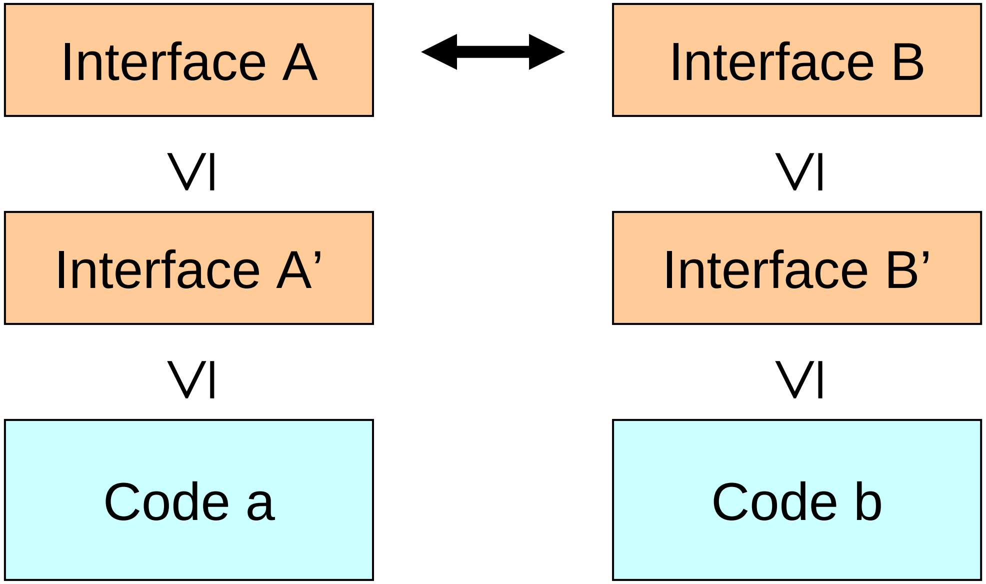
Interface-based Design 5

Tom Henzinger
EPFL and UC Berkeley



Interface A

Interface B

VI

VI

Interface A’

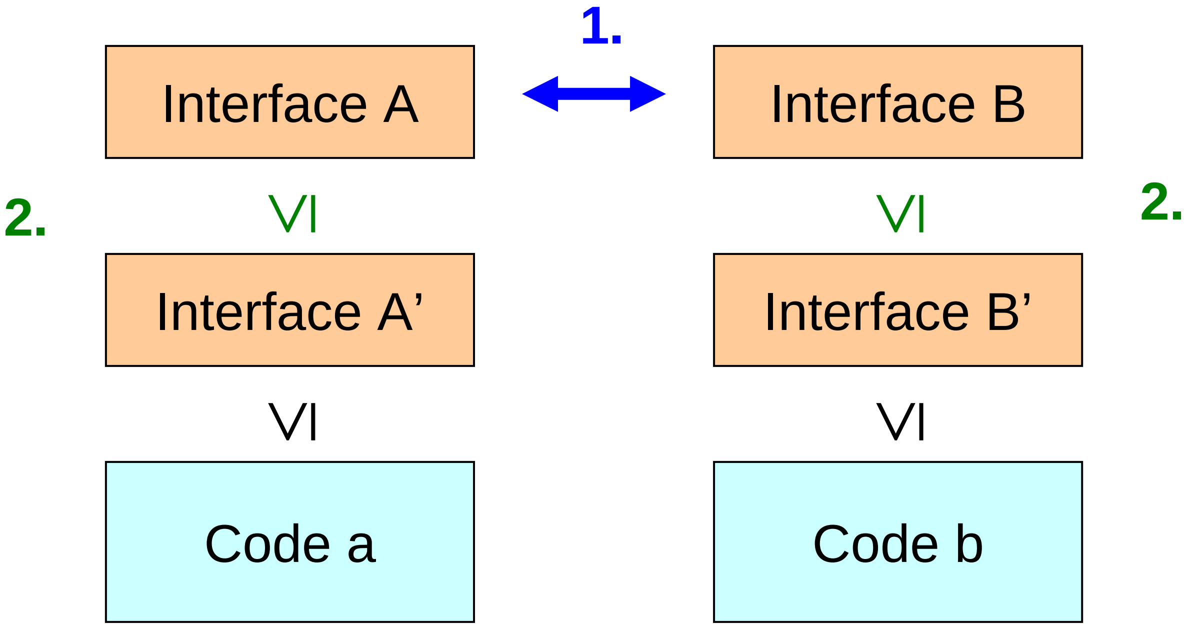
Interface B’

VI

VI

Code a

Code b



1.

Interface A

Interface B

2.

VI

VI

2.

Interface A’

Interface B’

VI

VI

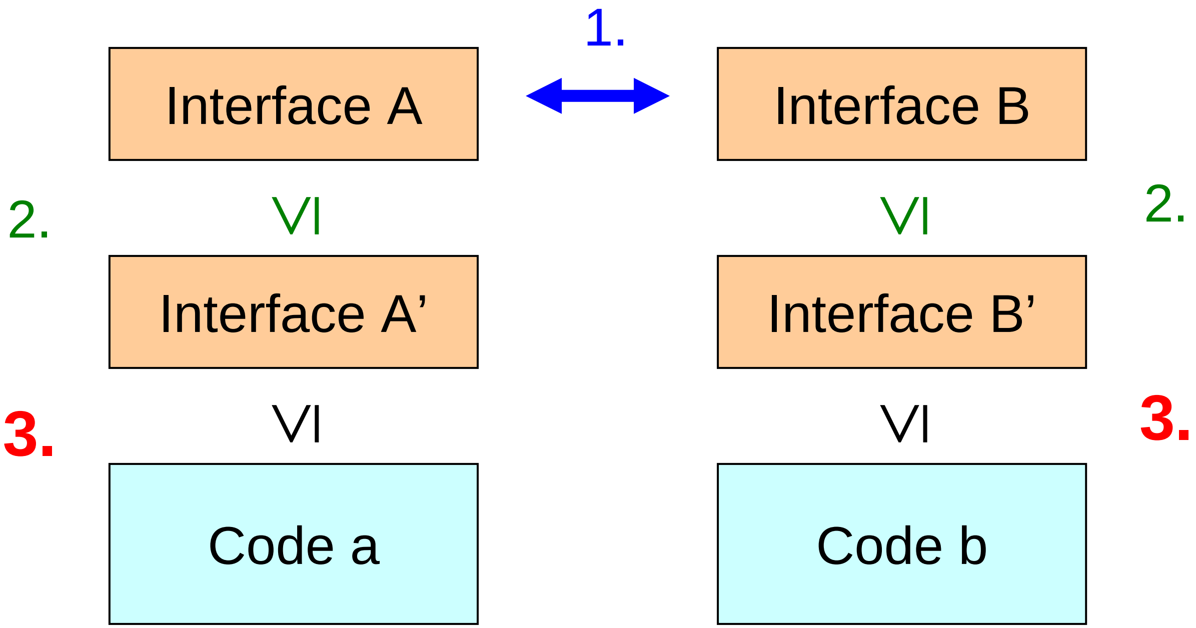
Code a

Code b

1. Interface compatibility checking:
   solving graph games

2. Interface refinement checking:
   alternating simulation relations

CHIC
[Chakrabarti]

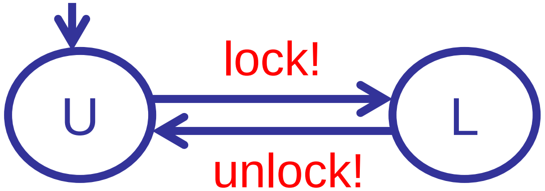


1. Interface compatibility checking:
   solving graph games
2. Interface refinement checking:
   alternating simulation relations
3. Conformance checking of code against interface

BLAST [Jhala,Majumdar,Sutre]

Interface Conformance Checking with BLAST

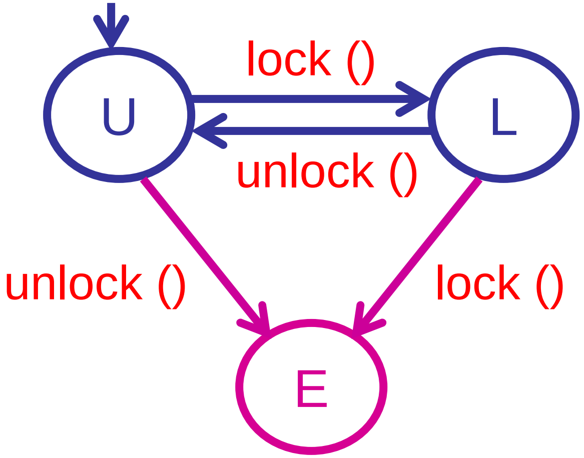
```
example () {
1:    do {
              lock ();
              old = new;
2:            if (*) {
3:                 unlock ();
                   new ++;
              }
4:    } while (new != old);
5:    unlock ();
?:   return;
 }
```



Interface Automaton.

Interface Conformance Checking with BLAST

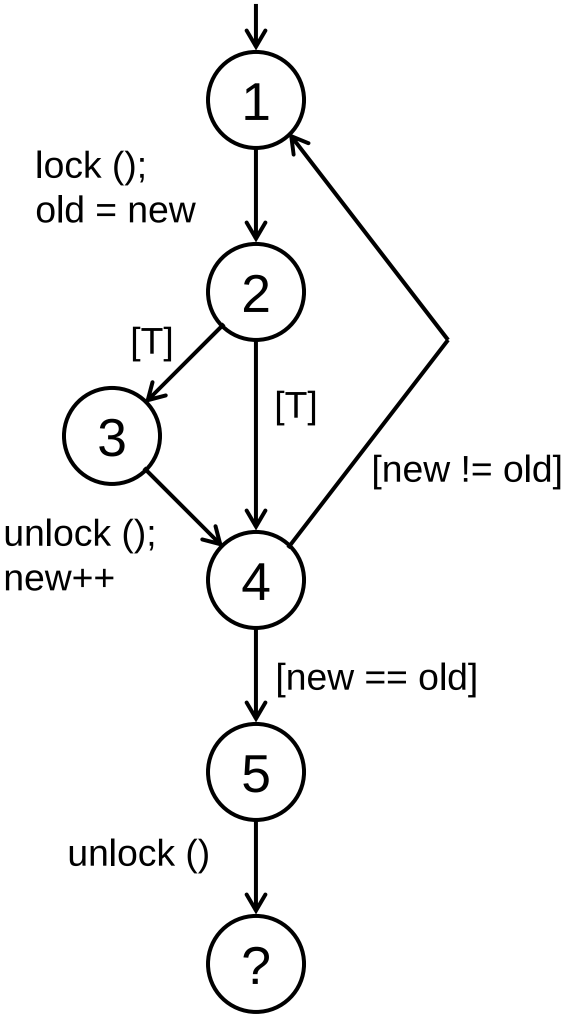
```
example () {
1:   do {
            lock ();
            old = new;
2:          if (*) {
3:                unlock ();
                  new ++;
            }
4:   } while (new != old);
5:   unlock ();
?:   return;
}
```

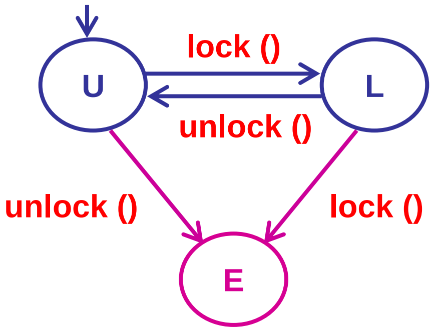


Monitor Automaton.

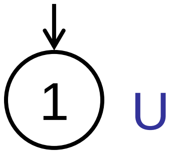
Control Flow Graph

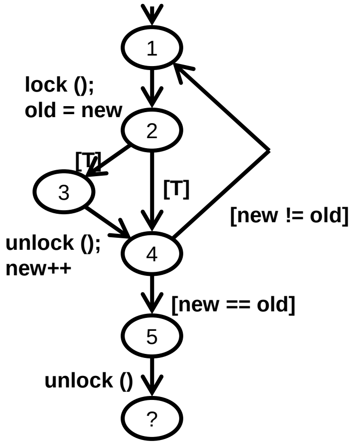
```
example () {
1:   do {
         lock ();
         old = new;
2:       if (*) {
            unlock ();
            new ++;
         }
4:   } while (new != old);
5:   unlock ();
?:  return;
 }
```

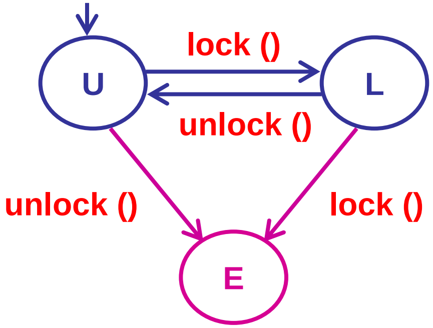




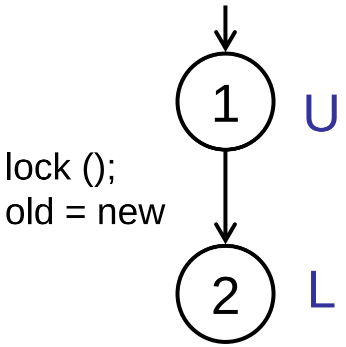
Abstract Reachability

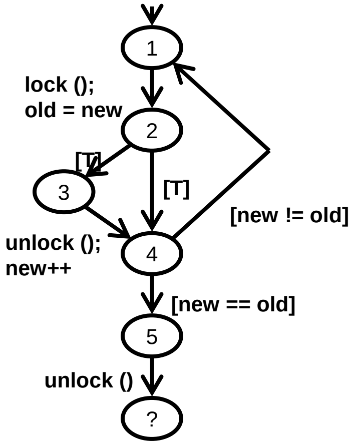


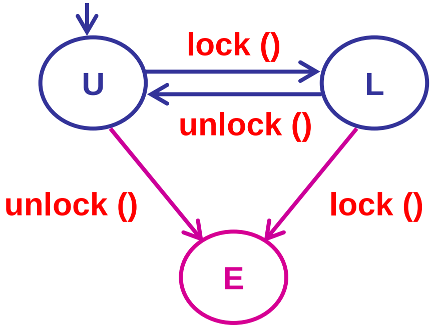




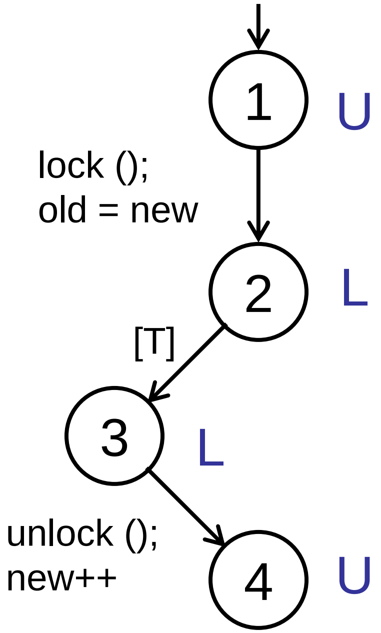
Abstract Reachability

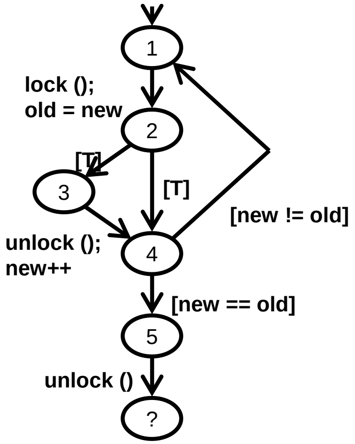


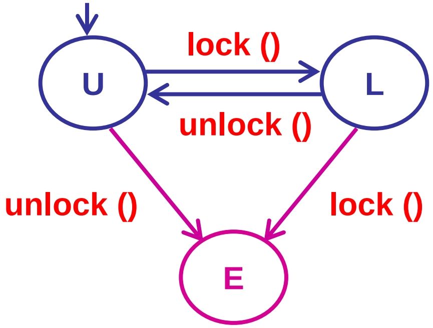




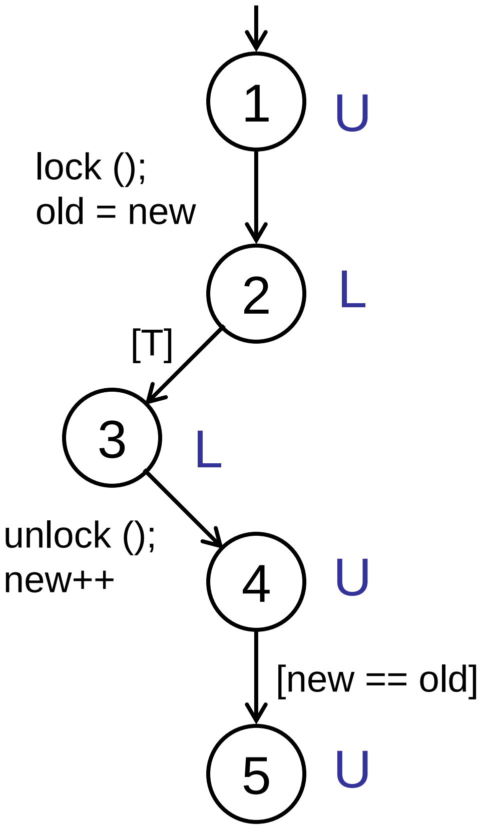
Abstract Reachability

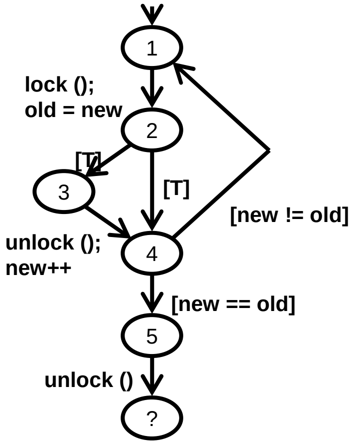


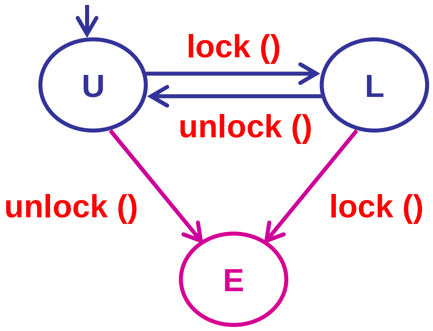




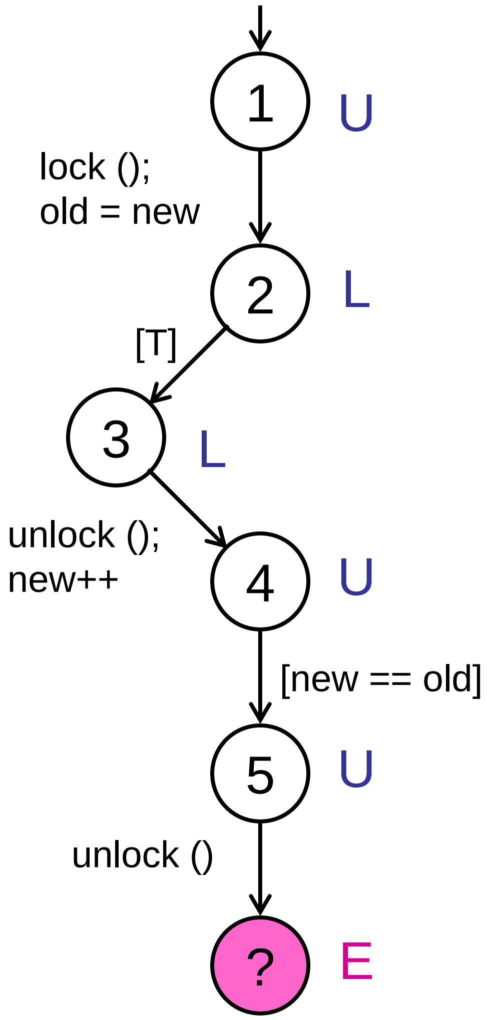
Abstract Reachability

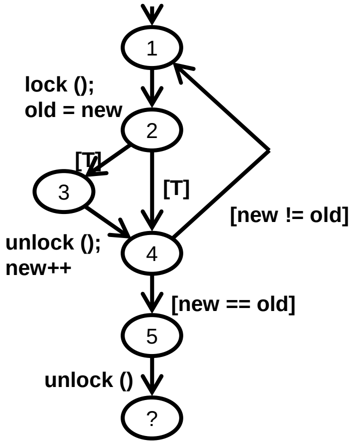


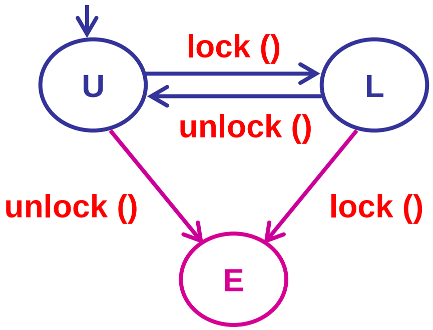




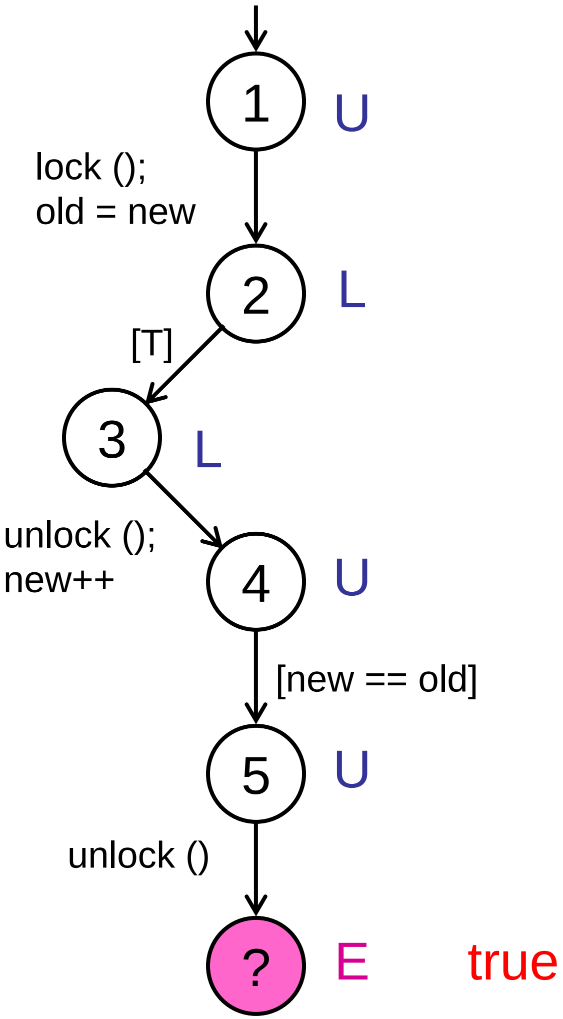
Abstract Reachability

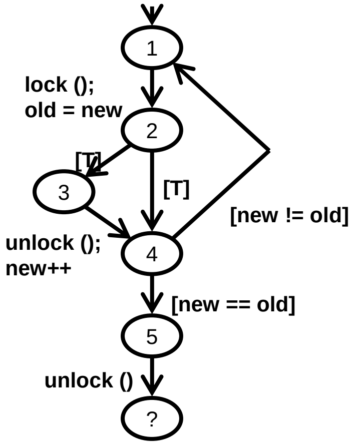


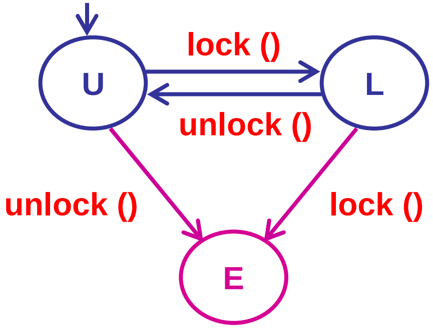




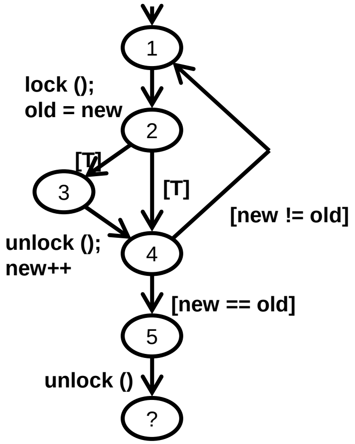
Concretize Error Trace

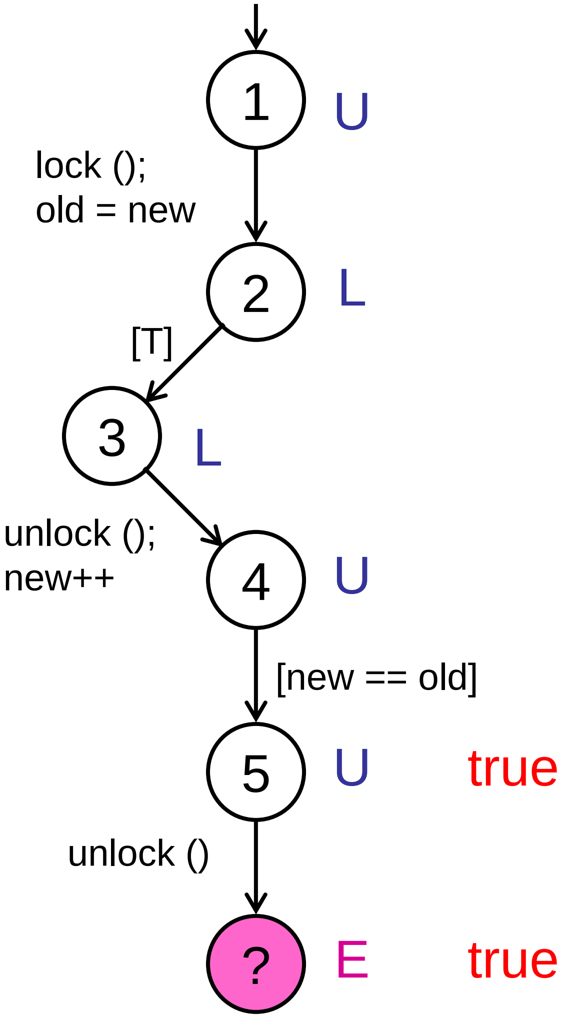


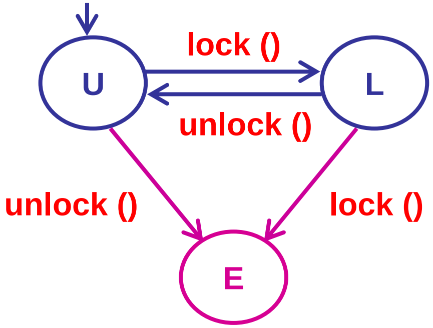




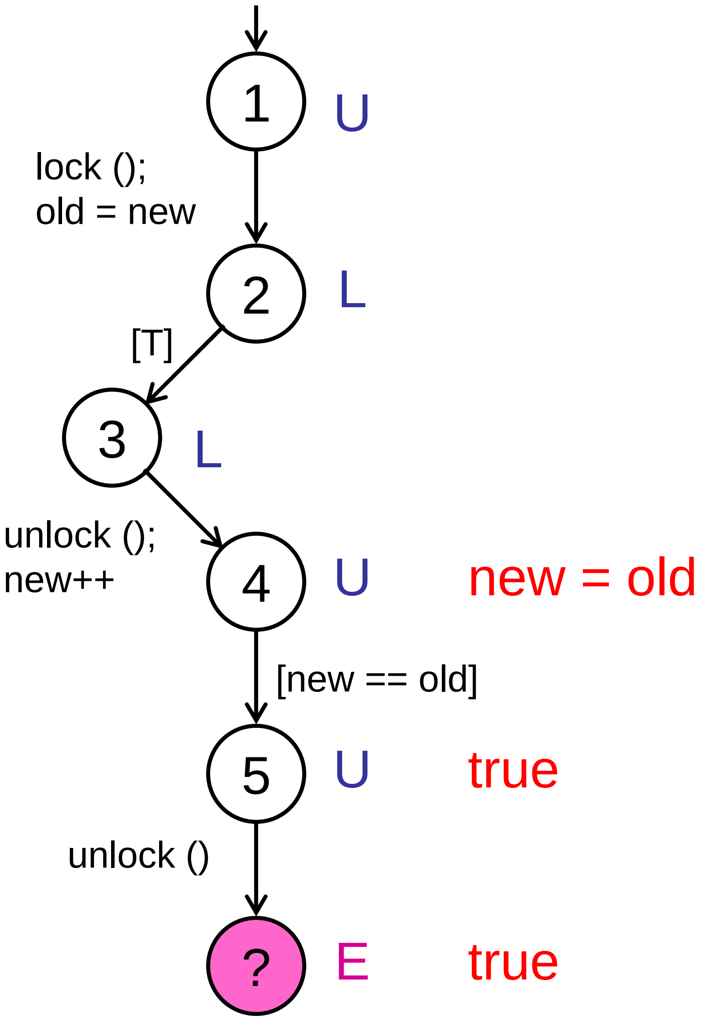
Concretize Error Trace

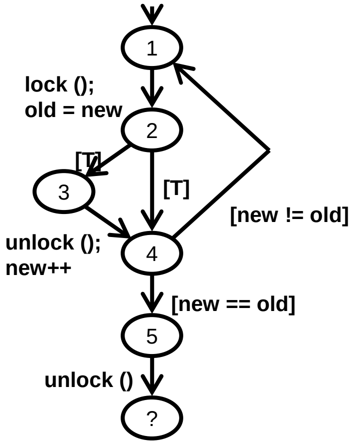


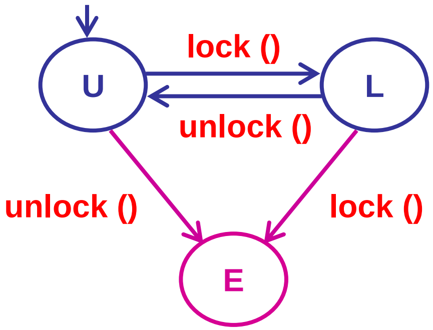




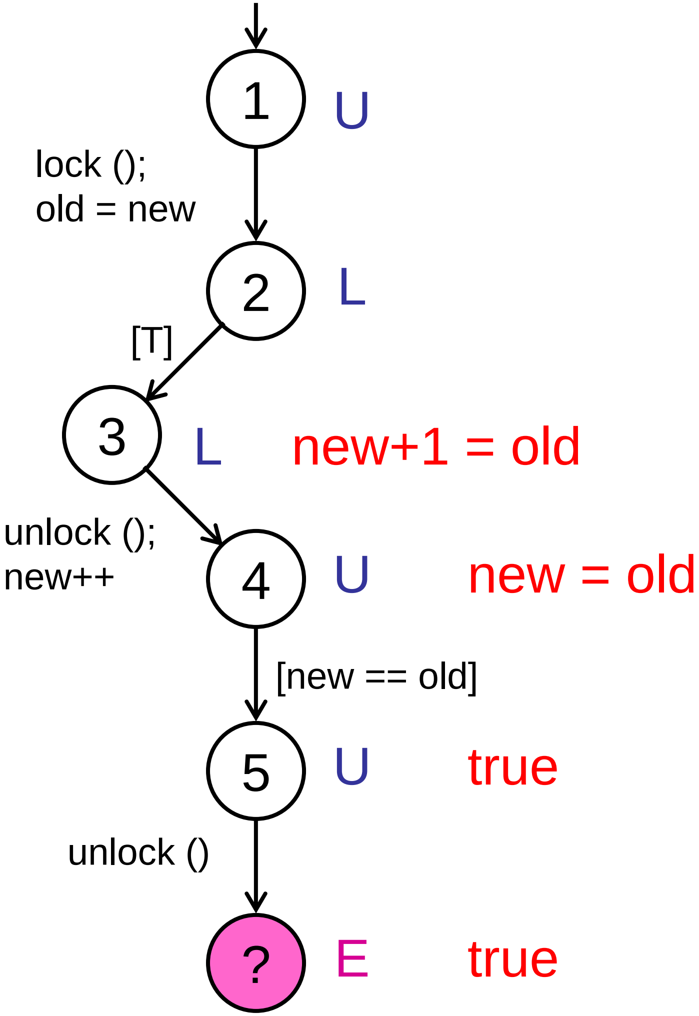
Concretize Error Trace

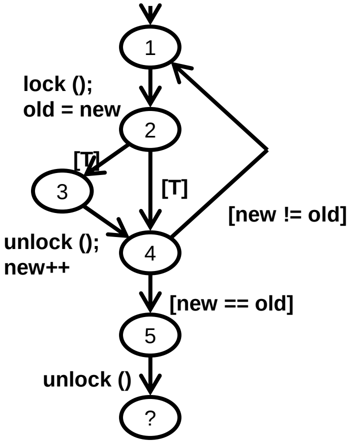


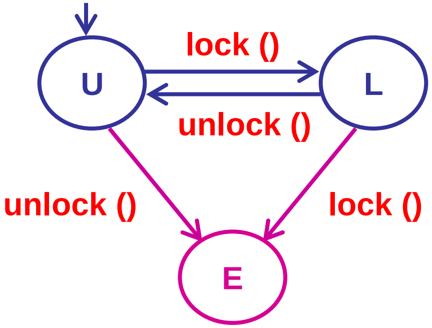




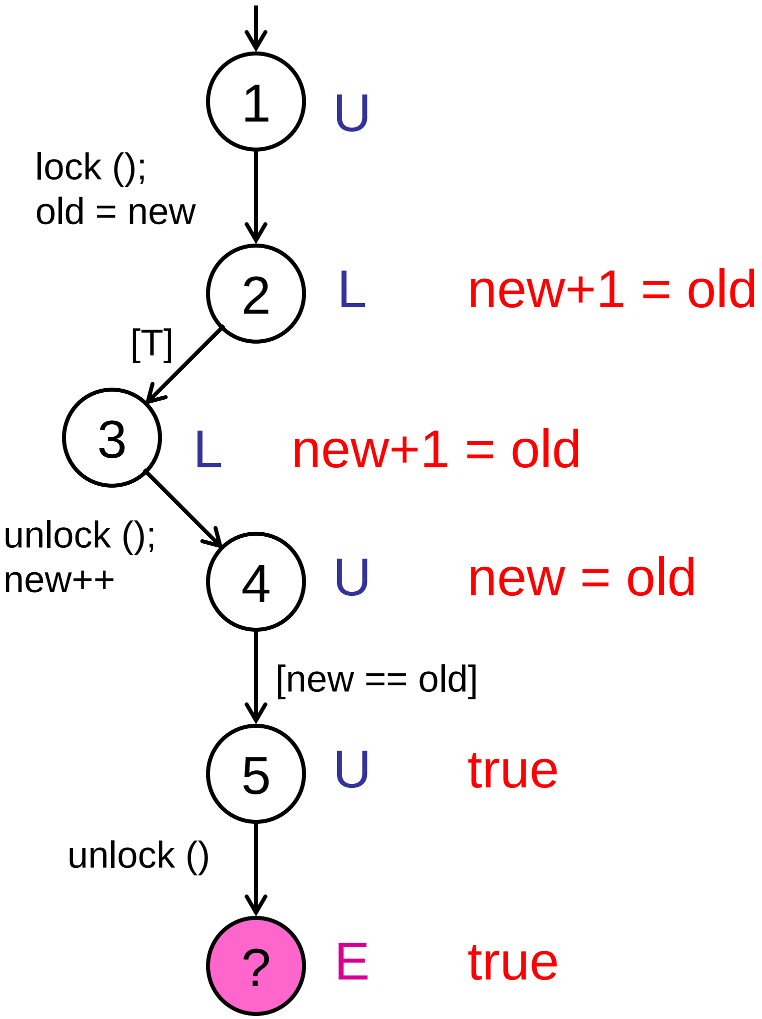
Concretize Error Trace

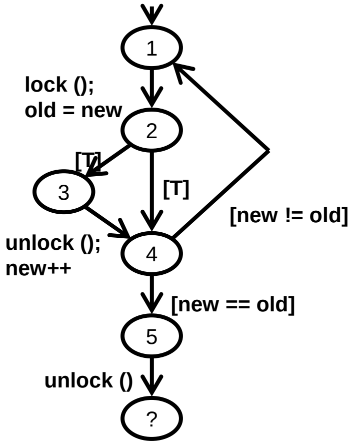


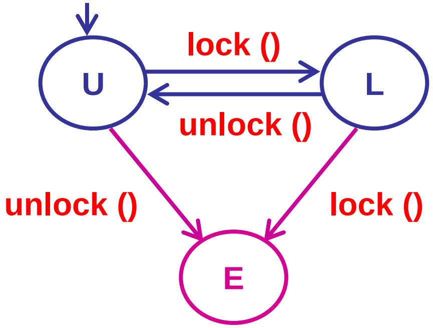




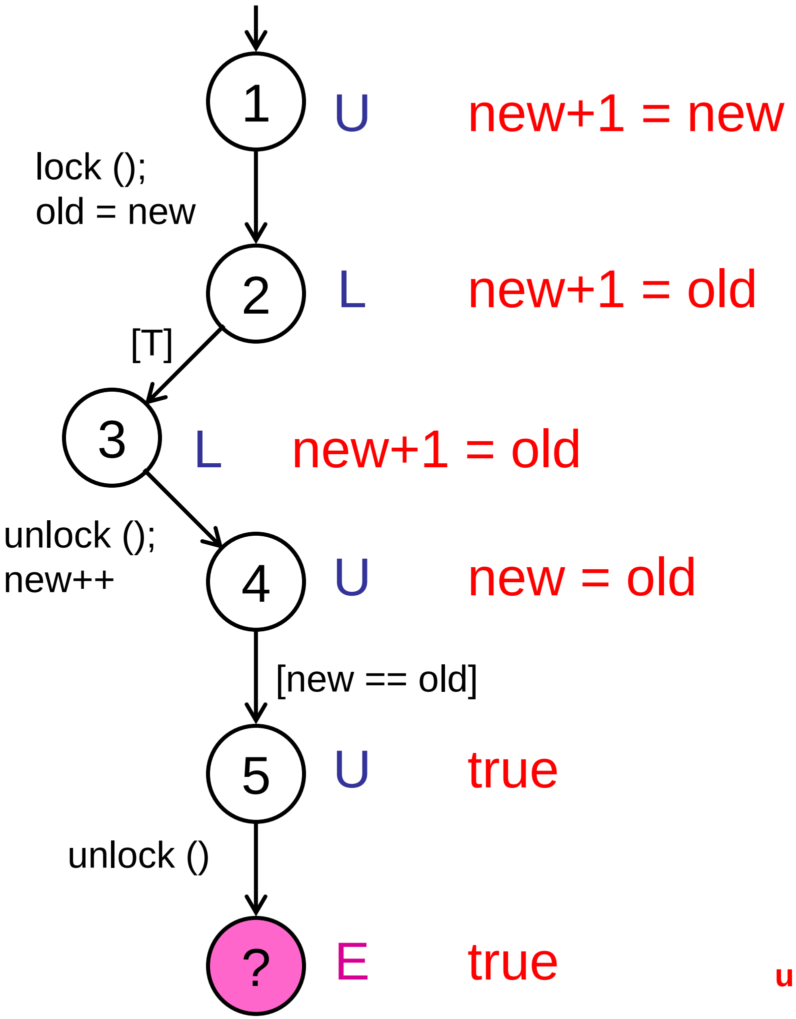
Concretize Error Trace







Concretize Error Trace



1 U new+1 = new

lock ();
old = new

2 L new+1 = old

[T]

3 L new+1 = old

unlock ();
new++

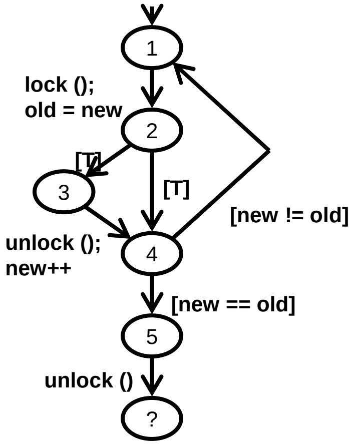
4 U new = old

[new == old]

5 U true

unlock ()

? E true



1

lock ();
old = new

2

[T]

3

[T]

[new != old]

unlock ();
new++

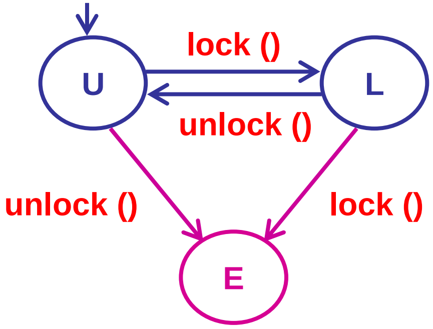
4

[new == old]

5

unlock ()

?



U

lock ()

L

unlock ()

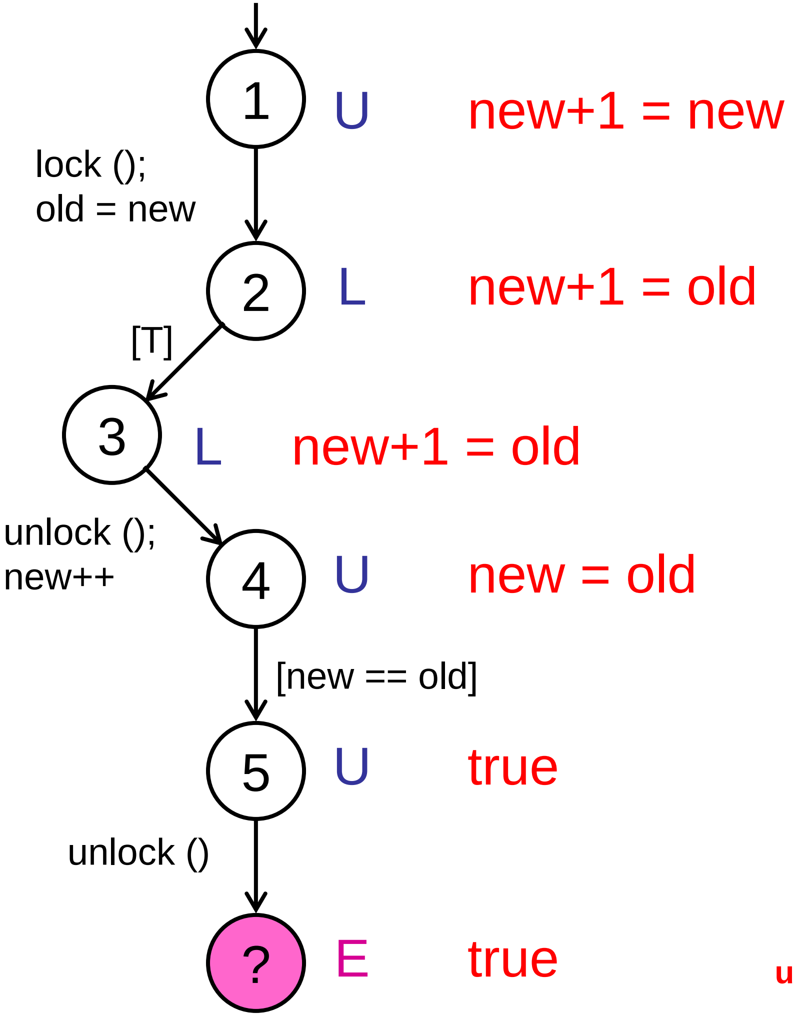
unlock ()

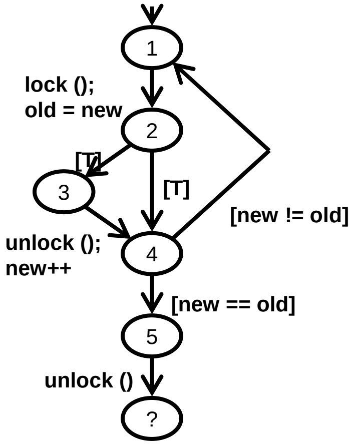
lock ()

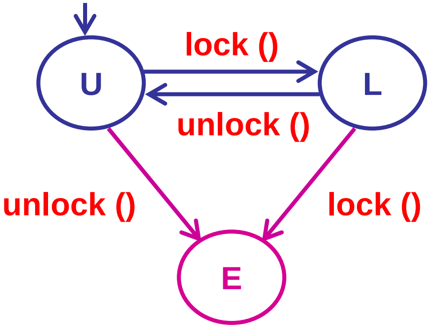
E

Concretize Error Trace

Spurious!



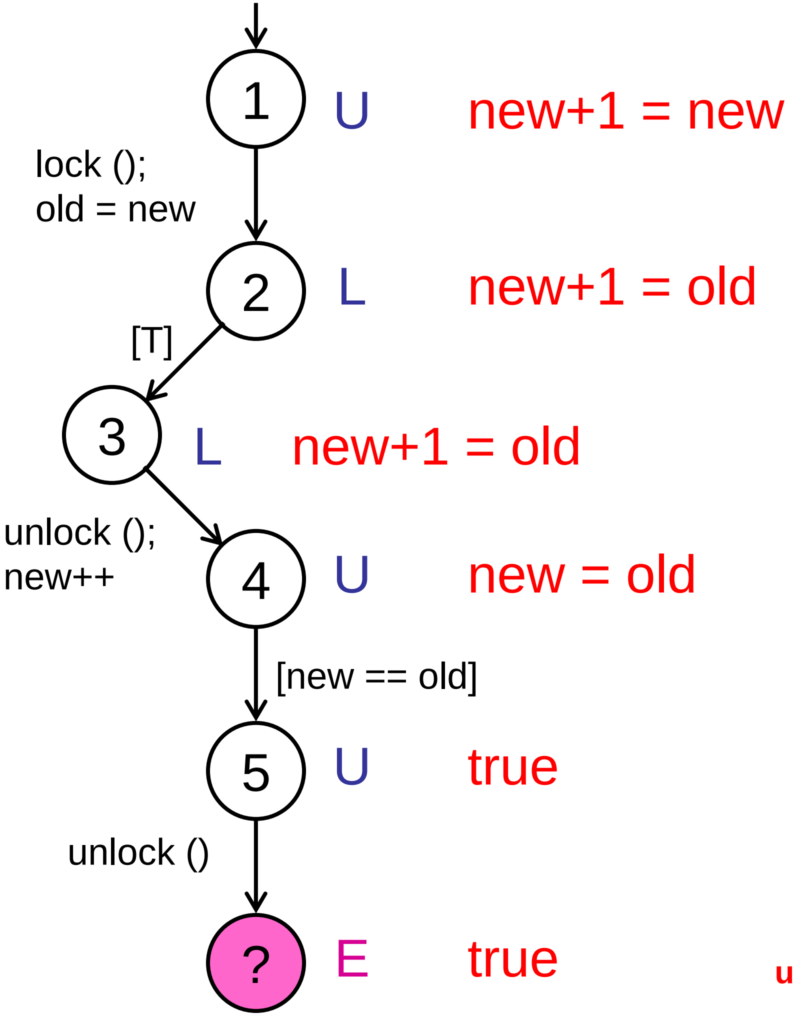




Concretize Error Trace

Spurious!

new = old
is a relevant predicate



1 U new+1 = new

lock ();
old = new

2 L new+1 = old

[T]

3 L new+1 = old

unlock ();
new++

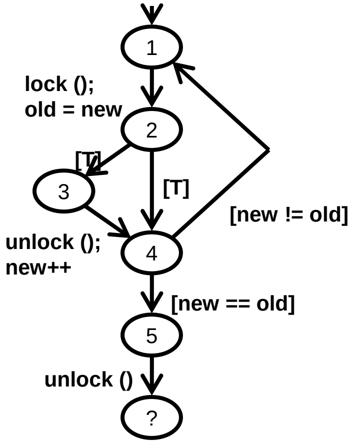
4 U new = old

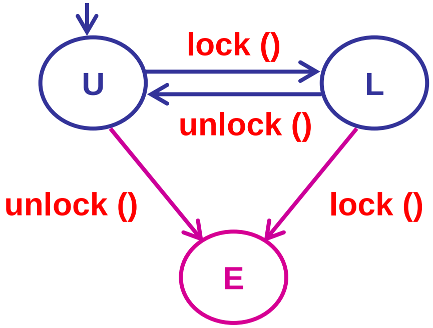
[new == old]

5 U true

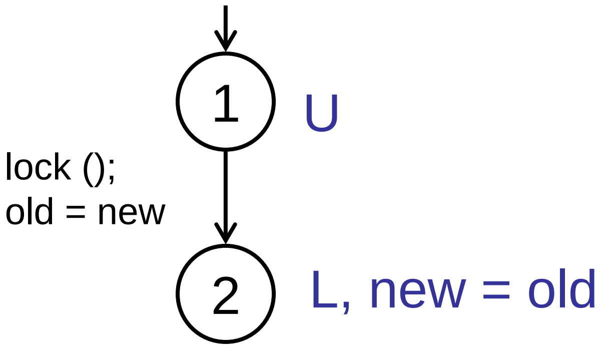
unlock ()

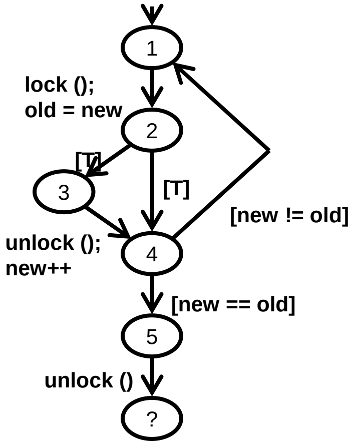
? E true

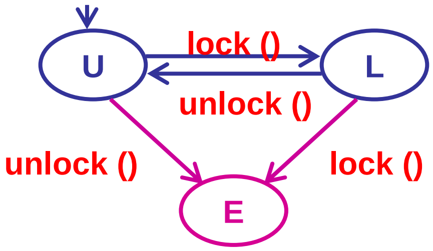




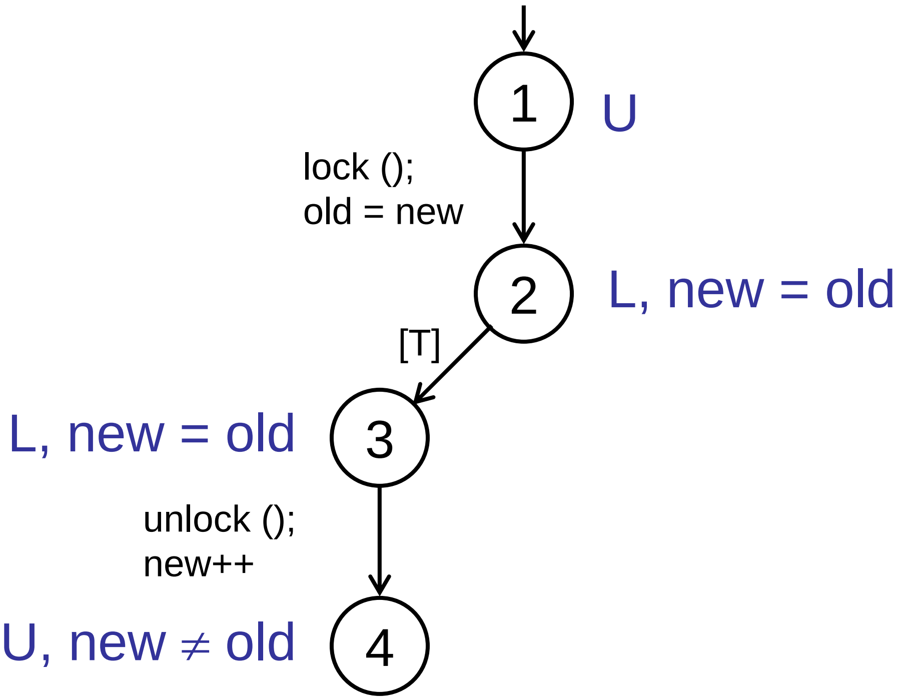
Refined Abstract Reachability

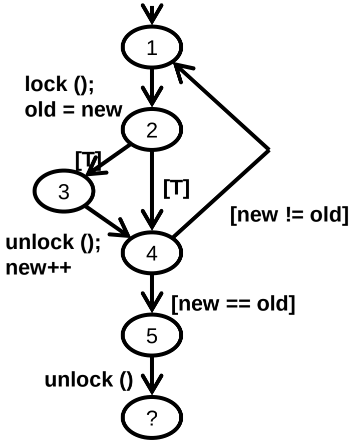


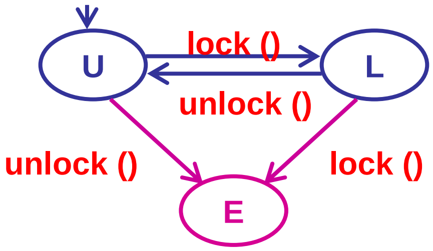




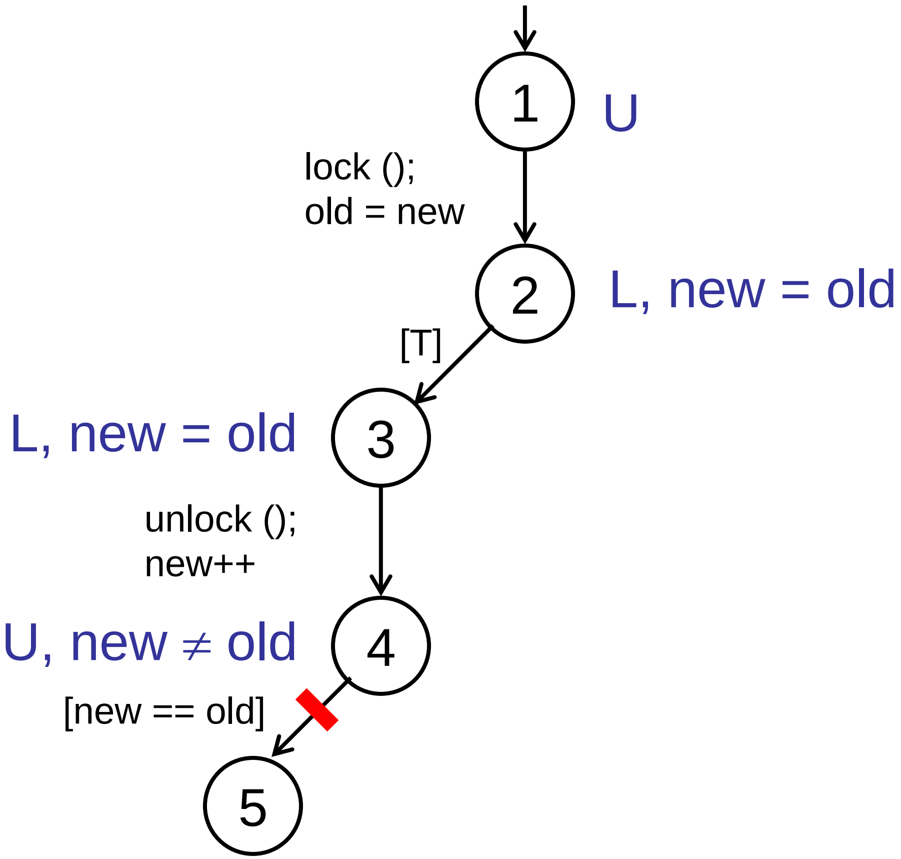
Refined Abstract Reachability

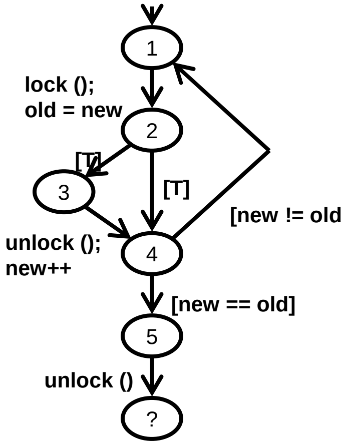


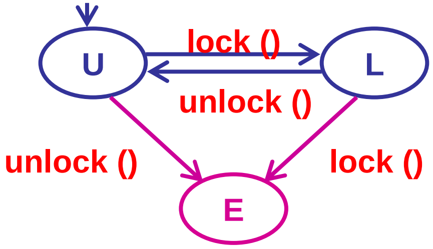




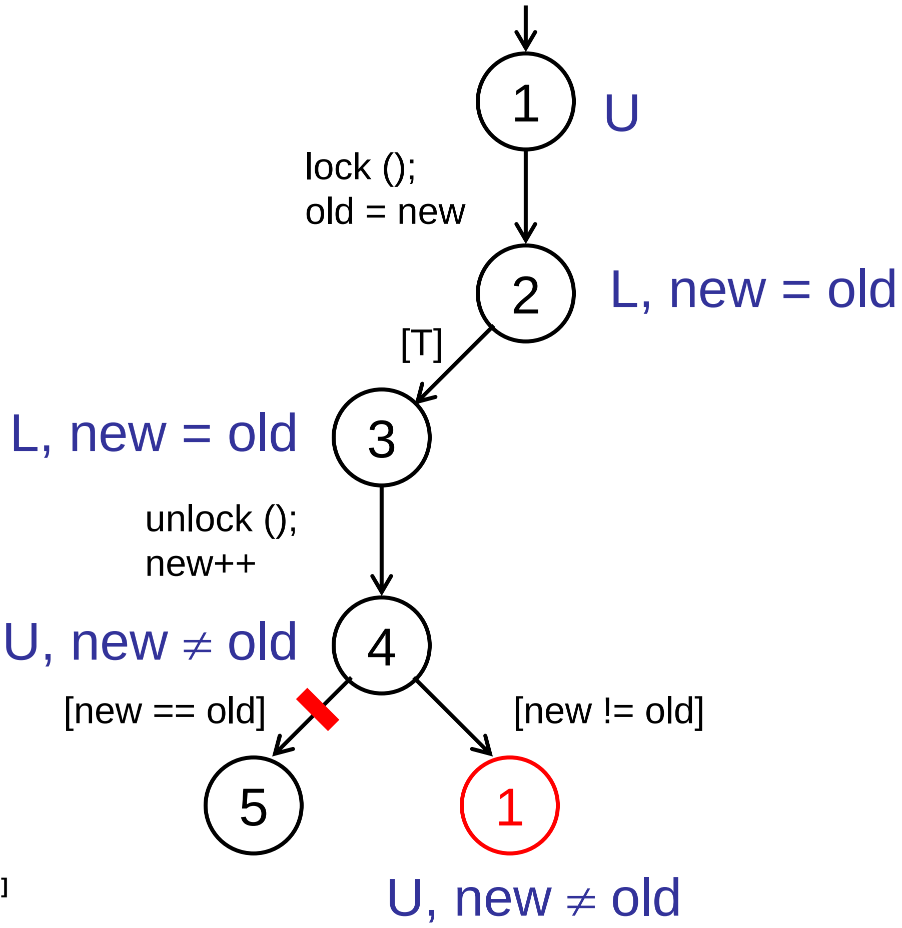
Refined Abstract Reachability

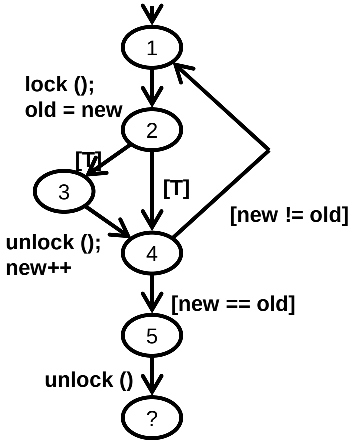


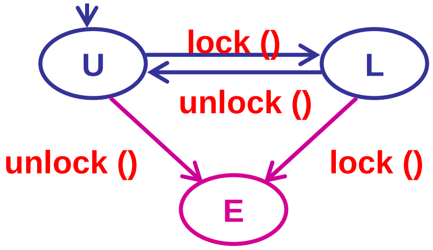




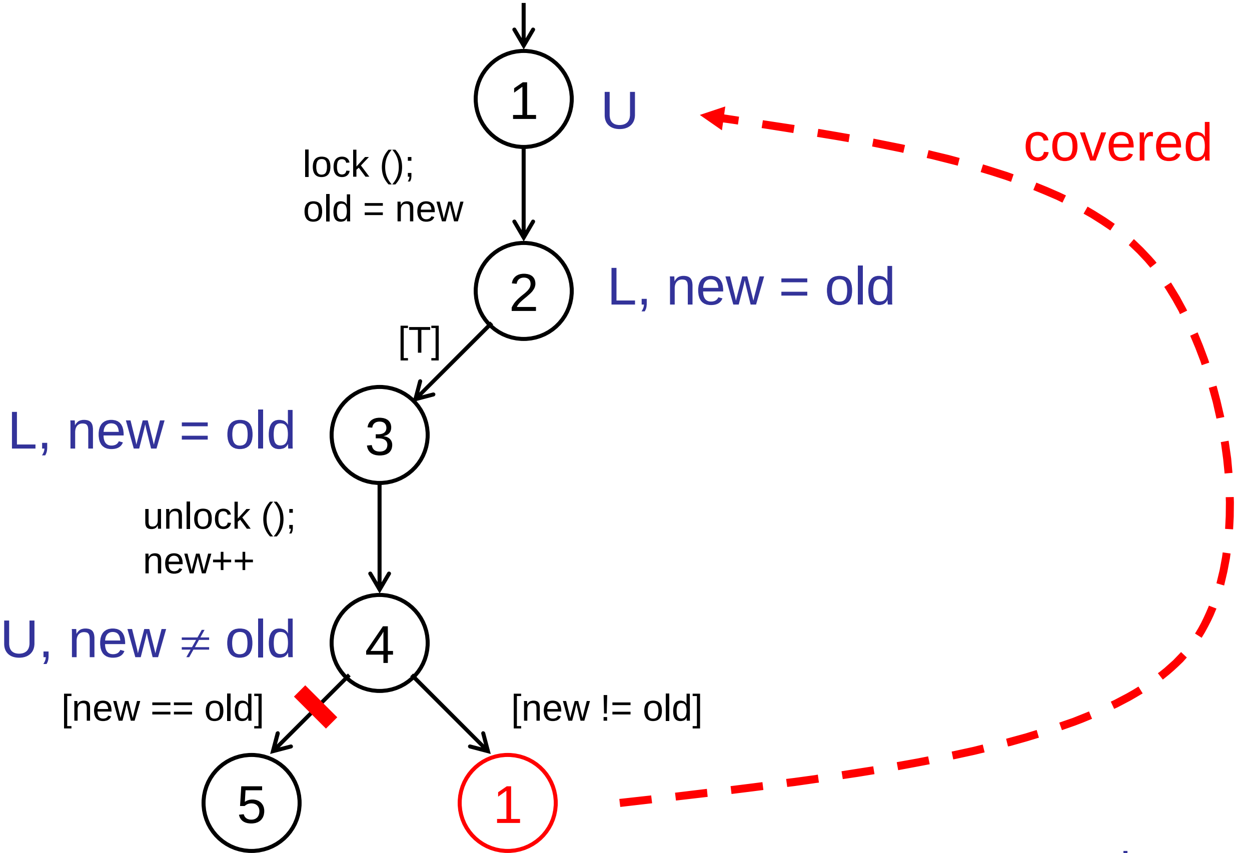
Refined Abstract Reachability







Refined Abstract Reachability



1 U

lock ();
old = new

2 L, new = old

[T]

3 L, new = old

unlock ();
new++

4 U, new ≠ old

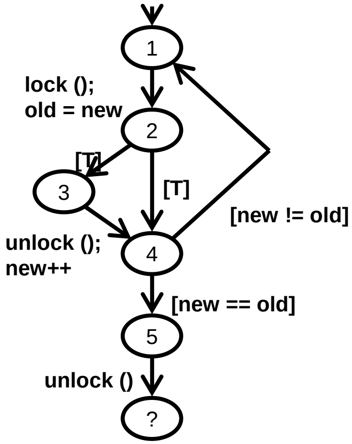
[new == old]

[new != old]

5

1 U, new ≠ old

covered



1

lock ();
old = new

2

[T]

3

[T]

[new != old]

unlock ();
new++

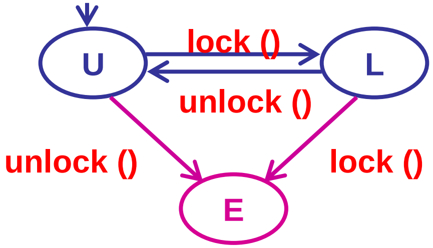
4

[new == old]

5

unlock ()

?



U

lock ()

L

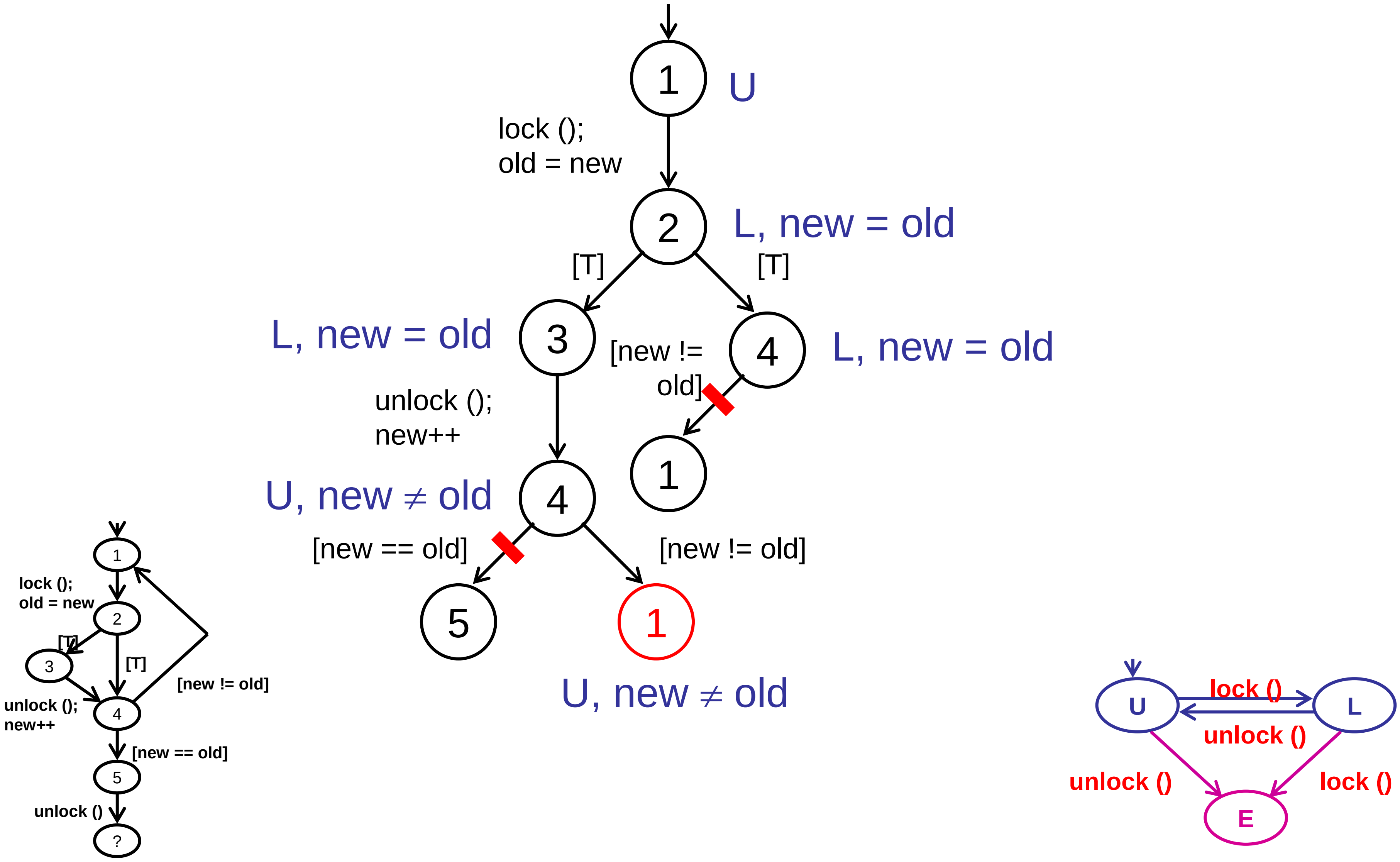
unlock ()

unlock ()

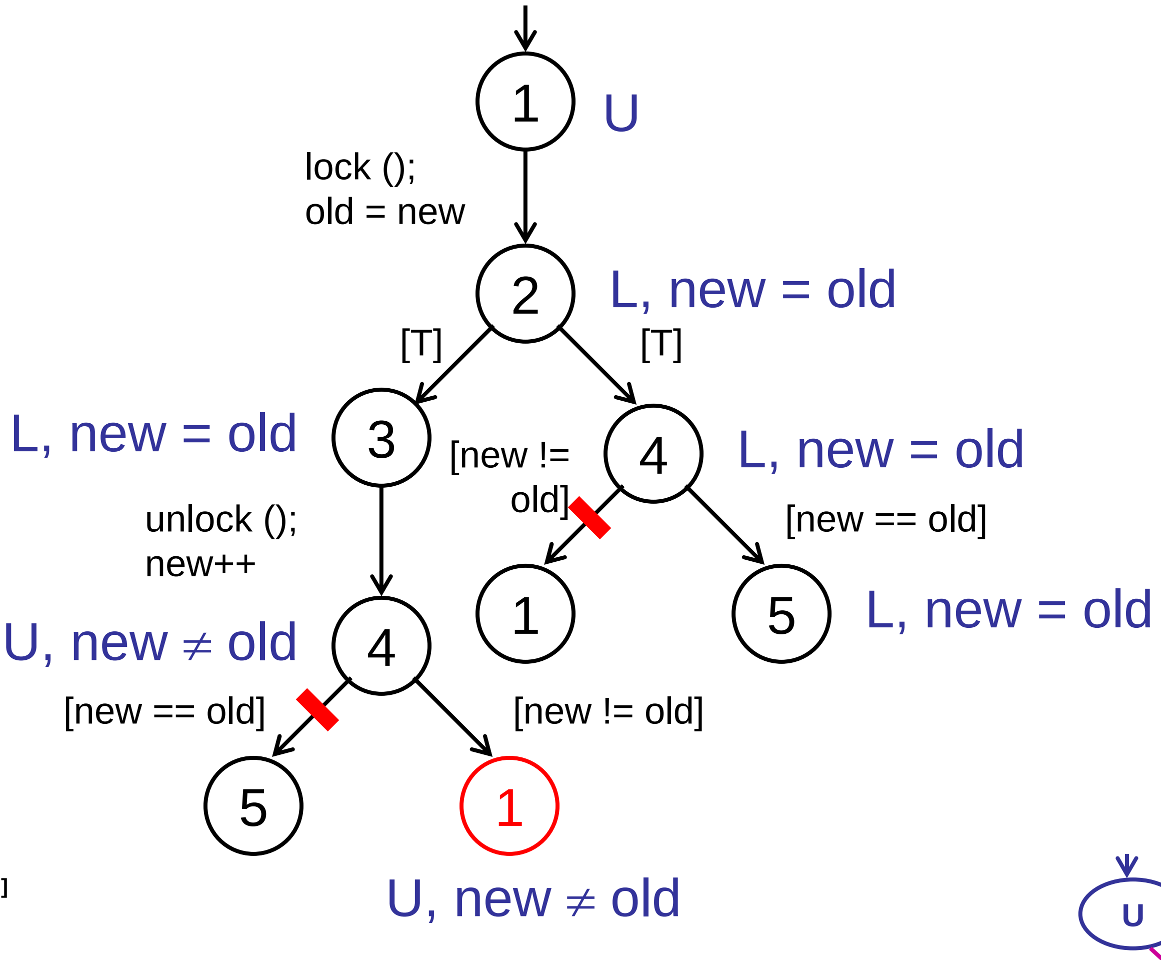
lock ()

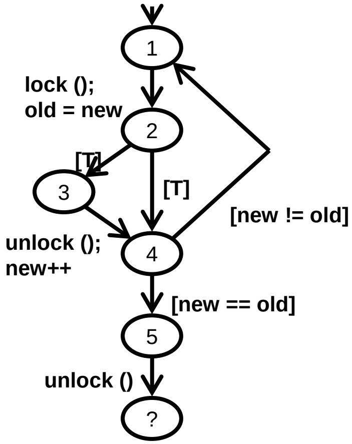
E

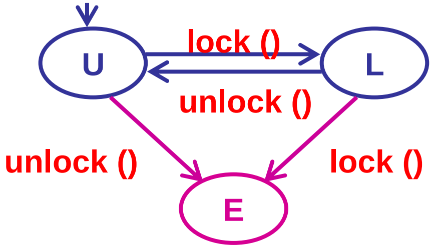
Refined Abstract Reachability



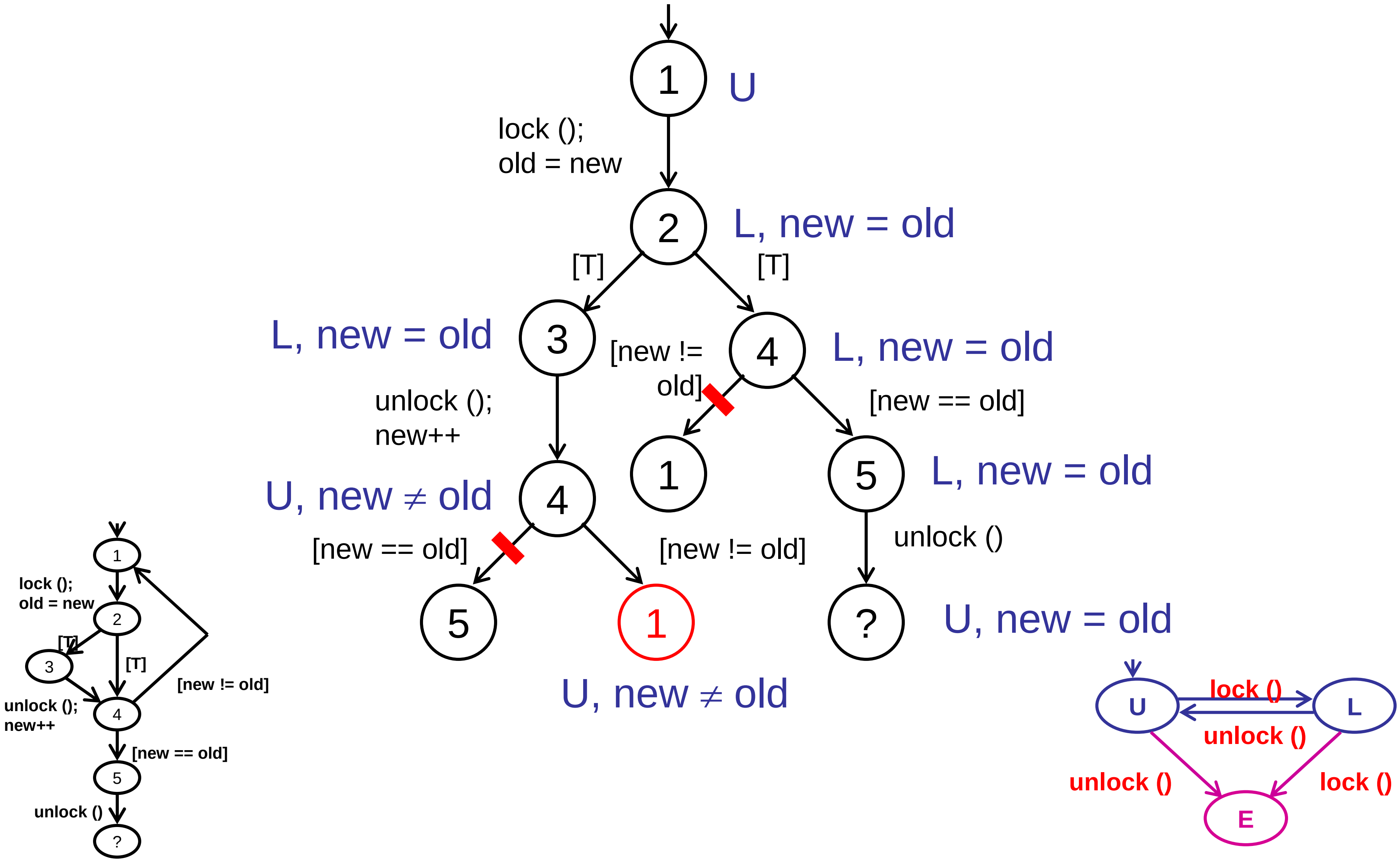
Refined Abstract Reachability







Abstract Reachability Tree



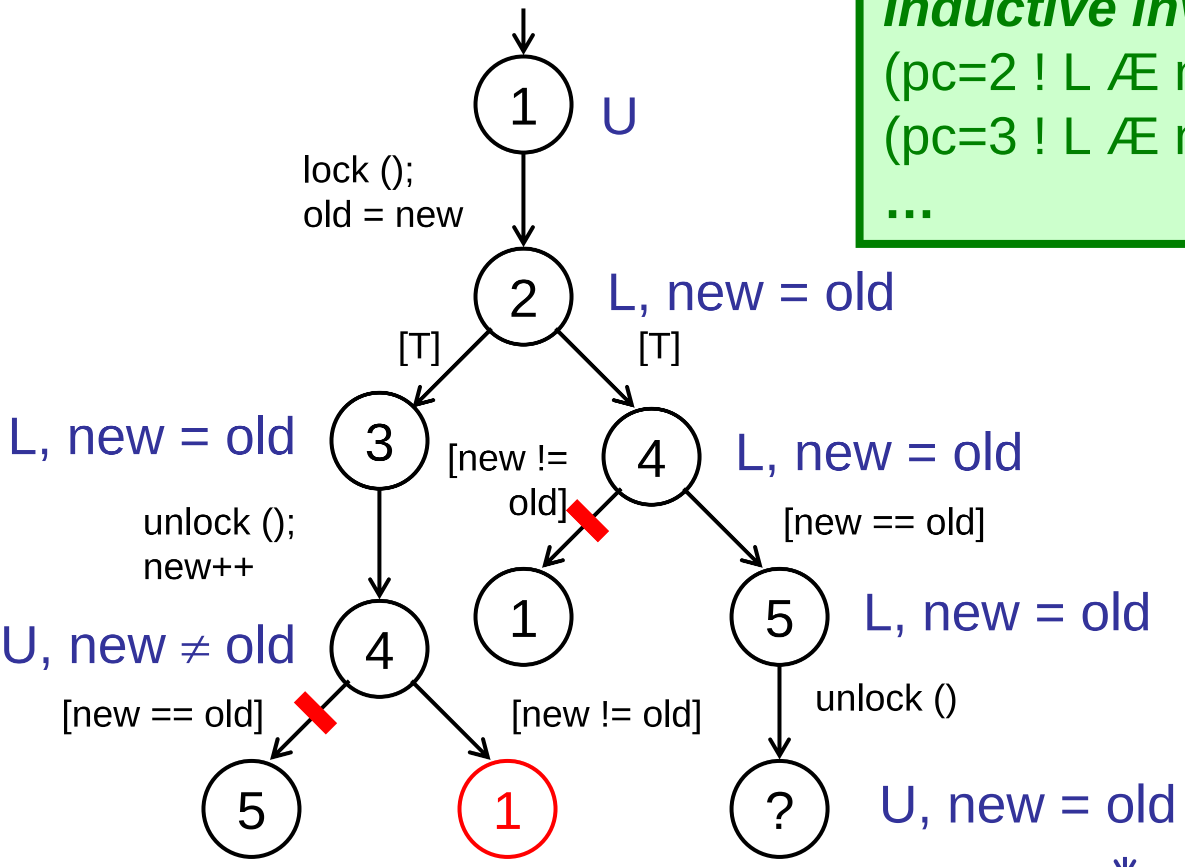
Abstract Reachability Tree = Proof

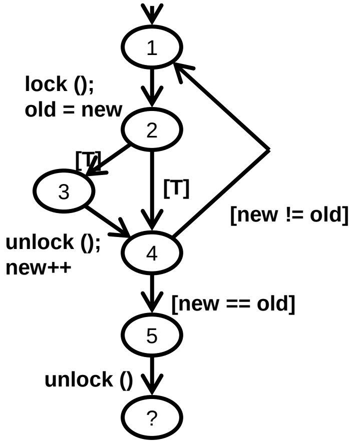
Inductive invariant:

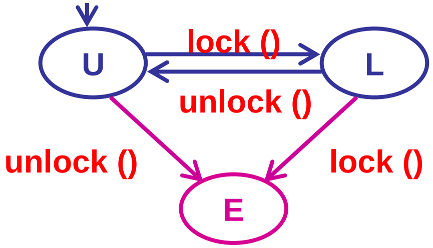
(pc=2 ! L Æ new=old) Æ

(pc=3 ! L Æ new=old) Æ

…







Lesson 5:

Automatic Program Verification

=

Abstract + Search

Identify relevant facts.

Track relevant facts.

A Brief History (and Future?) of Model Checking

1980s: **T**heory of finite-state model checking [Clarke/Emerson, Sifakis, et al.]

1990s: **T**echniques to combat state explosion

Finite-state model checking penetrates the hardware industry [Fujitsu, Intel, Motorola, Siemens, etc.]

Theory of infinite-state model checking

2000s: **T**echniqes for automatic abstraction

Infinite-state model checking penetrates the software industry ?!