From Non-preemptive to Preemptive Scheduling using Synchronization Synthesis

Roopsha Samanta

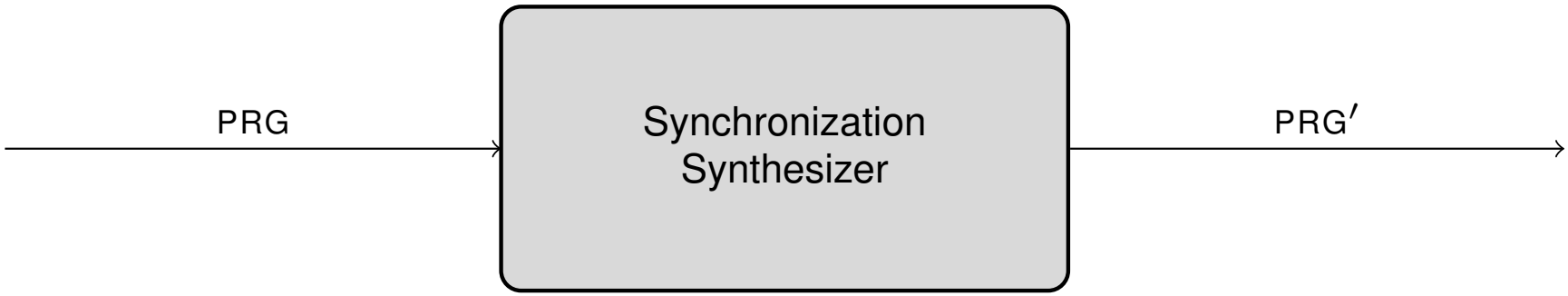
IST Austria

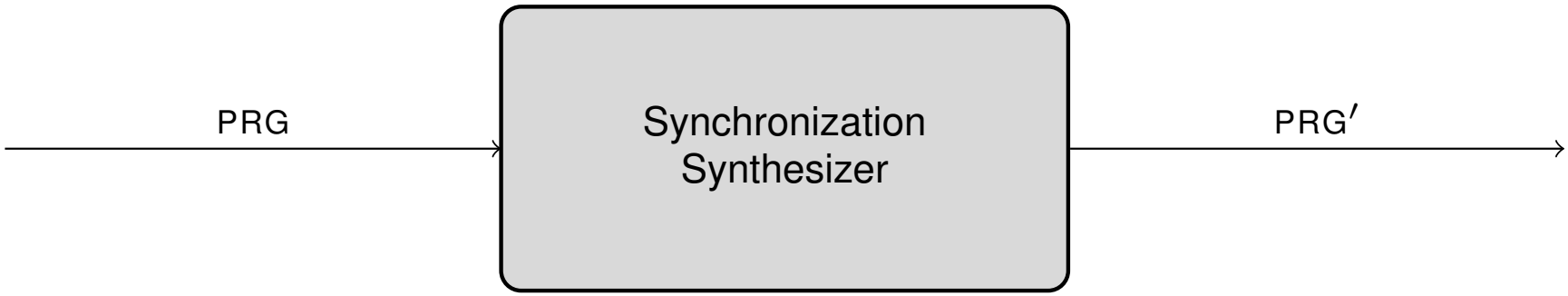
Joint work with Pavol Černý, Edmund M. Clarke, Tom Henzinger, Arjun Radhakrishna, Leonid Ryzhyk and Thorsten Tarrach

24 July, 2015

- Non-preemptive scheduler
- Preemptive scheduler

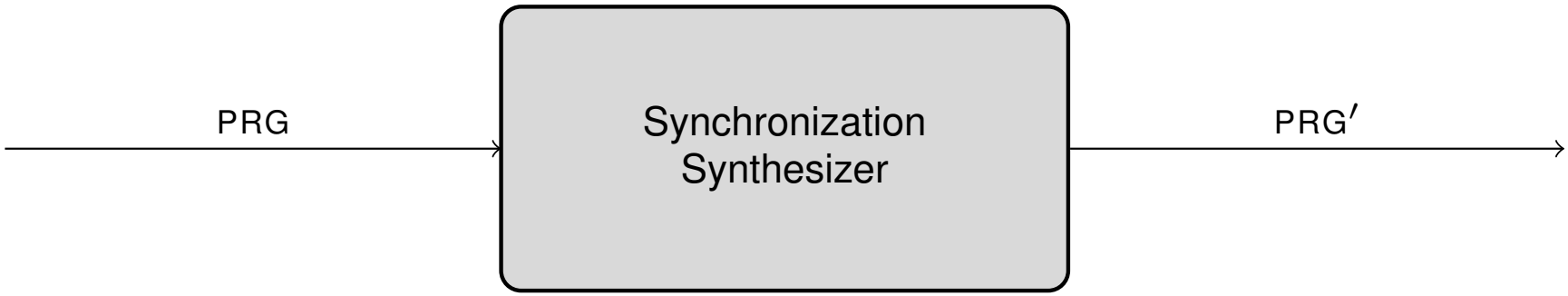
- Non-preemptive scheduler
  - Thread descheduled only if it yields control or terminates
- Preemptive scheduler
  - Thread can be descheduled at any point


PRG
Synchronization Synthesizer
PRG′



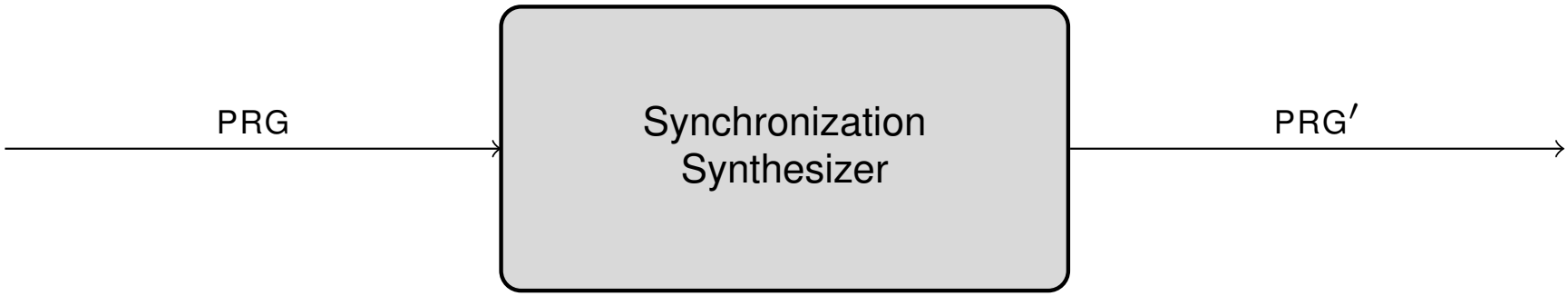
Programmer:

- Writes PRG



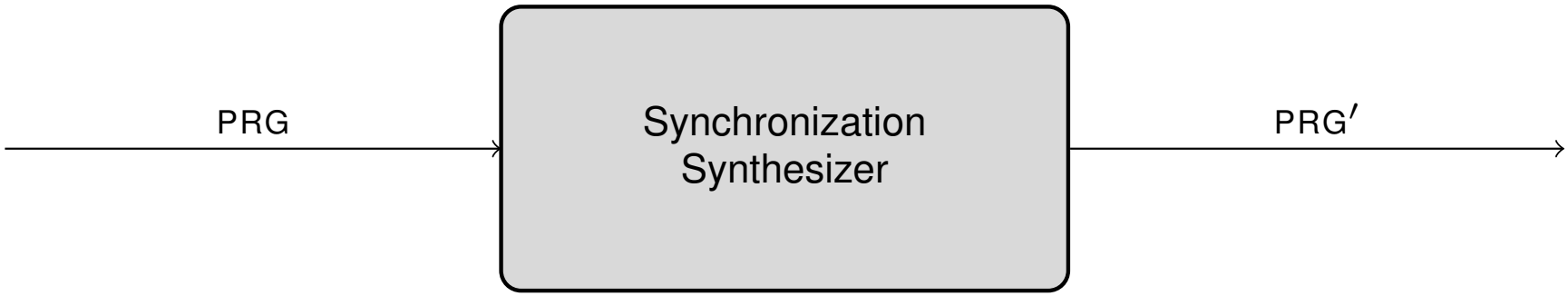
Programmer:

- Writes PRG
- Skips tricky synchronization



Programmer:

- Writes PRG
- Skips tricky synchronization
- Ensures:
  correct under nonpreemptive scheduler

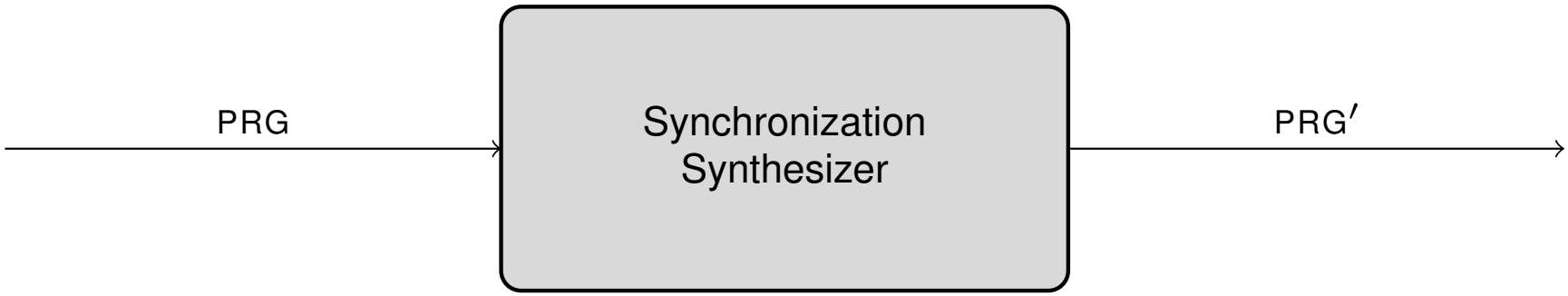


Programmer:

- Writes PRG
- Skips tricky synchronization
- Ensures:
  correct under nonpreemptive scheduler

Synthesizer:

- Automatically generates PRG′

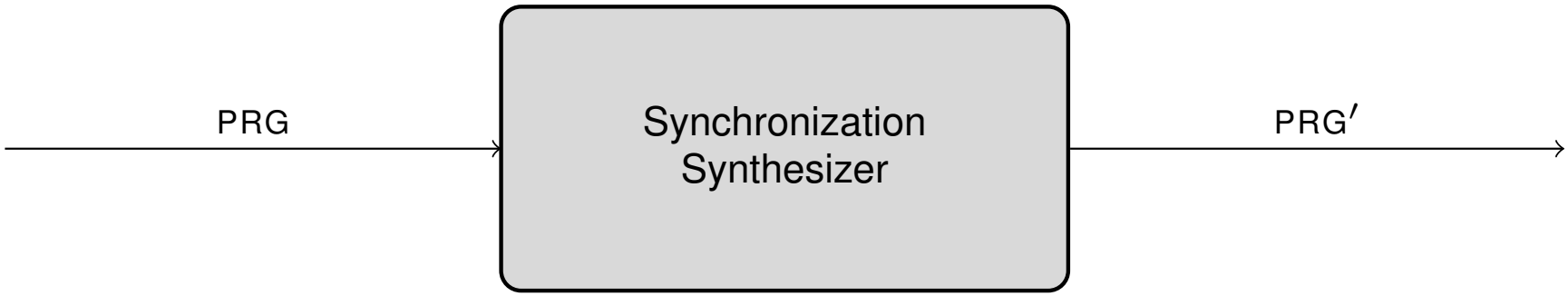


Programmer:

- Writes PRG
- Skips tricky synchronization
- Ensures: correct under nonpreemptive scheduler

Synthesizer:

- Automatically generates PRG′
- Inserts synchronization

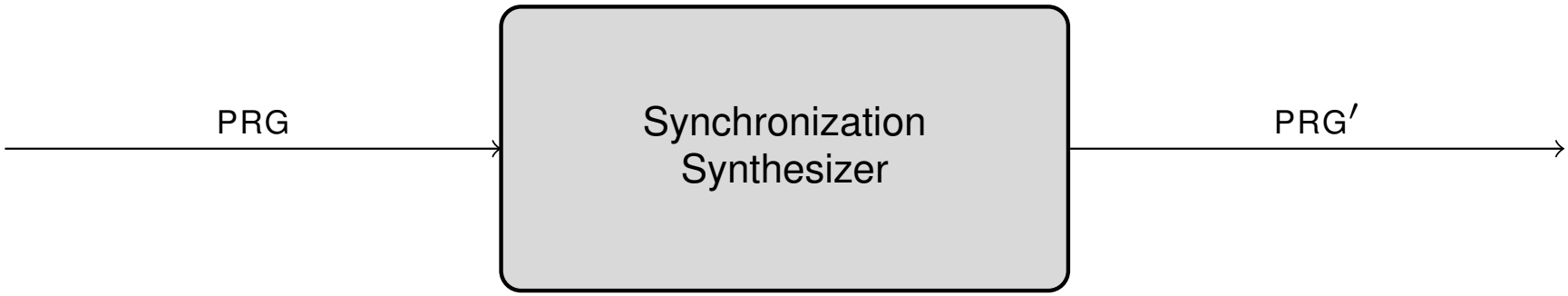


Programmer:

- Writes PRG
- Skips tricky synchronization
- Ensures:
  correct under nonpreemptive scheduler

Synthesizer:

- Automatically generates PRG′
- Inserts synchronization
- Ensures:
  correct under preemptive scheduler



Programmer:

- Writes PRG
- Skips tricky synchronization
- Ensures: correct under nonpreemptive scheduler

Synthesizer:

- Automatically generates PRG′
- Inserts synchronization
- Ensures: correct under preemptive scheduler

$$[\![\text{PRG}']\!]^{\texttt{preempt}} \subseteq [\![\text{PRG}]\!]^{\texttt{nonpreempt}}$$

Preemption-safety

- Why use an implicit specification?
- Why preemption-safety?

- Why use an implicit specification?
  - Hard to write explicit specifications
  - Hard to ensure explicit specification is complete
- Why preemption-safety?

- Why use an implicit specification?
  - Hard to write explicit specifications
  - Hard to ensure explicit specification is complete
- Why preemption-safety?
  - Intuitively appealing
  - Encouraging user study [Sadowski and Yi 2010]
  - Our evaluation: preemption-safety can replace assertions

Target scenario

Module with interfaces

Observations: calls to interfaces

```
void open_dev()
 if (open==0)
   power_up();
 open := open+1;
 yield;

void close_dev()
 if (open>0)
   open := open-1;
   if (open==0)
     power_down();
 yield;
```

open := 0;

open = 0

```
   void open_dev()
→   if (open==0)
      power_up();
    open := open+1;
    yield;

   void close_dev()
    if (open>0)
      open := open-1;
      if (open==0)
        power_down();
    yield;
```

```
   void open_dev()
→   if (open==0)
      power_up();
    open := open+1;
    yield;

   void close_dev()
    if (open>0)
      open := open-1;
      if (open==0)
        power_down();
    yield;
```

$[\![\mathsf{PRG}]\!]^{\mathtt{nonpreempt}}$:

```
open := 0;
```

open = 0

```
  void open_dev()
→  if (open==0)
     power_up();
    open := open+1;
    yield;

  void close_dev()
    if (open>0)
      open := open-1;
      if (open==0)
        power_down();
    yield;
```

```
  void open_dev()
→  if (open==0)
     power_up();
    open := open+1;
    yield;

  void close_dev()
    if (open>0)
      open := open-1;
      if (open==0)
        power_down();
    yield;
```

$[\![\text{PRG}]\!]^{\texttt{nonpreempt}}$:

```
                                    open := 0;

power_up()      void open_dev()                 void open_dev()
                 if (open==0)          →         if (open==0)
              →    power_up();                     power_up();
                 open := open+1;                 open := open+1;
                 yield;                          yield;

                void close_dev()                void close_dev()
open = 0         if (open>0)                     if (open>0)
                   open := open-1;                 open := open-1;
                   if (open==0)                    if (open==0)
                     power_down();                   power_down();
                 yield;                          yield;
```

$[\![\text{PRG}]\!]^{\texttt{nonpreempt}}$:

```
                                  open := 0;
```

power_up()

open = 1

```
void open_dev()                    void open_dev()
 if (open==0)                       if (open==0)
  power_up();                        power_up();
→ open := open+1;                   open := open+1;
 yield;                             yield;

void close_dev()                   void close_dev()
 if (open>0)                        if (open>0)
  open := open-1;                    open := open-1;
  if (open==0)                       if (open==0)
   power_down();                      power_down();
 yield;                             yield;
```

$[\![\text{PRG}]\!]^{\texttt{nonpreempt}}$:

```
                                  open := 0;

power_up()      void open_dev()          void open_dev()
                 if (open==0)      →      if (open==0)
                   power_up();             power_up();
                 open := open+1;          open := open+1;
              →  yield;                   yield;

                void close_dev()         void close_dev()
open = 1         if (open>0)              if (open>0)
                   open := open-1;          open := open-1;
                   if (open==0)             if (open==0)
                     power_down();            power_down();
                 yield;                   yield;
```

$[\![\text{PRG}]\!]^{\texttt{nonpreempt}}$:

```
                                  open := 0;

power_up()      void open_dev()                 void open_dev()
                 if (open==0)          →         if (open==0)
                   power_up();                     power_up();
                 open := open+1;                 open := open+1;
                 yield;                          yield;

                void close_dev()                void close_dev()
open = 1    →    if (open>0)                     if (open>0)
                   open := open-1;                 open := open-1;
                   if (open==0)                    if (open==0)
                     power_down();                   power_down();
                 yield;                          yield;
```

$[\![\text{PRG}]\!]^{\texttt{nonpreempt}}$:

```
                                open := 0;
```

power_up()

```
void open_dev()
 if (open==0)
   power_up();
 open := open+1;
 yield;
```

→

```
void open_dev()
 if (open==0)
   power_up();
 open := open+1;
 yield;
```

open = 2

```
void close_dev()
 if (open>0)
   open := open-1;
   if (open==0)
     power_down();
 yield;
```

→

```
void close_dev()
 if (open>0)
   open := open-1;
   if (open==0)
     power_down();
 yield;
```

$[\![\text{PRG}]\!]^{\texttt{nonpreempt}}$:

```
                                    open := 0;
```

power_up()

```
void open_dev()
 if (open==0)
   power_up();
 open := open+1;
 yield;
```

→

```
void open_dev()
 if (open==0)
   power_up();
 open := open+1;
 yield;
```

open = 2

→

```
void close_dev()
 if (open>0)
   open := open-1;
   if (open==0)
     power_down();
 yield;
```

```
void close_dev()
 if (open>0)
   open := open-1;
   if (open==0)
     power_down();
 yield;
```

$[\![\text{PRG}]\!]^{\texttt{nonpreempt}}$:

```
                                  open := 0;
```

power_up()

```
void open_dev()
 if (open==0)
   power_up();
 open := open+1;
 yield;

void close_dev()
 if (open>0)
   open := open-1;
   if (open==0)
     power_down();
 yield;
```

→

```
void open_dev()
 if (open==0)
   power_up();
 open := open+1;
 yield;

void close_dev()
 if (open>0)
   open := open-1;
   if (open==0)
     power_down();
 yield;
```

open = 2

$[\![\text{PRG}]\!]^{\texttt{nonpreempt}}$:

```
                                open := 0;
```

power_up()

```
void open_dev()
 if (open==0)
   power_up();
 open := open+1;
 yield;
```

```
void open_dev()
 if (open==0)
   power_up();
 open := open+1;
 yield;
```

open = 1

→

```
void close_dev()
 if (open>0)
   open := open-1;
   if (open==0)
     power_down();
 yield;
```

→

```
void close_dev()
 if (open>0)
   open := open-1;
   if (open==0)
     power_down();
 yield;
```

$[\![\text{PRG}]\!]^{\texttt{nonpreempt}}$:

```
                                  open := 0;
```

power_up()

```
void open_dev()
 if (open==0)
   power_up();
 open := open+1;
 yield;
```

```
void close_dev()
 if (open>0)
   open := open-1;
   if (open==0)
     power_down();
 yield;
```

open = 1

→

```
void open_dev()
 if (open==0)
   power_up();
 open := open+1;
 yield;
```

→

```
void close_dev()
 if (open>0)
   open := open-1;
   if (open==0)
     power_down();
 yield;
```

$[\![\text{PRG}]\!]^{\texttt{nonpreempt}}$:

```
                                  open := 0;

power_up()      void open_dev()          void open_dev()
                 if (open==0)             if (open==0)
                   power_up();              power_up();
                 open := open+1;          open := open+1;
                 yield;                   yield;

                void close_dev()         void close_dev()
                 if (open>0)              if (open>0)
open = 1           open := open-1;          open := open-1;
                   if (open==0)             if (open==0)
                     power_down();            power_down();
              →    yield;                   yield;
```

$[\![\text{PRG}]\!]^{\texttt{nonpreempt}}$:

```
                                    open := 0;

power_up()  →  void open_dev()          void open_dev()
                 if (open==0)             if (open==0)
                   power_up();              power_up();
               open := open+1;          open := open+1;
               yield;                   yield;

               void close_dev()         void close_dev()
open = 1        if (open>0)        →     if (open>0)
                  open := open-1;          open := open-1;
                  if (open==0)             if (open==0)
                    power_down();            power_down();
               yield;                   yield;
```

$[\![\text{PRG}]\!]^{\texttt{nonpreempt}}$:

```
                                open := 0;
```

power_up() →

```
void open_dev()
 if (open==0)
   power_up();
 open := open+1;
 yield;
```

```
void open_dev()
 if (open==0)
   power_up();
 open := open+1;
 yield;
```

open = 0

```
void close_dev()
 if (open>0)
   open := open-1;
   if (open==0)
     power_down();
 yield;
```

→

```
void close_dev()
 if (open>0)
   open := open-1;
   if (open==0)
     power_down();
 yield;
```

$[\![\text{PRG}]\!]^{\texttt{nonpreempt}}$:

```
                                    open := 0;
```

power_up() →

```
void open_dev()
 if (open==0)
   power_up();
 open := open+1;
 yield;
```

```
void open_dev()
 if (open==0)
   power_up();
 open := open+1;
 yield;
```

open = 0

```
void close_dev()
 if (open>0)
   open := open-1;
   if (open==0)
     power_down();
 yield;
```

→

```
void close_dev()
 if (open>0)
   open := open-1;
   if (open==0)
     power_down();
 yield;
```

$[\![\text{PRG}]\!]^{\texttt{nonpreempt}}$:

```
                                  open := 0;
```

power_up()
power_down() →

```
void open_dev()
 if (open==0)
   power_up();
 open := open+1;
 yield;
```

```
void open_dev()
 if (open==0)
   power_up();
 open := open+1;
 yield;
```

open = 0

```
void close_dev()
 if (open>0)
   open := open-1;
   if (open==0)
     power_down();
 yield;
```

→

```
void close_dev()
 if (open>0)
   open := open-1;
   if (open==0)
     power_down();
 yield;
```

$[\![\text{PRG}]\!]^{\texttt{nonpreempt}}$:

```
                                    open := 0;
```

power_up()
power_down()

```
void open_dev()
 if (open==0)
   power_up();
 open := open+1;
 yield;
```

```
void open_dev()
 if (open==0)
   power_up();
 open := open+1;
 yield;
```

```
void close_dev()
 if (open>0)
   open := open-1;
   if (open==0)
     power_down();
 yield;
```

open = 0

→

```
void close_dev()
 if (open>0)
   open := open-1;
   if (open==0)
     power_down();
 yield;
```

$[\![\text{PRG}]\!]^{\texttt{nonpreempt}}$:

```
open := 0;
```

power_up()
power_down()
power_up()

```
void open_dev()
 if (open==0)
   power_up();
 open := open+1;
 yield;

void close_dev()
 if (open>0)
   open := open-1;
   if (open==0)
     power_down();
 yield;
```

```
void open_dev()
 if (open==0)
   power_up();
 open := open+1;
 yield;

void close_dev()
 if (open>0)
   open := open-1;
   if (open==0)
     power_down();
 yield;
```

$[\![\text{PRG}]\!]^{\texttt{nonpreempt}}$:

```
                    open := 0;
```

```
power_up()
power_down()
power_up()
power_down()
```

```
void open_dev()
 if (open==0)
   power_up();
 open := open+1;
 yield;

void close_dev()
 if (open>0)
   open := open-1;
   if (open==0)
     power_down();
 yield;
```

```
void open_dev()
 if (open==0)
   power_up();
 open := open+1;
 yield;

void close_dev()
 if (open>0)
   open := open-1;
   if (open==0)
     power_down();
 yield;
```

$[\![\text{PRG}]\!]^{\texttt{nonpreempt}}$:

```
                    open := 0;
```

```
power_up()
power_down()
power_up()
power_down()
    :
```

```
void open_dev()
 if (open==0)
   power_up();
 open := open+1;
 yield;

void close_dev()
 if (open>0)
   open := open-1;
   if (open==0)
     power_down();
 yield;
```

```
void open_dev()
 if (open==0)
   power_up();
 open := open+1;
 yield;

void close_dev()
 if (open>0)
   open := open-1;
   if (open==0)
     power_down();
 yield;
```

open := 0;

open = 0

```
void open_dev()
 if (open==0)
   power_up();
 open := open+1;
 yield;

void close_dev()
 if (open>0)
   open := open-1;
   if (open==0)
     power_down();
 yield;
```

→

```
void open_dev()
 if (open==0)
   power_up();
 open := open+1;
 yield;

void close_dev()
 if (open>0)
   open := open-1;
   if (open==0)
     power_down();
 yield;
```

$[\![\text{PRG}]\!]^{\texttt{preempt}}$:

```
                    open := 0;

      void open_dev()          void open_dev()
  →    if (open==0)        →    if (open==0)
        power_up();              power_up();
       open := open+1;          open := open+1;
       yield;                   yield;

      void close_dev()         void close_dev()
       if (open>0)              if (open>0)
open = 0 open := open-1;         open := open-1;
        if (open==0)             if (open==0)
          power_down();            power_down();
       yield;                   yield;
```

$[\![\text{PRG}]\!]^{\texttt{preempt}}$:

```
open := 0;
```

```
void open_dev()
 if (open==0)
→  power_up();
 open := open+1;
 yield;

void close_dev()
 if (open>0)
  open := open-1;
  if (open==0)
   power_down();
 yield;
```

```
void open_dev()
→ if (open==0)
  power_up();
 open := open+1;
 yield;

void close_dev()
 if (open>0)
  open := open-1;
  if (open==0)
   power_down();
 yield;
```

`open = 0`

$[\![\text{PRG}]\!]^{\text{preempt}}$:

```
                         open := 0;

power_up()       void open_dev()          void open_dev()
                  if (open==0)             if (open==0)
             →     power_up();        →     power_up();
                  open := open+1;          open := open+1;
                  yield;                   yield;

                 void close_dev()         void close_dev()
                  if (open>0)              if (open>0)
open = 0           open := open-1;          open := open-1;
                   if (open==0)             if (open==0)
                     power_down();            power_down();
                  yield;                   yield;
```

$[\![\text{PRG}]\!]^{\texttt{preempt}}$:

```
                                  open := 0;
```

power_up()

```
void open_dev()
 if (open==0)
   power_up();
 open := open+1;
 yield;

void close_dev()
 if (open>0)
   open := open-1;
   if (open==0)
     power_down();
 yield;
```

→

```
void open_dev()
 if (open==0)
   power_up();
 open := open+1;
 yield;

void close_dev()
 if (open>0)
   open := open-1;
   if (open==0)
     power_down();
 yield;
```

open = 1

$[\![\text{PRG}]\!]^{\texttt{preempt}}$:

```
                                  open := 0;

power_up()      void open_dev()          void open_dev()
                 if (open==0)             if (open==0)
                   power_up();      →       power_up();
                 open := open+1;          open := open+1;
              →  yield;                   yield;

                void close_dev()         void close_dev()
                 if (open>0)              if (open>0)
 open = 1          open := open-1;          open := open-1;
                   if (open==0)             if (open==0)
                     power_down();            power_down();
                 yield;                   yield;
```

$[\![\text{PRG}]\!]^{\texttt{preempt}}$:

```
                                  open := 0;

power_up()      void open_dev()          void open_dev()
power_up()       if (open==0)             if (open==0)
                   power_up();      →       power_up();
                 open := open+1;          open := open+1;
                 yield;                   yield;

                void close_dev()         void close_dev()
             →   if (open>0)              if (open>0)
 open = 1          open := open-1;          open := open-1;
                   if (open==0)             if (open==0)
                     power_down();            power_down();
                 yield;                   yield;
```

$[\![\text{PRG}]\!]^{\texttt{preempt}}$:

```
                                  open := 0;
```

power_up()
power_up()

```
void open_dev()
 if (open==0)
   power_up();
 open := open+1;
 yield;
```

→

```
void open_dev()
 if (open==0)
   power_up();
 open := open+1;
 yield;
```

open = 2

```
void close_dev()
 if (open>0)
   open := open-1;
   if (open==0)
     power_down();
 yield;
```

→

```
void close_dev()
 if (open>0)
   open := open-1;
   if (open==0)
     power_down();
 yield;
```

$[\![\text{PRG}]\!]^{\texttt{preempt}}$:

```
                                  open := 0;

power_up()      void open_dev()          void open_dev()
power_up()       if (open==0)             if (open==0)
                   power_up();              power_up();
                 open := open+1;          open := open+1;
                 yield;              →    yield;

                void close_dev()         void close_dev()
            →    if (open>0)              if (open>0)
 open = 2         open := open-1;          open := open-1;
                  if (open==0)             if (open==0)
                    power_down();            power_down();
                 yield;                   yield;
```

$[\![\text{PRG}]\!]^{\texttt{preempt}}$:

```
open := 0;
```

```
power_up()
power_up()
   ⋮
```

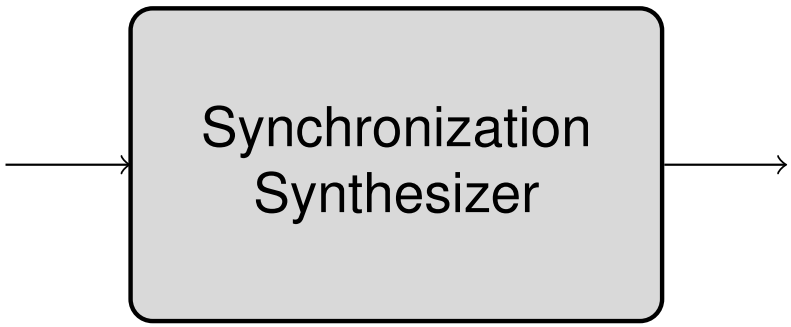
```
void open_dev()
 if (open==0)
  power_up();
 open := open+1;
 yield;

void close_dev()
→ if (open>0)
   open := open-1;
   if (open==0)
     power_down();
 yield;
```

```
void open_dev()
 if (open==0)
  power_up();
 open := open+1;
 yield;

void close_dev()
→ if (open>0)
   open := open-1;
   if (open==0)
     power_down();
 yield;
```

$[\![\text{PRG}]\!]^{\texttt{preempt}}$:

```
                                  open := 0;

power_up()          void open_dev()          void open_dev()
power_up()           if (open==0)             if (open==0)
    ⋮                  power_up();              power_up();
                     open := open+1;          open := open+1;
    ∉                yield;                   yield;

[[PRG]]^nonpreempt   void close_dev()         void close_dev()
              →       if (open>0)        →     if (open>0)
                        open := open-1;          open := open-1;
                        if (open==0)             if (open==0)
                          power_down();            power_down();
                      yield;                   yield;
```

```
void open_dev()
 if (open==0)
   power_up();
 open := open+1;
 yield;

void close_dev()
 if (open>0)
   open := open-1;
   if (open==0)
     power_down();
 yield;
```




```
void open_dev()
lock
 if (open==0)
   power_up();
 open := open+1;
unlock
 yield;

void close_dev()
lock
 if (open>0)
   open := open-1;
   if (open==0)
     power_down();
unlock
 yield;
```

$$[\![\text{PRG}']\!]^{\text{preempt}} \subseteq [\![\text{PRG}]\!]^{\text{nonpreempt}}$$

Abstraction

Data-oblivious abstraction

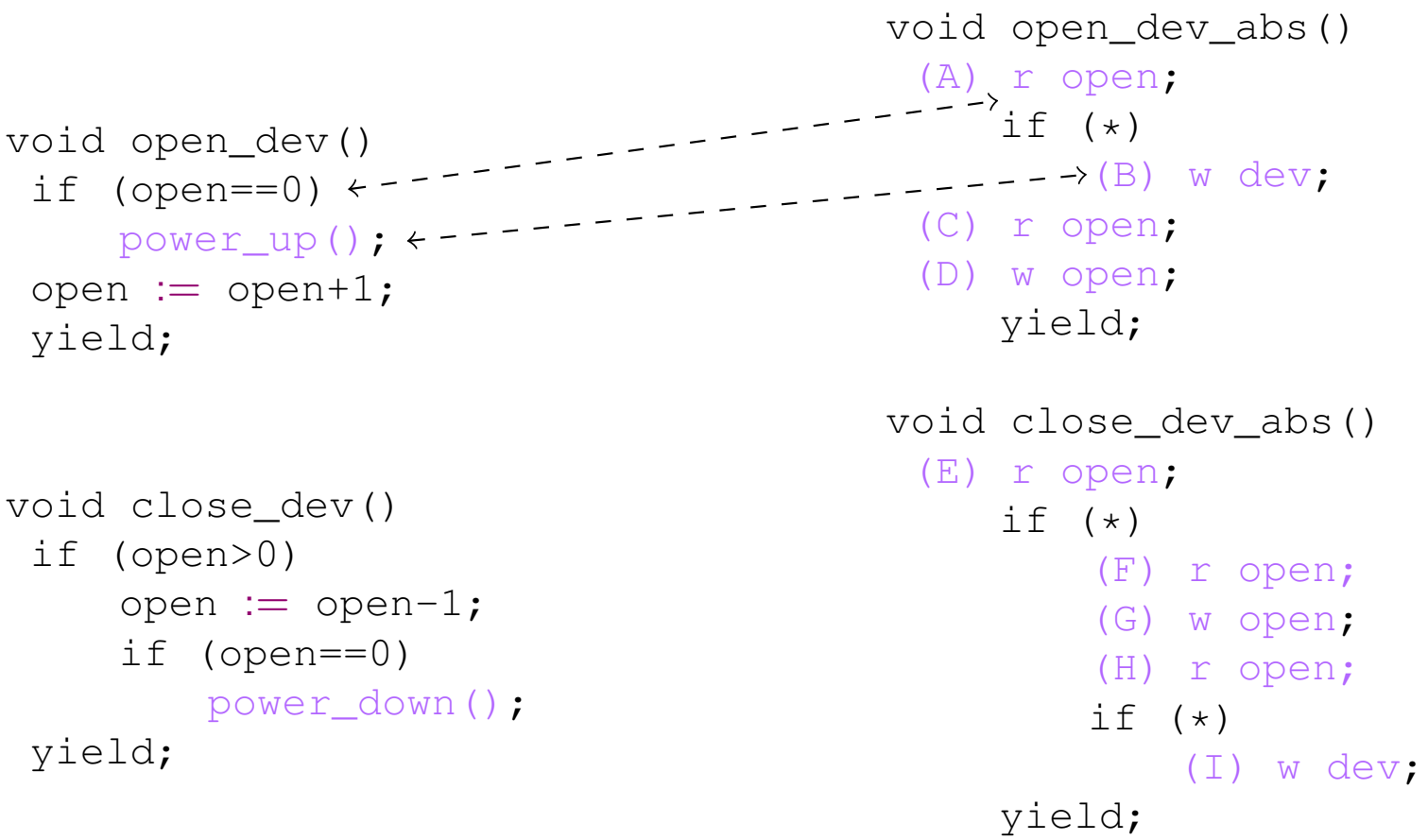
```
void open_dev()
 if (open==0)
     power_up();
 open := open+1;
 yield;
```

```
void close_dev()
 if (open>0)
     open := open-1;
     if (open==0)
         power_down();
 yield;
```

```
void open_dev_abs()
 (A) r open;
     if (*)
         (B) w dev;
 (C) r open;
 (D) w open;
     yield;
```

```
void close_dev_abs()
 (E) r open;
     if (*)
         (F) r open;
         (G) w open;
         (H) r open;
         if (*)
             (I) w dev;
     yield;
```

Data-oblivious abstraction



The Concrete Case:

Observations:

Calls to interface

The Abstract Case:

Observations:

Read/write memory locations

The Concrete Case:

Observations:
Calls to interface

Trace:
Sequence of observations

The Abstract Case:

Observations:
Read/write memory locations

Trace:
Sequence of observations

The Concrete Case:

Observations:
Calls to interface

Trace:
Sequence of observations

Trace equivalence:
Equality

The Abstract Case:

Observations:
Read/write memory locations

Trace:
Sequence of observations

Trace equivalence:
Equality modulo $\mathcal{I}$

The Concrete Case:

Observations:
Calls to interface

Trace:
Sequence of observations

Trace equivalence:
Equality

Correctness:
Preemption-safety

The Abstract Case:

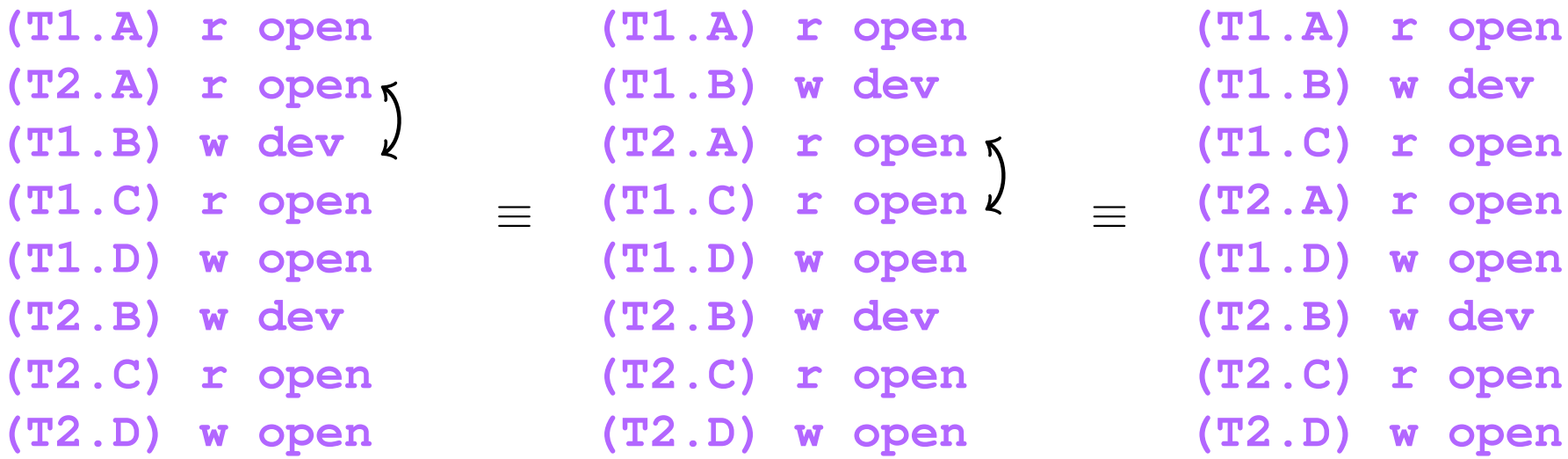
Observations:
Read/write memory locations

Trace:
Sequence of observations

Trace equivalence:
Equality modulo $\mathcal{I}$

Correctness:
Preemption-safety

Equality modulo $\mathcal{I}$



Classical independence relation $\mathcal{I}$:

- Accesses to different locations
- Read accesses to the same location

Soundness of Abstraction

Preemption-safety under abstraction $\Rightarrow$ Preemption safety

(T1.A) r open
(T2.A) r open
(T1.B) w dev
(T1.C) r open
(T1.D) w open
(T2.B) w dev
(T2.C) r open
(T2.D) w open
⋮

$\notin$

$[\![\text{PRG}]\!]^{\text{nonpreempt}}_{\text{ABS}}$

```
void open_dev_abs()
(A) r open;
    if (*)
        (B) w dev;
(C) r open;
(D) w open;
    yield;

void close_dev_abs()
(E) r open;
    if (*)
        (F) r open;
        (G) w open;
        (H) r open;
        if (*)
            (I) w dev;
    yield;
```

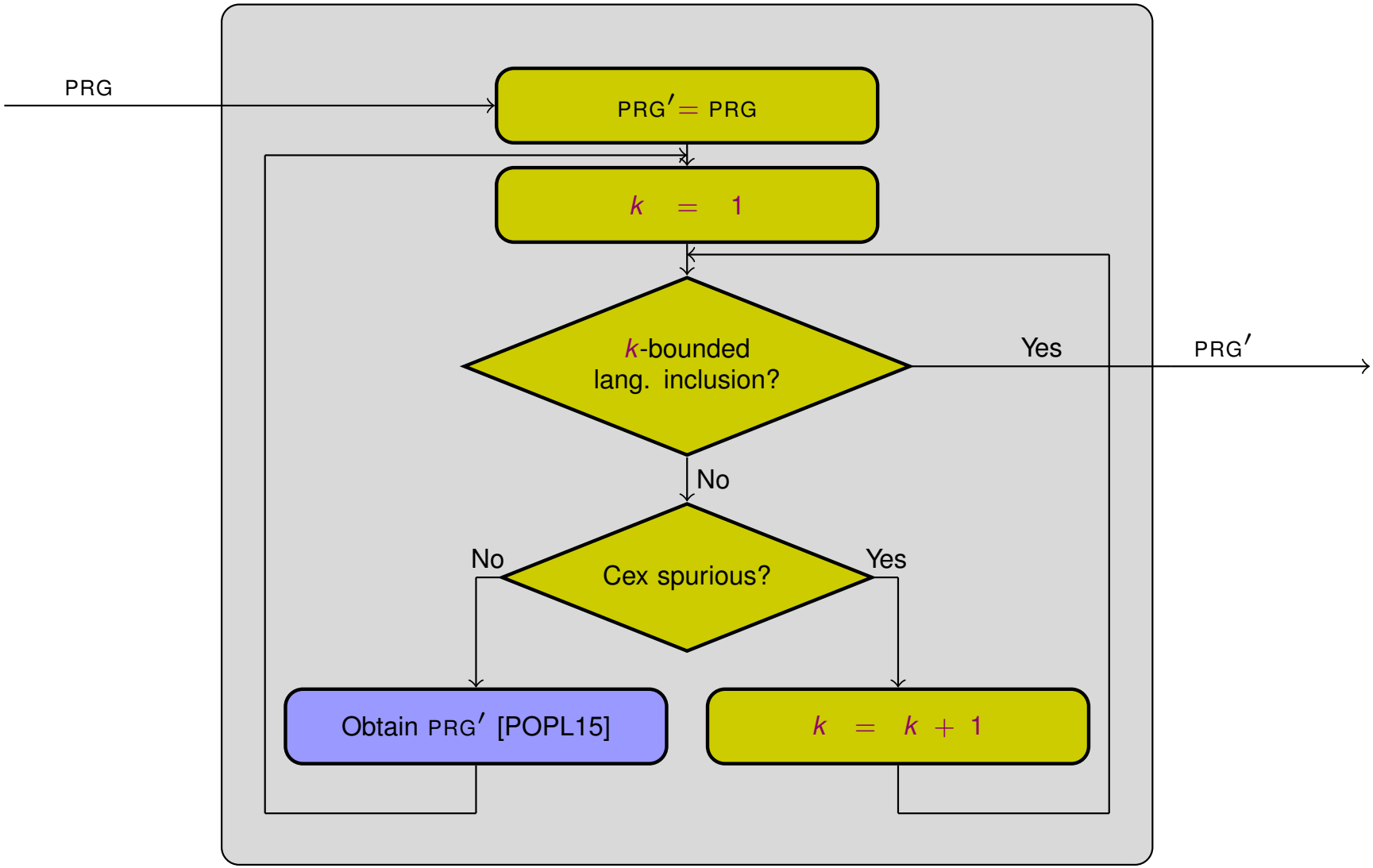
```
void open_dev_abs()
(A) r open;
    if (*)
        (B) w dev;
(C) r open;
(D) w open;
    yield;

void close_dev_abs()
(E) r open;
    if (*)
        (F) r open;
        (G) w open;
        (H) r open;
        if (*)
            (I) w dev;
    yield;
```

Brief Note on The Rest

- Abstract semantics to automata

- Abstract semantics to automata
- Reduce preemption-safety to language inclusion modulo $\mathcal{I}$
  - Known to be undecidable [Bertoni et al. 1982]

- Abstract semantics to automata
- Reduce preemption-safety to language inclusion modulo $\mathcal{I}$
  - Known to be undecidable [Bertoni et al. 1982]
- We define k-bounded language inclusion modulo $\mathcal{I}$
  - And develop decision procedure


PRG
PRG′ = PRG
k = 1
k-bounded lang. inclusion?
Yes
PRG′
No
No
Cex spurious?
Yes
Obtain PRG′ [POPL15]
k = k + 1

Evaluation

LISS

- Evaluated on device driver benchmarks
- Implicit vs explicit specification

- Efficacy of abstraction

LISS

- Evaluated on device driver benchmarks
- Implicit vs explicit specification
  - LISS eliminated all races (except 2)
  - LISS detected and eliminated a race missed by assertion-based approach
- Efficacy of abstraction

LISS

- Evaluated on device driver benchmarks
- Implicit vs explicit specification
  - LISS eliminated all races (except 2)
  - LISS detected and eliminated a race missed by assertion-based approach
- Efficacy of abstraction
  - Highly efficient
  - Very precise

Experiments

Name	#Loc	#Th	#Iter	Max k	Bug(s)	Syn(s)	Ver(s)
ex1.c	18	2	1	1	<1	<1	<1
ex2.c	23	2	1	1	<1	<1	<1
ex3.c	37	2	1	1	<1	<1	<1
ex5.c	42	2	3	1	<1	<1	2
lc-rc.c	35	4	0	1	-	-	<1
dv1394.c	37	2	1	1	<1	<1	<1
em28xx.c	20	2	1	1	<1	<1	<1
f_acm.c	80	3	1	1	<1	<1	<1
i915_irq.c	17	2	1	1	<1	<1	<1
ipath.c	23	2	1	1	<1	<1	<1
iwl3945.c	26	3	1	1	<1	<1	<1
md.c	35	2	1	1	<1	<1	<1
myri10ge.c	60	4	2	1	-	-	<1

Experiments

Name	#Loc	#Th	#Iter	Max k	Bug(s)	Syn(s)	Ver(s)
usb-serial.bug1.c	357	7	2	1	0.4	3.1	3.4
usb-serial.bug2.c	355	7	1	3	0.7	2.1	12.9
usb-serial.bug3.c	352	7	1	4	3.8	1.3	111.1
usb-serial.bug4.c	351	7	1	4	93.9	2.4	123.1
usb-serial.c	357	7	1	4	-	-	103.2
cpmac.bug1.c	1275	5	1	1	1.3	113.4	21.9
cpmac.bug2.c	1275	5	1	1	3.3	68.4	27.8
cpmac.bug3.c	1270	5	1	1	5.4	111.3	8.7
cpmac.bug4.c	1276	5	2	1	2.4	124.8	31.5
cpmac.bug5.c	1275	5	1	1	2.8	112.0	58.0
cpmac.c	1276	5	1	1	-	-	17.4

Conclusion

Contributions:

- Synchronization synthesis for preemption-safety
- LISS can handle realistic device driver code

Contributions:

- Synchronization synthesis for preemption-safety
- LISS can handle realistic device driver code
- Novel data-oblivious abstraction
- Reduce preemption safety to language inclusion modulo $\mathcal{I}$
- New algorithm for k-bounded language inclusion modulo $\mathcal{I}$

Contributions:

- Synchronization synthesis for preemption-safety
- LISS can handle realistic device driver code
- Novel data-oblivious abstraction
- Reduce preemption safety to language inclusion modulo $\mathcal{I}$
- New algorithm for k-bounded language inclusion modulo $\mathcal{I}$

Ongoing work: Quality of synthesized solution

- Deadlock-freedom
- Performance