Skink Refinement Loop with Predicate Abstraction

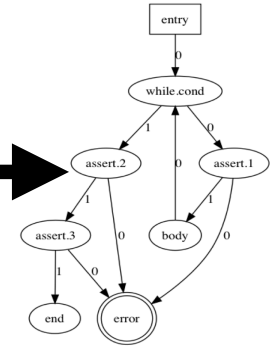
Pongsak Suvanpong, Anthony Sloane and Franck Cassez

Department of Computing
Macquarie University

Skink refinement loop

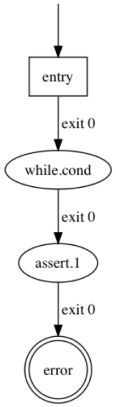
```
define i32 @main()
{
  entry:
    br label %do.body

  do.body:
    %0 = phi i32 [ %add, %do.body ], [ 0, %entry ]
    %add = add nsw i32 %0, 1
    %cmp = icmp slt i32 %add, 100
    br i1 %cmp, label %do.body, label %do.end

  do.end:
    %cmp1 = icmp eq i32 %add, 100
    br i1 %cmp1, label %ok, label %err

  ok:
    ret i32 0

  err:
    call void (...) @__VERIFIER_error()
    unreachable
}
```

Program (e.g. LLVM IR)



CFG

No more error trace



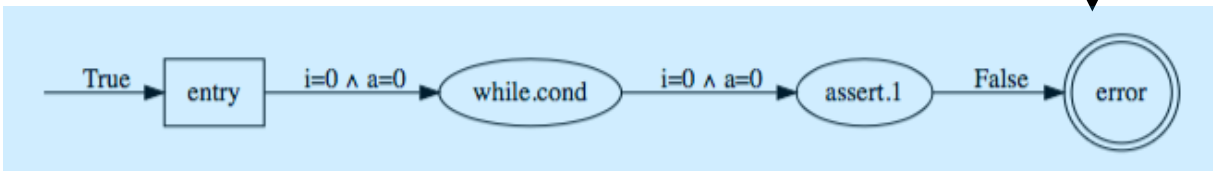
An error trace

the error trace infeasible?

NO

Yes

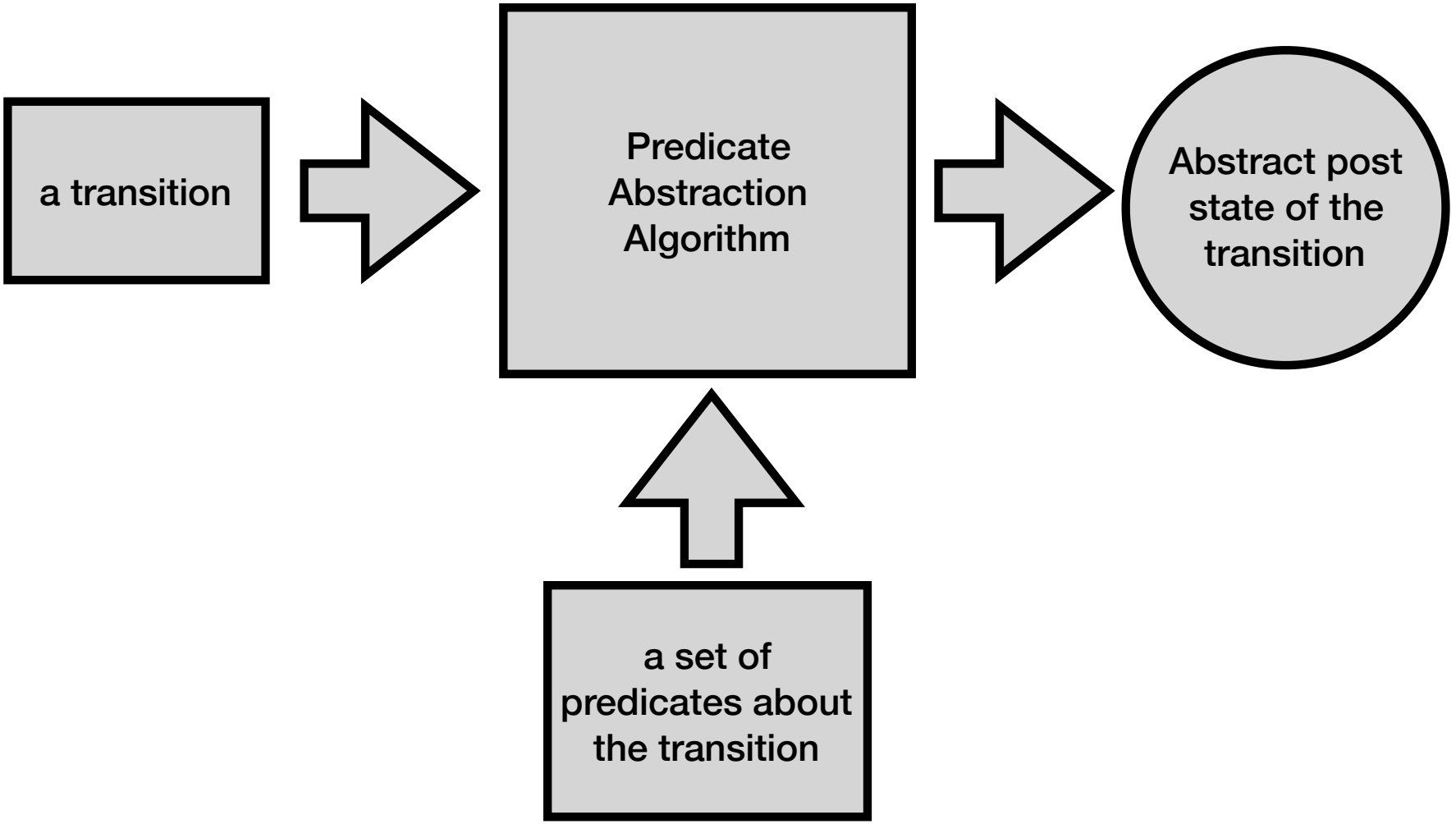
Refine CFG



MACQUARIE University

MACQUARIE University

Predicate Abstraction



Graf, S. and Saïdi, H. 1997. Construction of abstract state graphs with PVS. *International Conference on Computer Aided Verification* (1997), 72–83.

Flanagan, C. and Qadeer, S. 2002. Predicate abstraction for software verification. *ACM SIGPLAN Notices* (2002), 191–202.

Predicate Abstraction

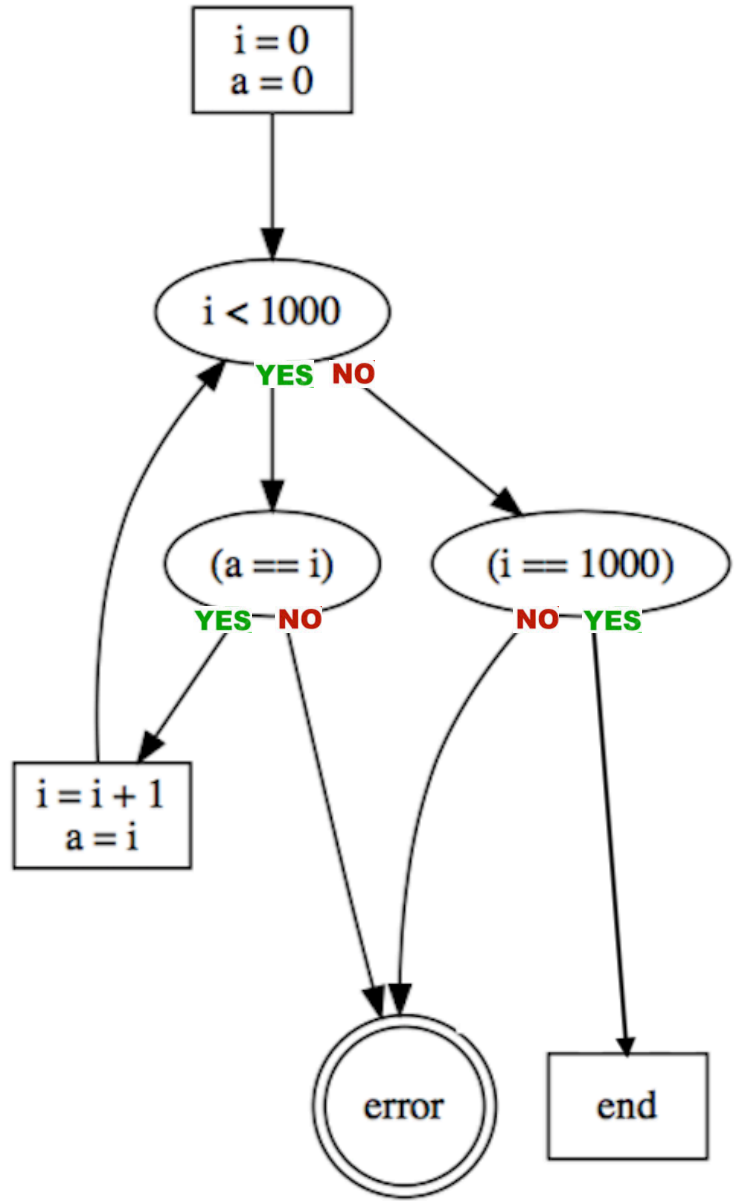
- SLAM (Static Driver Verifier: Technology Transfer of Formal Methods inside Microsoft)
- BLAST is a software model checker for C programs. (http://cseweb.ucsd.edu/~rjhala/blast.html)
- MAGIC: Modular Analysis of proGrams In C (http://www.cs.cmu.edu/~chaki/magic/)

Example

YES indicates an exit point where the condition in the block is **true**

NO indicates an exit point where the condition in the block is **false**

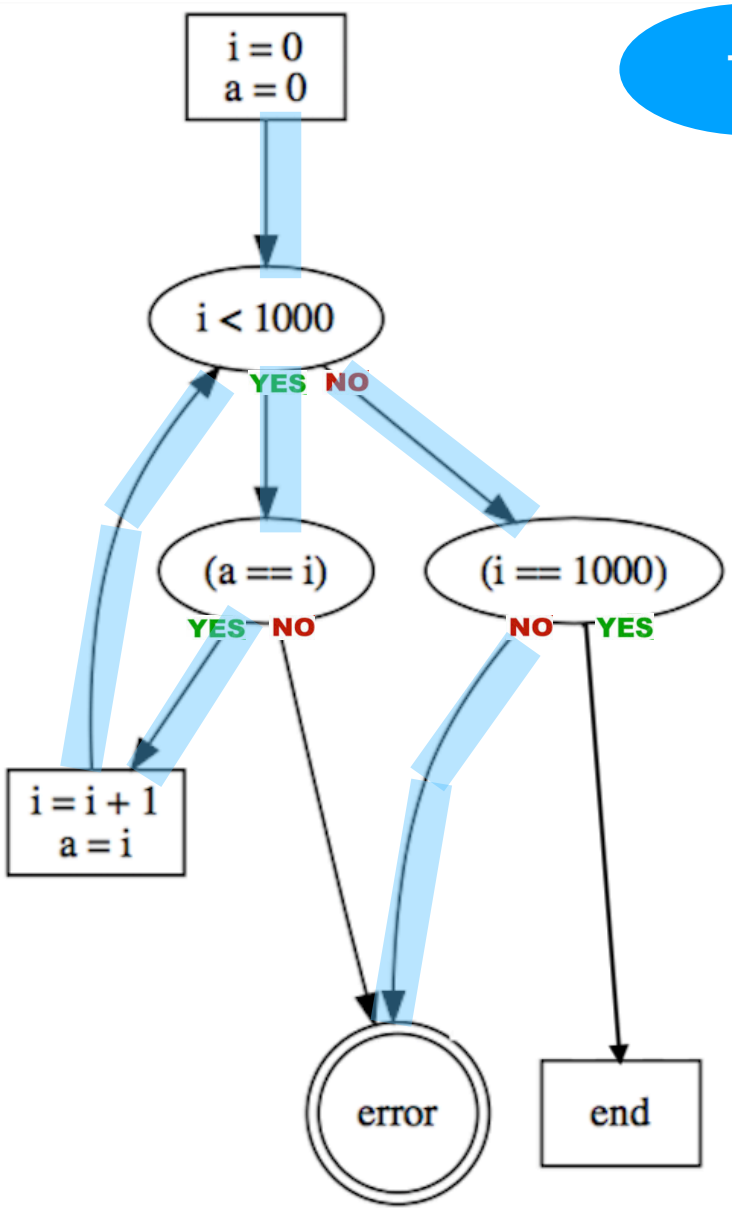
CFG



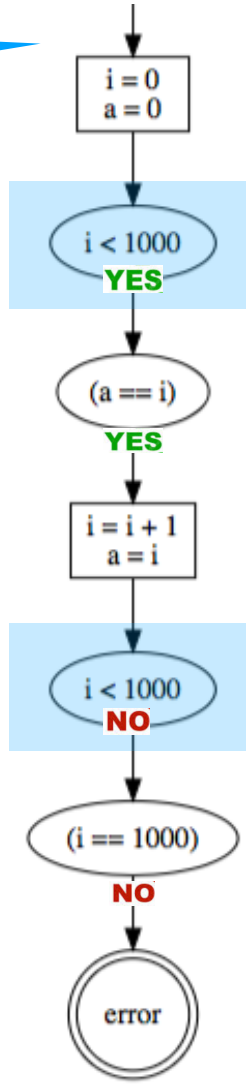
pseudo code

```
var i:Int, a:Int
i = 0, a = 0
while(i < 1000) do
    assert(a == i)
    i = i + i, a = i
assert(i == 1000)
```

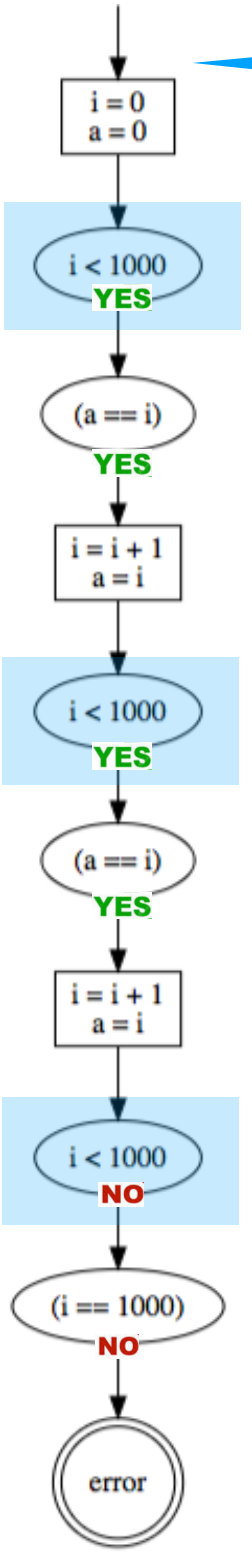
Example



T1

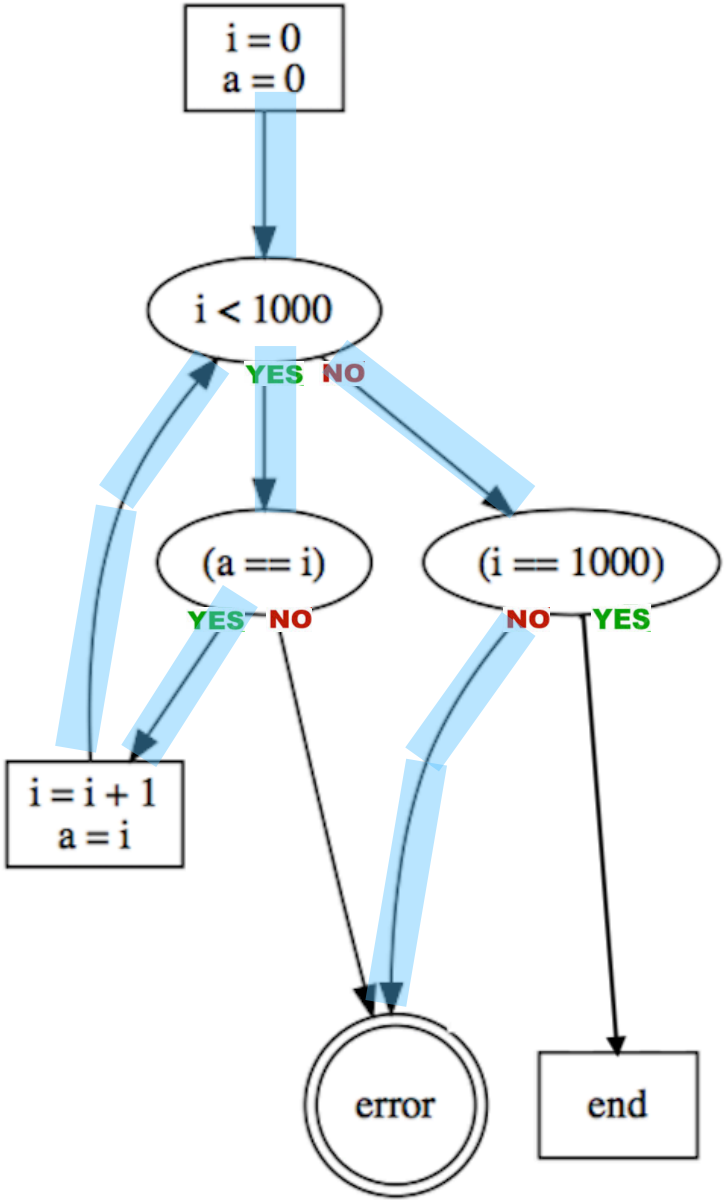


T2

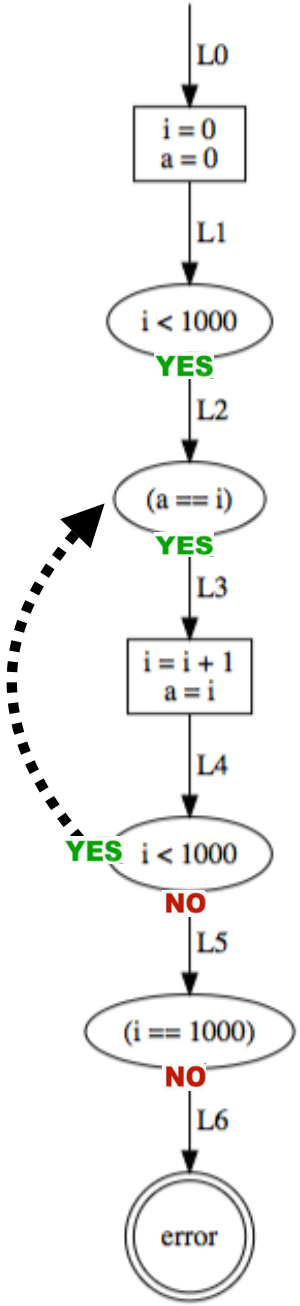


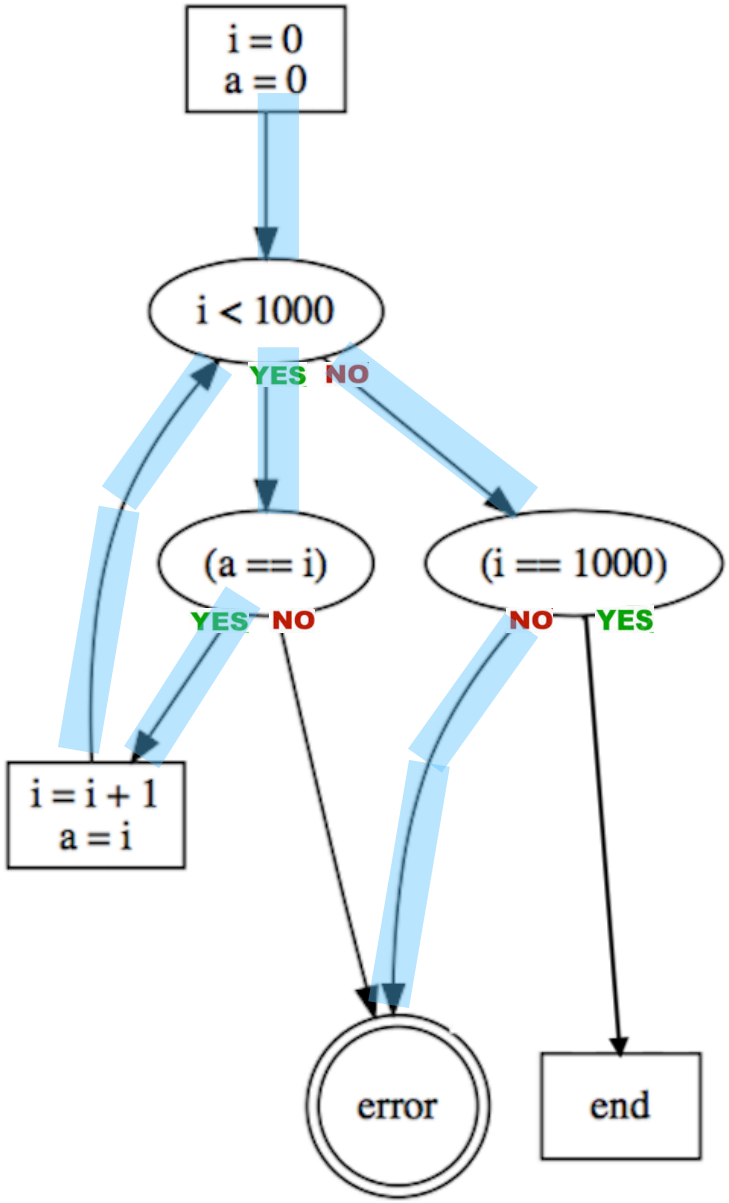
.....

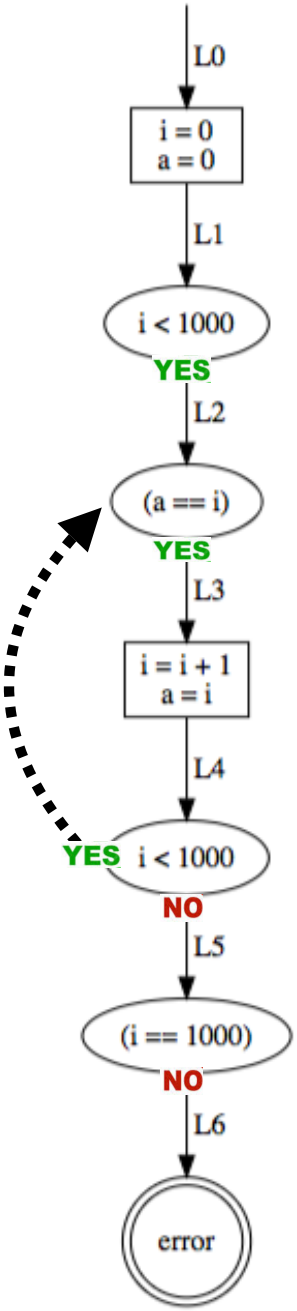
Example: back edge



add new edge to the linear trace




i = 0
a = 0
i < 1000
YES
NO
(a == i)
YES
NO
(i == 1000)
NO
YES
i = i + 1
a = i
error
end


L0
i = 0
a = 0
L1
i < 1000
YES
L2
(a == i)
YES
L3
i = i + 1
a = i
L4
YES
i < 1000
NO
L5
(i == 1000)
NO
L6
error

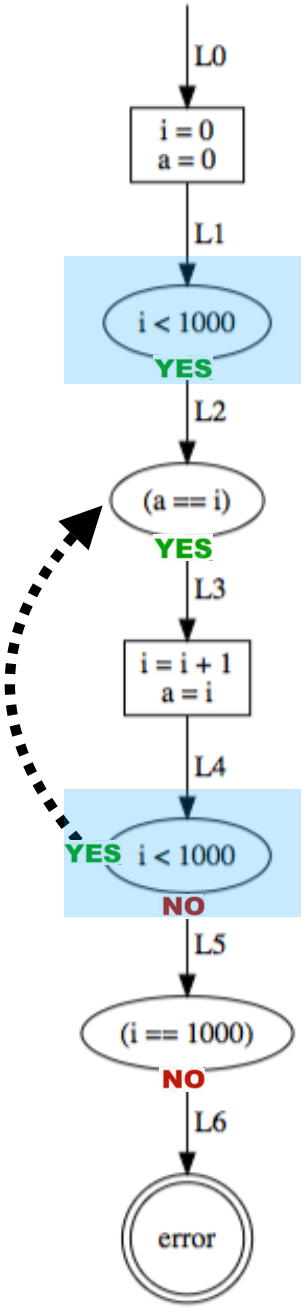
Example: Safe to add?

Triples for this error trace

{L0 (True)} *i = 0; a = 0* {L1}
{L1} *i < 1000* {L2}
{L2} *a == i* {L3}
{L3} *i = i + 1; a = i* {L4}
{L4} *!(i < 1000)* {L5}
{L5} *!(a == 1000)* {L6 (False)}

Triple for the new edge

{L4} i < 1000 {L2}



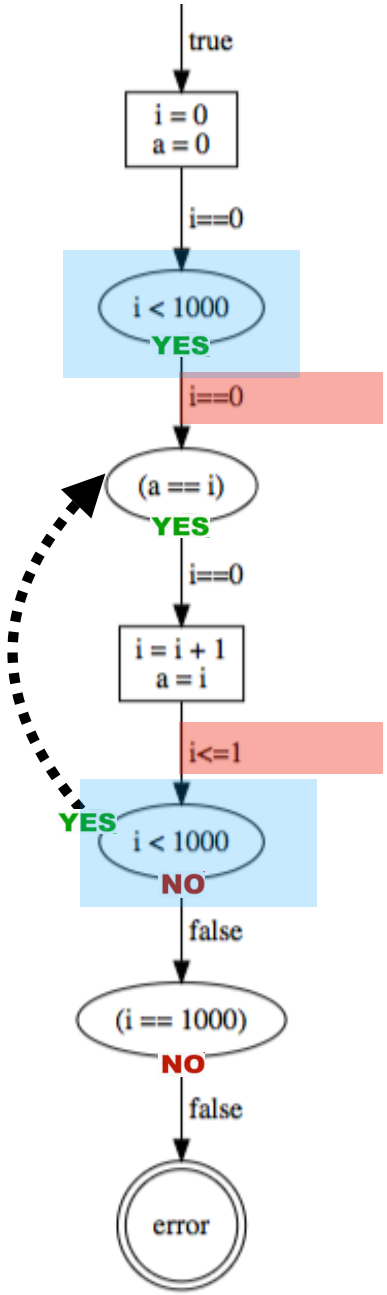
Compute predicates at L1 to L6 such that all the triples hold.

Example

Interpolating SMT Solver

{i <= 1} i < 1000 {i == 0}

✗



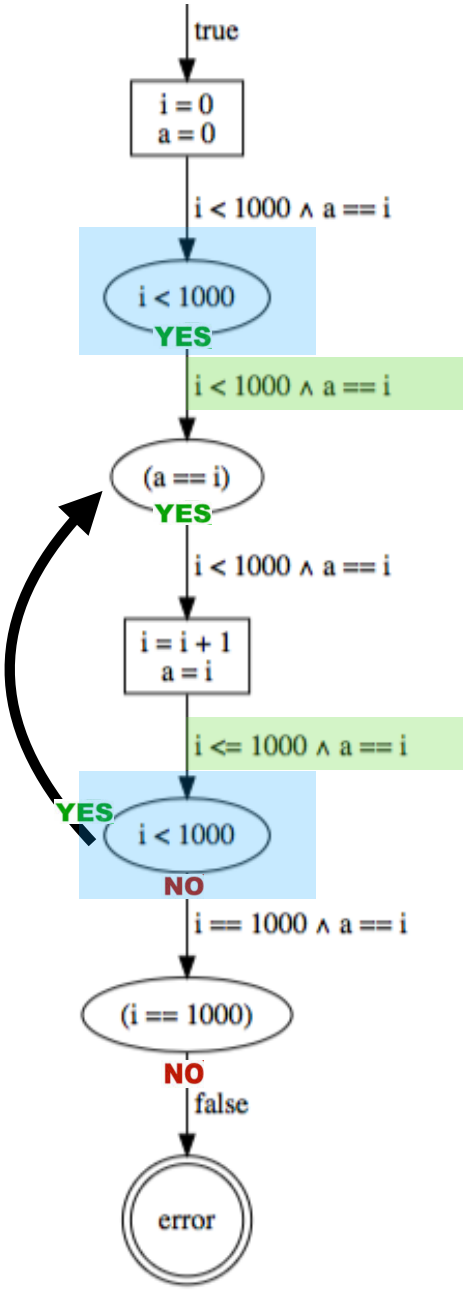
McMillan, K. 2005. Applications of Craig interpolation to model checking. *International Conference on Application and Theory of Petri Nets* (2005), 15–16.

Example

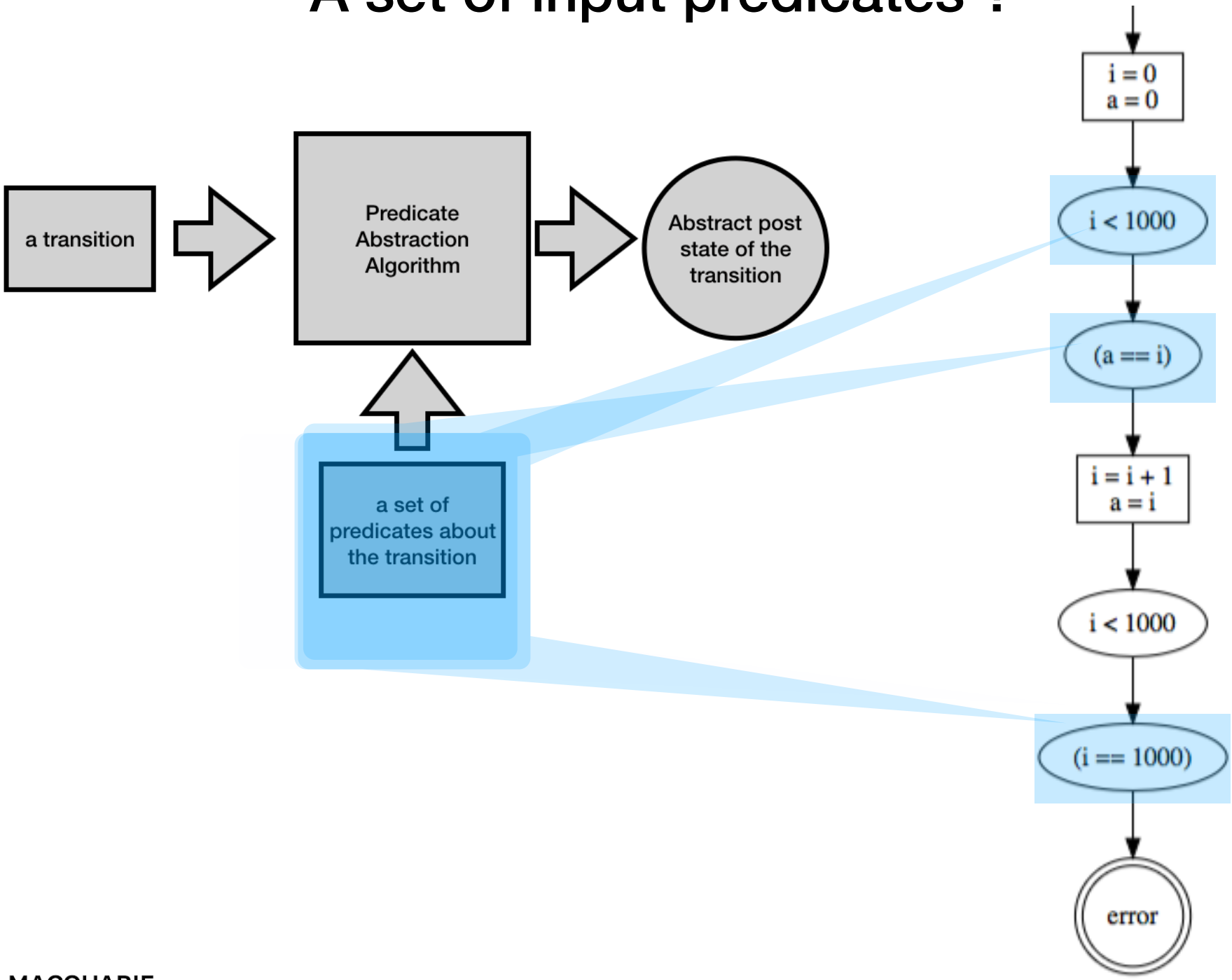
Using predicate abstraction

{i ≤ 1000 ∧ a == i} i < 1000 {i < 1000 ∧ a == i}

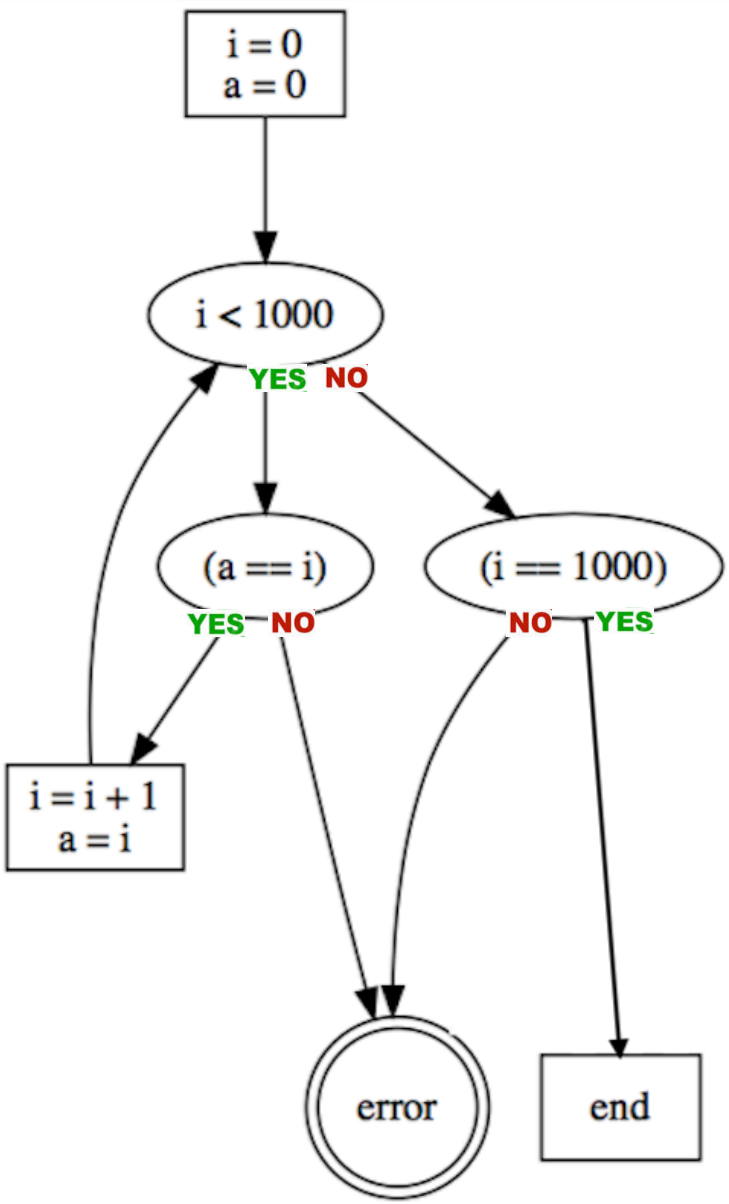
✔



A set of input predicates ?



Result (from the example)



	Number of refinement Iteration
Predicate Abstraction	4
Interpolating SMT Solver (z3)	> 20

Result (preliminary)

Loop Acceleration	Predicate Abstraction	Interpolating SMT Solver
const_true-unreach-call1.c	2	2
diamond_true-unreach-call1.c	7	> 20
nested_true-unreach-call1.c	7	> 20
overflow_true-unreach-call1.c	2	3
multivar_true-unreach-call1.c	2	2
phases_true-unreach-call1.c	> 20	> 20
simple_true-unreach-call1.c	2	> 20
simple_true-unreach-call2.c	2	2
simple_true-unreach-call3.c	2	> 20
simple_true-unreach-call4.c	2	> 20
underapprox_true-unreach-call1.c	2	8
underapprox_true-unreach-call2.c	2	7
phases_true-unreach-call2.c	unknown	unknown

ETAPS
EUROPEAN JOINT CONFERENCES ON THEORY & PRACTICE OF SOFTWARE
TACAS 2017
Competition on Software Verification (SV-COMP)

MACQUARIE University

Next Step

- Program with Array (require quantified invariants)
- Better heuristic to automatically generate input predicates for Predicate Abstraction.

????

MACQUARIE
University