Lazily Analysing the Result of Abstracting "Counting Processes"

Ahmed Rezine

joint work with: Zeinab Ganjei, Petru Eles, Zebo Peng

Linköping University, Sweden

Madrid 2015

Table of Contents

Goal

Verify safety of multi-threaded programs:

- involving an arbitrary number of threads (i.e. parameterized)
- synchronizing using "process counting variables", barriers, semaphores, etc.

Safety as in reachability of bad:

- individual configurations: assertion violation, mutual exclusion, etc
- global configurations : deadlock, number of threads at certain "states" is (e.g.,) smaller than the number of threads at some other "states", etc

Example

```
/* Only one main thread,
* no other threads    */

bool rd, op:=true;
int wt:=0,cn:=0;

main
{
  while(*)
  {
    atomic
    {
      spawn(thread);
      cn++;
    }
    op:=false;
  }
}
```

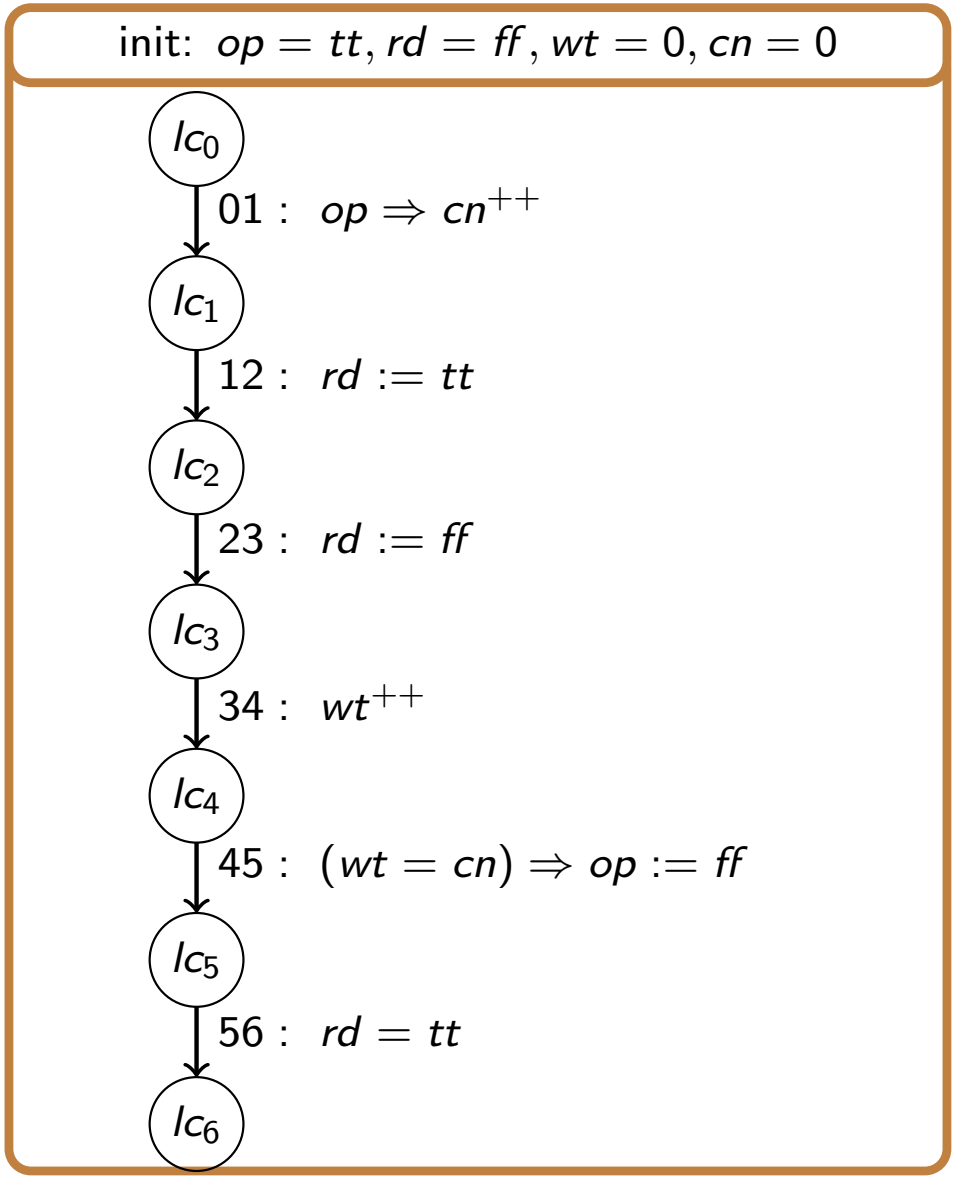
```
thread
{
  rd := true;

  //do read

  rd := false;
  wt++;
  assume(!op && (wt=cn));
  assert(!rd);
}
```

Example

- "Reformulation:" with identical threads.
- Initially, arbitrary many threads are at location lc_0
- Location lc_6 corresponds to the violation of the assertion



Goal and motivation

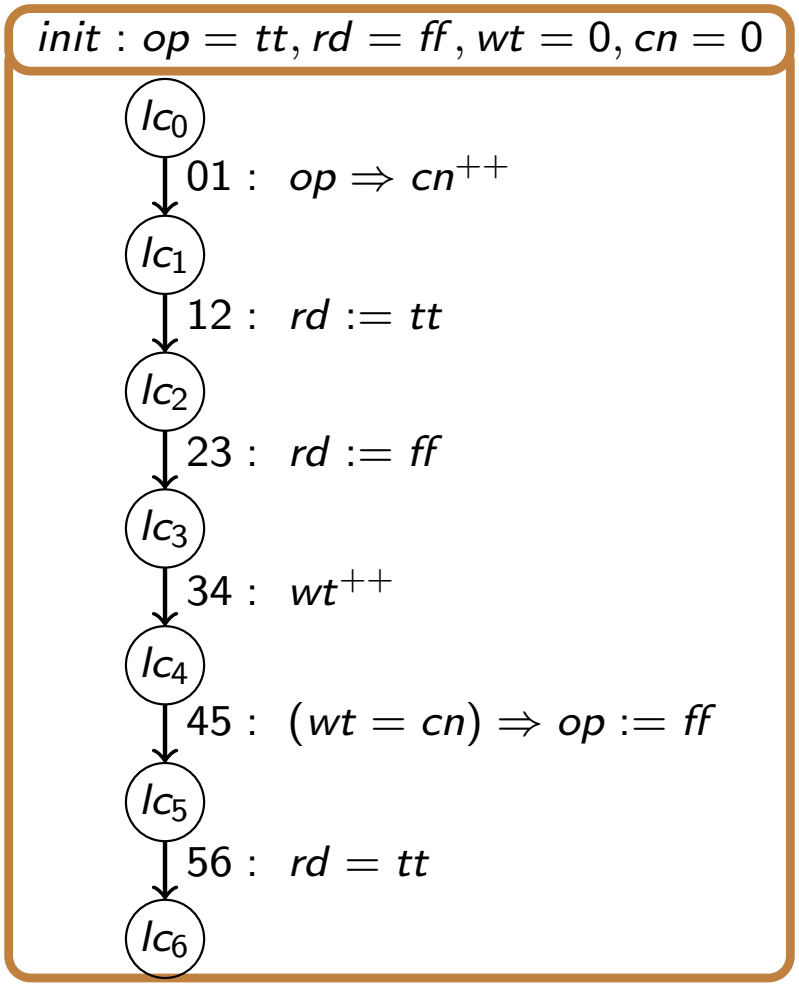
Predicate Abstraction

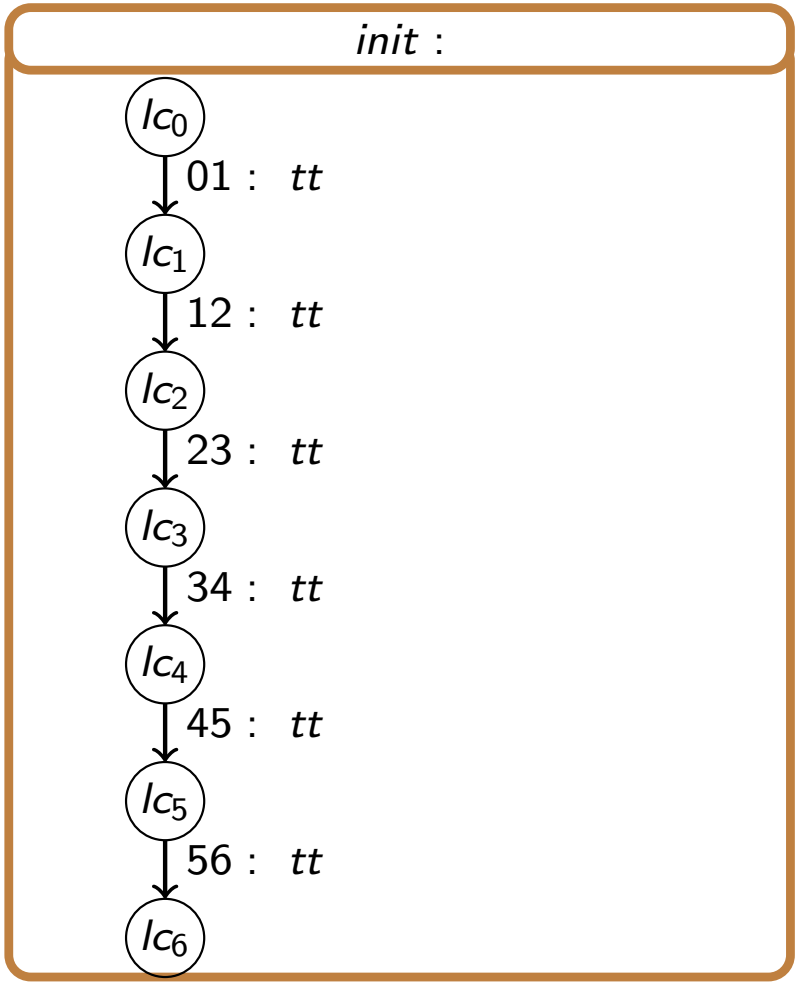
Strengthening

Monotonic abstraction

Table of Contents

Predicate abstraction



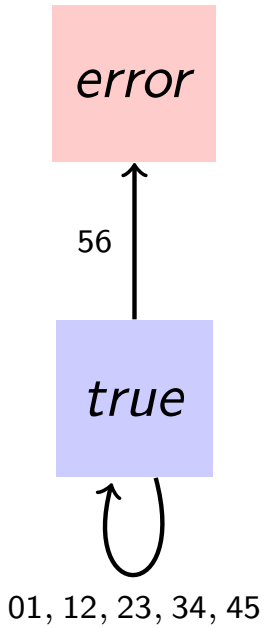


Counter Machine

Counter abstraction of the boolean program

- Control States
  - One state for each shared valuation of the Boolean program as well as error state
- Counters
  - One counter for each local valuation of the Boolean program
- Transitions
  - Translation of all possible transitions in the Boolean program
  - Local, global, transfer

Precisely captures the behavior of the boolean program


error
56
true
01, 12, 23, 34, 45

Counter Machine

Configuration

- State
- Value for each counter

Ordering on configurations

- $c_2 \geq c_1$ iff
  - Same state, larger counters in c_2

Closing upwards

- Intuitively amounts to adding more threads to a configuration

Counter Machine

A counter machine induces a well structured transition system, if

- Monotonic
  - The predecessor of every upward closed set is an upward closed set
- Well-quasi-ordering
  - Every increasing sequence of upward closed sets will eventually stabilize

Reachability Analysis

Check whether a control state is reachable from an initial configuration

Well structured systems:

- Classical backward reachability analysis

Reachability Analysis

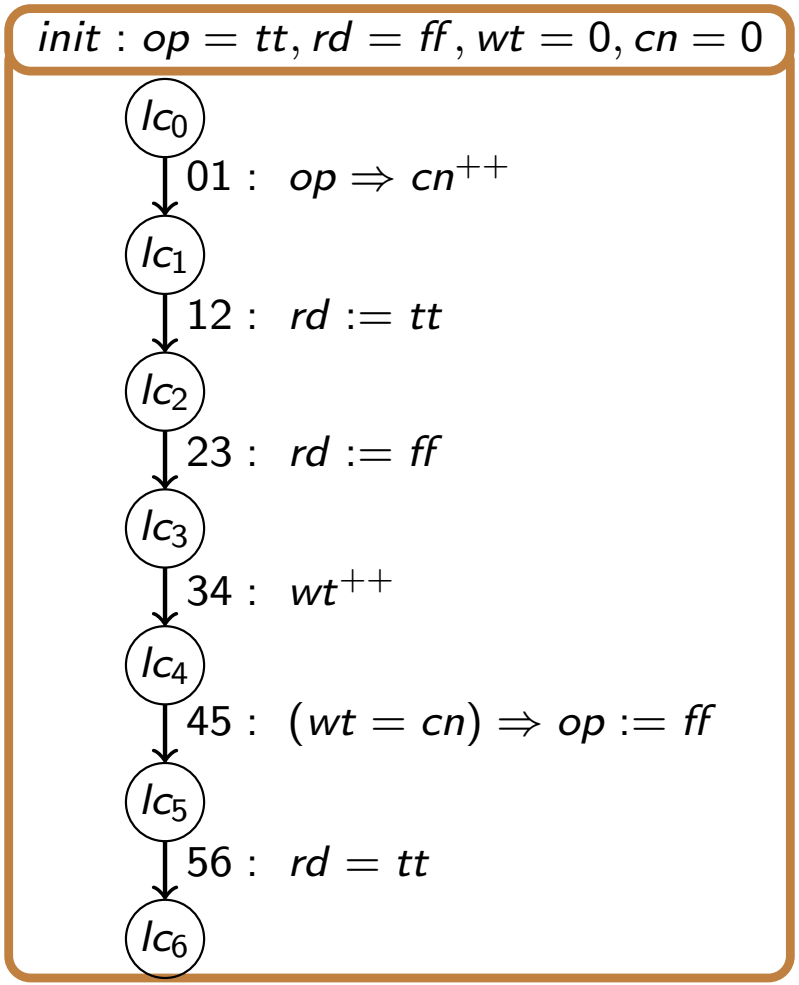
Check whether a control state is reachable from an initial configuration

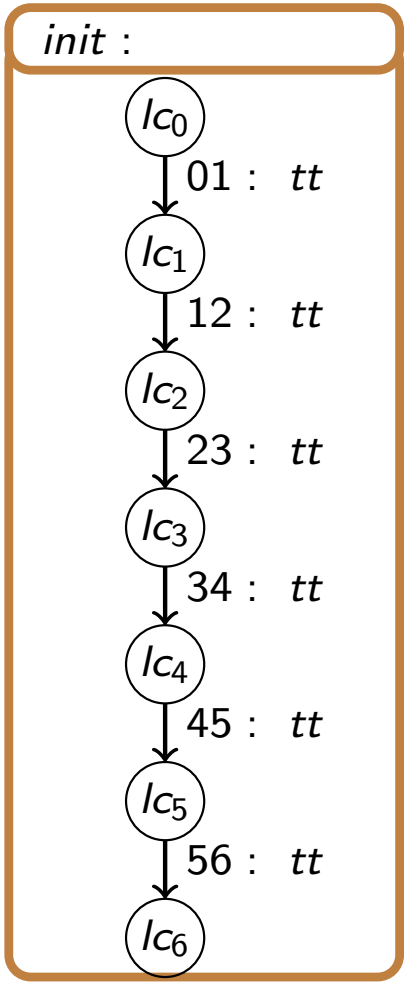
Well structured systems:

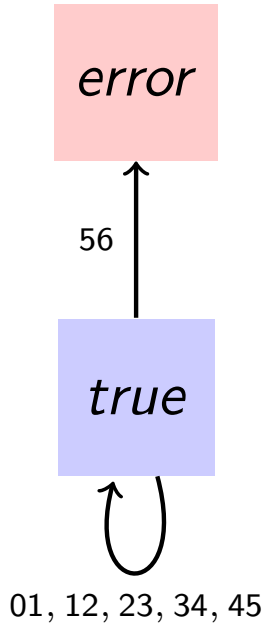
- Classical backward reachability analysis

  - Monotonicity: exactness
  - Well-quasi-ordering: termination

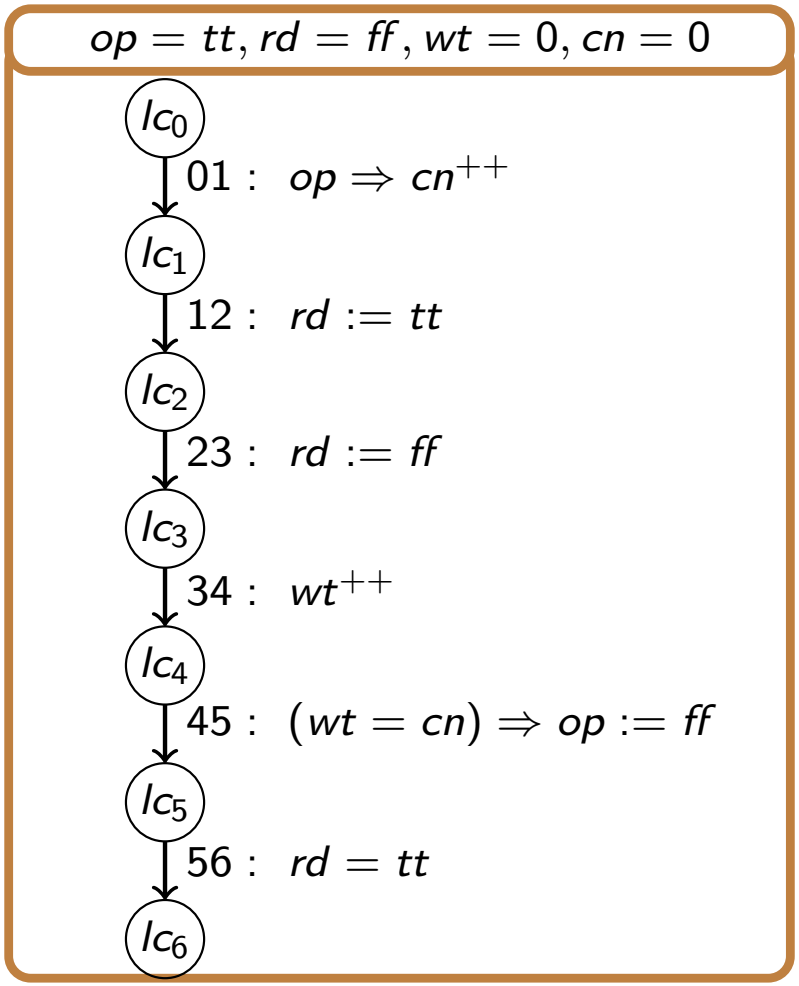
Predicate abstraction:

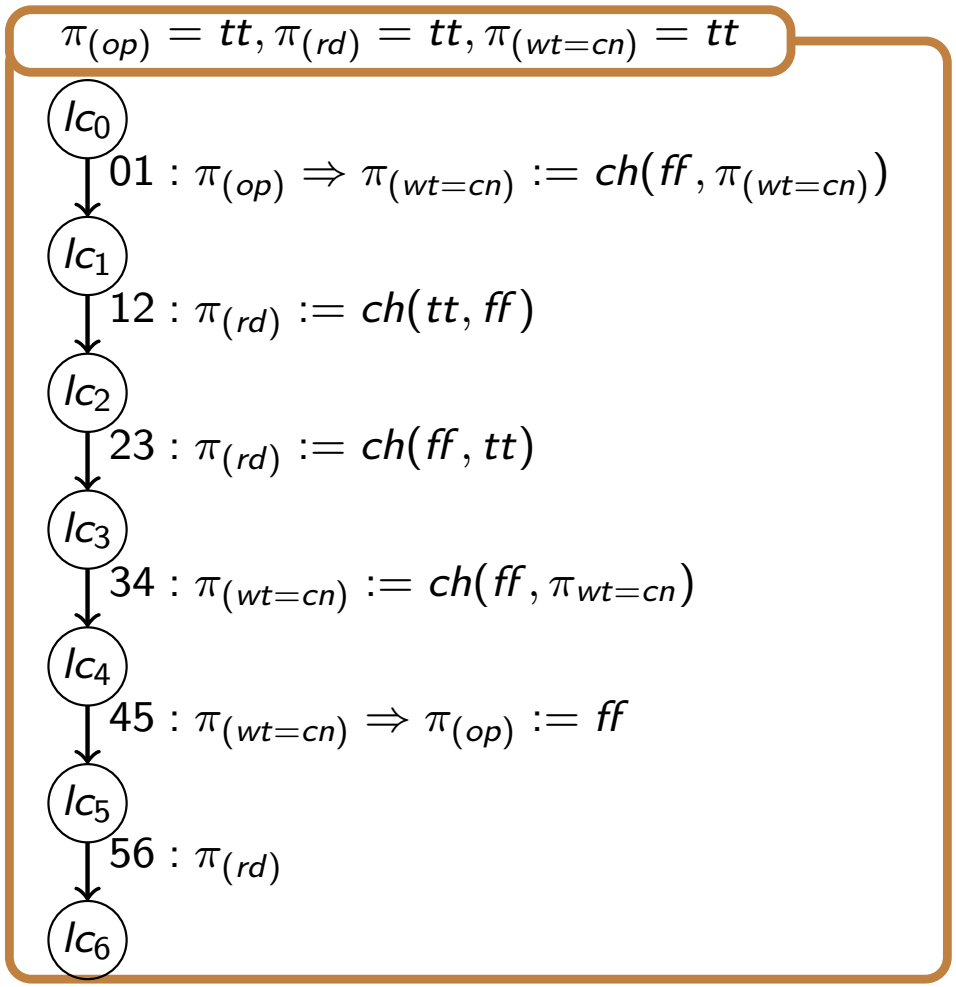


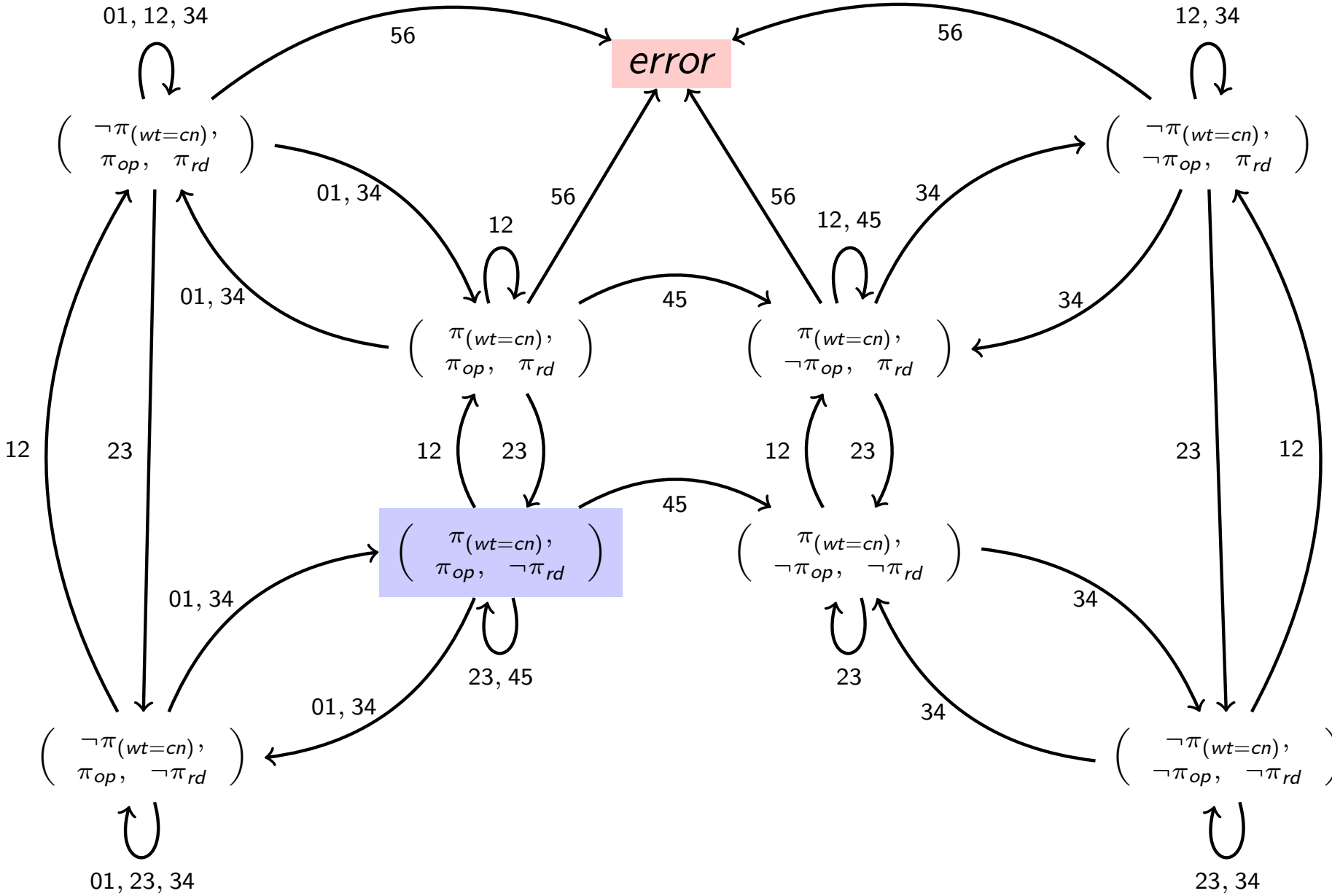


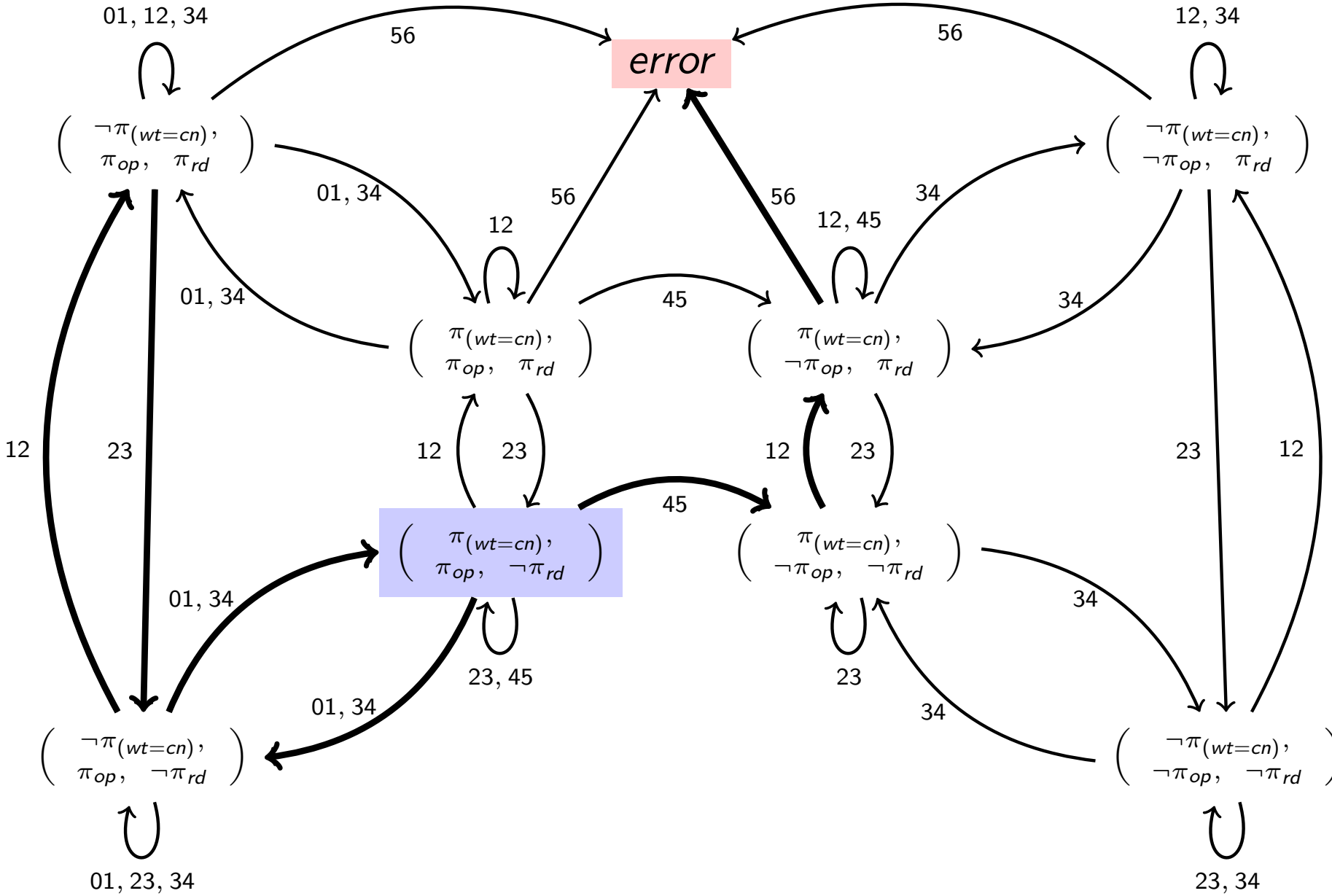


Predicate abstraction: (cont.)



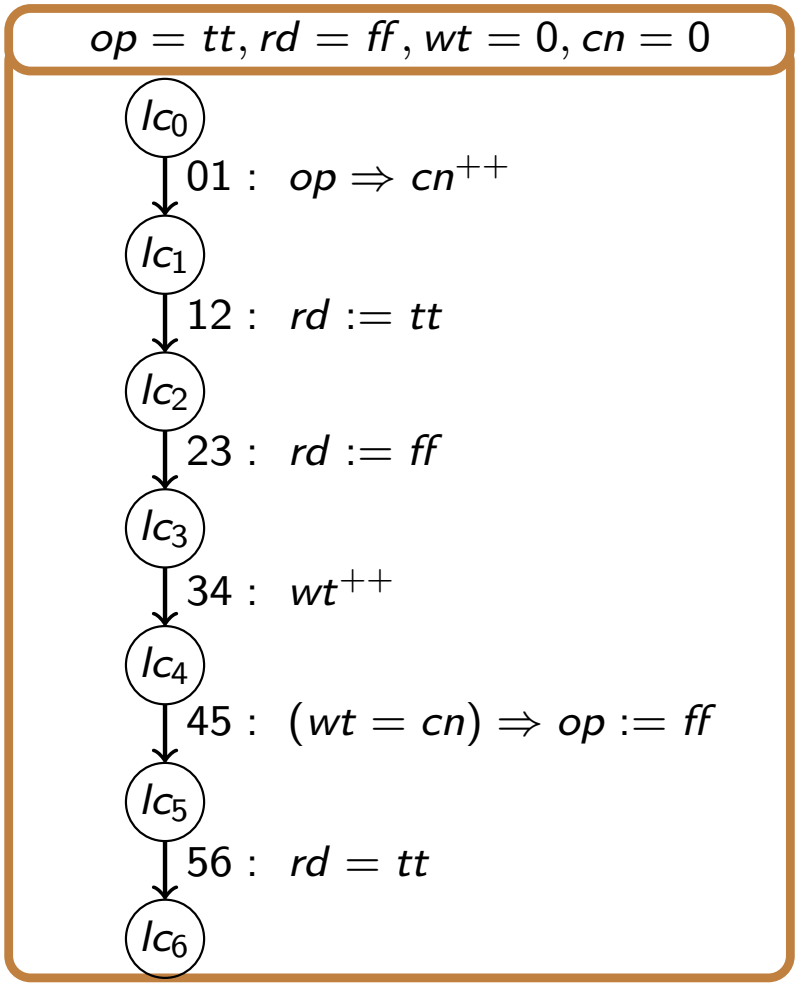


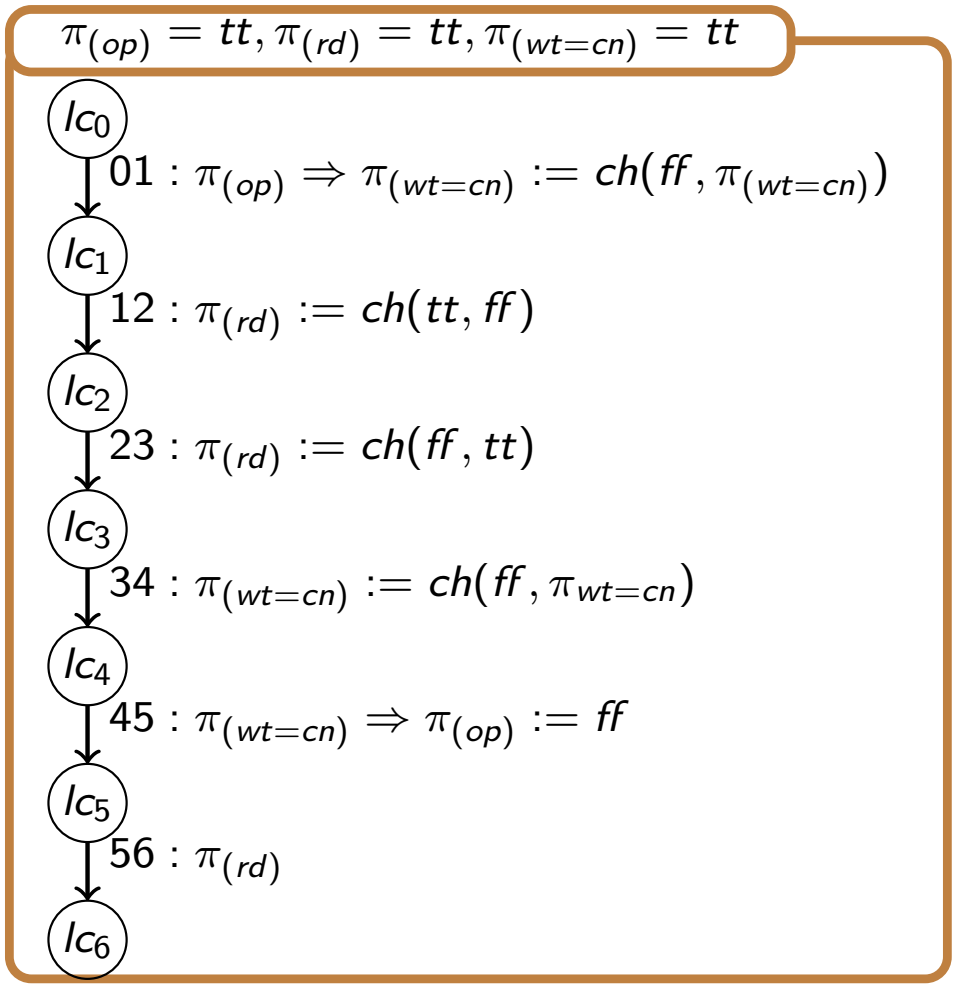

error
$\neg\pi_{(wt=cn)}$, π_{op}, π_{rd}
$\pi_{(wt=cn)}$, π_{op}, π_{rd}
$\pi_{(wt=cn)}$, $\neg\pi_{op}$, π_{rd}
$\pi_{(wt=cn)}$, π_{op}, $\neg\pi_{rd}$
$\pi_{(wt=cn)}$, $\neg\pi_{op}$, $\neg\pi_{rd}$
$\neg\pi_{(wt=cn)}$, π_{op}, $\neg\pi_{rd}$
$\neg\pi_{(wt=cn)}$, $\neg\pi_{op}$, π_{rd}
$\neg\pi_{(wt=cn)}$, $\neg\pi_{op}$, $\neg\pi_{rd}$
01, 12, 34
56
56
12, 34
01, 34
56
56
12
12, 45
34
01, 34
45
34
12
23
12
23
12
23
23
12
45
01, 34
34
23, 45
23
01, 34
34
01, 23, 34
23, 34


error
$\neg\pi_{(wt=cn)}$, π_{op}, π_{rd}
$\pi_{(wt=cn)}$, π_{op}, π_{rd}
$\pi_{(wt=cn)}$, $\neg\pi_{op}$, π_{rd}
$\pi_{(wt=cn)}$, π_{op}, $\neg\pi_{rd}$
$\pi_{(wt=cn)}$, $\neg\pi_{op}$, $\neg\pi_{rd}$
01, 12, 34
56
56
12, 34
01, 34
56
56
12, 45
34
12
01, 34
45
34
12
23
12
23
12
23
23
12
45
01, 34
34
23, 45
23
01, 34
34
01, 23, 34
23, 34

Predicate abstraction: (cont.)

The relation between the number of processes at locations lc_1, lc_2, lc_3 and the predicate $(wt = cn)$ is crucial (barrier).





Goal and motivation

Predicate Abstraction

Strengthening

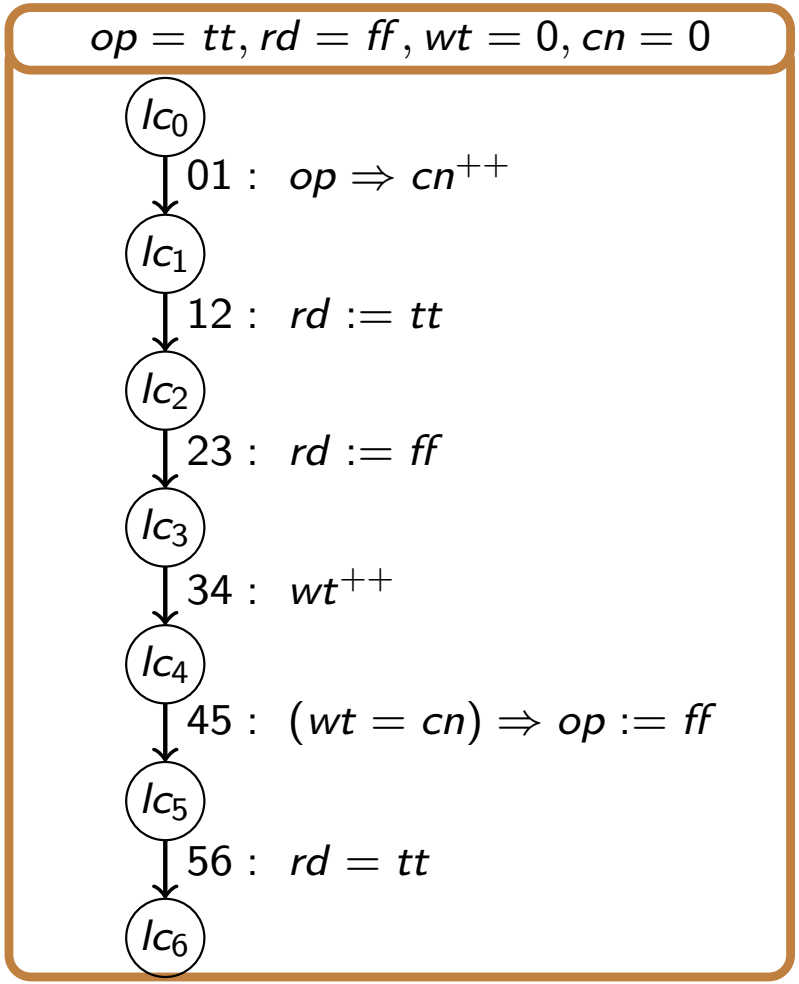
Monotonic abstraction

Table of Contents

Counting Predicates

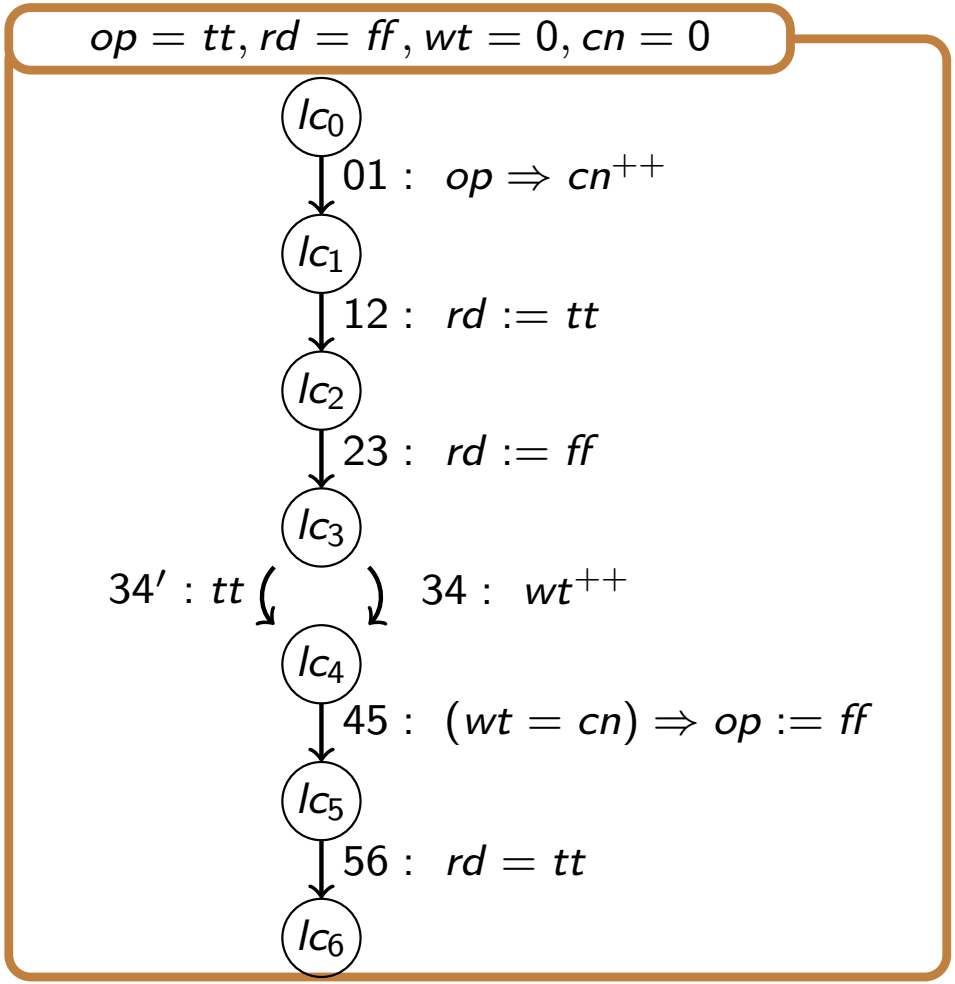
- Threads have states defined by their location and the predicates they satisfy on the program variables
- A counting predicate is a predicate that can involve:
  - shared program variables,
  - "ghost" counting variables $\mathcal{N}_{(\pi)}$ representing the number of threads satisfying some predicate π on the program variables
- We use them to define:
  - Properties to be checked, whether individual or global
  - Invariants that can be leveraged on during the analysis

Counting predicates for local properties:



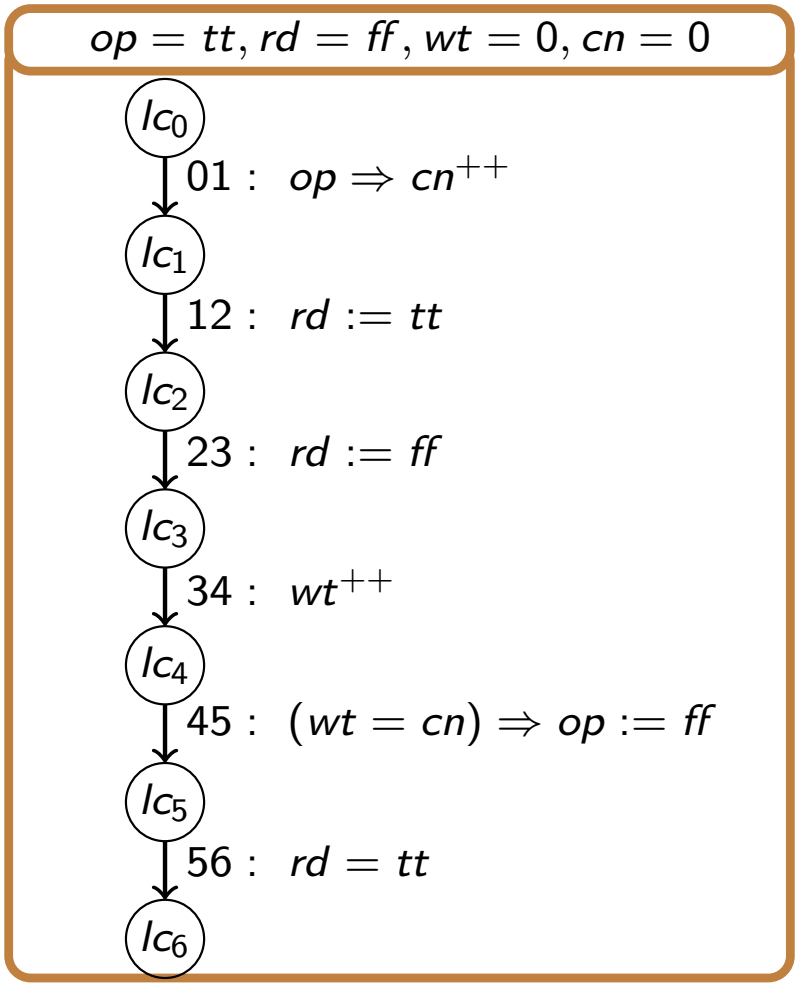
$\mathcal{N}_{(@lc_5 \wedge rd)} \geq 1$ reachable?

Counting predicates for global properties:



$\left(\Sigma_{i \neq 4} \mathcal{N}_{(@lc_i)} + \mathcal{N}_{(@lc_4 \wedge op \wedge wt = cn)} = 0\right)$ reachable?

Strengthening with Counting Predicates as Invariants



An appropriate thread modular analysis gives:

Invariants

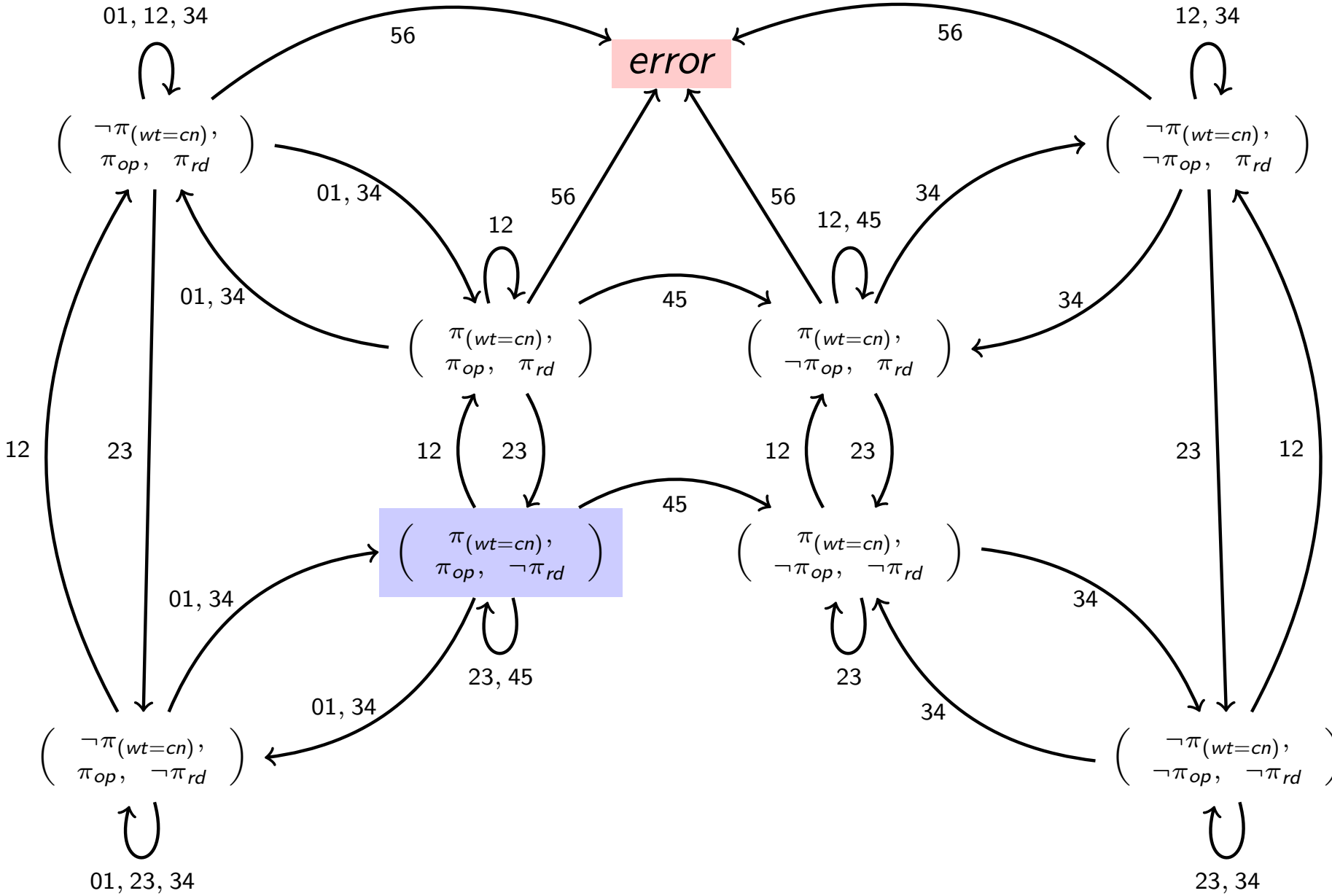
$cn = \sum_{i \geq 1}(\mathcal{N}_{(@lc_i)})$

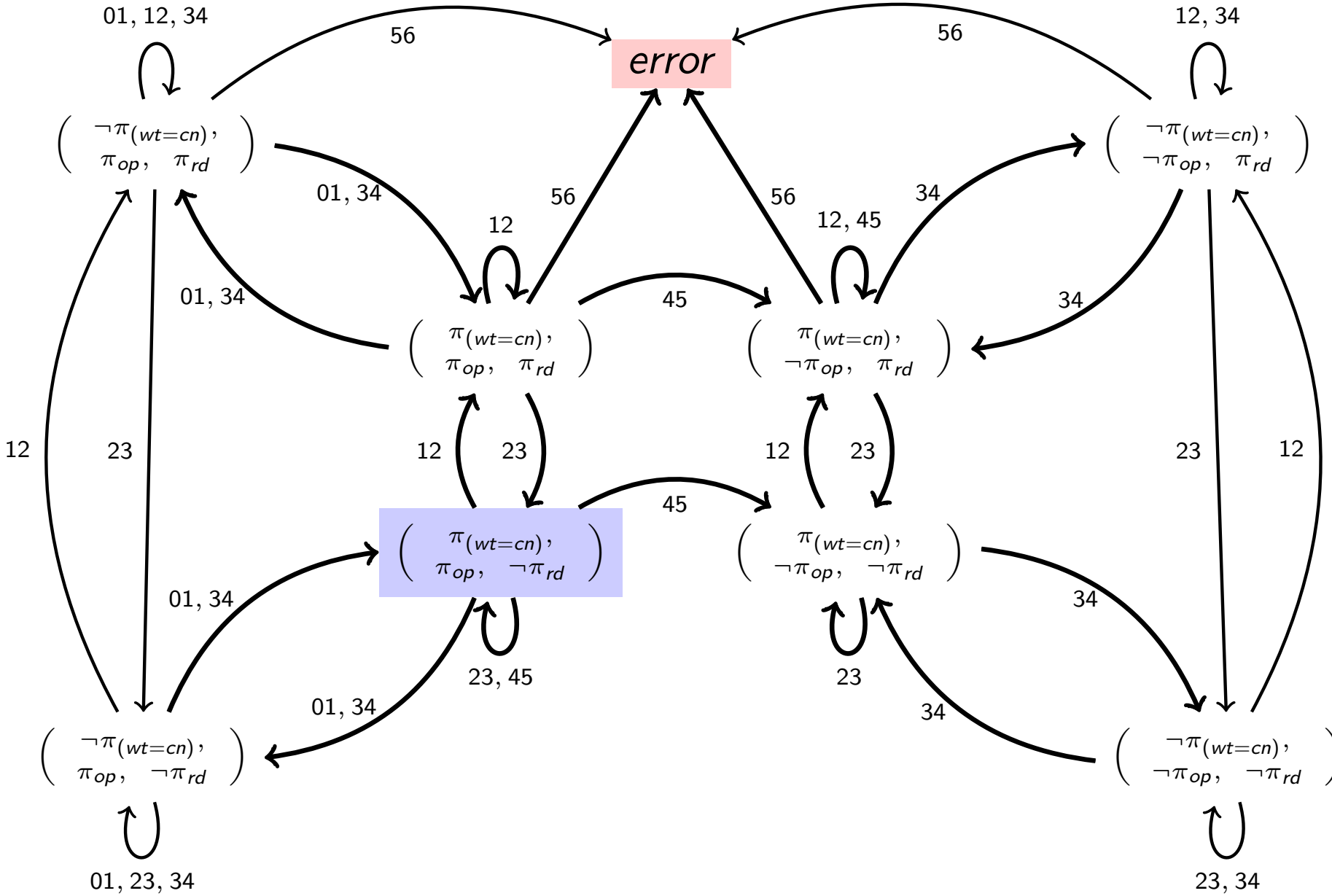
$wt = \sum_{i \geq 3}(\mathcal{N}_{(@lc_i)})$

So each time ($wait = count$):

$\mathcal{N}_{(@lc_1)} + \mathcal{N}_{(@lc_2)} + \mathcal{N}_{(@lc_3)} = 0$

...source of non-monotonicity


error
$\left(\begin{array}{l}\neg\pi_{(wt=cn)},\\ \pi_{op},\quad \pi_{rd}\end{array}\right)$
$\left(\begin{array}{l}\neg\pi_{(wt=cn)},\\ \neg\pi_{op},\quad \pi_{rd}\end{array}\right)$
$\left(\begin{array}{l}\pi_{(wt=cn)},\\ \pi_{op},\quad \pi_{rd}\end{array}\right)$
$\left(\begin{array}{l}\pi_{(wt=cn)},\\ \neg\pi_{op},\quad \pi_{rd}\end{array}\right)$
$\left(\begin{array}{l}\pi_{(wt=cn)},\\ \pi_{op},\quad \neg\pi_{rd}\end{array}\right)$
$\left(\begin{array}{l}\pi_{(wt=cn)},\\ \neg\pi_{op},\quad \neg\pi_{rd}\end{array}\right)$
$\left(\begin{array}{l}\neg\pi_{(wt=cn)},\\ \pi_{op},\quad \neg\pi_{rd}\end{array}\right)$
$\left(\begin{array}{l}\neg\pi_{(wt=cn)},\\ \neg\pi_{op},\quad \neg\pi_{rd}\end{array}\right)$
01, 12, 34
12, 34
01, 23, 34
23, 34
56
01, 34
12
12, 45
45
34
23
23, 45


error
$\neg\pi_{(wt=cn)}$, π_{op}, π_{rd}
$\pi_{(wt=cn)}$, π_{op}, π_{rd}
$\pi_{(wt=cn)}$, $\neg\pi_{op}$, π_{rd}
$\neg\pi_{(wt=cn)}$, $\neg\pi_{op}$, π_{rd}
$\pi_{(wt=cn)}$, π_{op}, $\neg\pi_{rd}$
$\pi_{(wt=cn)}$, $\neg\pi_{op}$, $\neg\pi_{rd}$
$\neg\pi_{(wt=cn)}$, π_{op}, $\neg\pi_{rd}$
$\neg\pi_{(wt=cn)}$, $\neg\pi_{op}$, $\neg\pi_{rd}$
01, 12, 34
56
56
12, 34
01, 34
56
56
12
12, 45
34
01, 34
45
34
12
23
12
23
12
23
23
12
45
01, 34
34
23, 45
23
01, 34
34
01, 23, 34
23, 34

Global Configurations

- Checking reachability of global configurations amounts to adding tests in the counter machine,
- For deadlocks we test that the number of enabled threads is zero,
- This makes the system non-monotonic

Goal and motivation

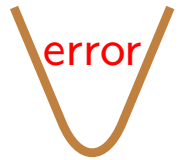
Predicate Abstraction

Strengthening

Monotonic abstraction

Table of Contents

Backward Reachability Analysis



$\pi_{(op)} = tt, \pi_{(rd)} = tt, \pi_{(wt=cn)} = tt$

lc_0

$01 : \pi_{(op)} \Rightarrow \pi_{(wt=cn)} := ch(ff, \pi_{(wt=cn)})$

lc_1

$12 : \pi_{(rd)} := ch(tt, ff)$

lc_2

$23 : \pi_{(rd)} := ch(ff, tt)$

lc_3

$34 : \pi_{(wt=cn)} := ch(ff, \pi_{wt=cn})$

lc_4

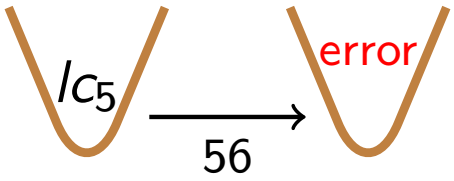
$45 : \pi_{(wt=cn)} \Rightarrow \pi_{(op)} := ff$

lc_5

$56 : \pi_{(rd)}$

lc_6

Backward Reachability Analysis



$\pi_{(op)} = tt, \pi_{(rd)} = tt, \pi_{(wt=cn)} = tt$

lc_0

$01 : \pi_{(op)} \Rightarrow \pi_{(wt=cn)} := ch(ff, \pi_{(wt=cn)})$

lc_1

$12 : \pi_{(rd)} := ch(tt, ff)$

lc_2

$23 : \pi_{(rd)} := ch(ff, tt)$

lc_3

$34 : \pi_{(wt=cn)} := ch(ff, \pi_{wt=cn})$

lc_4

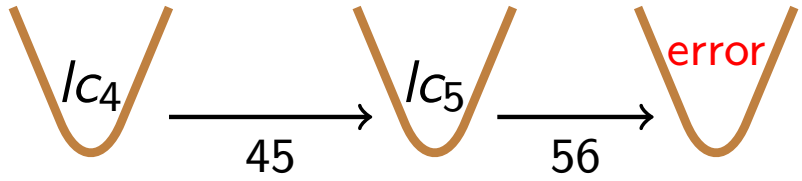
$45 : \pi_{(wt=cn)} \Rightarrow \pi_{(op)} := ff$

lc_5

$56 : \pi_{(rd)}$

lc_6

Backward Reachability Analysis



$\pi_{(op)} = tt, \pi_{(rd)} = tt, \pi_{(wt=cn)} = tt$

lc_0

$01 : \pi_{(op)} \Rightarrow \pi_{(wt=cn)} := ch(ff, \pi_{(wt=cn)})$

lc_1

$12 : \pi_{(rd)} := ch(tt, ff)$

lc_2

$23 : \pi_{(rd)} := ch(ff, tt)$

lc_3

$34 : \pi_{(wt=cn)} := ch(ff, \pi_{wt=cn})$

lc_4

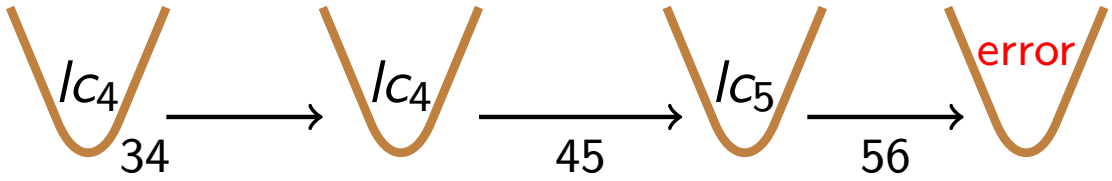
$45 : \pi_{(wt=cn)} \Rightarrow \pi_{(op)} := ff$

lc_5

$56 : \pi_{(rd)}$

lc_6

Backward Reachability Analysis



$\pi_{(op)} = tt, \pi_{(rd)} = tt, \pi_{(wt=cn)} = tt$

lc_0

$01 : \pi_{(op)} \Rightarrow \pi_{(wt=cn)} := ch(ff, \pi_{(wt=cn)})$

lc_1

$12 : \pi_{(rd)} := ch(tt, ff)$

lc_2

$23 : \pi_{(rd)} := ch(ff, tt)$

lc_3

$34 : \pi_{(wt=cn)} := ch(ff, \pi_{wt=cn})$

lc_4

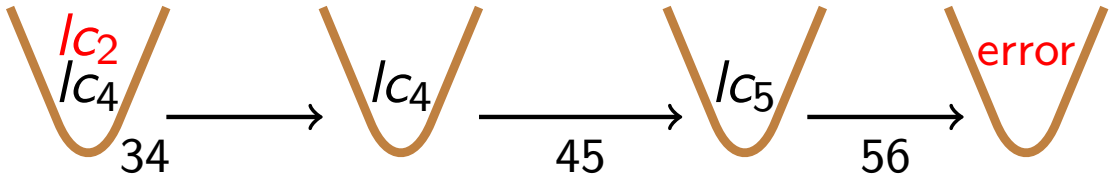
$45 : \pi_{(wt=cn)} \Rightarrow \pi_{(op)} := ff$

lc_5

$56 : \pi_{(rd)}$

lc_6

Backward Reachability Analysis



$\pi_{(op)} = tt, \pi_{(rd)} = tt, \pi_{(wt=cn)} = tt$

lc_0

01 : $\pi_{(op)} \Rightarrow \pi_{(wt=cn)} := ch(ff, \pi_{(wt=cn)})$

lc_1

12 : $\pi_{(rd)} := ch(tt, ff)$

lc_2

23 : $\pi_{(rd)} := ch(ff, tt)$

lc_3

34 : $\pi_{(wt=cn)} := ch(ff, \pi_{wt=cn})$

lc_4

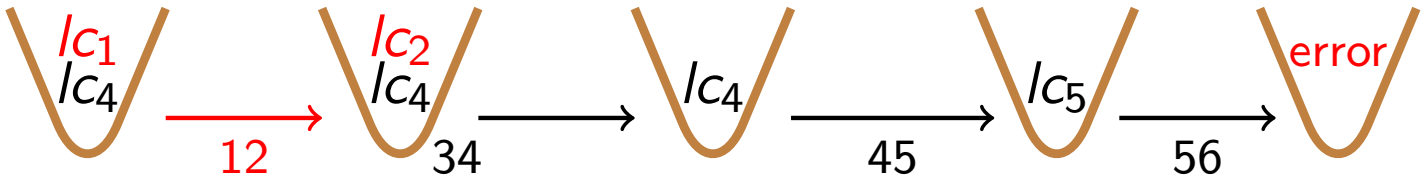
45 : $\pi_{(wt=cn)} \Rightarrow \pi_{(op)} := ff$

lc_5

56 : $\pi_{(rd)}$

lc_6

Backward Reachability Analysis



$\pi_{(op)} = tt, \pi_{(rd)} = tt, \pi_{(wt=cn)} = tt$

lc_0

01 : $\pi_{(op)} \Rightarrow \pi_{(wt=cn)} := ch(ff, \pi_{(wt=cn)})$

lc_1

12 : $\pi_{(rd)} := ch(tt, ff)$

lc_2

23 : $\pi_{(rd)} := ch(ff, tt)$

lc_3

34 : $\pi_{(wt=cn)} := ch(ff, \pi_{wt=cn})$

lc_4

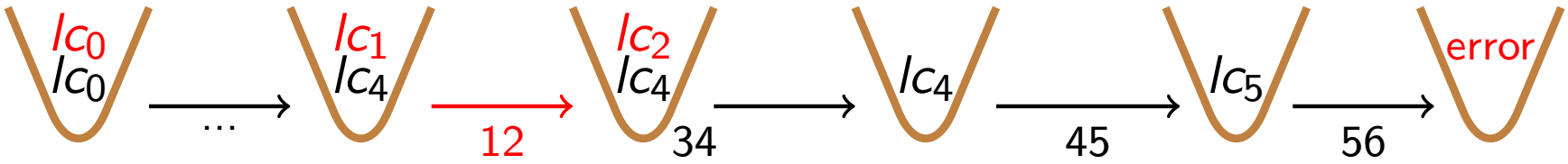
45 : $\pi_{(wt=cn)} \Rightarrow \pi_{(op)} := ff$

lc_5

56 : $\pi_{(rd)}$

lc_6

Backward Reachability Analysis



$\pi_{(op)} = tt, \pi_{(rd)} = tt, \pi_{(wt=cn)} = tt$

lc_0

$01 : \pi_{(op)} \Rightarrow \pi_{(wt=cn)} := ch(ff, \pi_{(wt=cn)})$

lc_1

$12 : \pi_{(rd)} := ch(tt, ff)$

lc_2

$23 : \pi_{(rd)} := ch(ff, tt)$

lc_3

$34 : \pi_{(wt=cn)} := ch(ff, \pi_{wt=cn})$

lc_4

$45 : \pi_{(wt=cn)} \Rightarrow \pi_{(op)} := ff$

lc_5

$56 : \pi_{(rd)}$

lc_6

PACMAN: PredicAted Constrained Monotonic AbstractioN