CPALockator

Thread-Modular Approach with Transition Abstraction for Analysis of Multithreaded Software

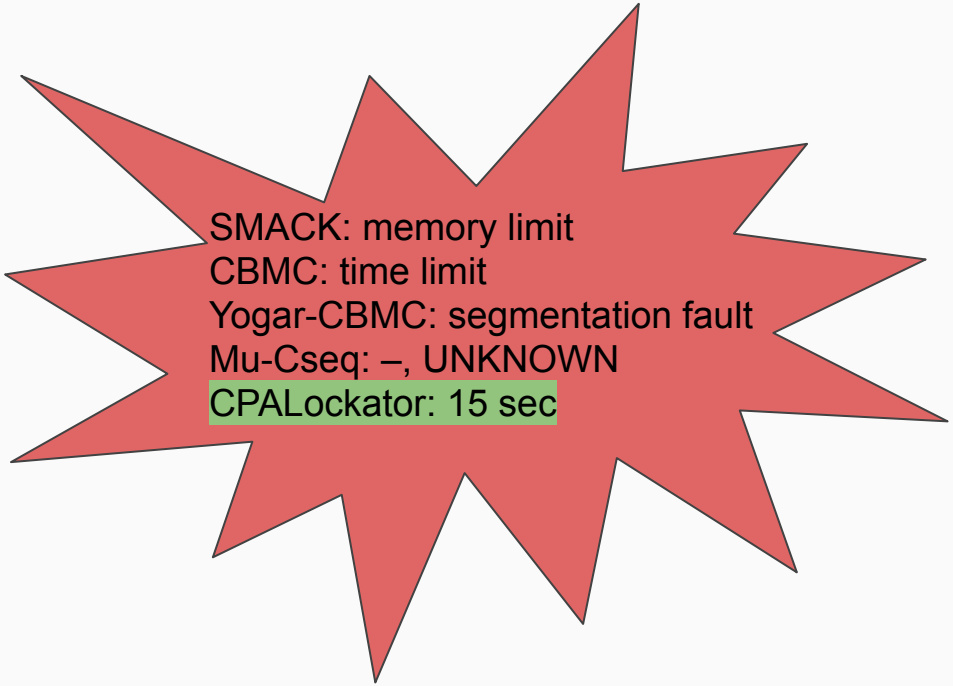
Pavel Andrianov, andrianov@ispras.ru

Motivation

Linux module drivers/net/irda/w83977af_ir.ko: 10 000 LOC

```
static void w83977af_change_speed(struct
w83977af_ir *self , __u32 speed ){
  ...
  self->io.speed = speed;
  ...
}
```

```
static void w83977af_hard_xmit(struct
sk_buff *skb , struct net_device *dev){
  ...
  speed = irda get next speed(skb);
  tmp speed = self->io.speed;
  assert(self->io.speed == tmp_speed);
  if ((speed != self->io.speed) && ...) {
  …
  }
}
```



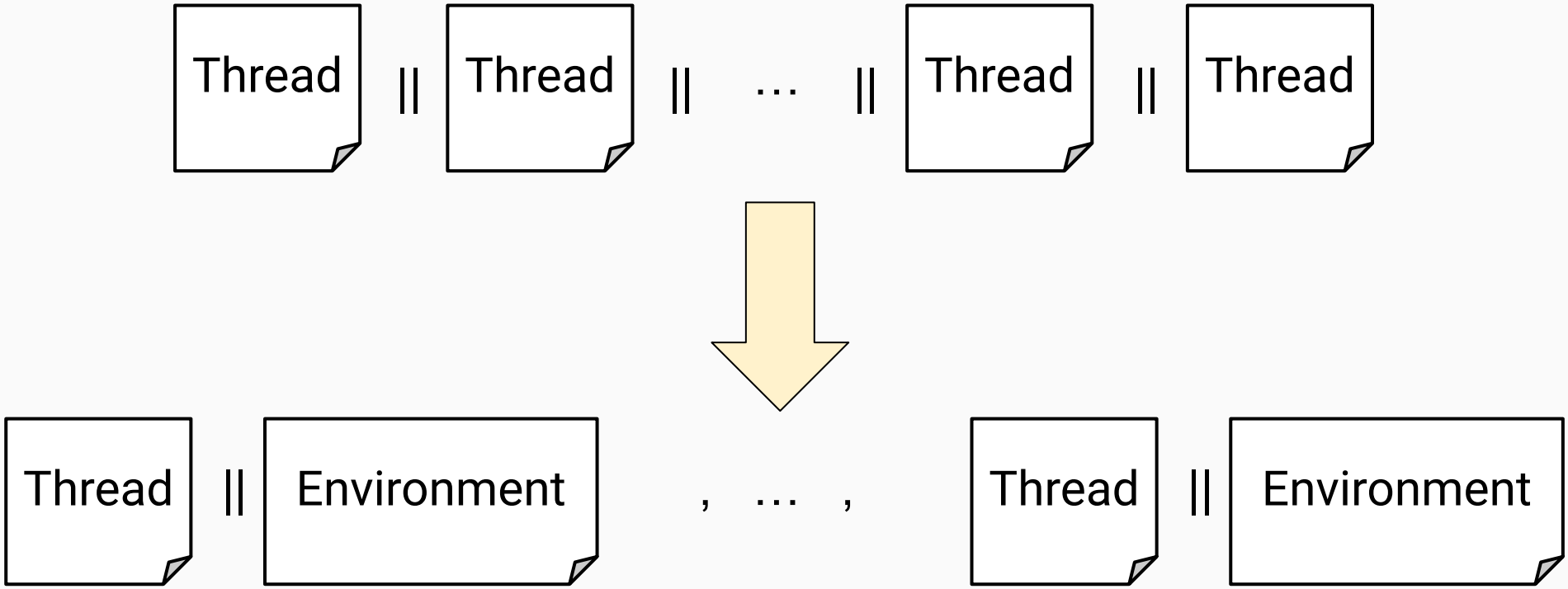
The goals of a new theory

Scaling on a real software

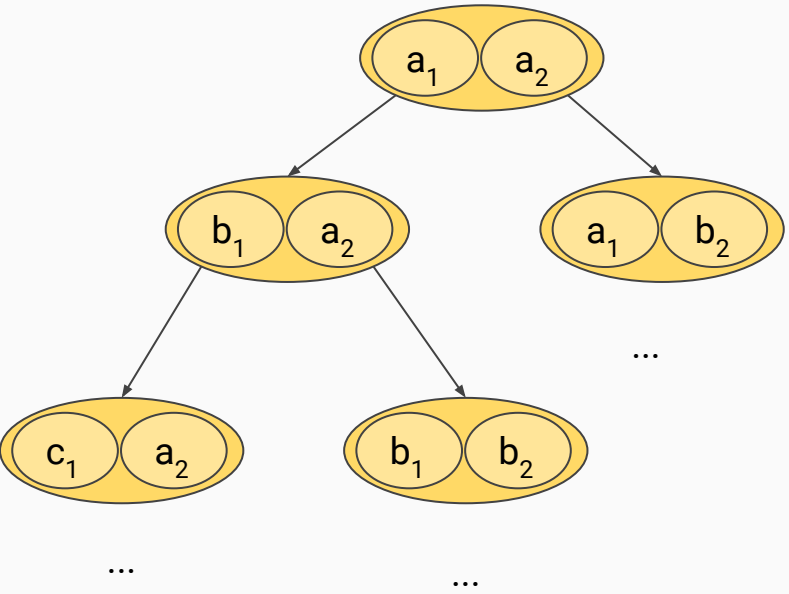
Small amount of false alarms

Flexible balance between speed and precision

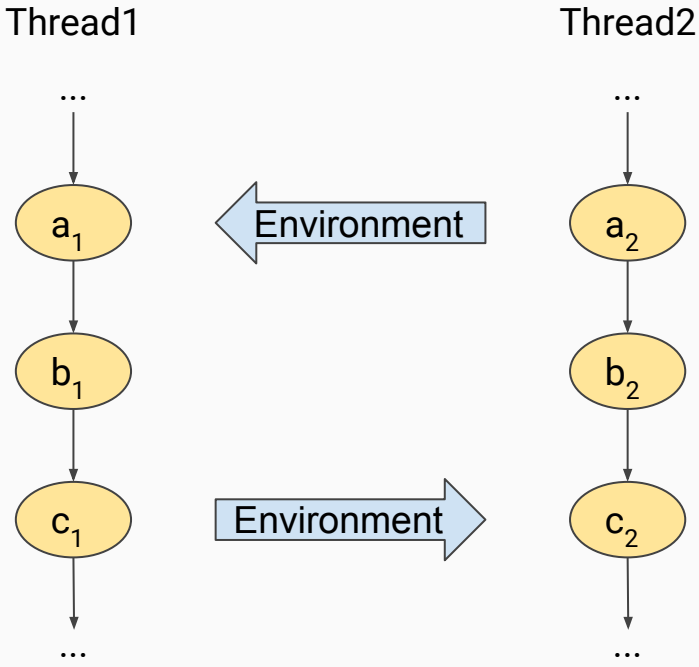
Thread-Modular Approach



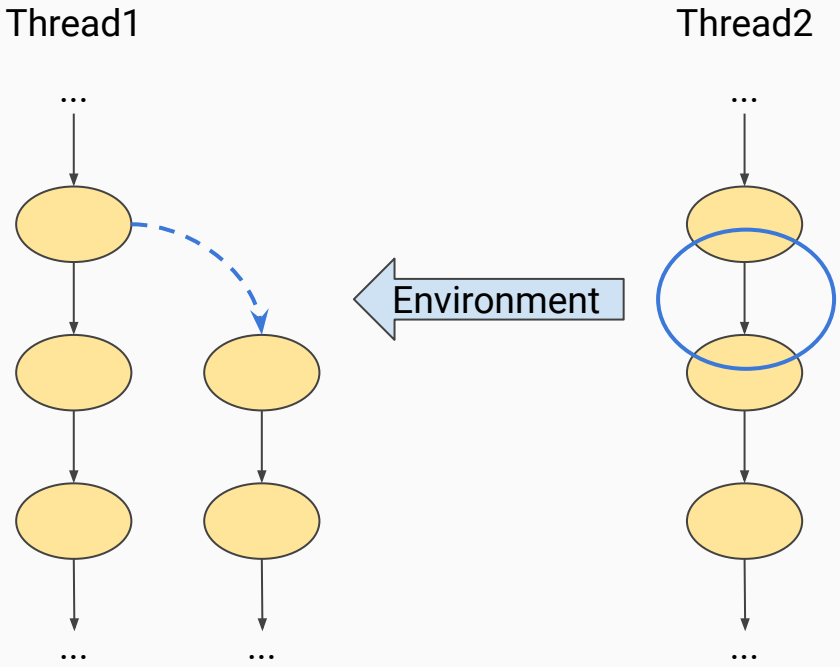
Interleavings



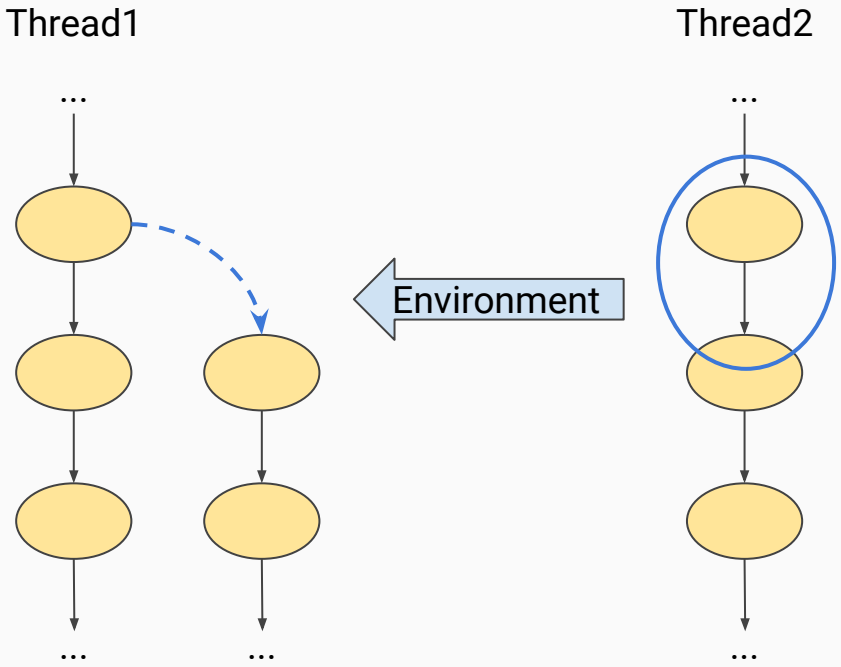
Partial states



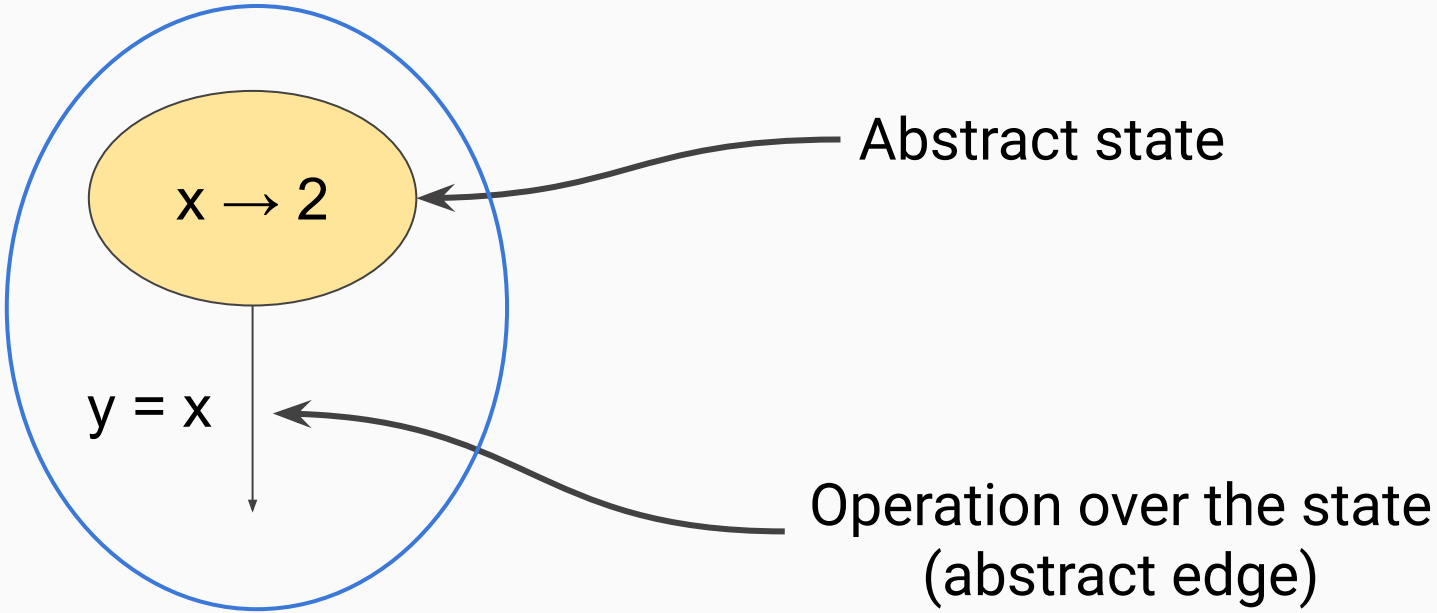
Environment actions based on inference objects



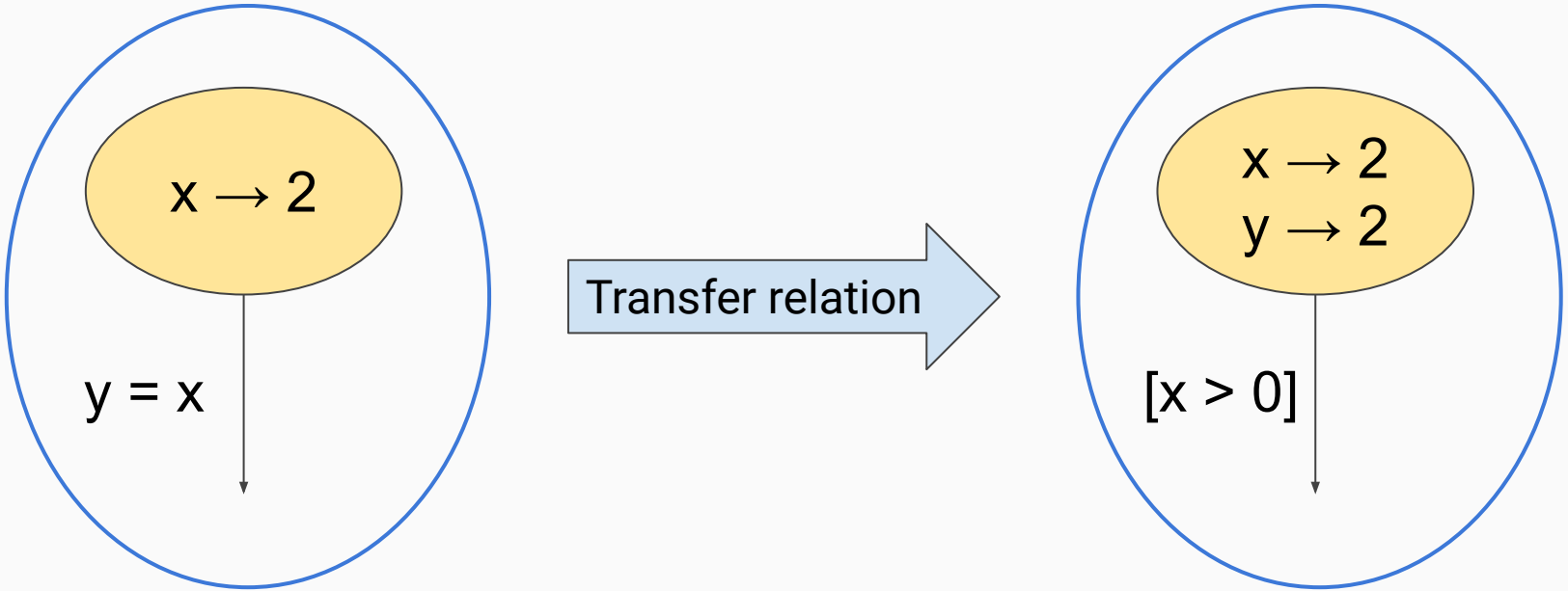
Environment actions based on abstract transitions



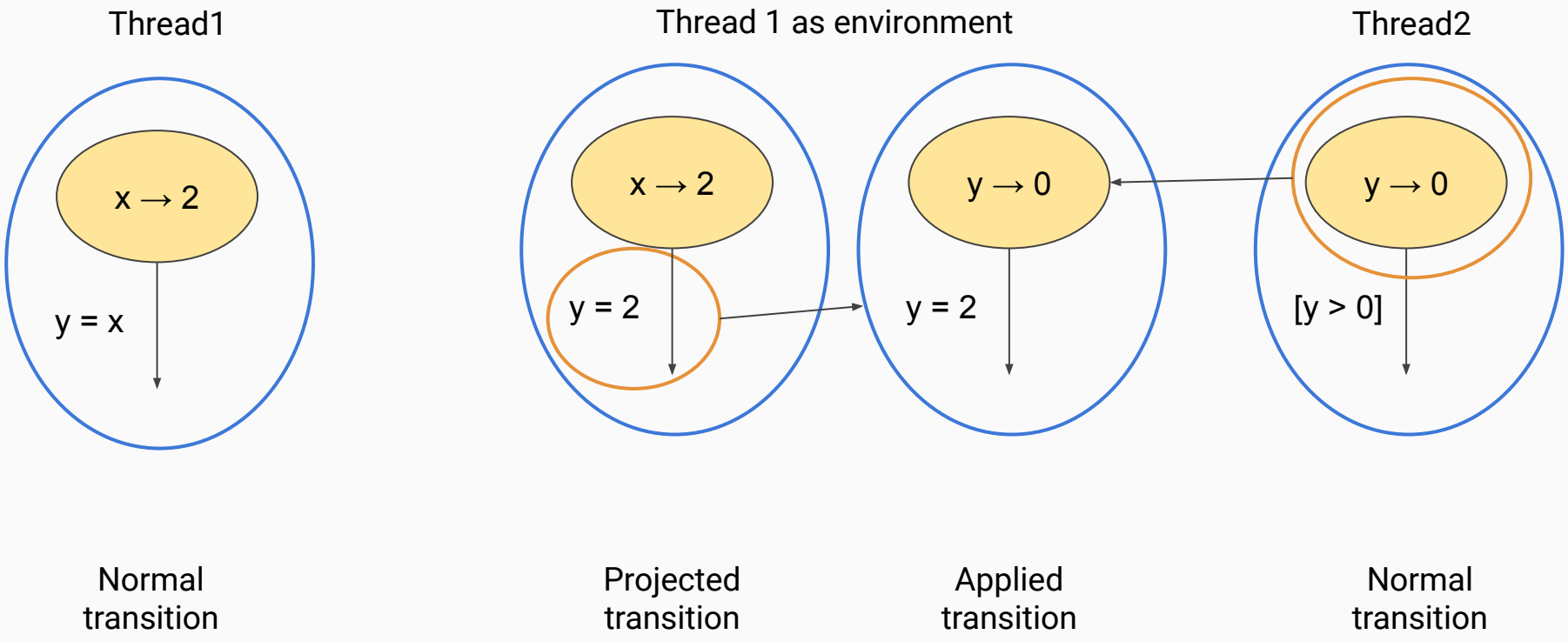
Abstract transitions



Abstract transitions



Environment computation



Extension of the theory

$waitlist := \{e_0\}$ $reached := \{(e_0, \pi_0)\}$;
while $waitlist \neq \emptyset$ **do**
  pop e from $waitlist$;
  for *each* e' *in* $(e, reached) \rightsquigarrow (e', \pi')$ **do**
    $(\widehat{e}, \widehat{\pi}) = prec(e', \pi', reached)$;
    for *each* $(e'', \pi'') \in reached$ **do**
      $e_{new} = merge(\widehat{e}, e'', \widehat{\pi})$;
      if $e_{new} \neq e''$ **then**
        $waitlist := waitlist \setminus \{(e'', \pi'')\} \cup \{(e_{new}, \widehat{\pi})\}$;
        $reached := reached \setminus \{(e'', \pi'')\} \cup \{(e_{new}, \widehat{\pi})\}$;
      end
    end
    if $!stop(\widehat{e}, reached, \widehat{\pi})$ **then**
      $waitlist := waitlist \cup \{(\widehat{e}, \widehat{\pi})\}$;
      $reached := reached \cup \{(\widehat{e}, \widehat{\pi})\}$;
    end
  end
end

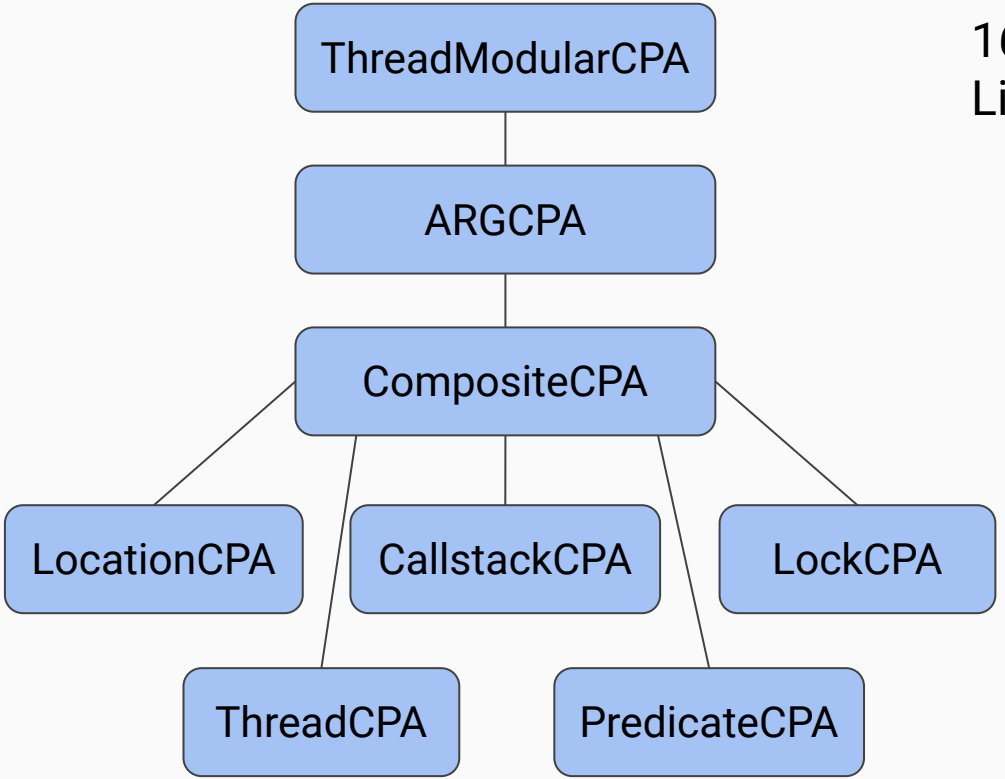
Transfer Relation of ThreadModularCPA

for $each\ \widehat{e} : e_0 \rightsquigarrow_I \widehat{e}$ **do**
  $result := result \cup \{\widehat{e}\}$;
  for $each\ e' \in reached$ **do**
    $result := result \cup \{apply(e', \widehat{e}|_p)\}$;
    $result := result \cup \{apply(\widehat{e}, e'|_p)\}$;
  end
end

Optimized Transfer Relation

for *each* $e' \in reached$ **do**
  $transitions := transitions \cup \{apply(e', e_0)\}$;
  $transitions := transitions \cup \{apply(e_0, e')\}$;
end
for *each* $e \in transitions$ **do**
  for *each* $\widehat{e} : e \rightsquigarrow_I \widehat{e}$ **do**
    $result := result \cup \{\widehat{e}\} \cup \{\widehat{e}|_p\}$;
  end
end

Experiments



16 tasks x {true, false} = 32 benchmarks
Limits: 15 min, 8 Gb

Results

Approach	Theory with inference objects	Theory with abstract transitions
False verdicts		
Correct	10	12
Incorrect	2	2
True verdicts	12	11
Unknowns	8	7
Time(s)	10200	9820

Refinement of environment formula encoding

```
int f() {
  int tmp = 2;
  g = 1;
  if (g != 0) {
    g++;
  }
  if (tmp < 10) {
    ERROR();
  }
}
```

Error Path

```
tmp = 2;
g = 1;
[ g == 0 ]
[ tmp < 10 ]
ERROR();
```

Precise formula

- tmp = 2
- g = 1
- [g == 0]
- tmp < 10

Imprecise formula

- tmp = 2
- g = *
- [g == 0]
- tmp < 10

Interpolants

- [g != 0]
- [tmp == 2]

Results

Approach	Base refinement	Imprecise+Precise combination
False verdicts		
Correct	12	12
Incorrect	2	2
True verdicts	11	11
Unknowns	7	7
Time(s)	9820	9790

Refinement with iterative proactive effects application

```
int f() {
  int tmp = 2;
  g = 1;
  if (g != 0) {
    g++;
  }
  if (tmp < 10) {
    ERROR();
  }
}
```

Error Path

tmp = 2;

g = 1;

[g == 0]

[tmp < 10]

ERROR();

Precise formula

tmp = 2

g = 1

[g == 0]

tmp < 10

Precise formula with effects

tmp = 2

g = 1

[g == 0]

tmp < 10

g = *

Interpolants

[g != 0]

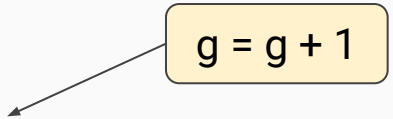
[tmp == 2]

Results

Approach	Base version	Proactive refinement
False verdicts		
Correct	12	15
Incorrect	2	2
True verdicts	11	12
Unknowns	7	3
Time(s)	9820	7780

Predicate Abstract edge (effect)

Origin Edge:



Statement

Projected Edge:

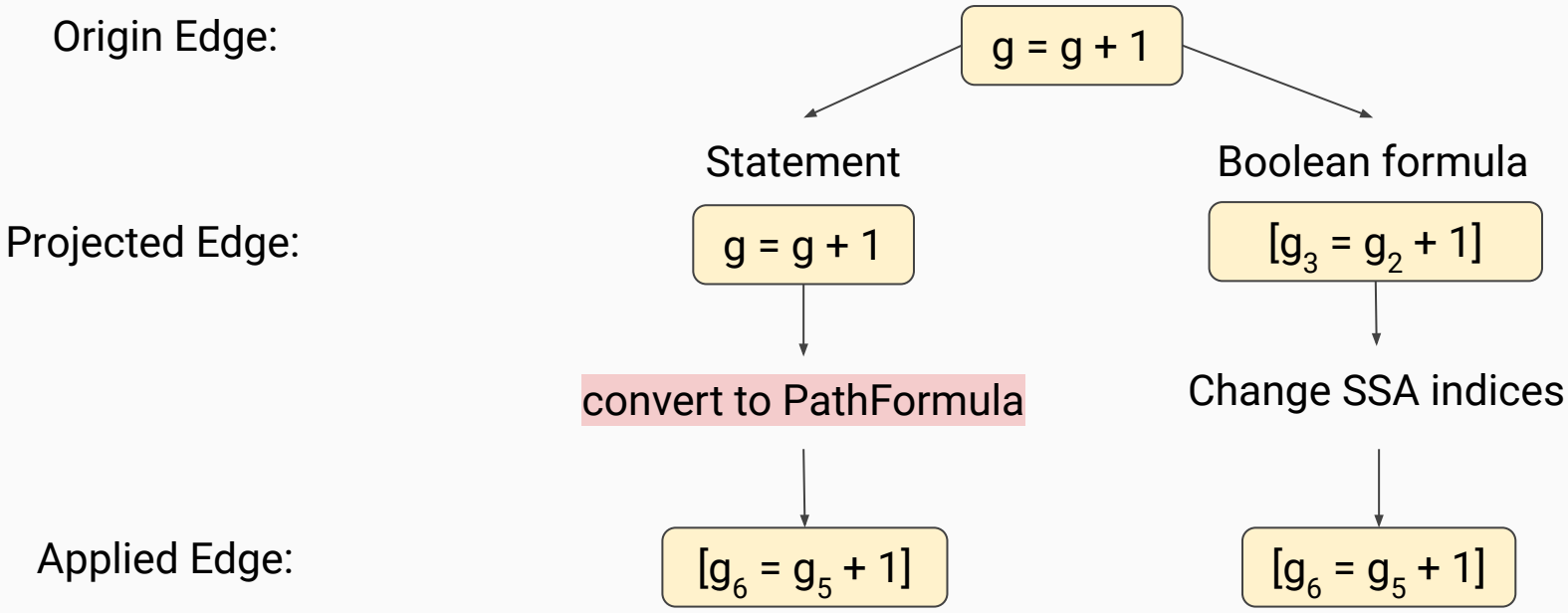
g = g + 1

convert to PathFormula

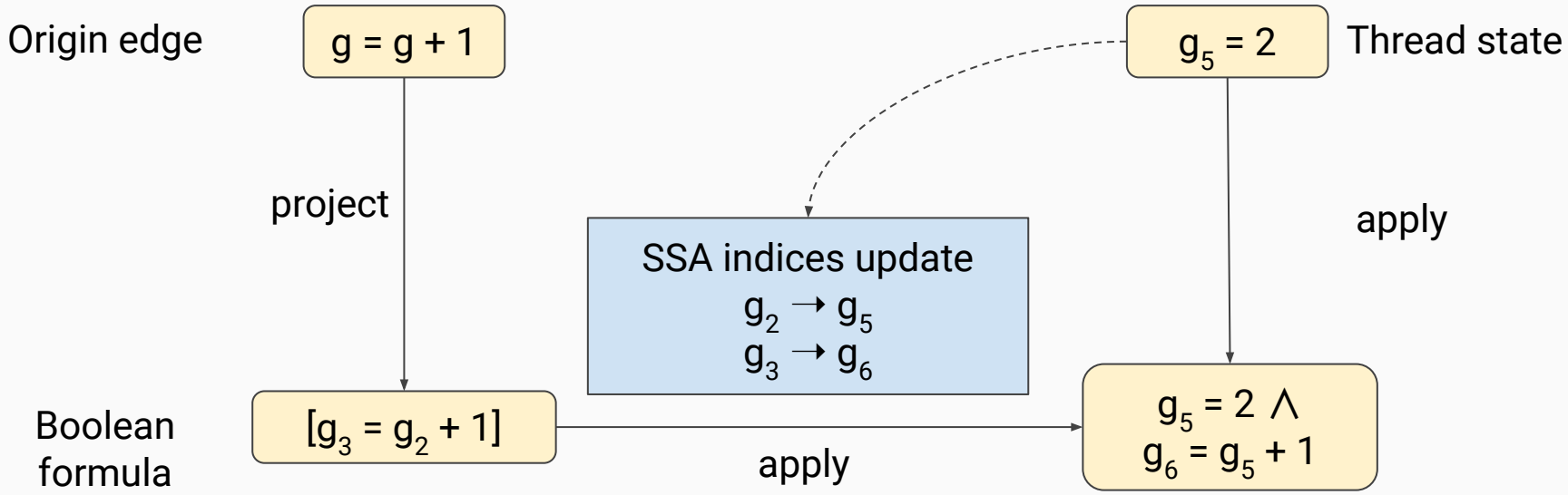
Applied Edge:

$[g_6 = g_5 + 1]$

Predicate Abstract edge (effect)



Instantiating of formulas



Results

Approach	Base version	Formula effects
False verdicts		
Correct	12	13
Incorrect	2	2
True verdicts	11	12
Unknowns	7	5
Time(s)	9820	6600

Results

Approach	Base version	Proactive refinement + Formula effects
False verdicts		
Correct	12	16
Incorrect	2	2
True verdicts	11	13
Unknowns	7	2
Time(s)	9820	3950

Adjustable block encoding