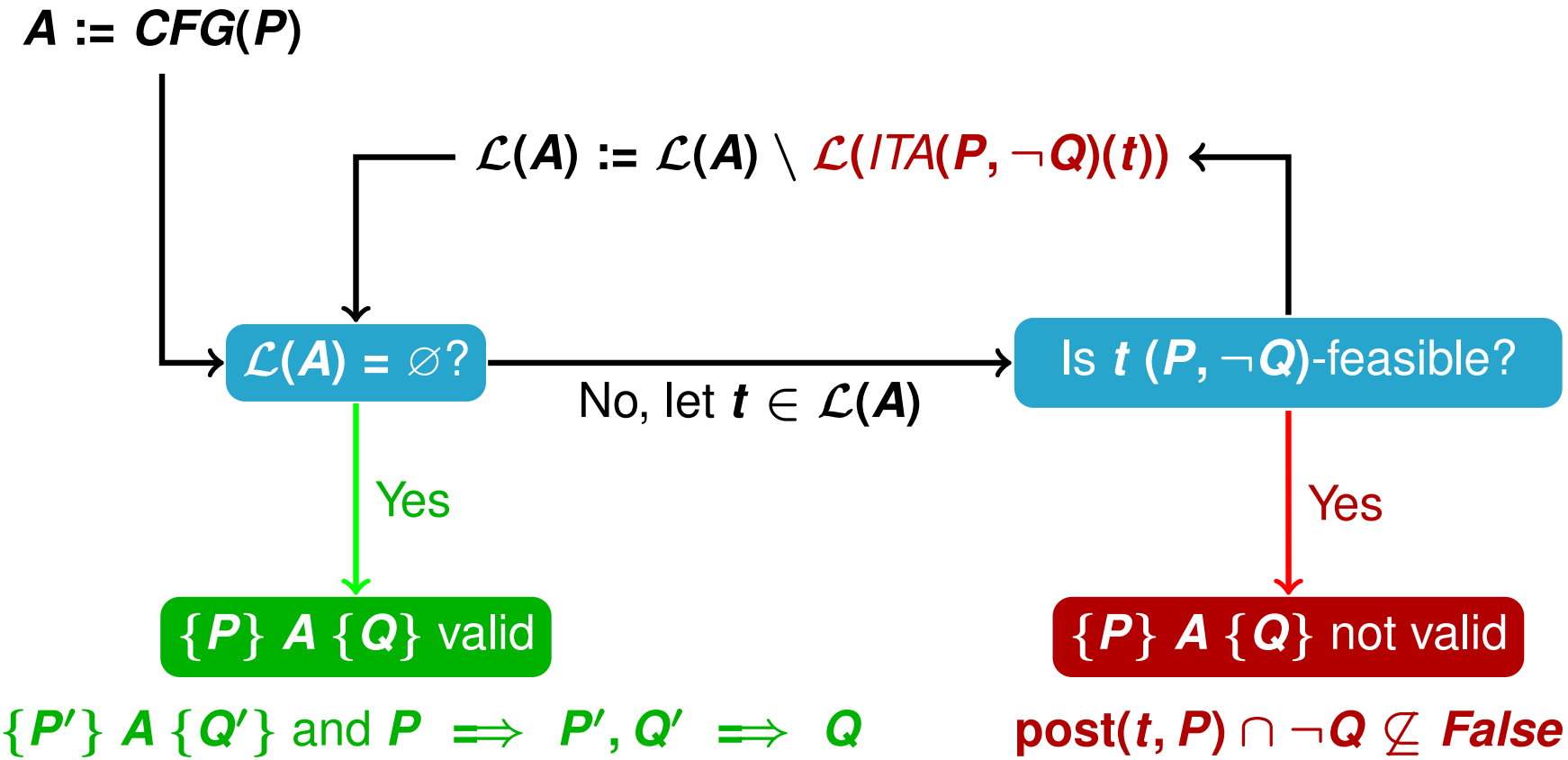
$\{\neg\_\_error\} \rightarrow \{\neg\_\_error\}$

$\{p \geq 0 \wedge \neg\_\_error\}\ \boldsymbol{A}_{inc}\ \{\neg\_\_error\}$

$\{p = 1 \wedge \neg\_\_error\}\ \boldsymbol{A}_{inc}\ \{\neg\_\_error\}$

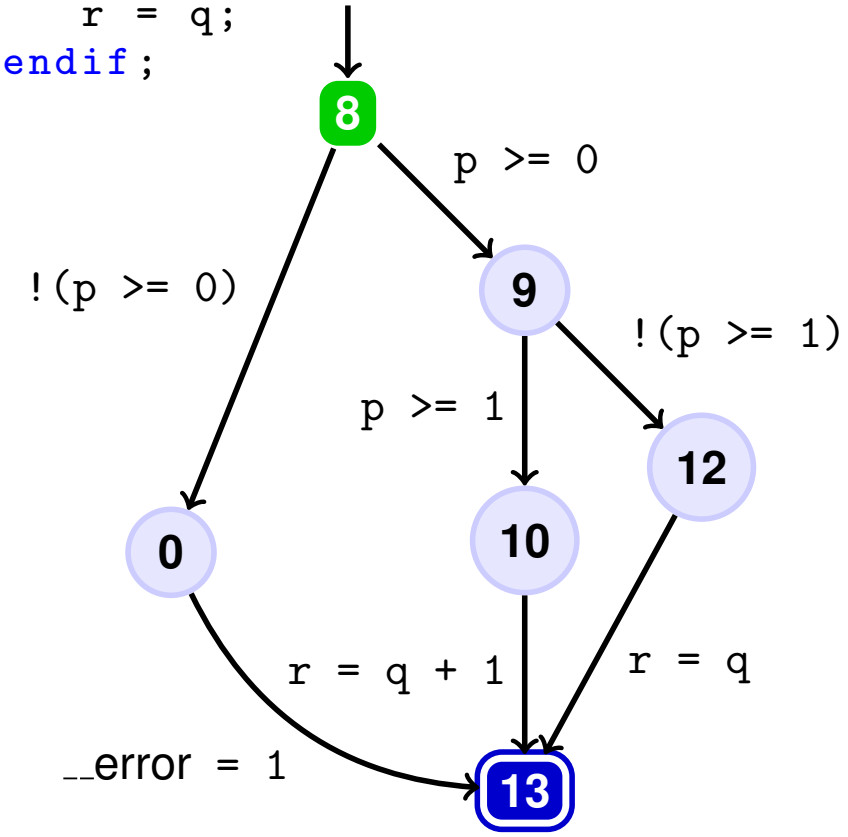
Algorithm 1

Check triple $\{P\}\ A\ \{Q\}$

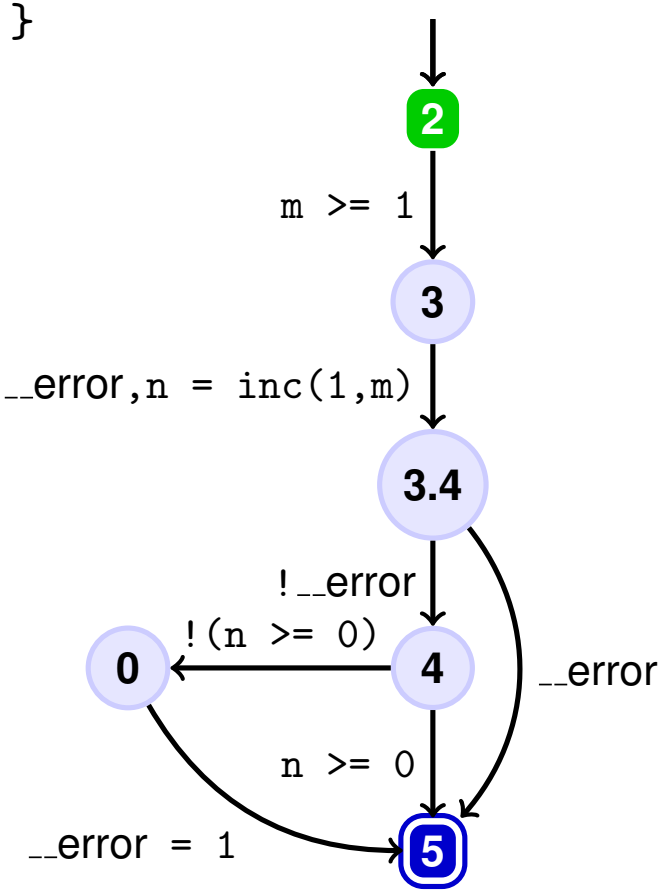


Extended CFGs

```
 7  proc inc(p,q) : (r) {
 8    assert(p >= 0);
 9    if (p >= 1)
10       r = q + 1;
11    else
12       r = q;
13    endif;
14  }
```

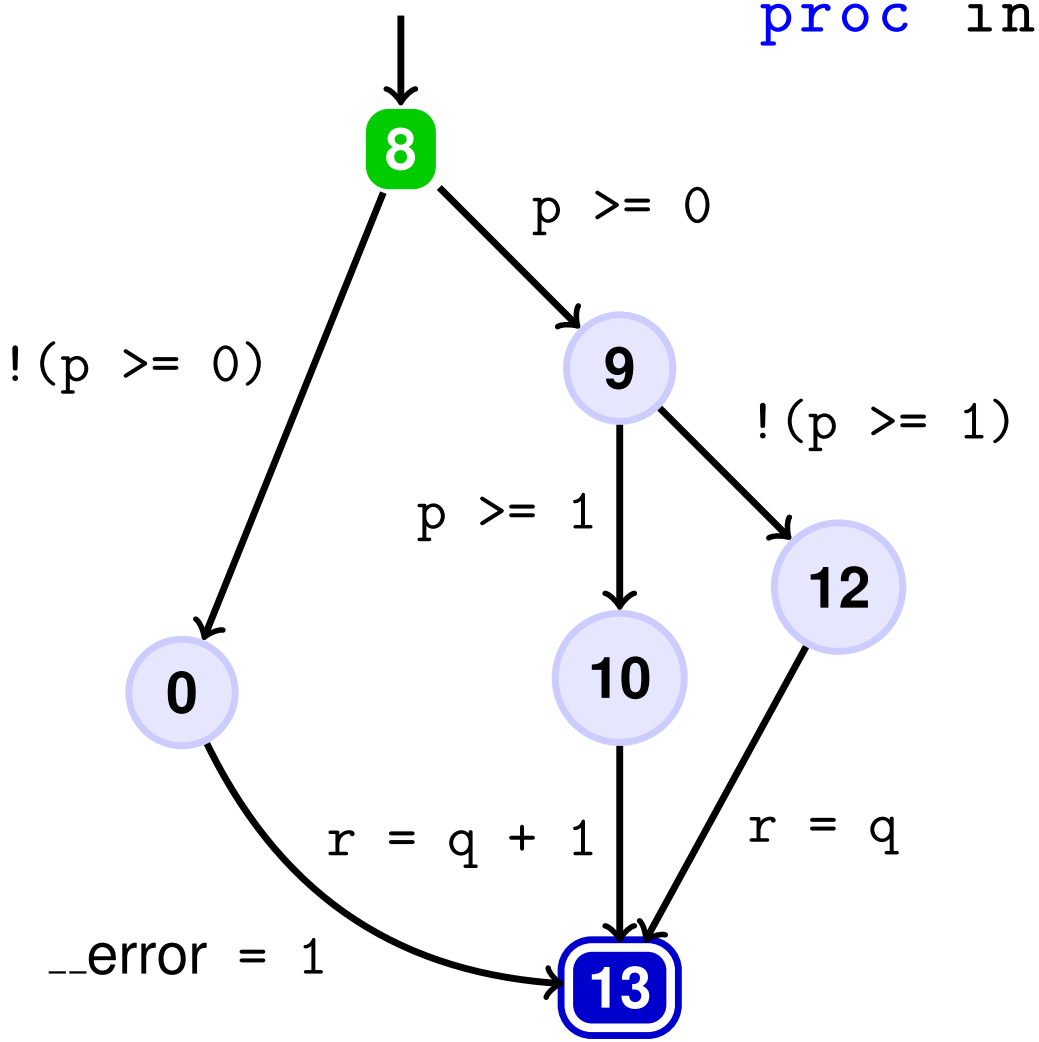



```
1  proc main() : (n) {
2    assume(m >= 1);
3    n = inc(1, m);
4    assert(n >= 0);
5  }
```



Function Summaries

```
proc inc(p, q) : r
```



$(p \geq 1, r \geq q + 1)$

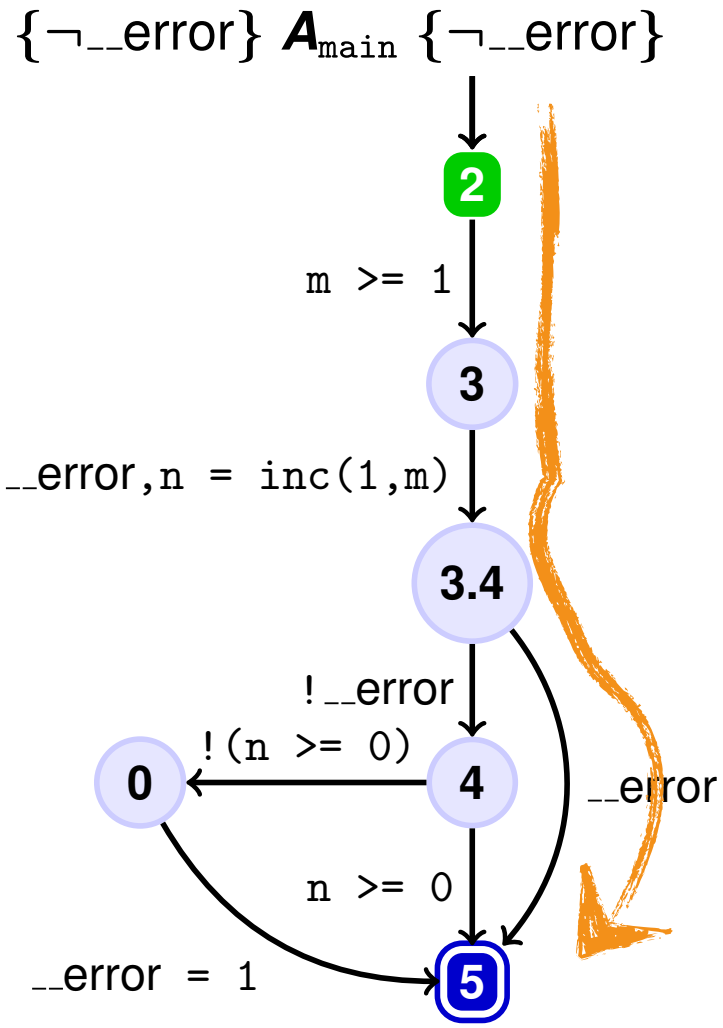
$(p \geq 1, r \leq q + 1)$

$(p \geq 0, r > q)$

$(p \geq 1 \wedge \neg\_\_error, \neg\_\_error)$

(***True***, ***True***)

Inter-procedural Trace Refinement (1)



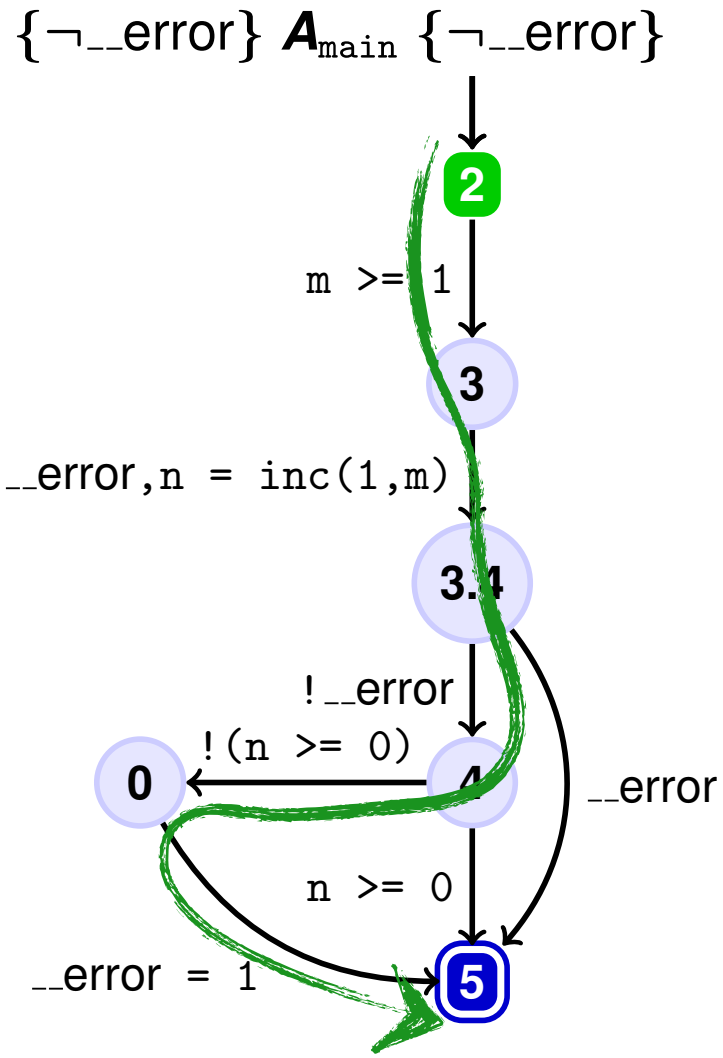
A_{inc}

(*True*, *True*)

$\{\neg \_\_error \wedge p = q = 1\}\ A_{inc}\ \{\neg \_\_error\}$

$\{\neg \_\_error \wedge p \geq 0\}\ A_{inc}\ \{\neg \_\_error\}$

Inter-procedural Trace Refinement (2)



A_{inc}

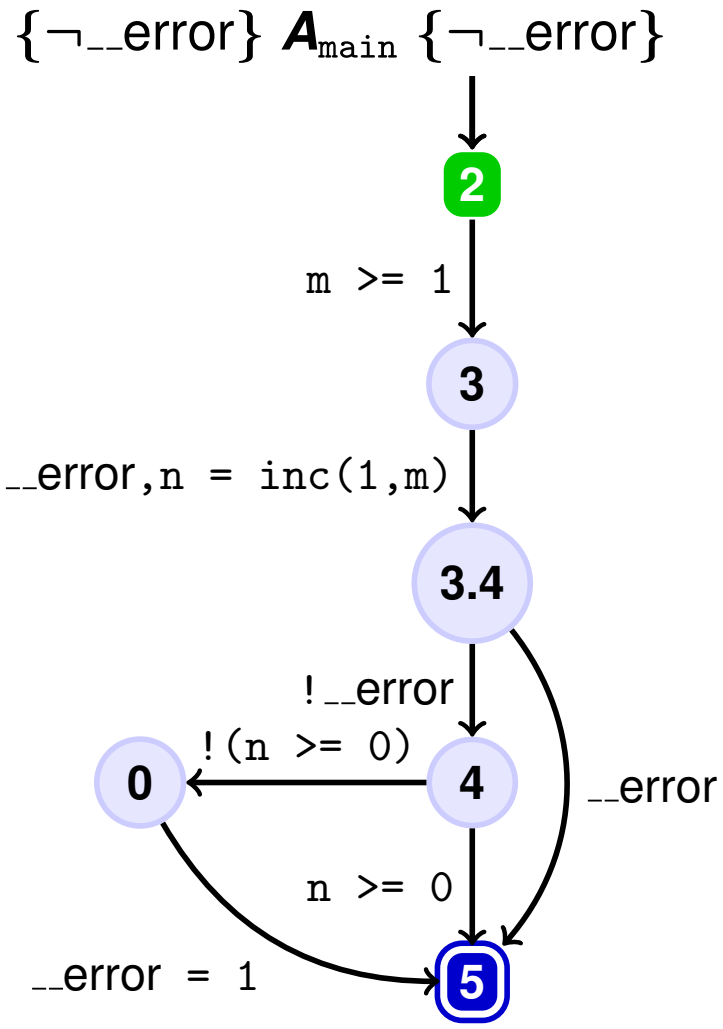
(*True*, *True*)

$(\neg\_\_error \wedge p \geq 0, \neg\_\_error)$

$\{p = q = 1\}\ A_{inc}\ \{not(n = -1)\}$

$\{p \geq 1 \wedge q \geq 1\}\ A_{inc}\ \{r \geq q + 1\}$

Inter-procedural Trace Refinement (3)



A_{inc}

(*True*, *True*)

$(\neg\_\_error \wedge p \geq 0, \neg\_\_error)$

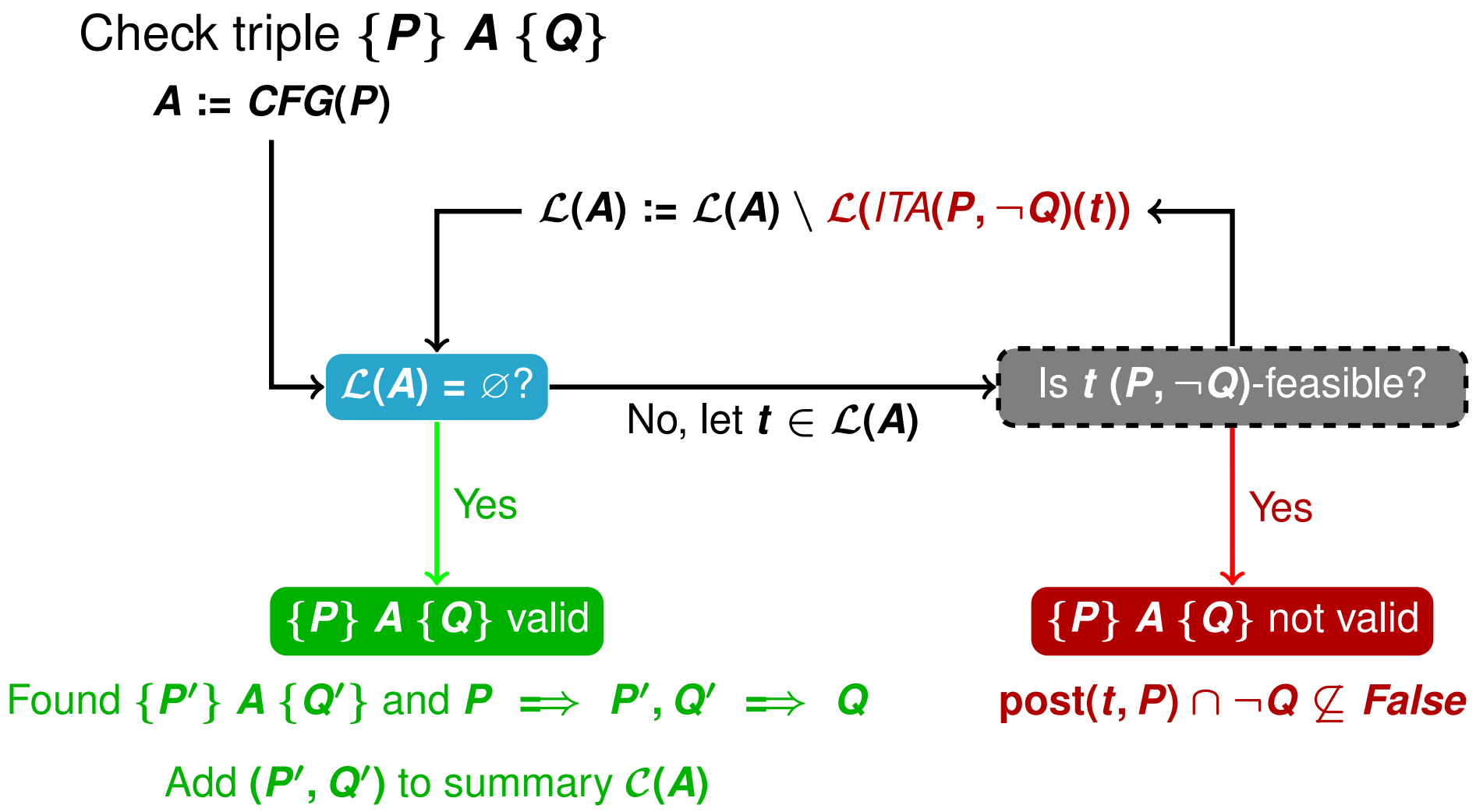
$(p \geq 1 \wedge q \geq 1, r \geq q + 1)$

$\{\neg\_\_error\}\ A_{main}\ \{\neg\_\_error \wedge n \geq m + 1\}$

Cassez F., Müller C. and Burnett K.
Summary-Based Inter-Procedural Analysis via Modular Trace Refinement, FSTTCS 2014

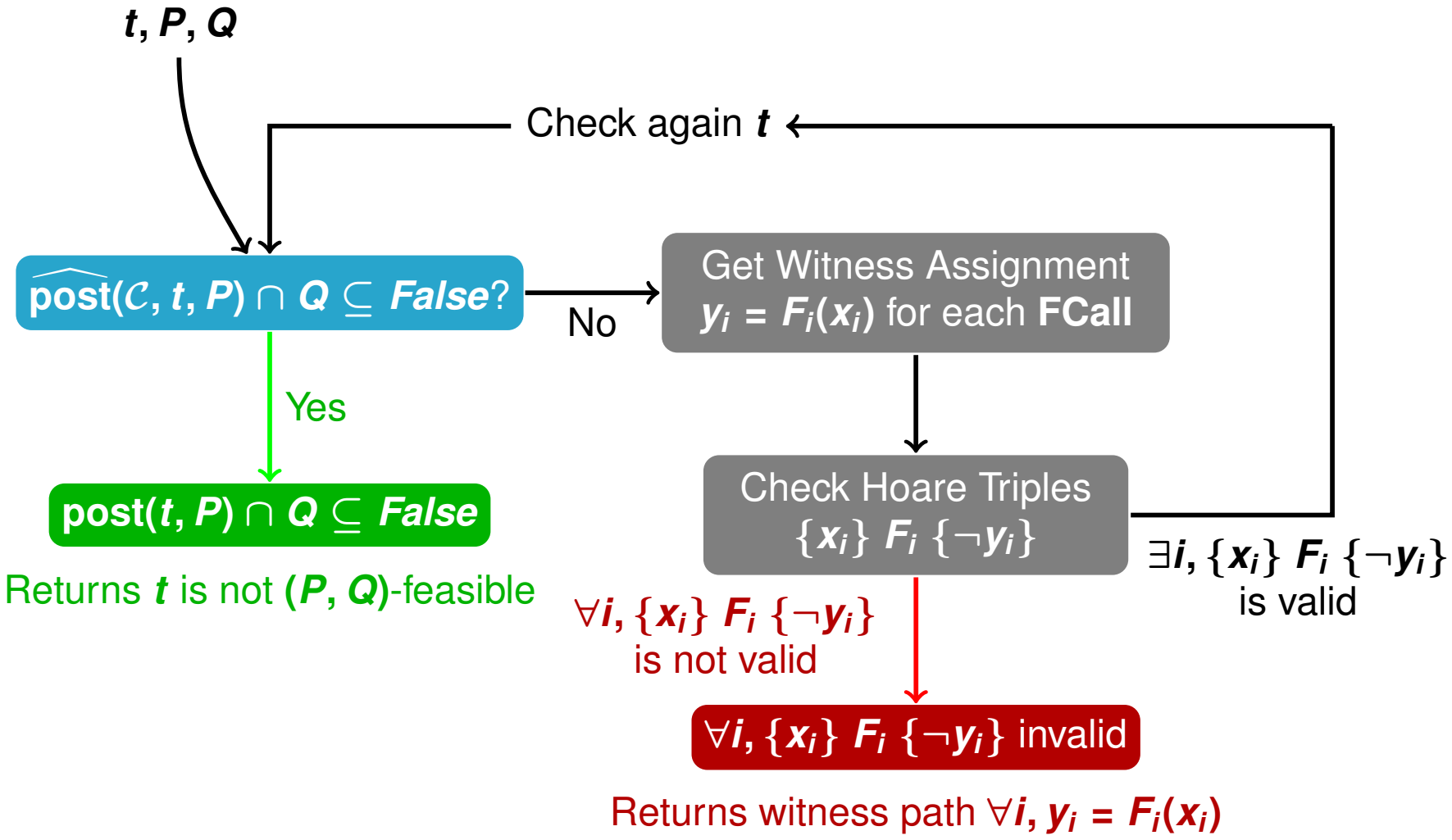
Cassez F. and Müller C.
Analysis of program code
US Patent Application (2014). 14/540,929

Algorithm 2


Check triple $\{P\}\ A\ \{Q\}$
$A := CFG(P)$
$\mathcal{L}(A) := \mathcal{L}(A) \setminus \mathcal{L}(ITA(P, \neg Q)(t))$
$\mathcal{L}(A) = \varnothing$?
No, let $t \in \mathcal{L}(A)$
Is t $(P, \neg Q)$-feasible?
Yes
Yes
$\{P\}\ A\ \{Q\}$ valid
$\{P\}\ A\ \{Q\}$ not valid
Found $\{P'\}\ A\ \{Q'\}$ and $P \implies P', Q' \implies Q$
$\mathbf{post}(t, P) \cap \neg Q \not\subseteq \mathit{False}$
Add (P', Q') to summary $\mathcal{C}(A)$

Inter-Procedural Algorithm

Is t (*P*, *Q*) – feasible ?



Skink

C/C++

Clang

LLVM-IR Parser

Automata

Partial Order Reduction

Verification Engine

Invariant Synthesis

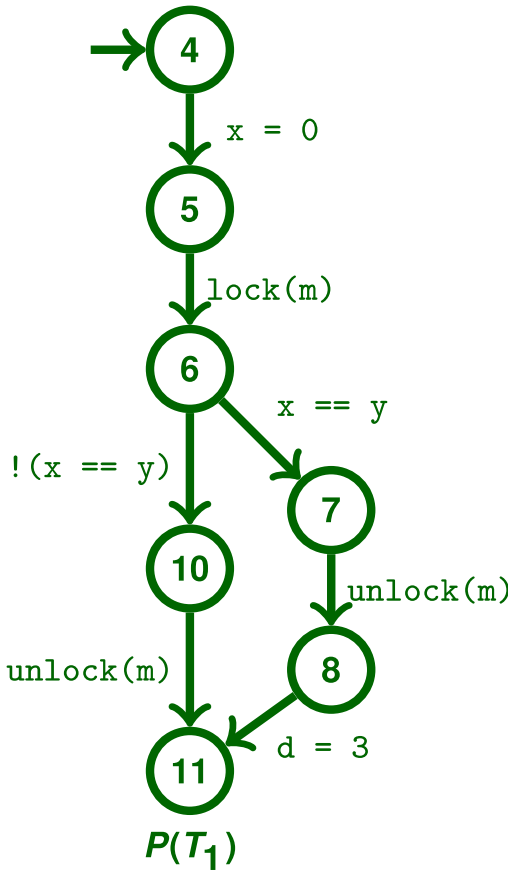
Abstract SMT solver

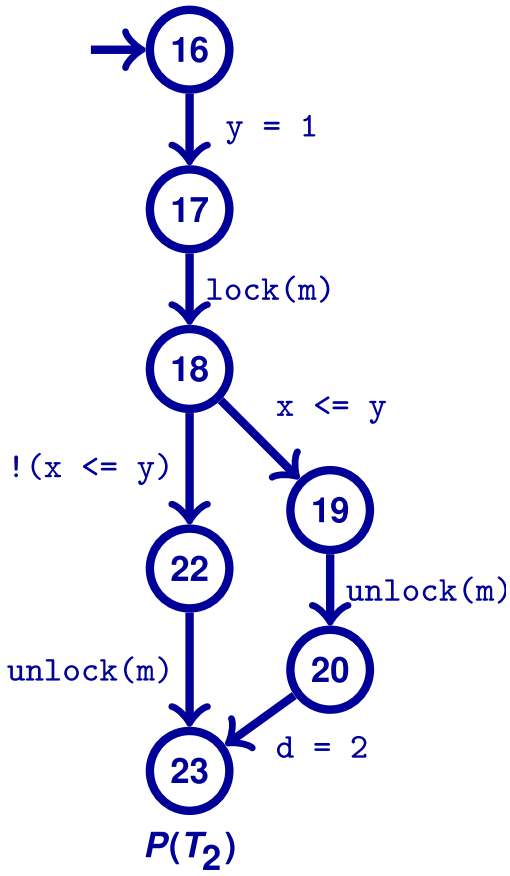
correct

incorrect witness

Concurrent Programs

```
2   // thread T1
3   thread T1
4   x = 0;
5   lock(m);
6   if (x == y) {
7     unlock(m);
8     d = 3;
9   } else {
10    unlock(m);
11  }
12  /* end */
```






```
14  // Thread T2
15  thread T2
16  y = 1;
17  lock(m);
18  if (x <= y) {
19    unlock(m);
20    d = 2;
21  } else {
22    unlock(m);
23  }
```

$\mathcal{L}(P(T_1) \times P(T_2))$ **Is there a *feasible* error trace?**

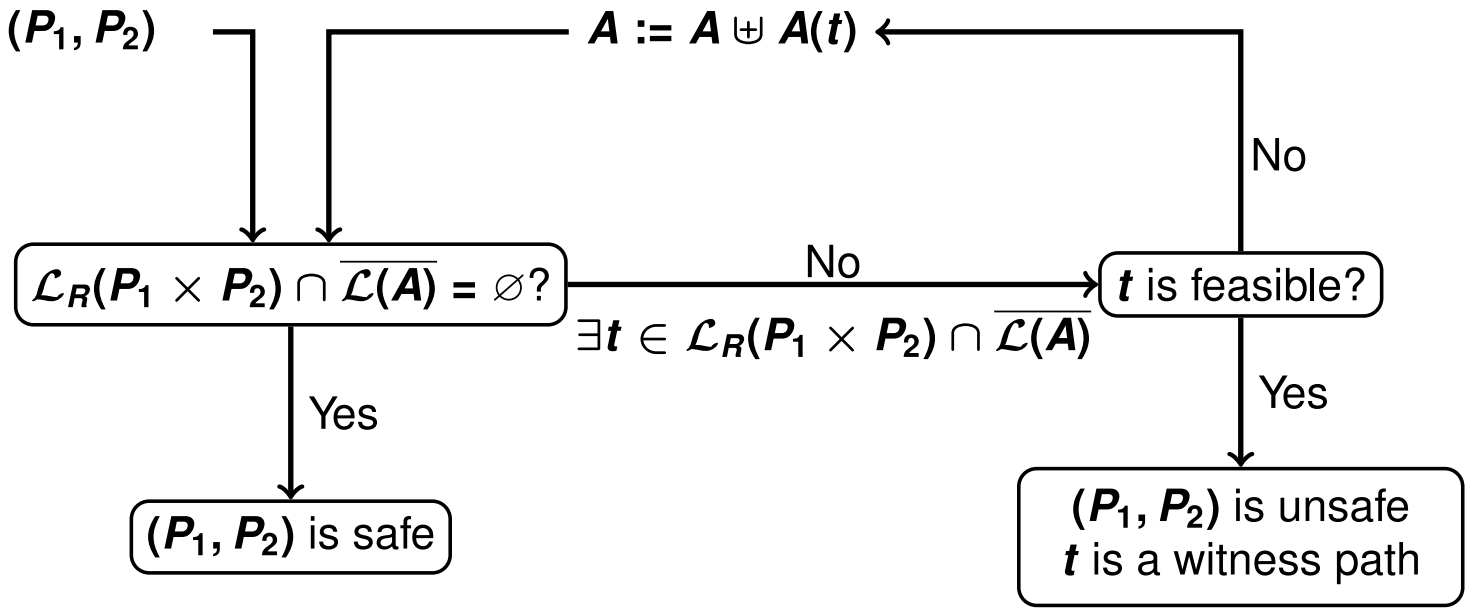
Trace Refinement for Concurrent Programs

$\exists$an error trace $t \in \mathcal{L}(P_1 \times P_2)$

$\Longleftrightarrow$

$\exists$an error trace $t' \in \mathcal{L}_R(P_1 \times P_2)$

feasible



Cassez, F. and Ziegler F.
Verification of Concurrent Programs Using Trace Abstraction Refinement
LPAR 2015