Formal Methods for the Verification of Distributed Algorithms

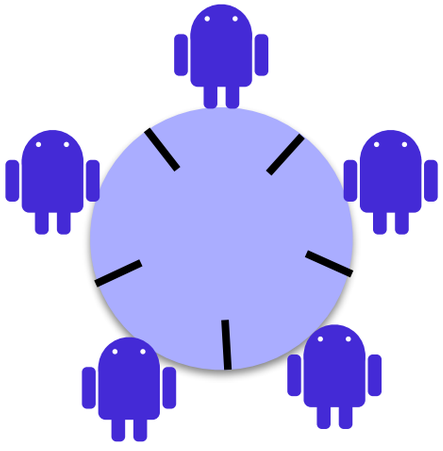
Paul Gastin

Laboratoire Spécification et Vérification
ENS Cachan, CNRS & Inria

Joint work with C. Aiswarya & Benedikt Bollig
CONCUR'15

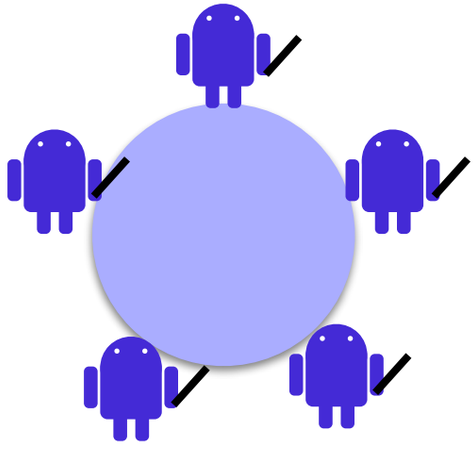
DRV, Bertinoro, May 18, 2016

Formal methods & verification



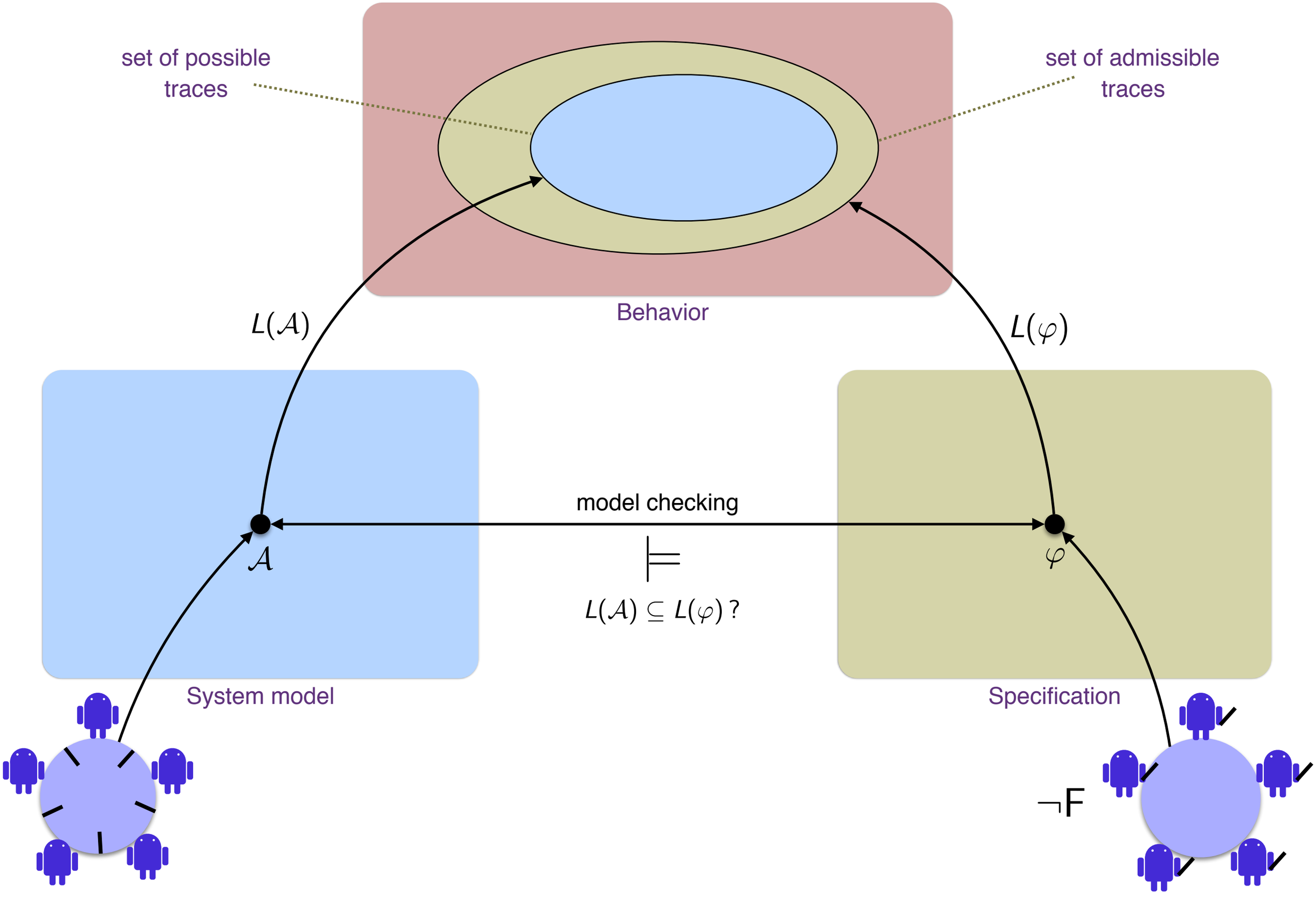
distributed system

¬F



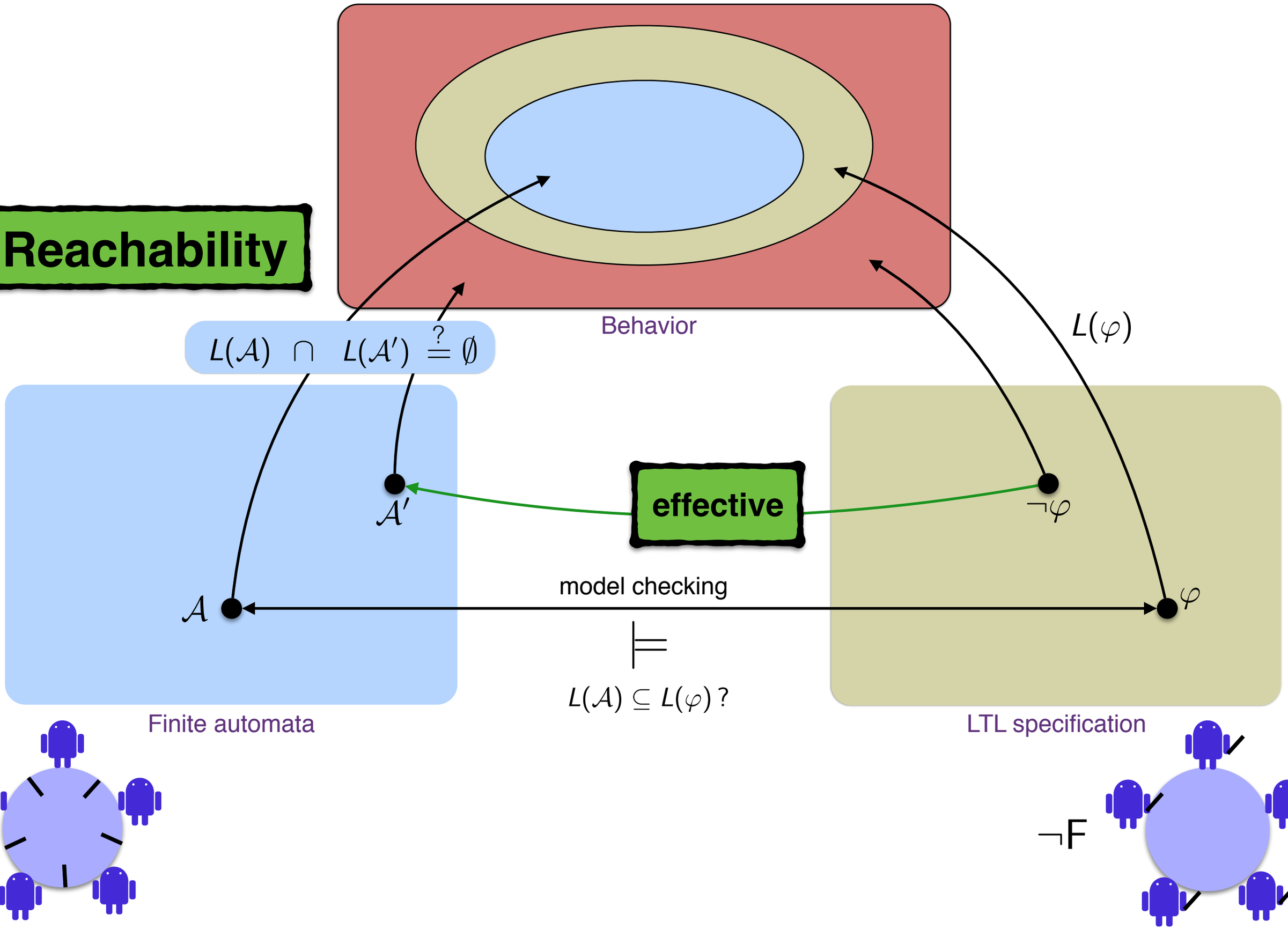
specification

Formal methods & verification



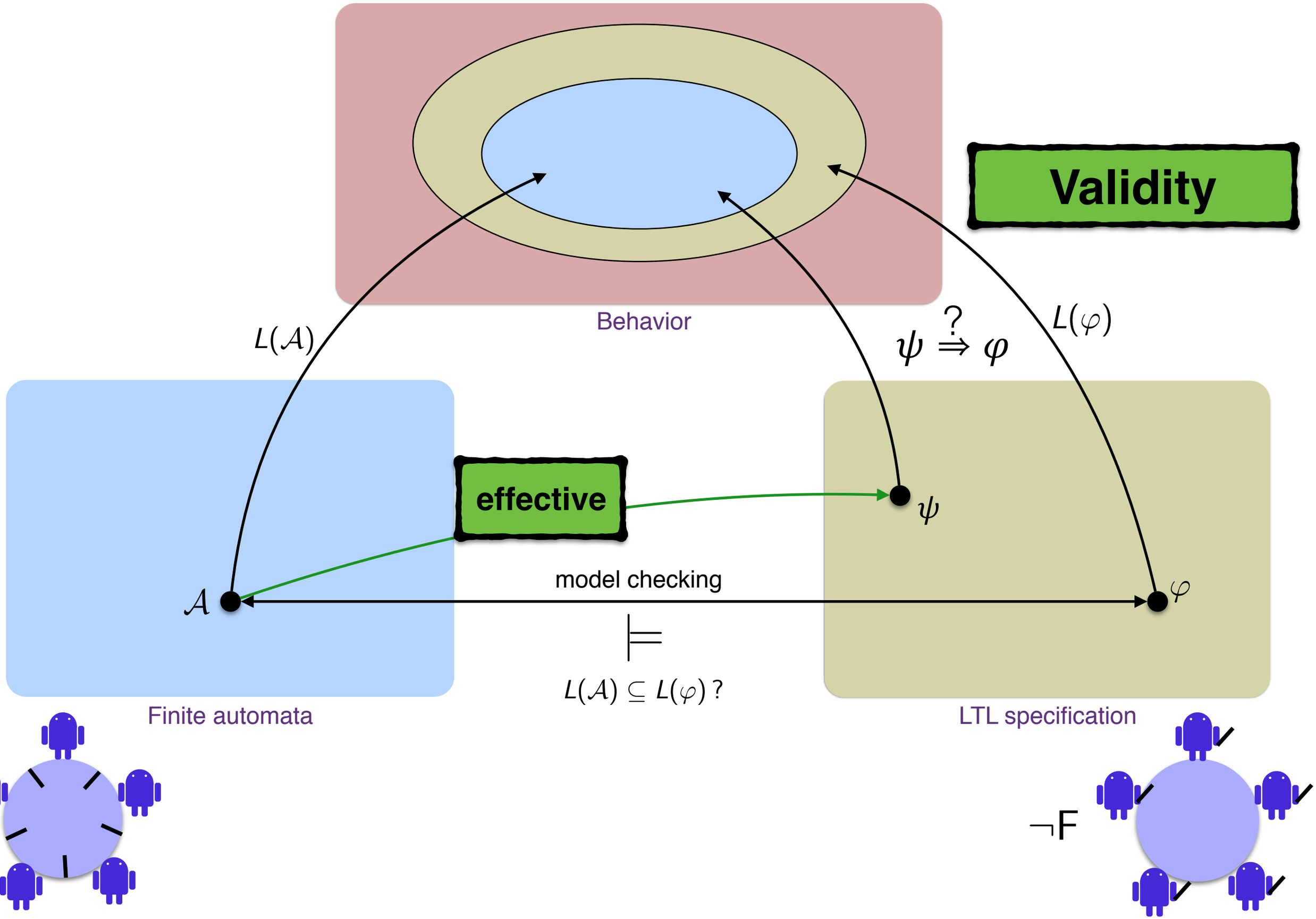
Formal methods & verification

Reachability

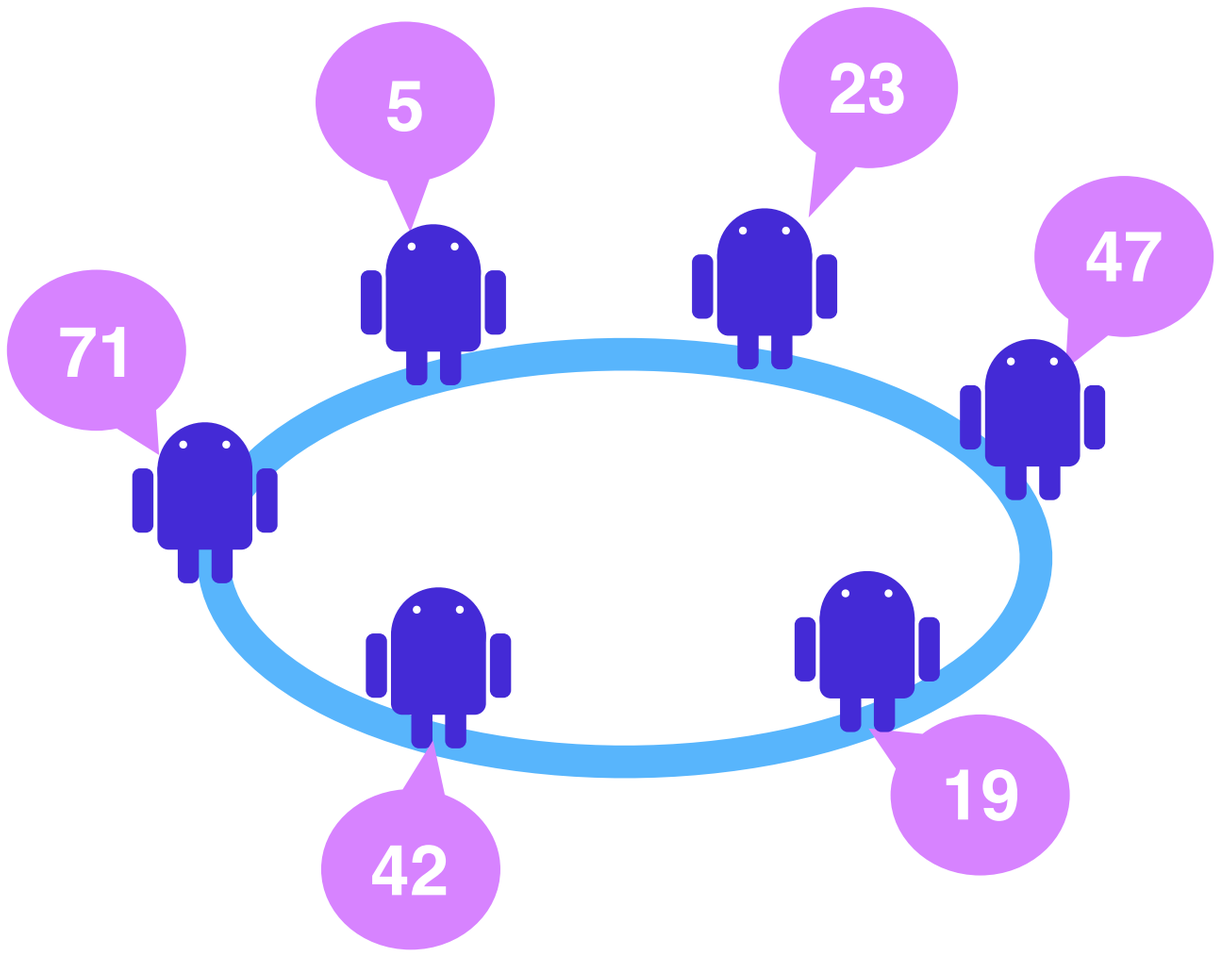


$\neg$F

Formal methods & verification



Models of Distributed Systems



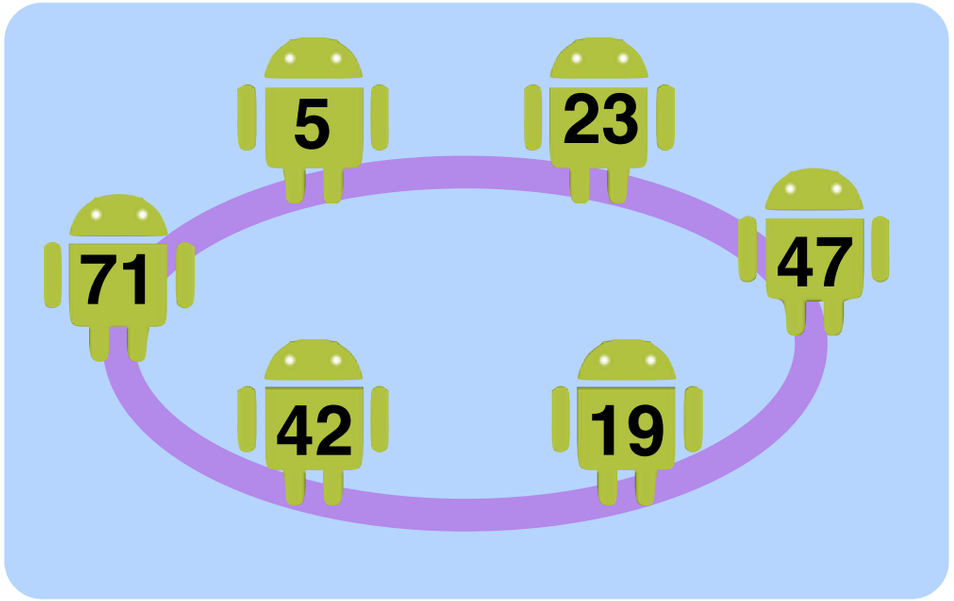
Distributed Algorithms

Topology
- tree, ring, star, …

Number of processes
- fixed & static
- non-fixed & unbounded
  - static (parameterized)
  - dynamic

Identification
- (partly) indistinguishable
- unique process identifiers (pids)
  - test for equality
  - test for linear order

Communication
- broadcast
- shared variable
- point-to-point
  - rendez-vous
  - FIFO queues



System model

$\models$

φ

Specification

Several sources of infinity / undecidability

Distributed Algorithms

Topology
- tree, ring, star, …

Number of processes
- fixed & static
- non-fixed & unbounded

→ static (parameterized)

dynamic

Identification
- (partly) indistinguishable
- unique process identifiers (pids)

test for equality

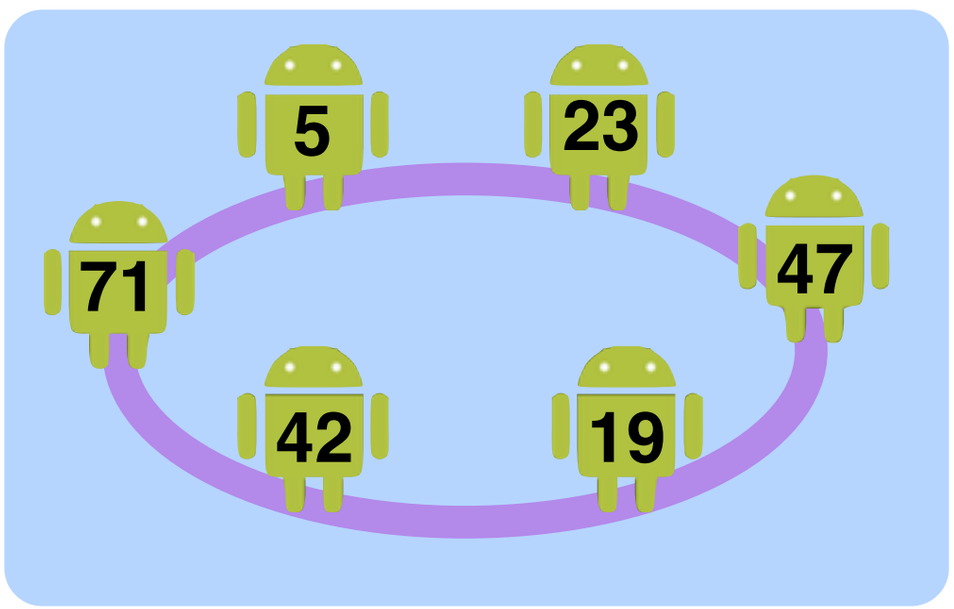
→ test for linear order

Communication
- broadcast
- shared variable
- point-to-point

→ rendez-vous

FIFO queues

Our Model



System model

$\models$

φ

Specification

Several sources of infinity / undecidability

Distributed algorithms

Leader election [Franklin ’82]

arbitrary number of identical processes

Behavior

Distributed algorithm

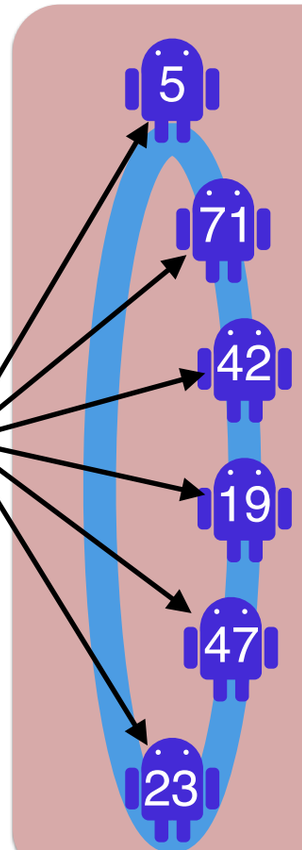
Distributed algorithms

Leader election [Franklin ’82]

arbitrary distribution
of process identifiers

5 < 19 < 23 < …

elect process
with maximum id

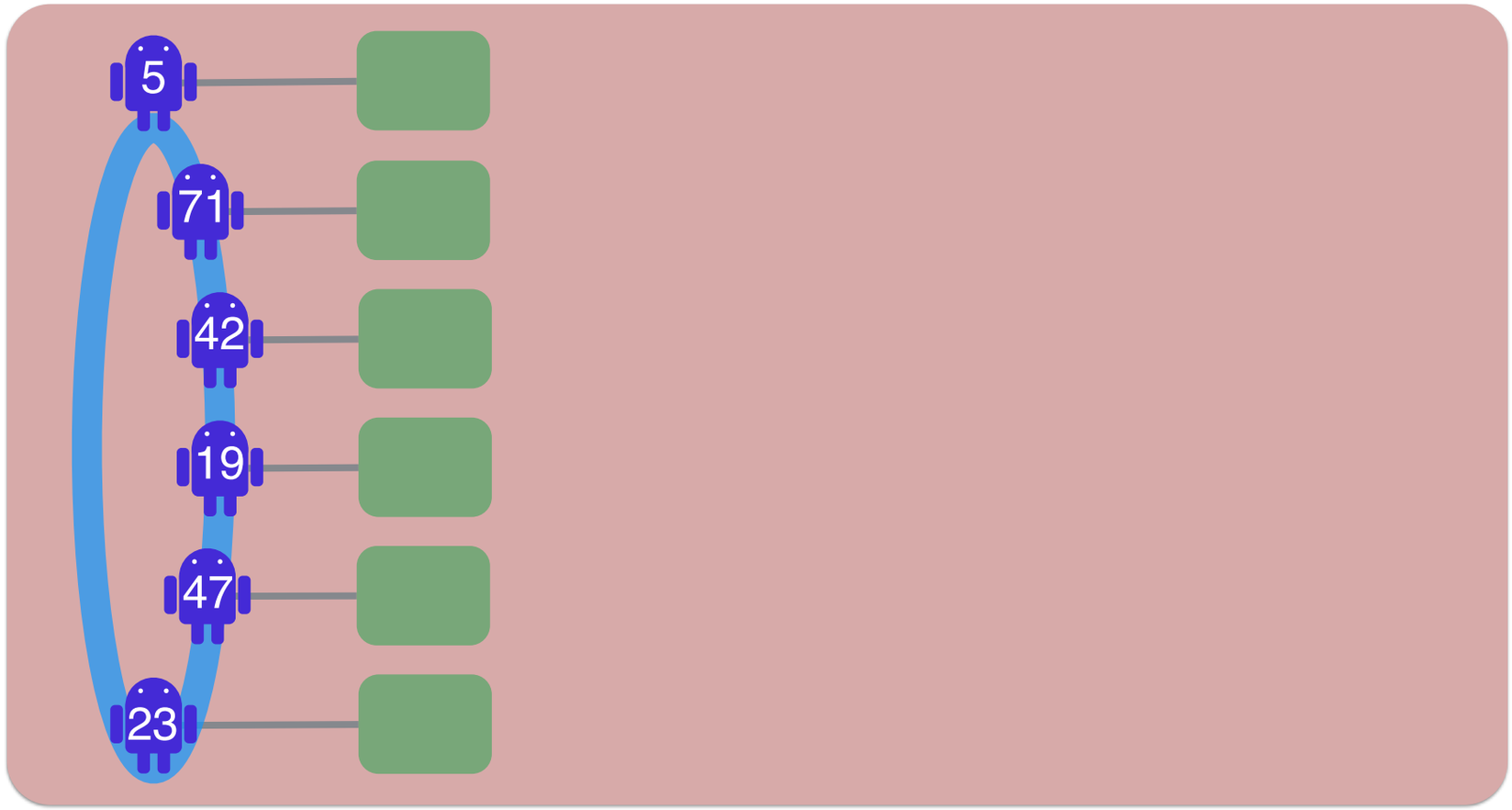


Behavior

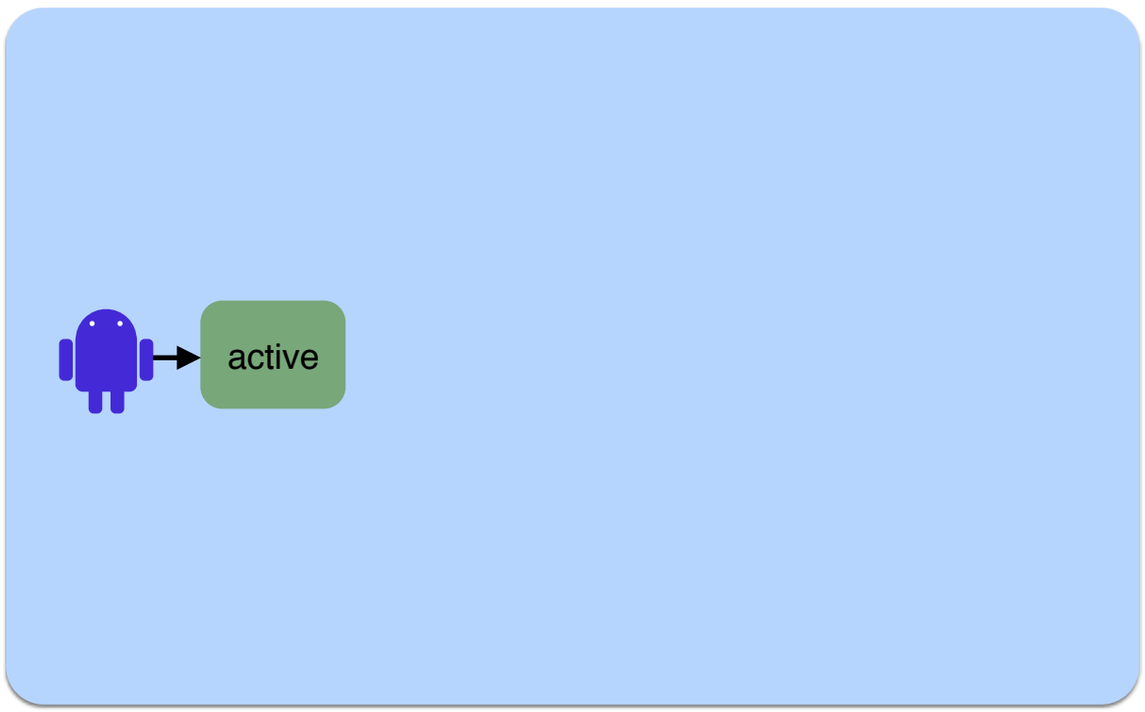
Distributed algorithm

Distributed algorithms

Leader election [Franklin ’82]



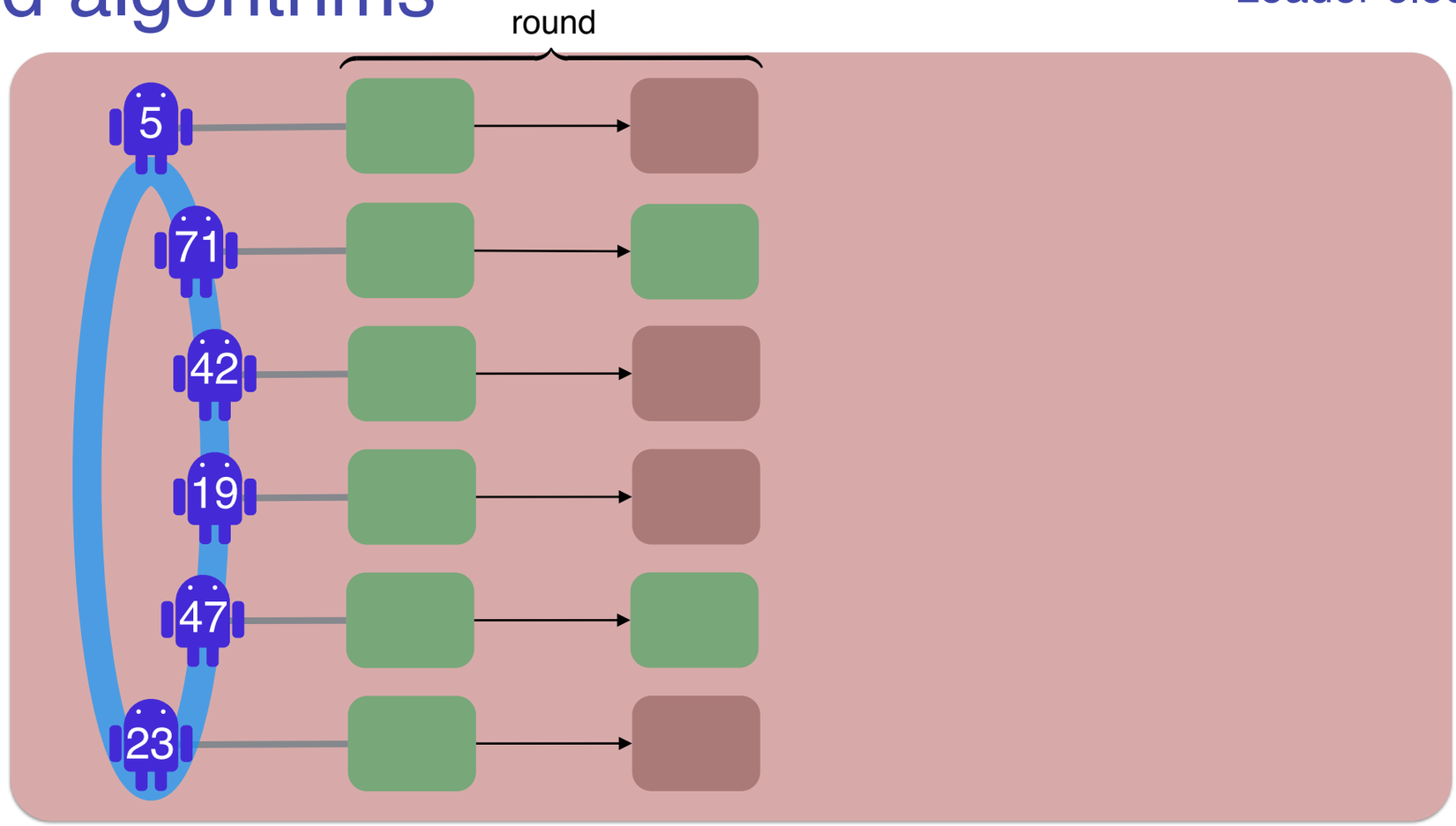
Behavior



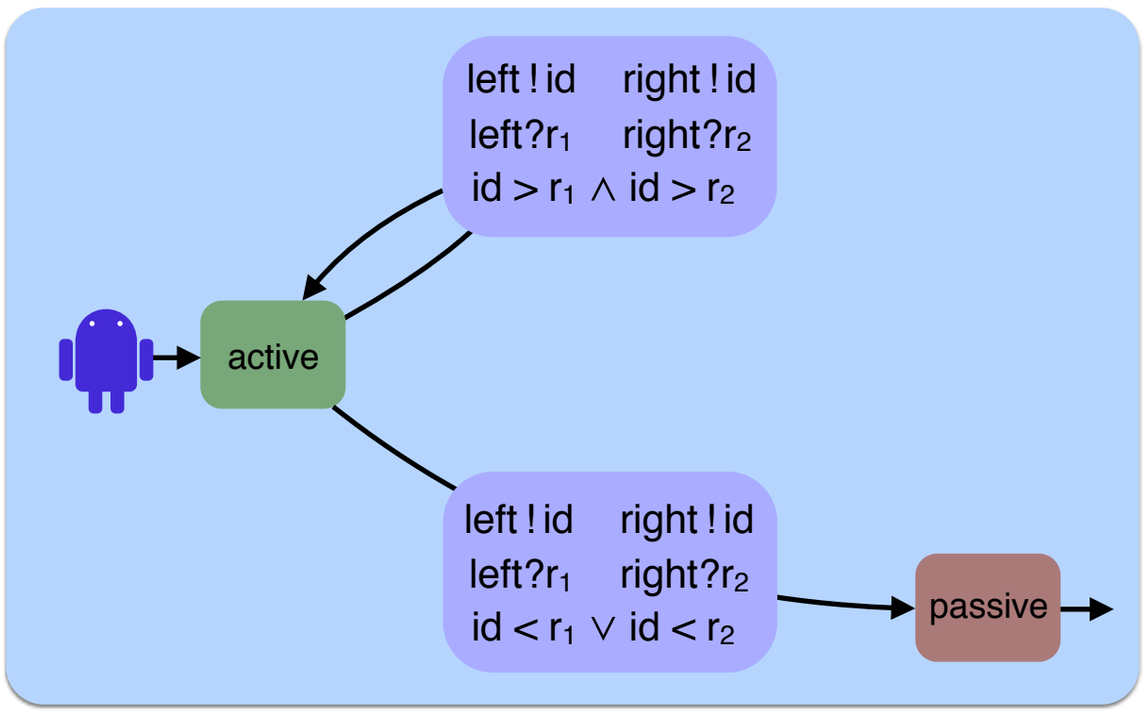
Distributed algorithm

Distributed algorithms

Leader election [Franklin ’82]



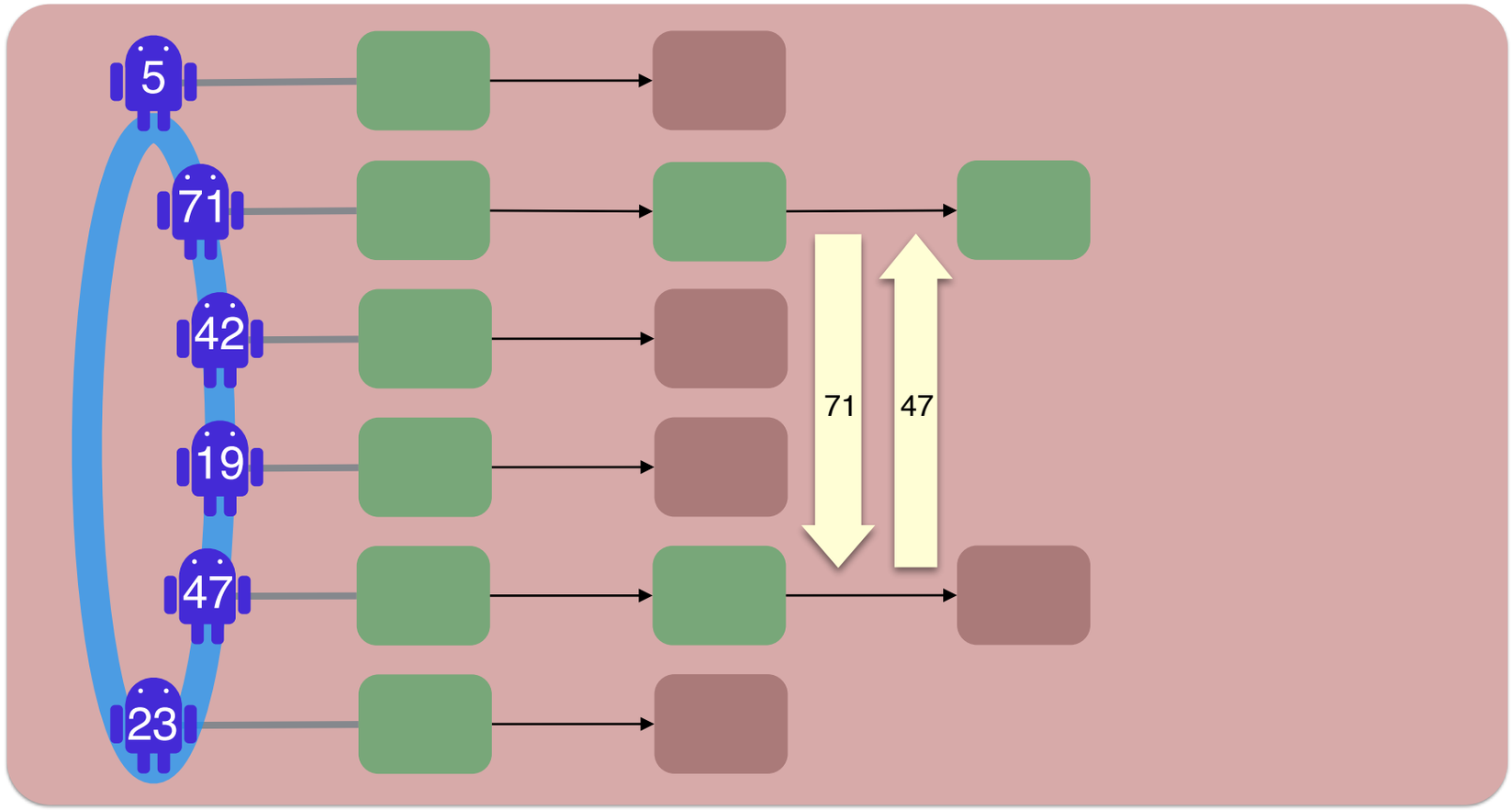
Behavior



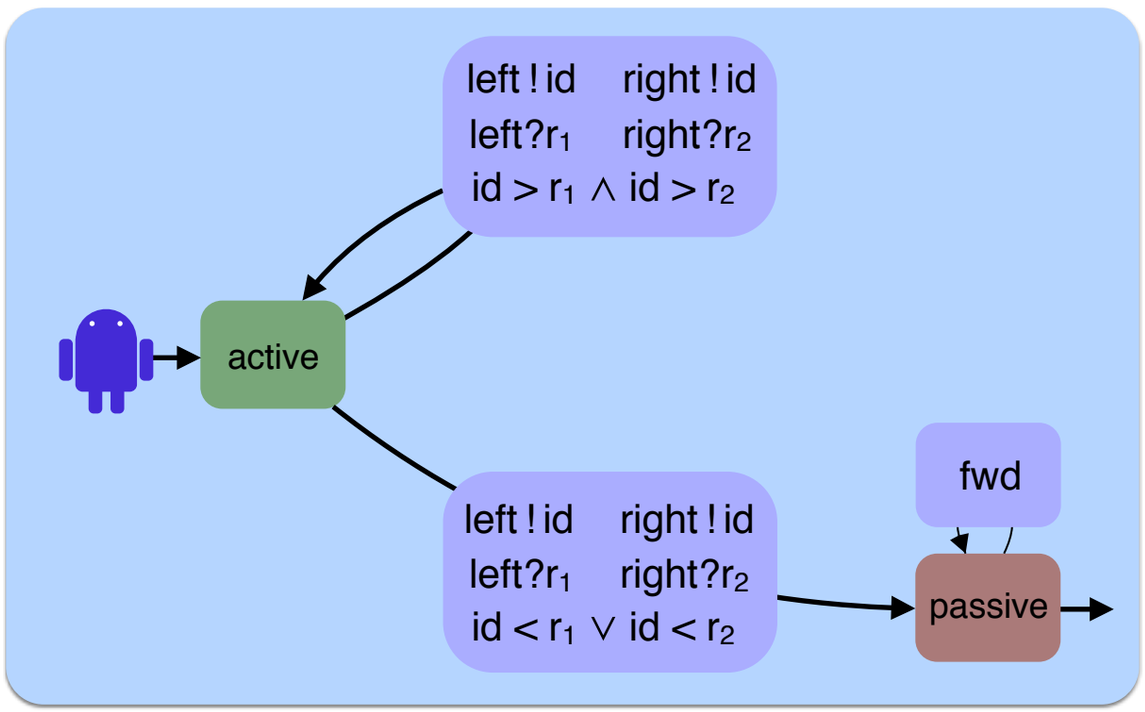
Distributed algorithm

Distributed algorithms

Leader election [Franklin ’82]



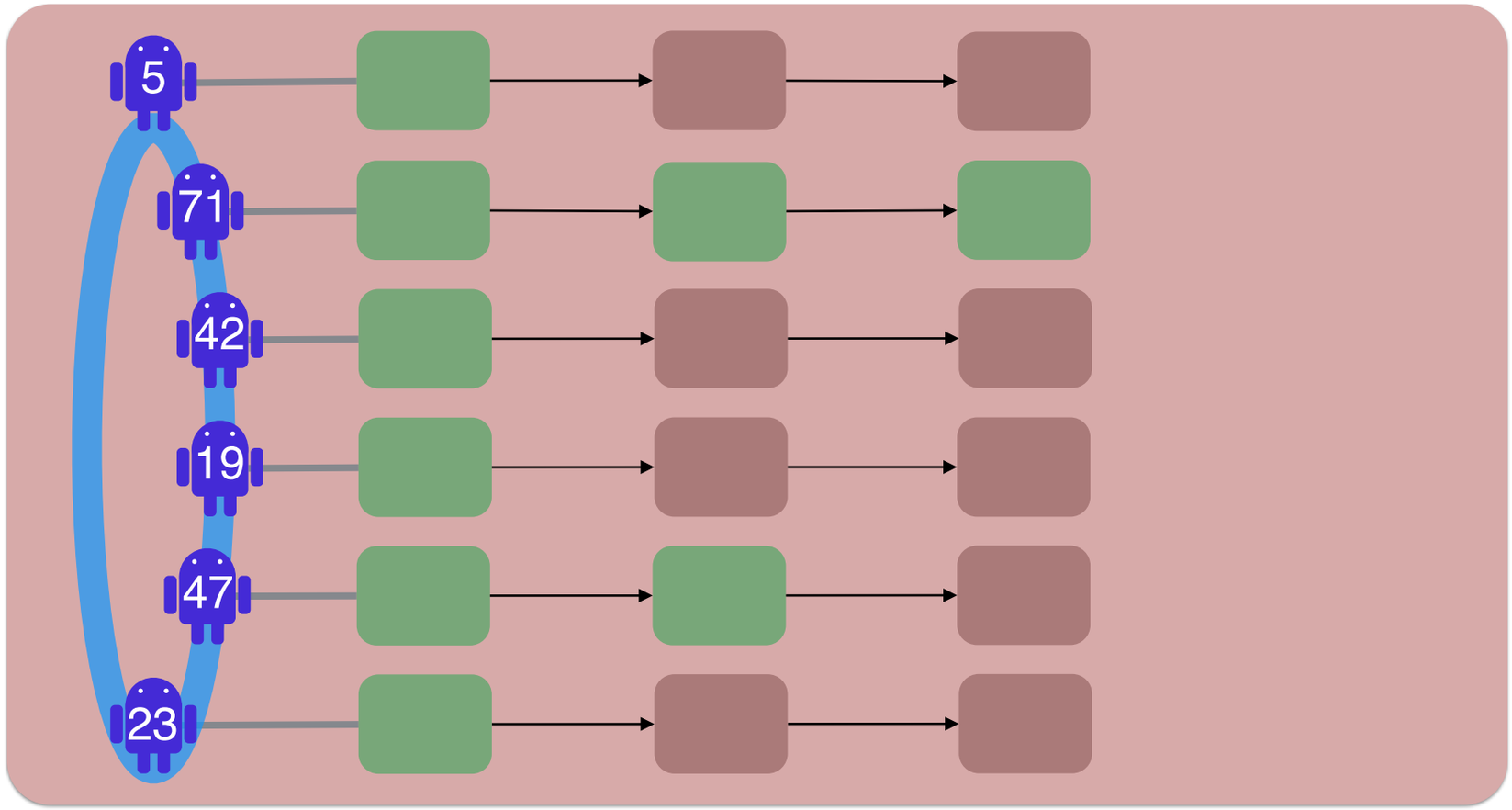
Behavior



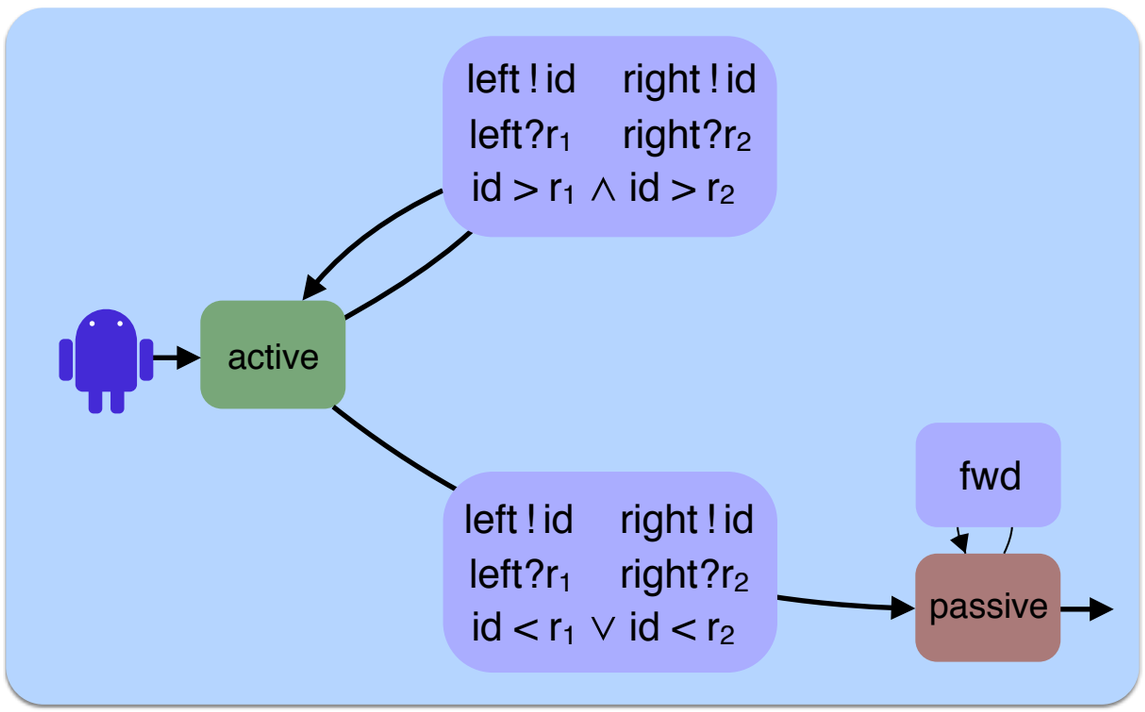
Distributed algorithm

Distributed algorithms

Leader election [Franklin ’82]



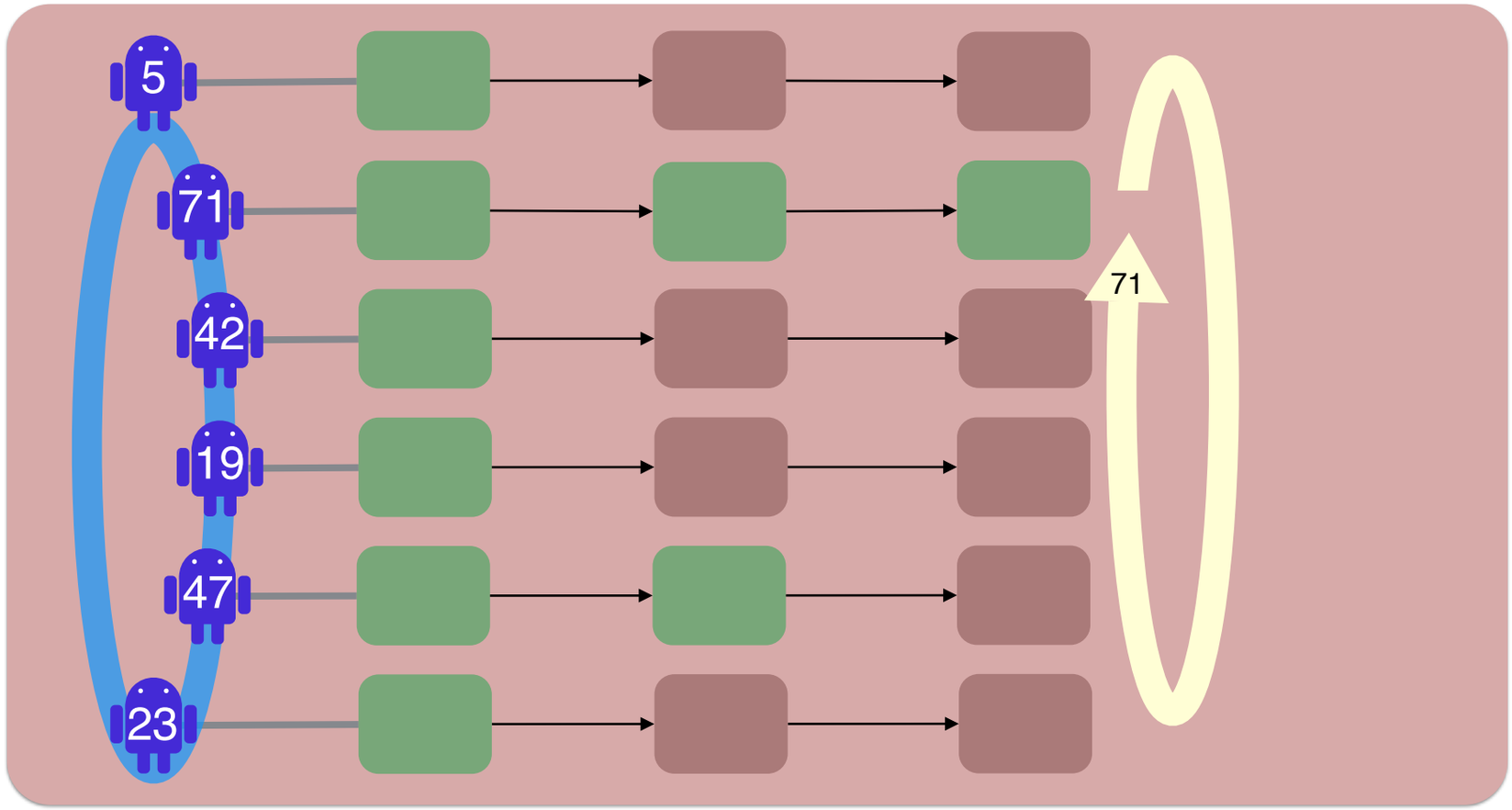
Behavior



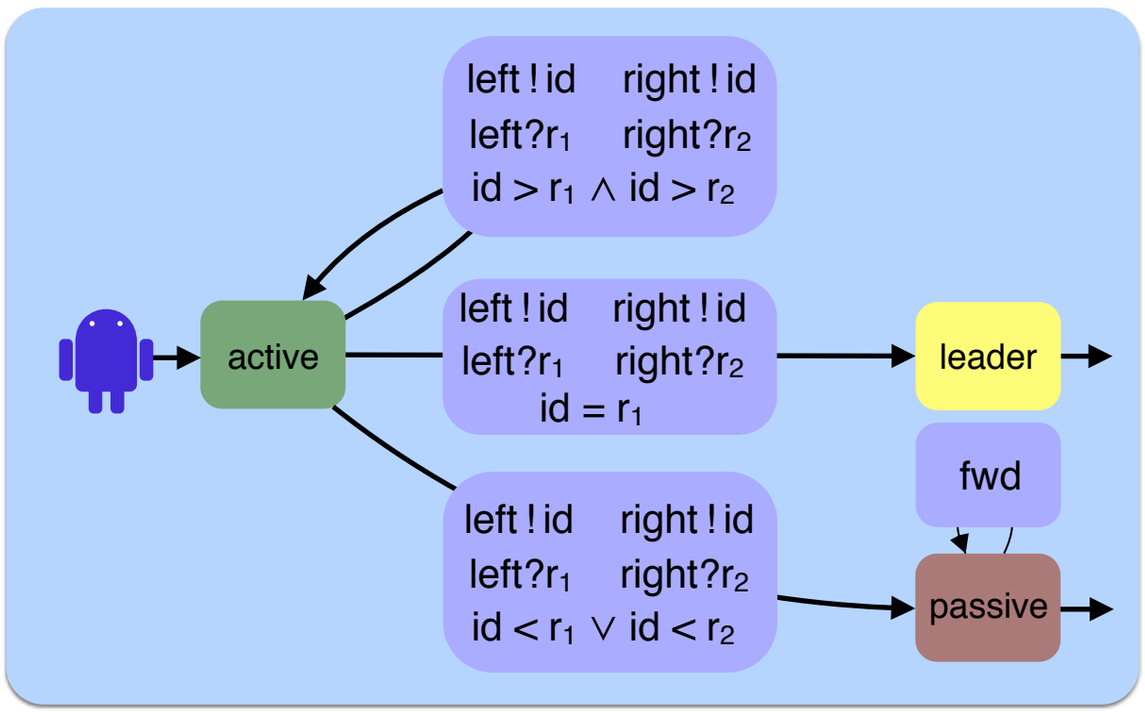
Distributed algorithm

Distributed algorithms

Leader election [Franklin ’82]



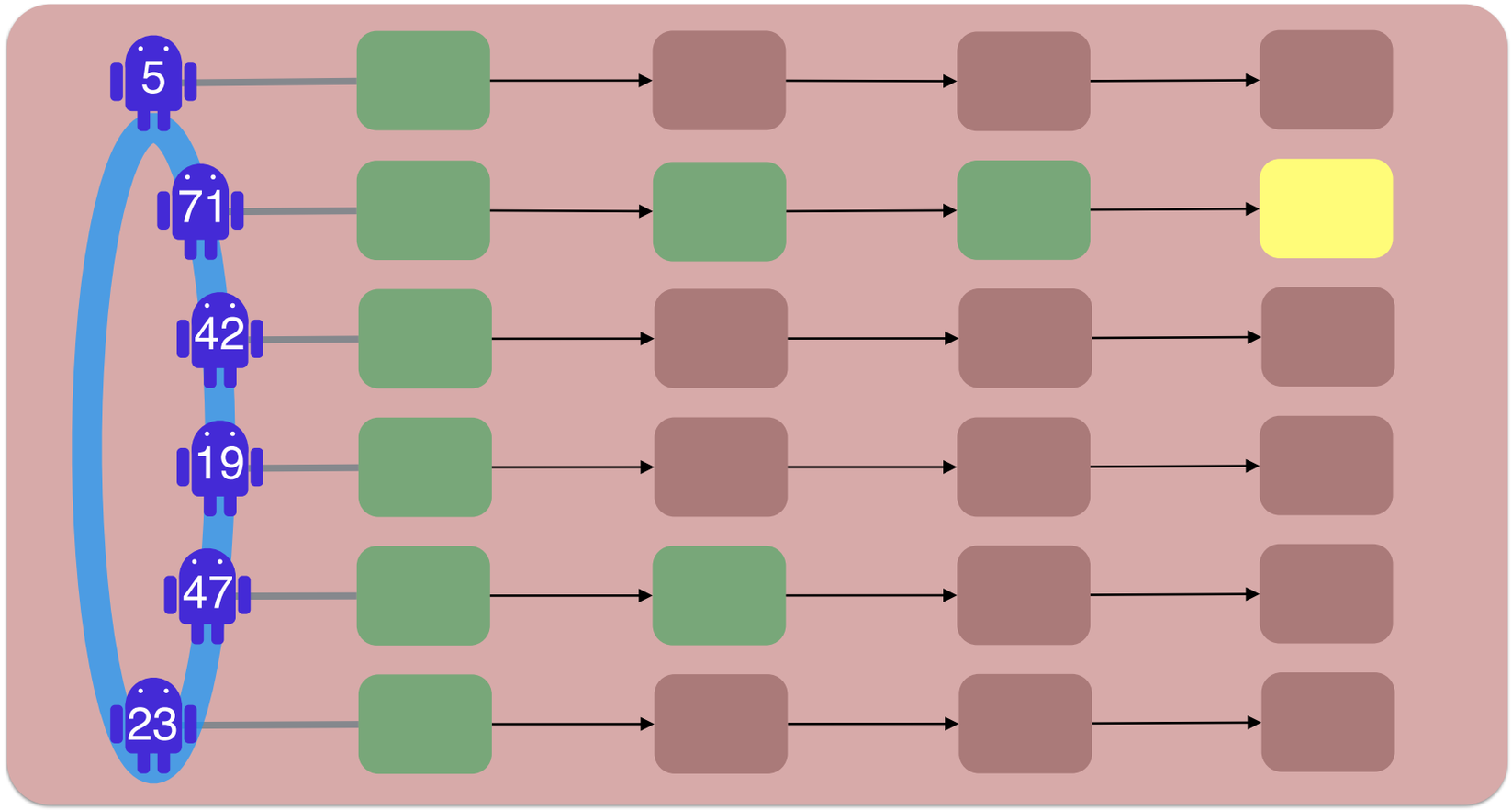
Behavior



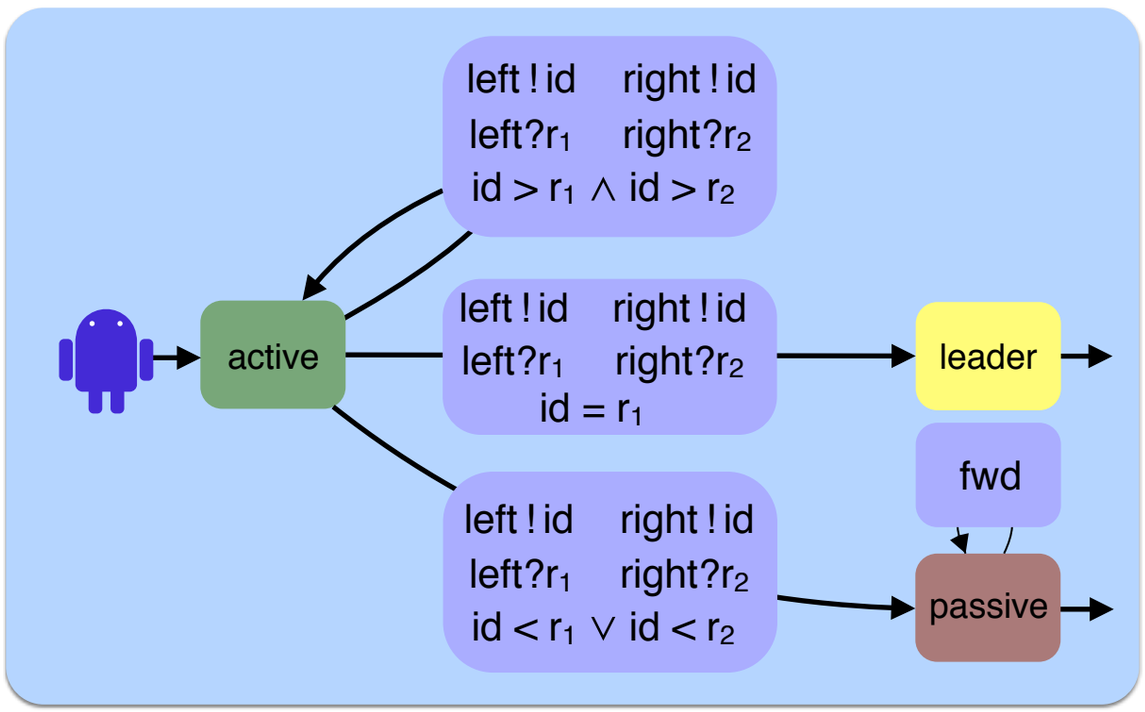
Distributed algorithm

Distributed algorithms

Leader election [Franklin ’82]

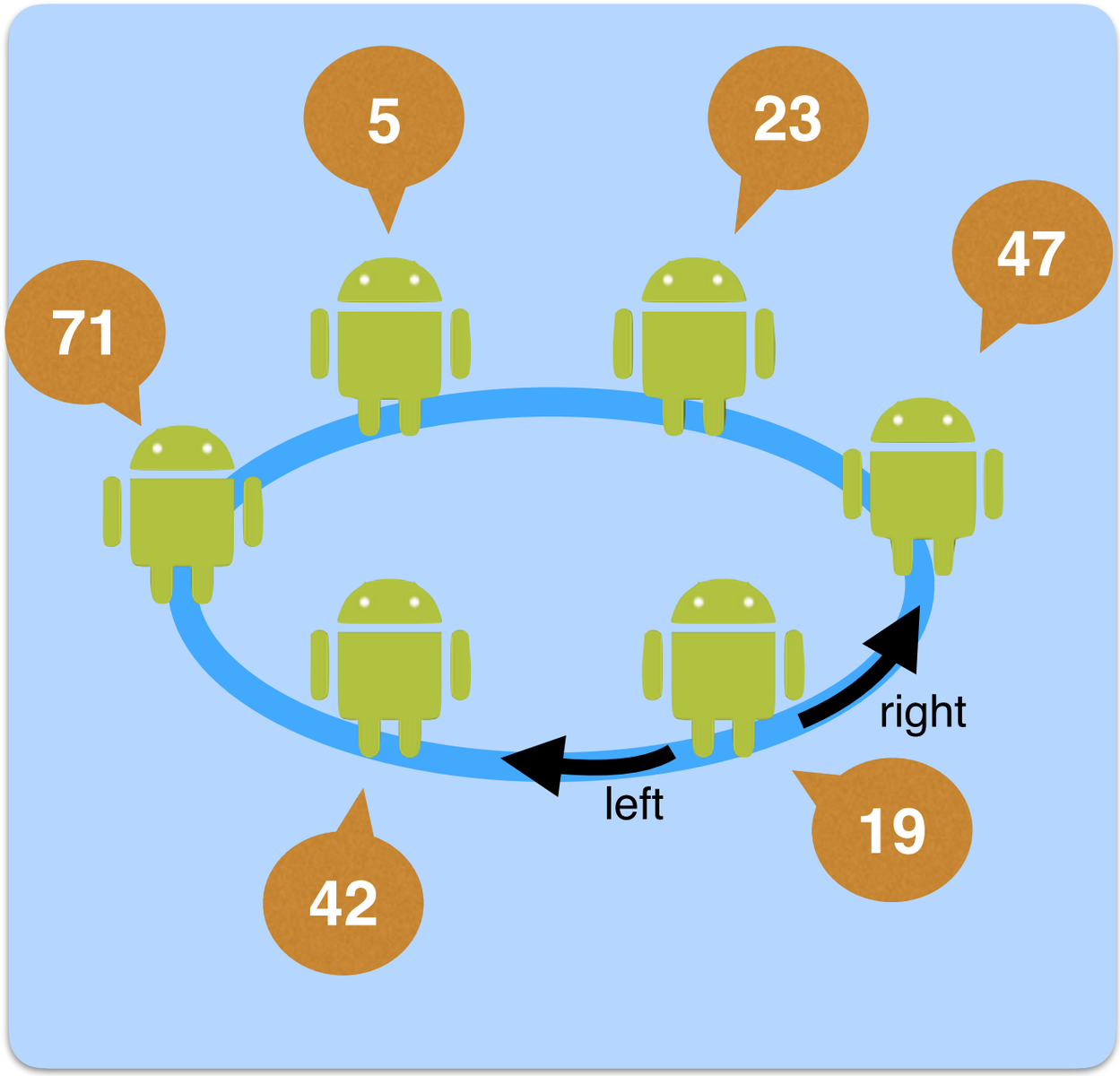


Behavior



Distributed algorithm

Distributed algorithms



- Identical finite-state processes
- Number of processes is unknown and unbounded
- Processes have unique pids (integers — unbounded data)

A formal model for distributed algorithms

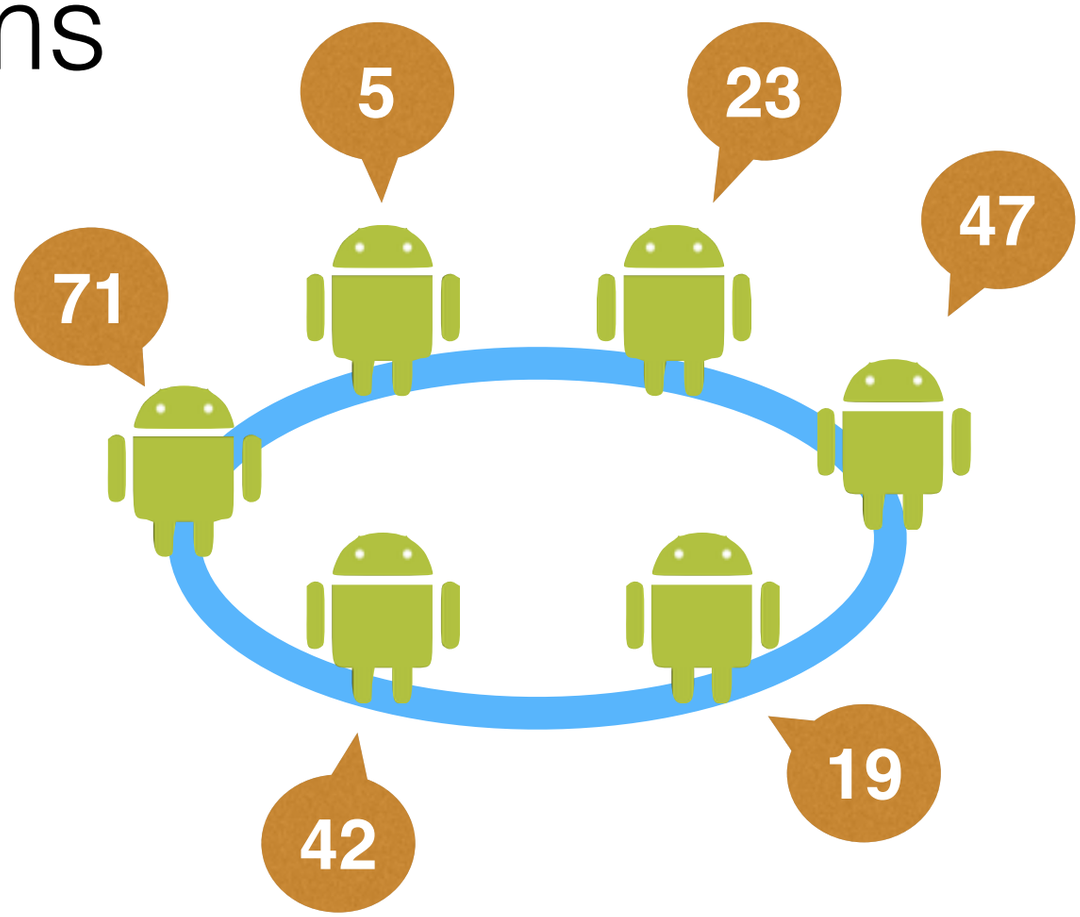
An automata-like way of writing DA

Every process  can be described by:

- Set of states
- Initial state
- Set of registers
  - stores pid
- Set of transitions
  - send pids to neighbours
  - receive pids from neighbours, and store in registers
  - compare registers
  - update registers

Leader Election Algorithms

Dolev-Klawe-Rhodeh82



states: $active_0, active_1$

$passive, found$

initial state: $active_0$

registers: id, r, r', r''

$t_1 = \langle active_0\text{: } \mathbf{right}!r\,;\mathbf{left}?r'\,;\mathbf{goto}\ active_1\rangle$

$t_2 = \langle active_1\text{: } \mathbf{right}!r'\,;\mathbf{left}?r''\,;r'' < r'\,;r < r'\,;r := r'\,;\mathbf{goto}\ active_0\rangle$

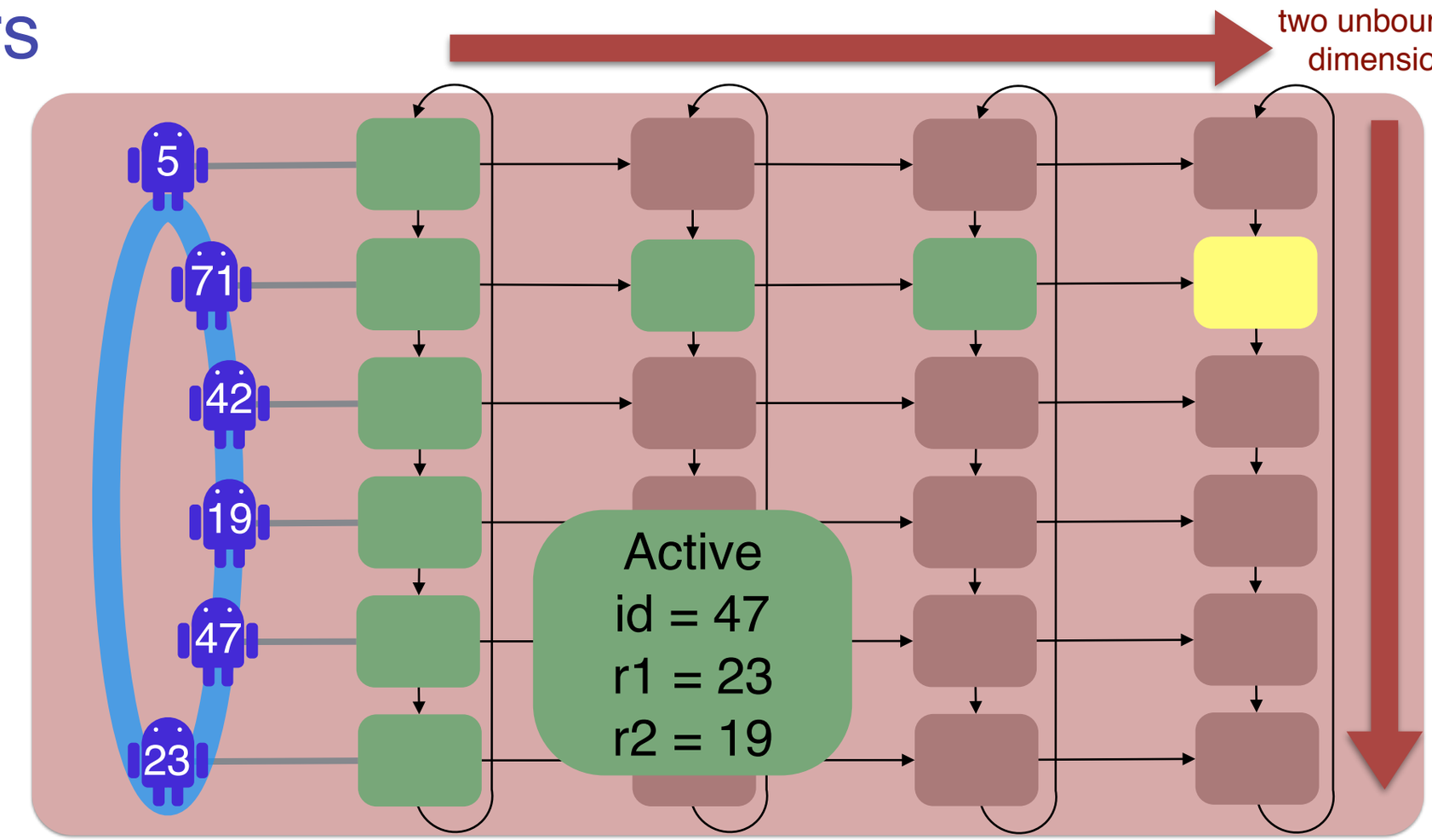
$t_3 = \langle active_1\text{: } \_\_\_\_\_\,;r' < r\,;\mathbf{goto}\ passive\rangle$

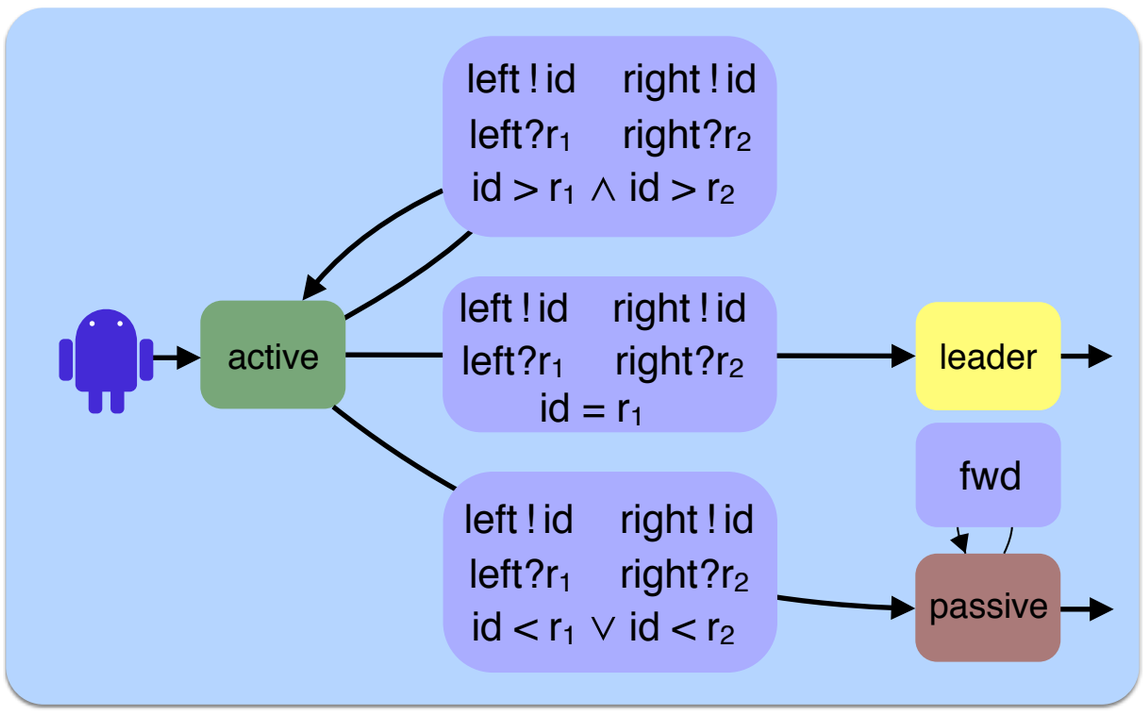
$t_4 = \langle active_1\text{: } \_\_\_\_\_\,;r' < r''\,;\mathbf{goto}\ passive\rangle$

$t_5 = \langle active_1\text{: } \_\_\_\_\_\,;r = r'\,;\mathbf{goto}\ found\rangle$

$t_6 = \langle passive\text{: } \mathbf{fwd}\,;\mathbf{left}?r\,;\mathbf{goto}\ passive\rangle$

Behaviors



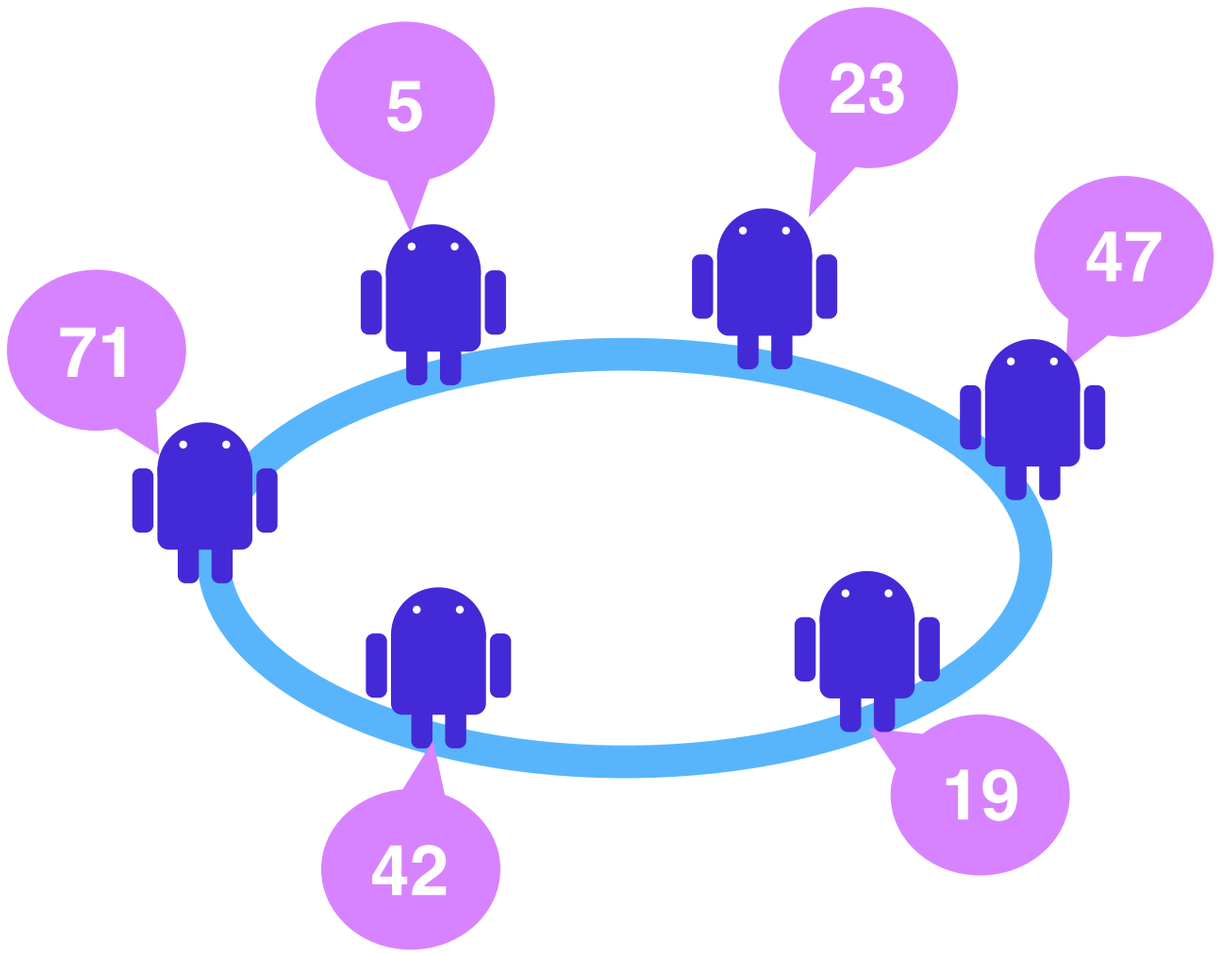


Distributed algorithm

Cylinders
Arbitrary length and width
Labelled with data
from an infinite domain

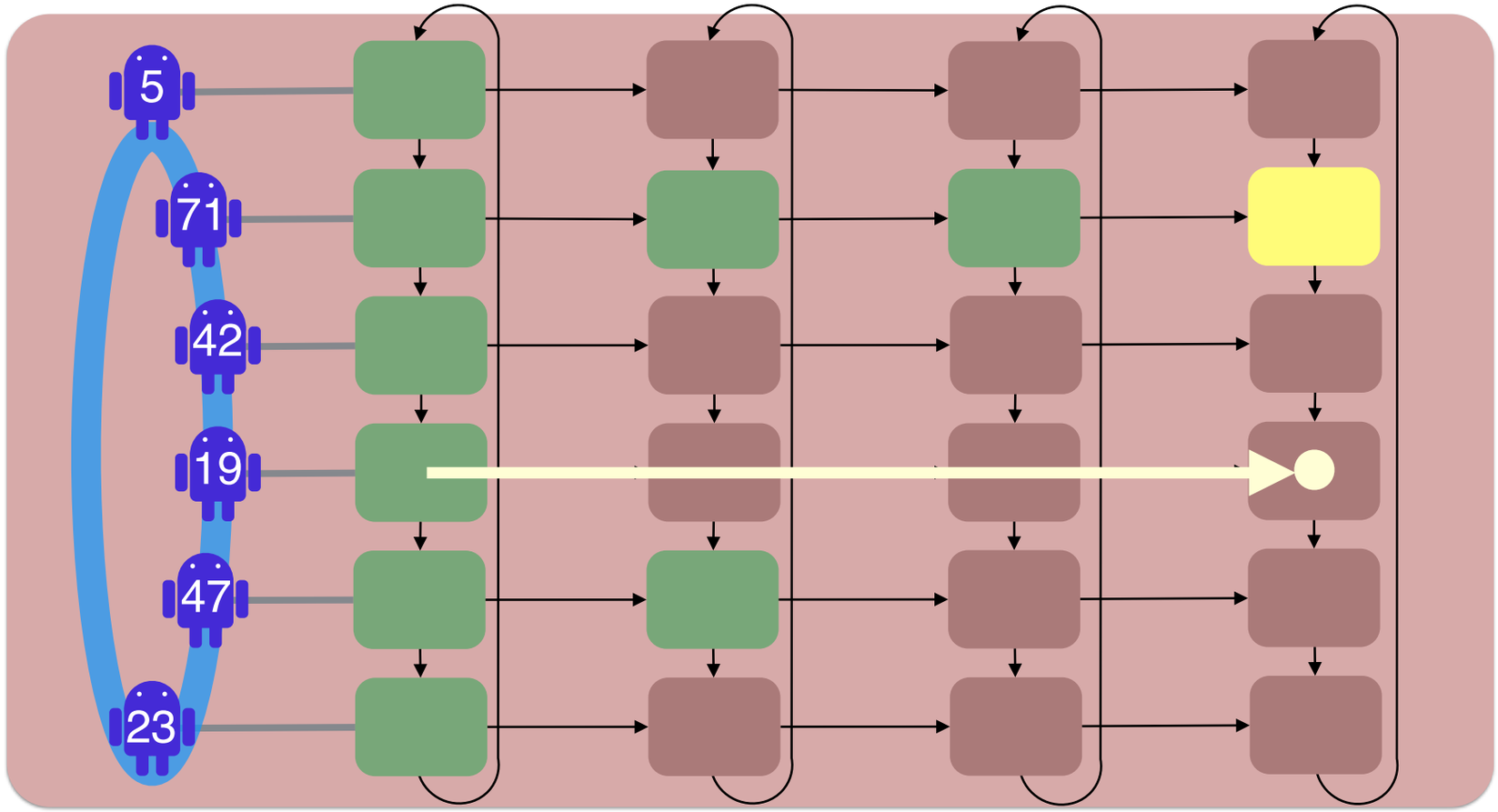
3 sources of infinity

Specification language



Distributed algorithms

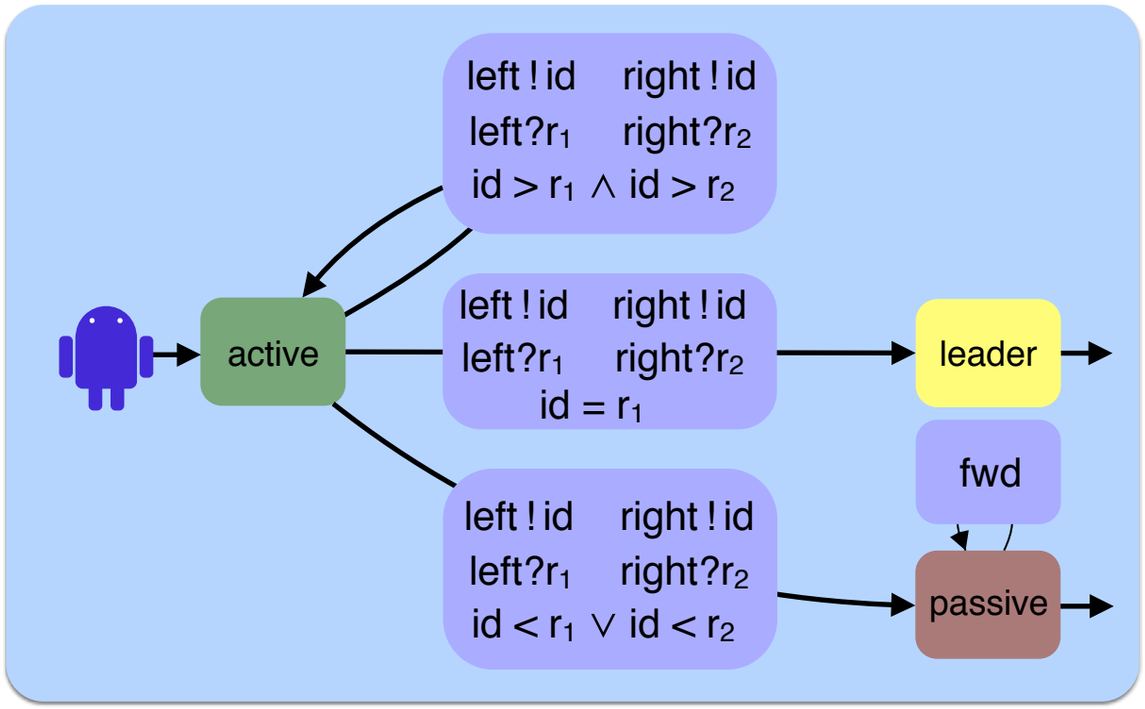
Leader election [Franklin ’82]



move in the cylinder

compare values at different nodes

Behavior



Distributed algorithm

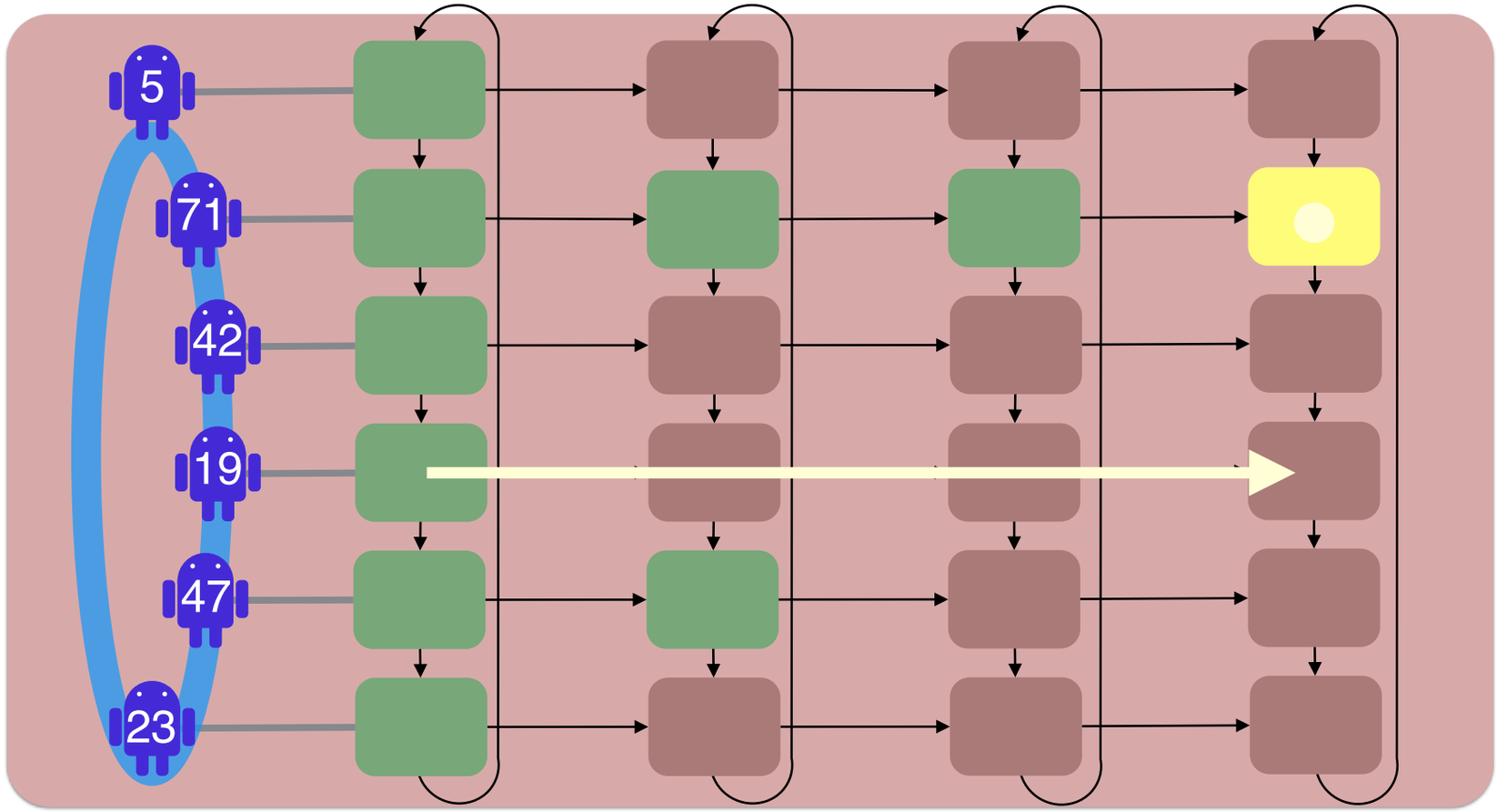
«At the end, there is a leader, and
the leader is the process with the maximum id.»

$$\langle \longrightarrow^* \rangle \left(\neg \langle \longrightarrow \rangle \wedge \langle \text{go-to-} \square \rangle \right.$$
$$\left. \wedge\ [\downarrow^*]\ (\mathbf{id} \leq \langle \text{go-to-} \square \rangle\, \mathbf{id}) \right)$$

Data Propositional Dynamic Logic
[Bojanczyk et al. ’09; Figueira-Segoufin ‘11]

Distributed algorithms

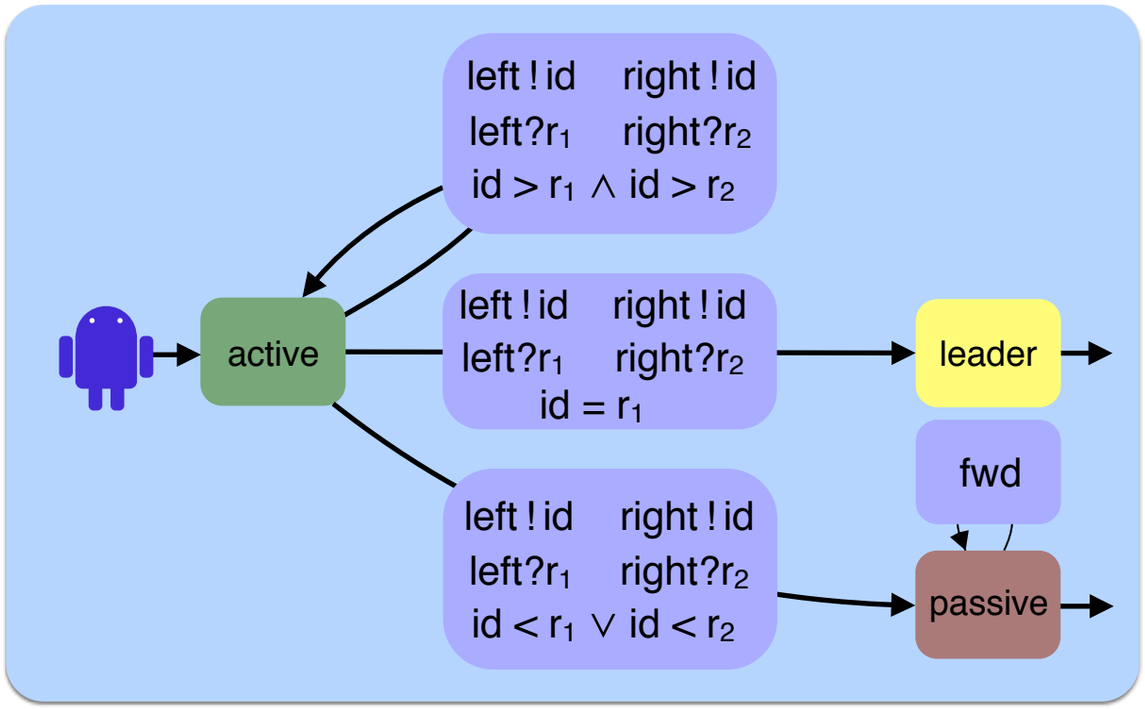
Leader election [Franklin '82]



move in the cylinder

compare values at different nodes

Behavior



Distributed algorithm

«At the end, there is a leader, and
the leader is the process with the maximum id.»

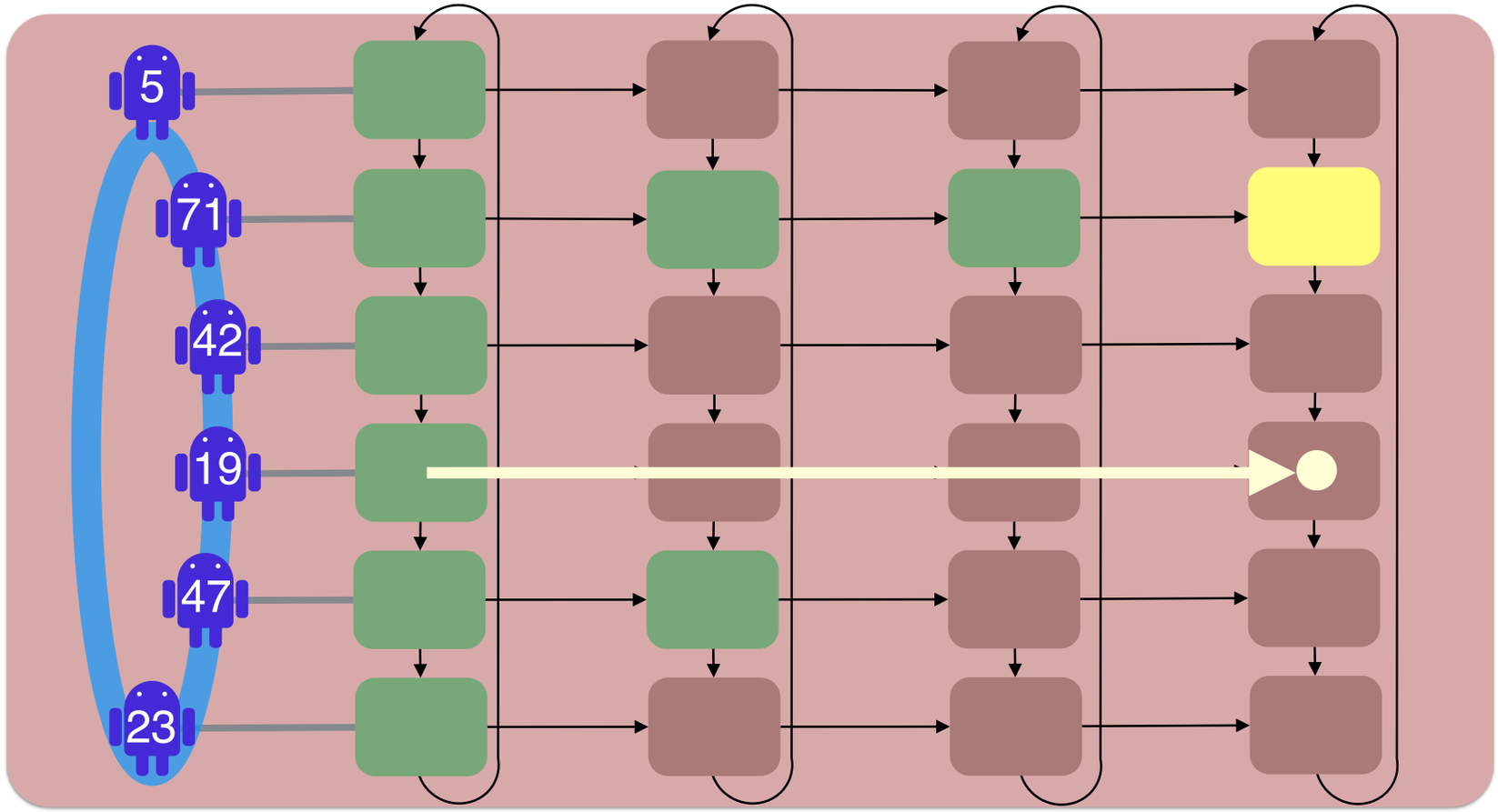
$$\langle \rightarrow^* \rangle \left(\neg\langle \rightarrow \rangle \wedge \langle \text{go-to-}\square \rangle \right.$$
$$\left. \wedge\ [\downarrow^*]\ (\mathbf{id} \leq \langle \text{go-to-}\square \rangle\, \mathbf{id}) \right)$$

$$\text{go-to-}\square \;=\; (\neg\, \square\, \downarrow)^*\ \square$$

Data Propositional Dynamic Logic
[Bojanczyk et al. '09; Figueira-Segoufin '11]

Distributed algorithms

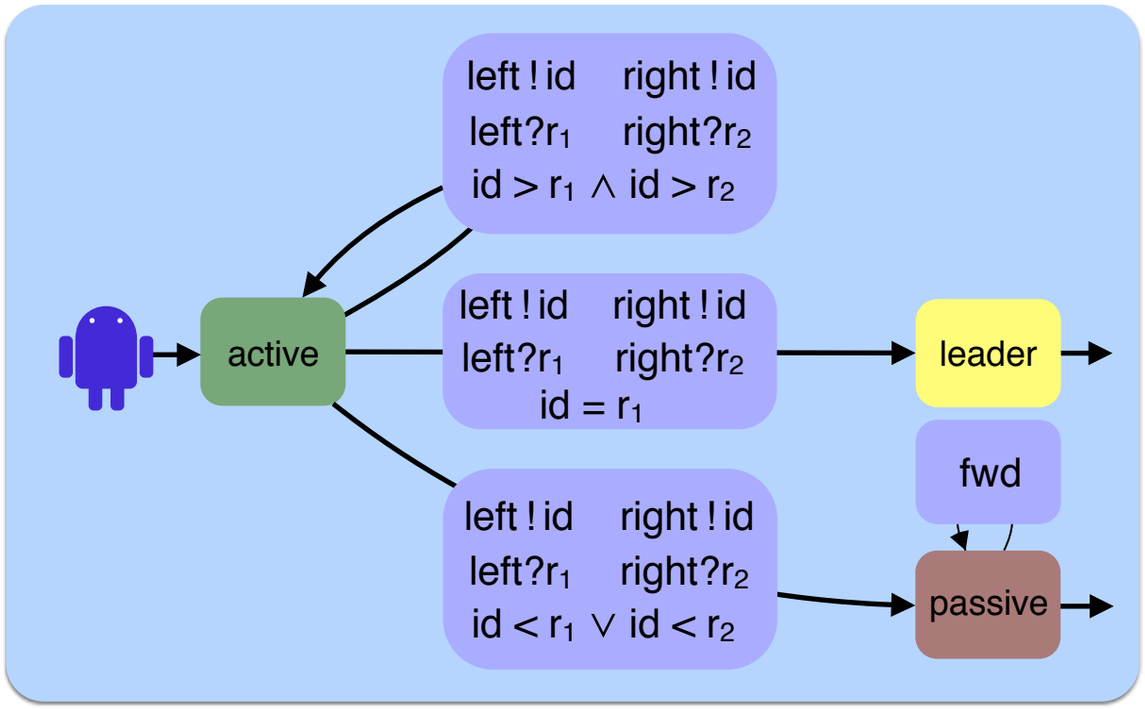
Leader election [Franklin '82]



move in the cylinder

compare values at different nodes

Behavior



Distributed algorithm

«At the end, there is a leader, and
the leader is the process with the maximum id.»

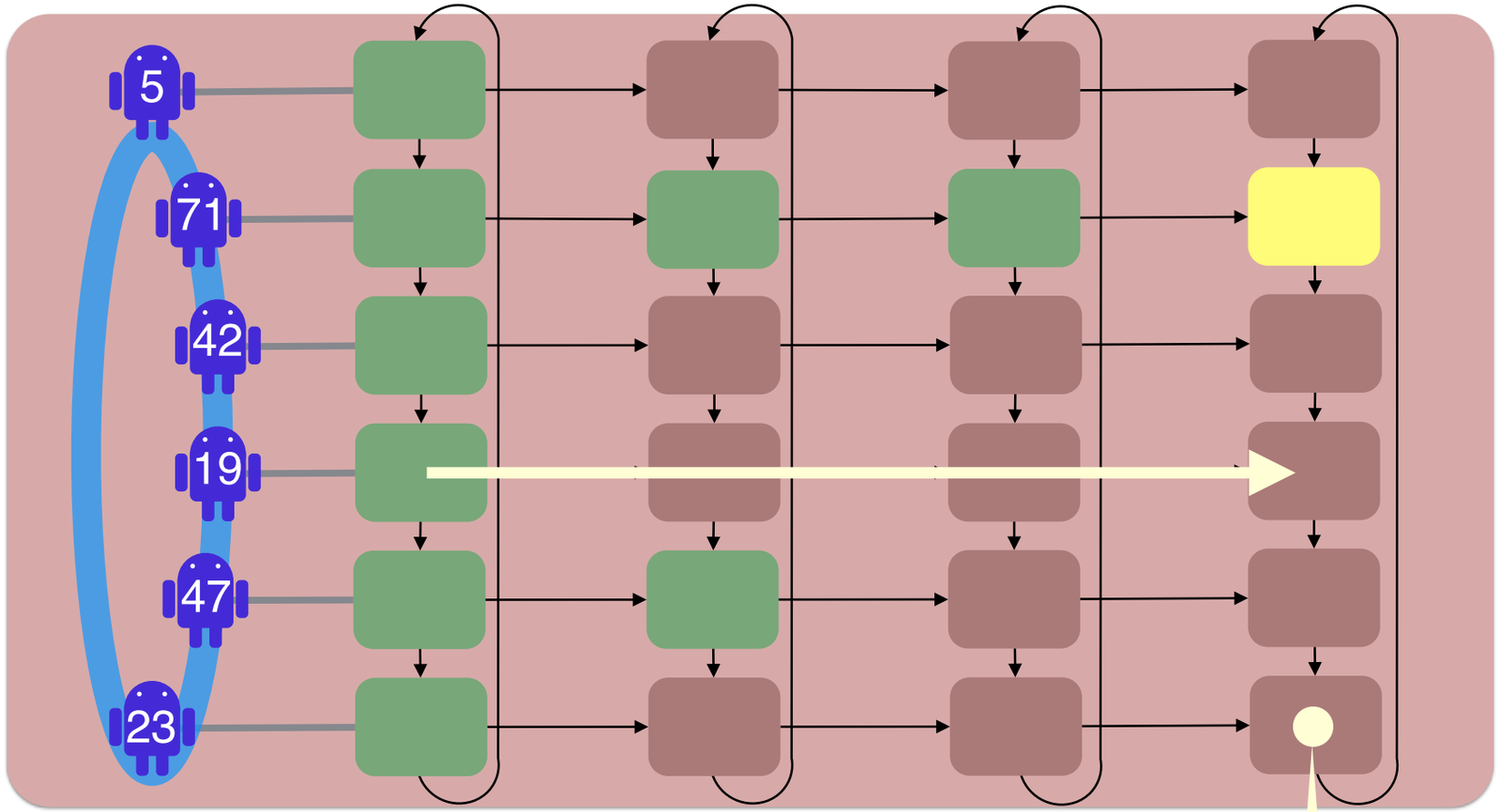
$$\langle \rightarrow^* \rangle \big(\quad \neg\langle \rightarrow \rangle \ \wedge \ \langle \text{go-to-}\square \rangle$$
$$\wedge \ [\downarrow^*] \ (\mathbf{id} \leq \langle \text{go-to-}\square \rangle \mathbf{id})\big)$$

$$\text{go-to-}\square \ = \ (\neg\, \square \downarrow)^* \ \square$$

Data Propositional Dynamic Logic
[Bojanczyk et al. '09; Figueira-Segoufin '11]

Distributed algorithms

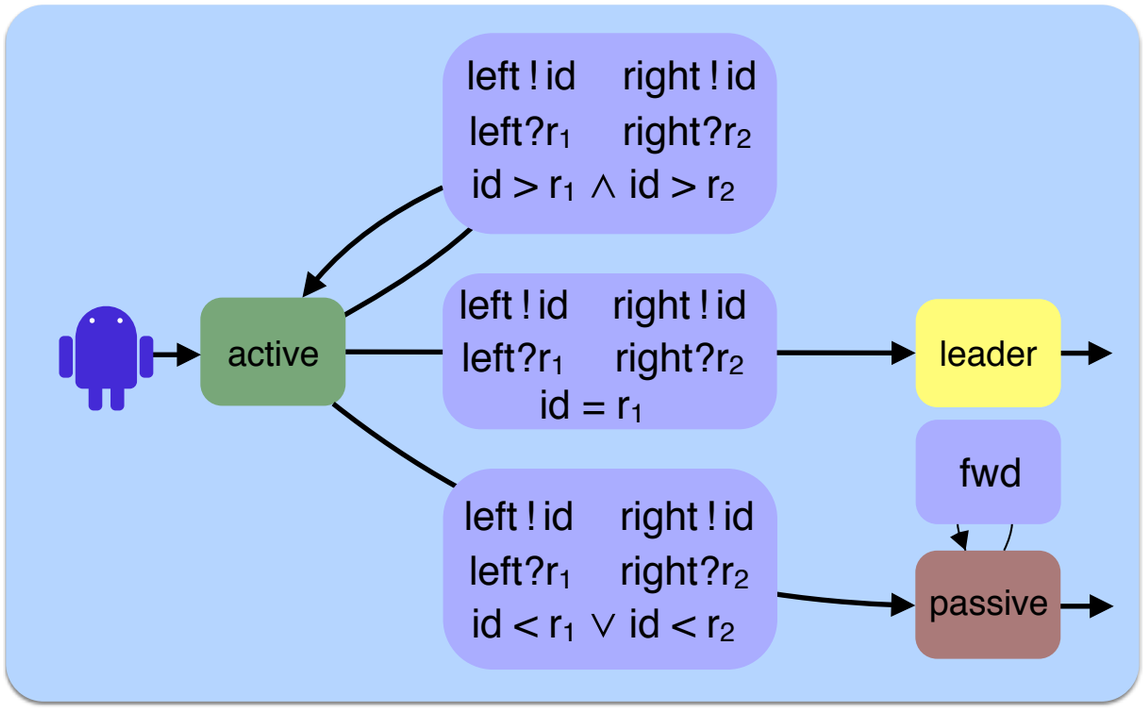
Leader election [Franklin '82]



move in the cylinder

compare values at different nodes

Behavior



Distributed algorithm

$\models$

«At the end, the is a leader, and
the leader is the proces with the maximum id.»

For all *n*, pid distributions, **accepting runs**, and processes:

$$\langle \rightarrow^* \rangle \left(\quad \neg \langle \rightarrow \rangle \quad \langle \text{go-to-} \square \rangle \right.$$
$$\left. \wedge\ [\downarrow^*]\ (\mathbf{id} \leq \langle \text{go-to-} \square \rangle\, \mathbf{id}) \right)$$

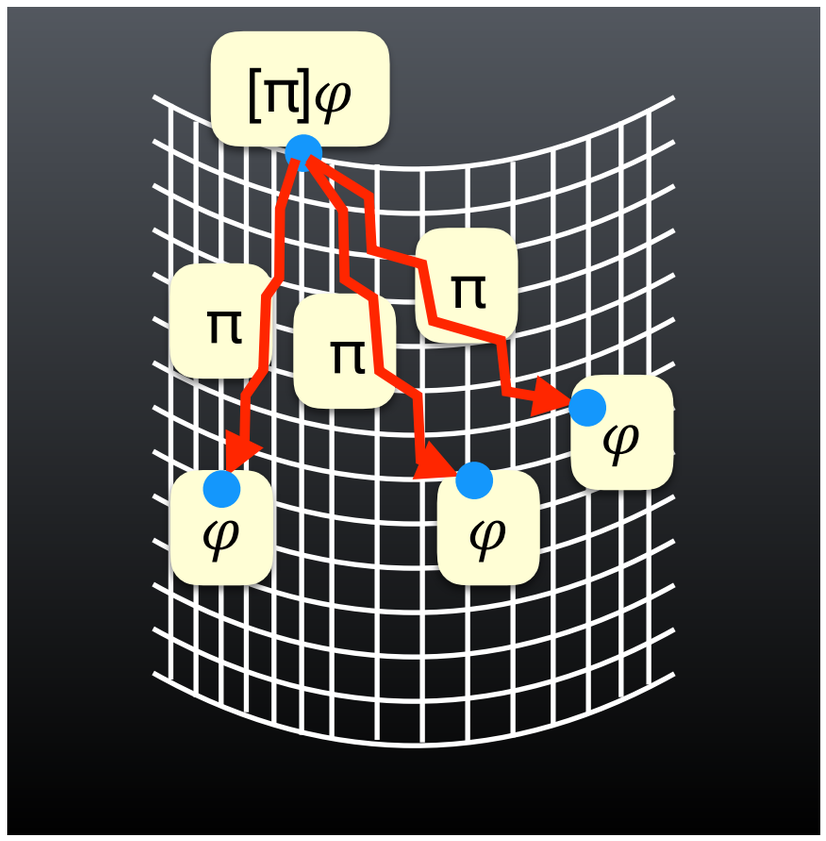
$$\text{go-to-}\square \;=\; (\neg\, \square\, \downarrow)^*\, \square$$

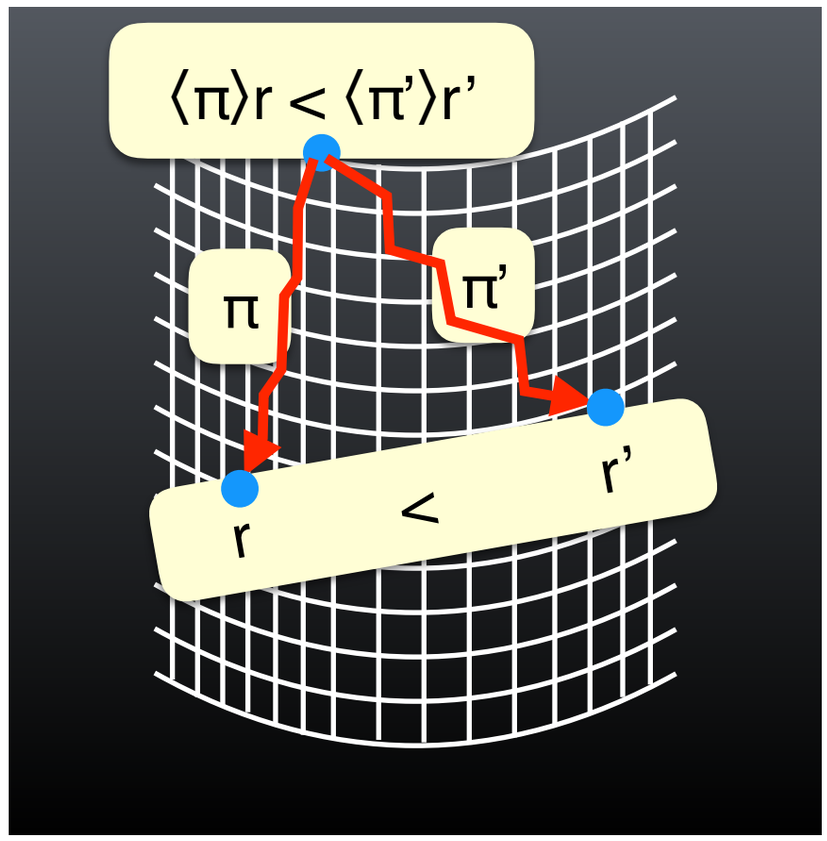
Data Propositional Dynamic Logic
[Bojanczyk et al. '09; Figueira-Segoufin '11]

Specifications
Data PDL

move in the cylinder

compare values at different nodes





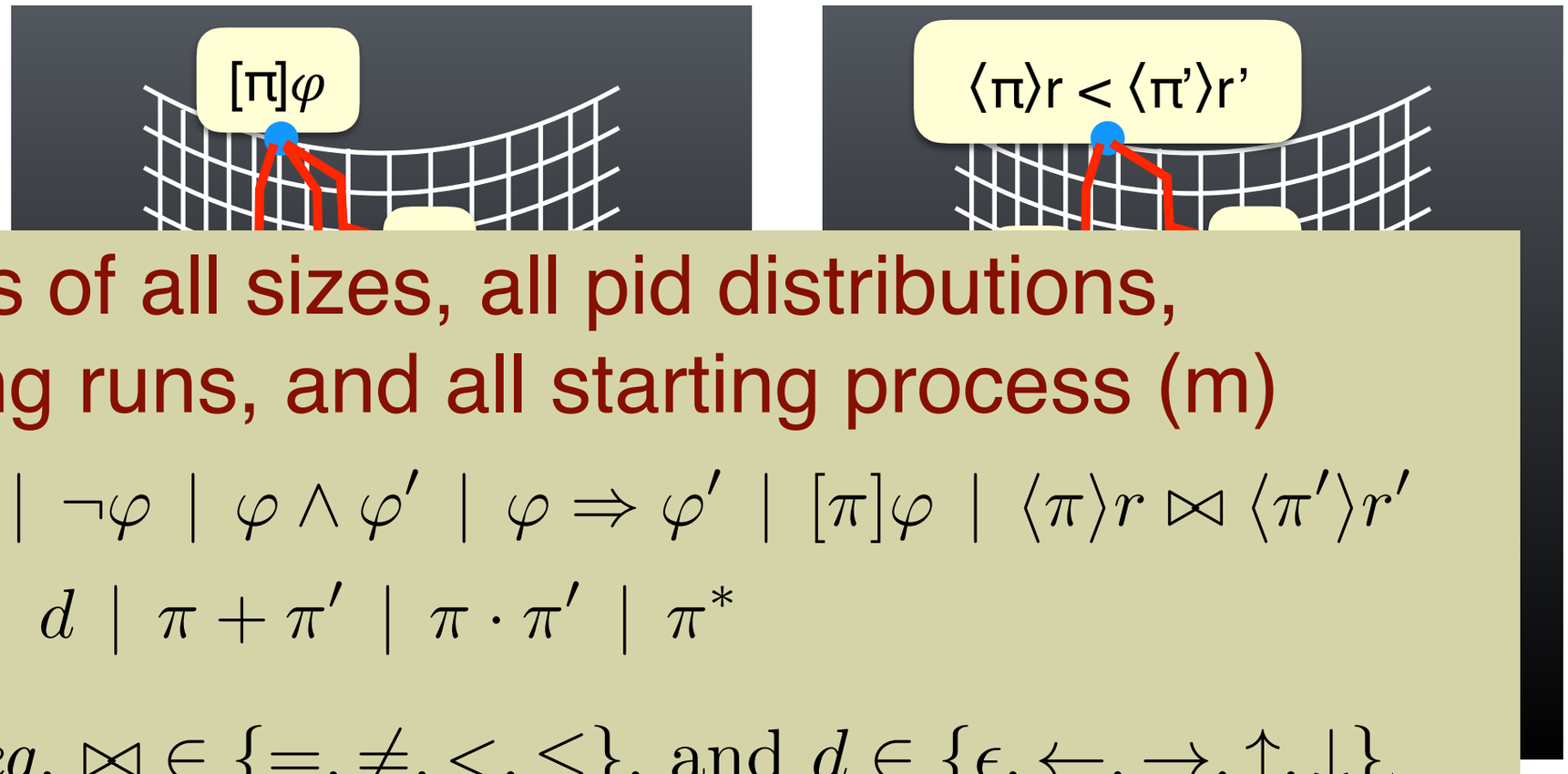
Path π = rational expression over directions

Inspired by [Bojanczyk et al. '09; Figueira-Segoufin '11]

Specifications Data PDL

move in the cylinder

compare values at different nodes



For rings of all sizes, all pid distributions, all accepting runs, and all starting process (m)

$$\varphi, \varphi' ::= \mathsf{m} \mid s \mid \neg\varphi \mid \varphi \wedge \varphi' \mid \varphi \Rightarrow \varphi' \mid [\pi]\varphi \mid \langle\pi\rangle r \bowtie \langle\pi'\rangle r'$$

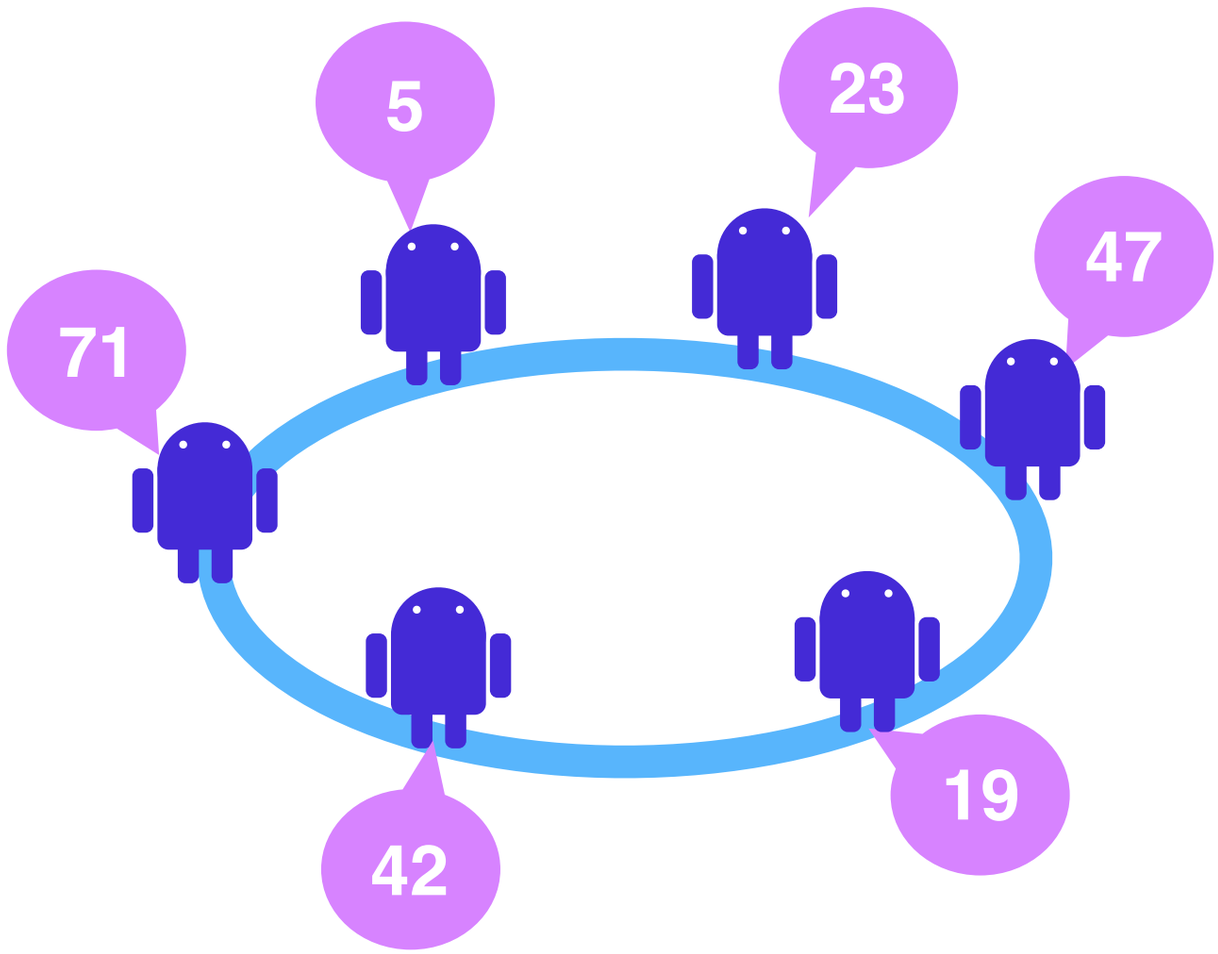
$$\pi, \pi' ::= \{\varphi\}? \mid d \mid \pi + \pi' \mid \pi \cdot \pi' \mid \pi^*$$

$$s \in S,\ r, r' \in Reg,\ \bowtie \in \{=, \neq, <, \leq\},\ \text{and}\ d \in \{\epsilon, \leftarrow, \rightarrow, \uparrow, \downarrow\}$$

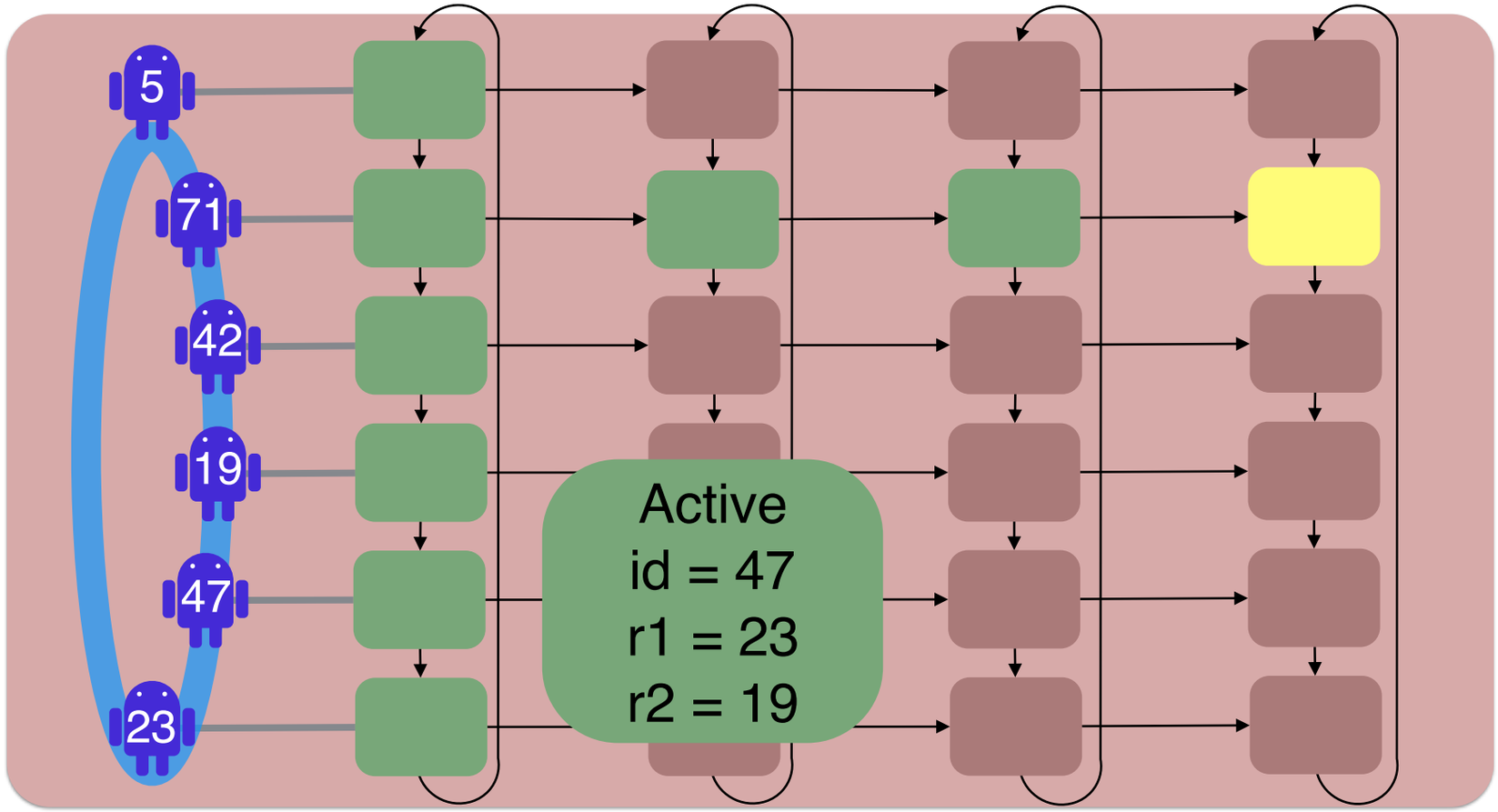
Path π = rational expression over directions

Inspired by [Bojanczyk et al. '09; Figueira-Segoufin '11]

Model Checking 1



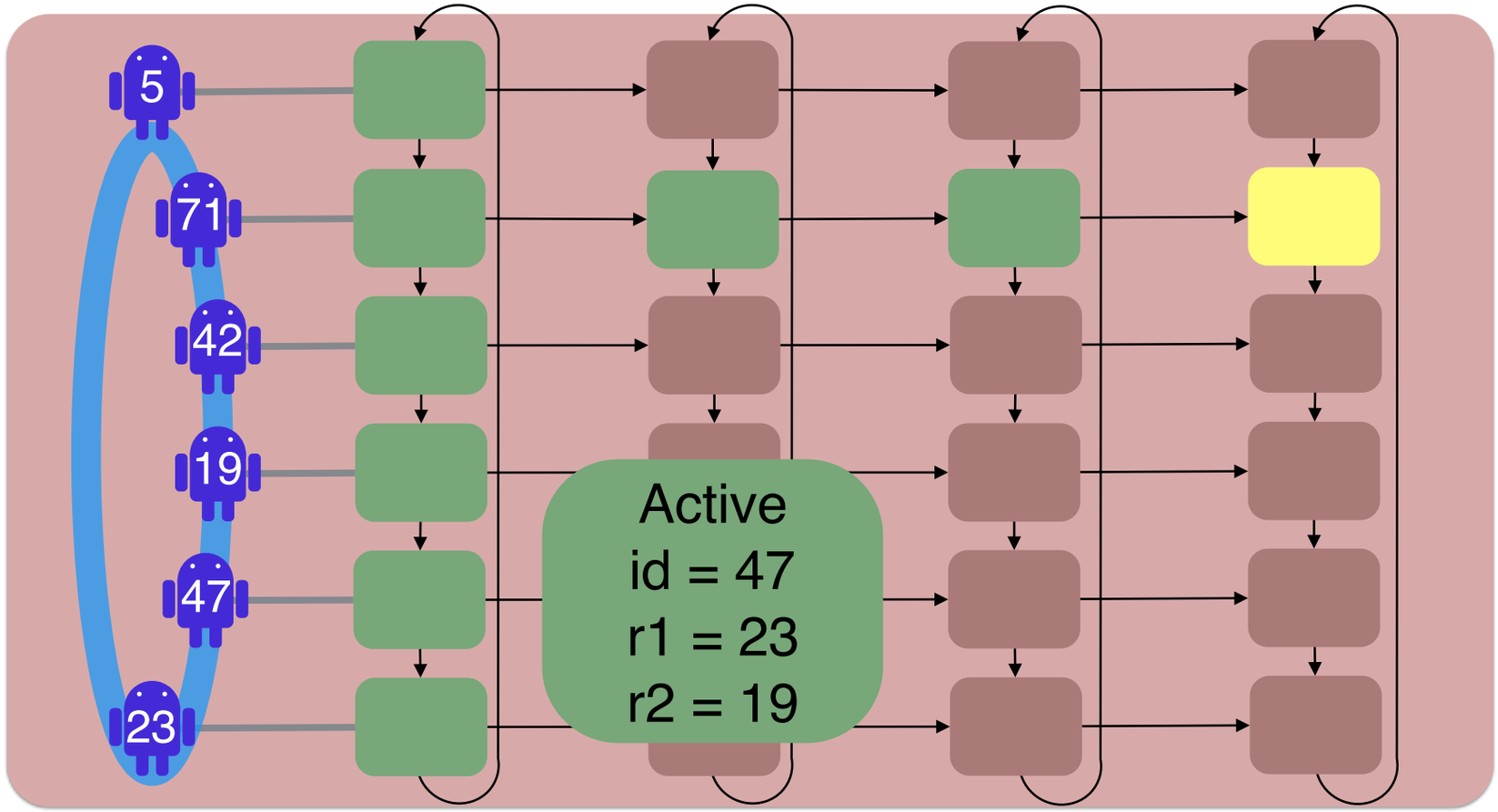
Model Checking Distributed algorithms



UNDECIDABLE

- Behaviors: Cylinders of arbitrary width and length
  Data from an infinite domain
- System: Register automata with data comparisons
- Specification: Data PDL with data comparisons

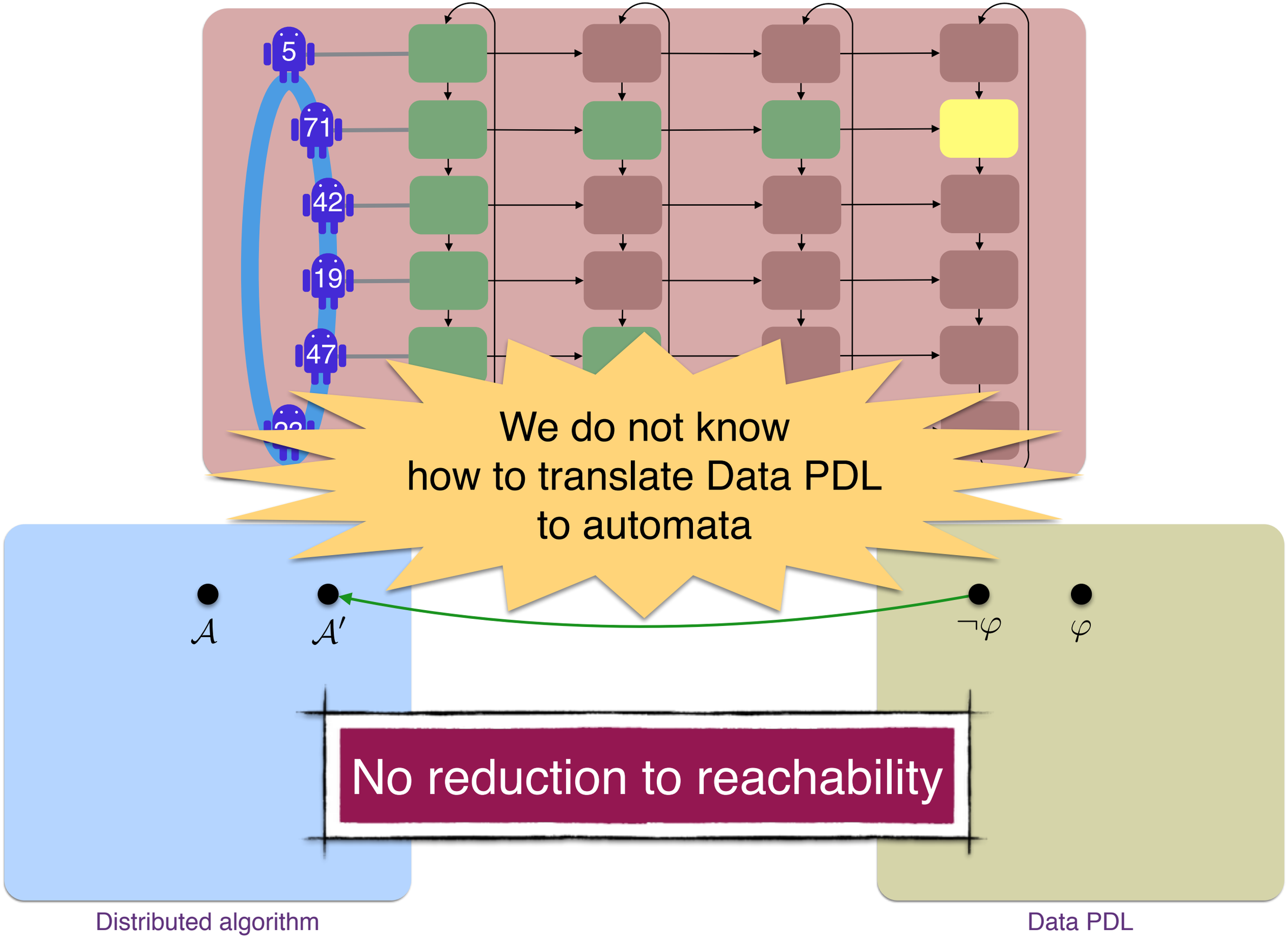
Model Checking Distributed algorithms



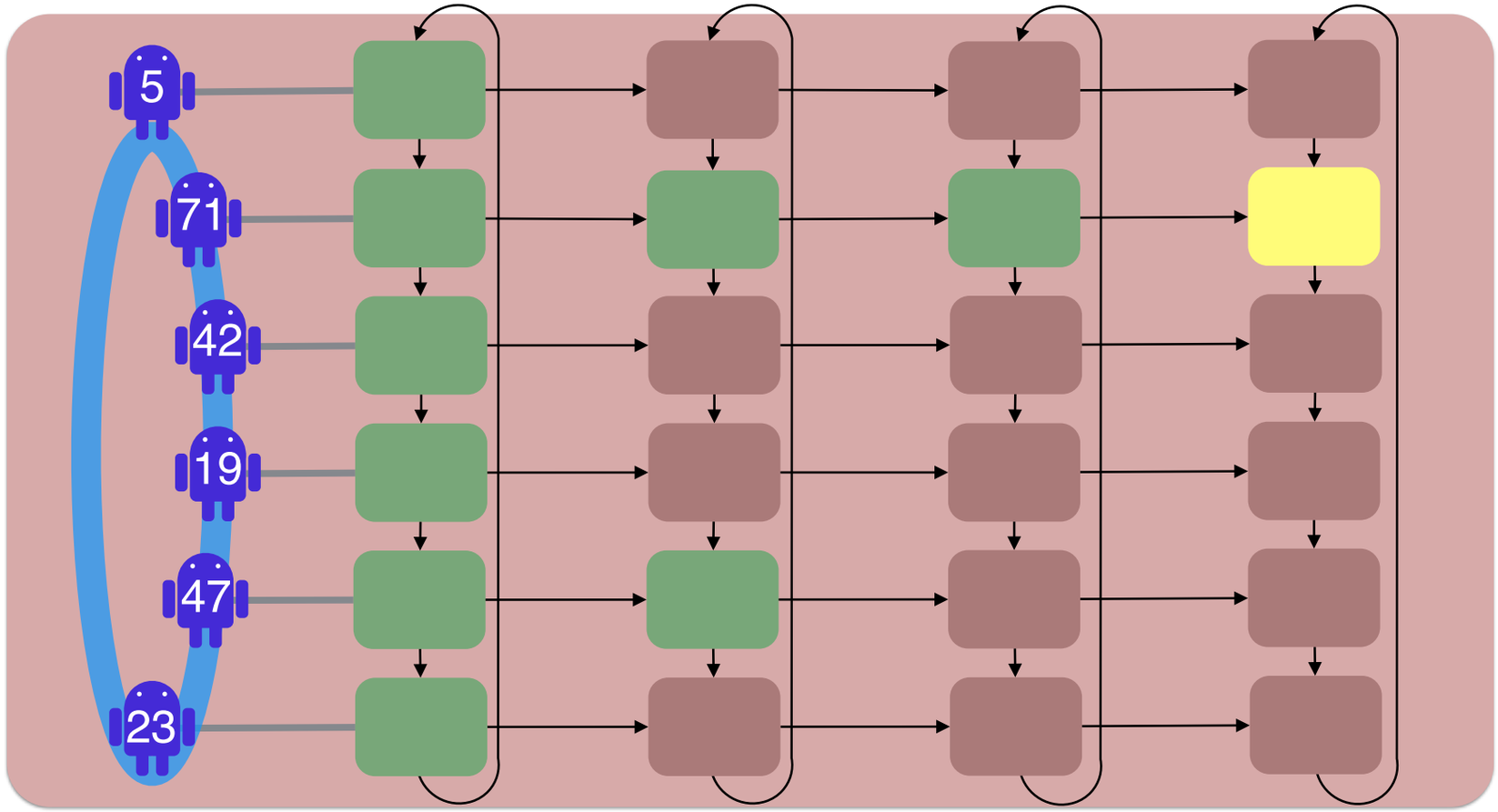
UNDECIDABLE

- Behaviors: Cylinders of arbitrary width and length
  Data from an infinite domain
- System: Register automata with data comparisons
- Specification: Data PDL with data comparisons

Reduction to automata?



Reduction to Satisfiability of LCPDL: Data abstraction



$\alpha \wedge \neg\beta$

unsatisfiable over grids

$\mathcal{A}$

φ

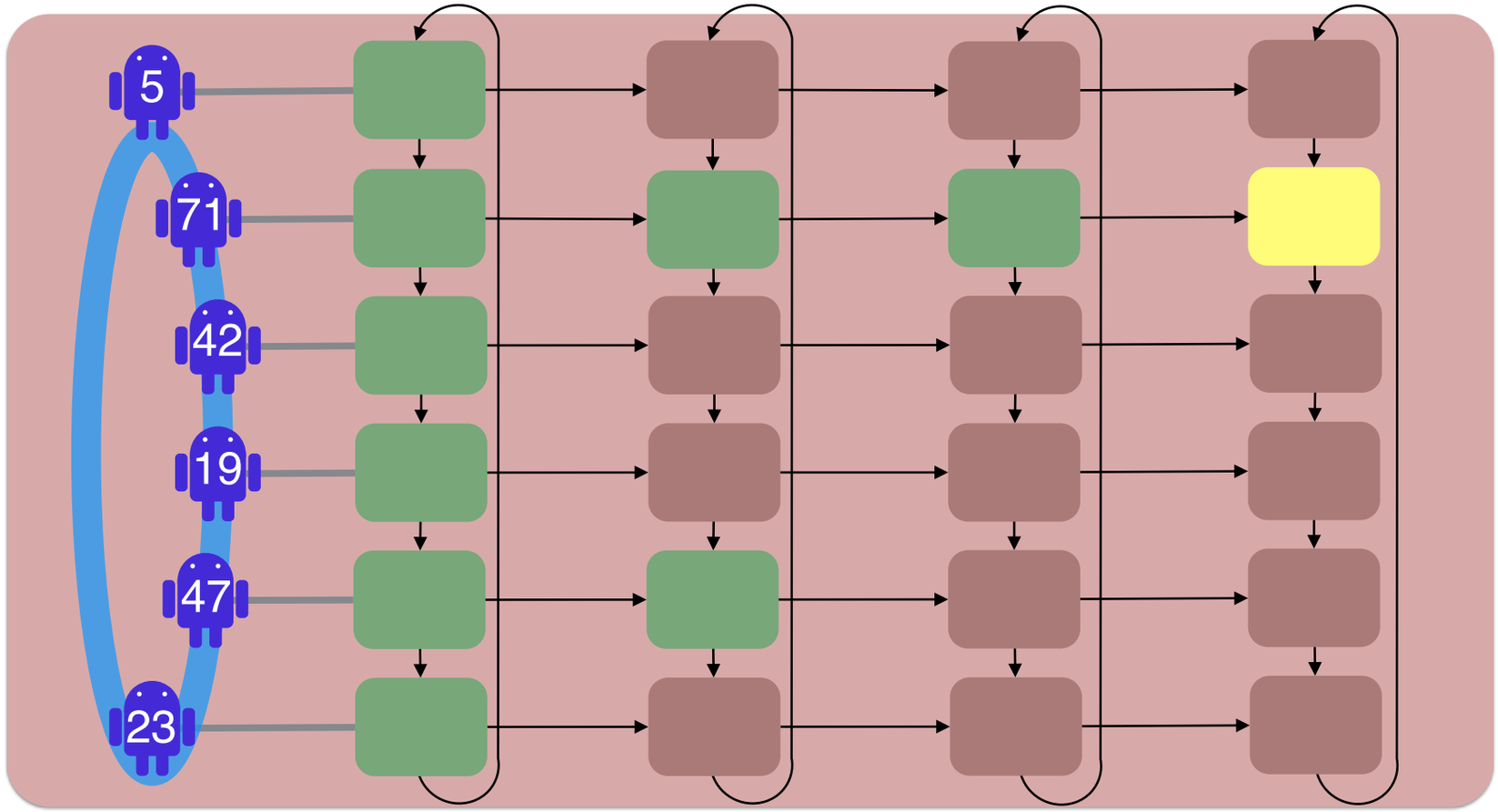
$\Longleftrightarrow \quad \mathcal{A} \models \varphi$

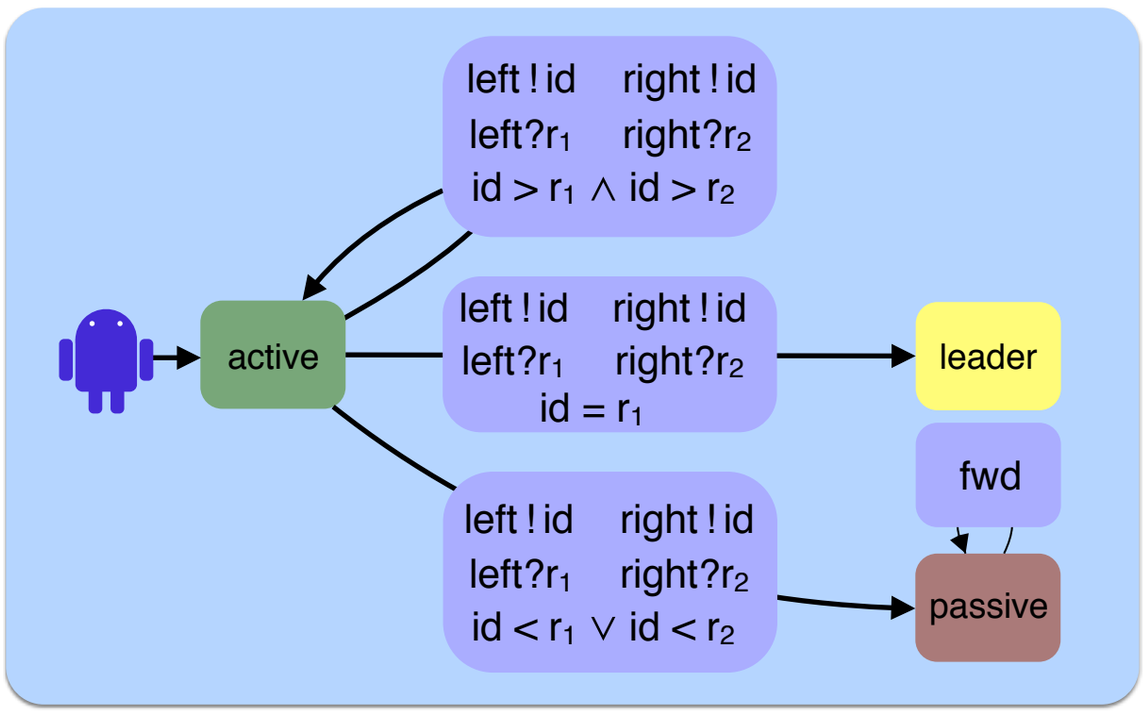
Distributed algorithm

PDL with loop (over finite alphabet)

Data PDL

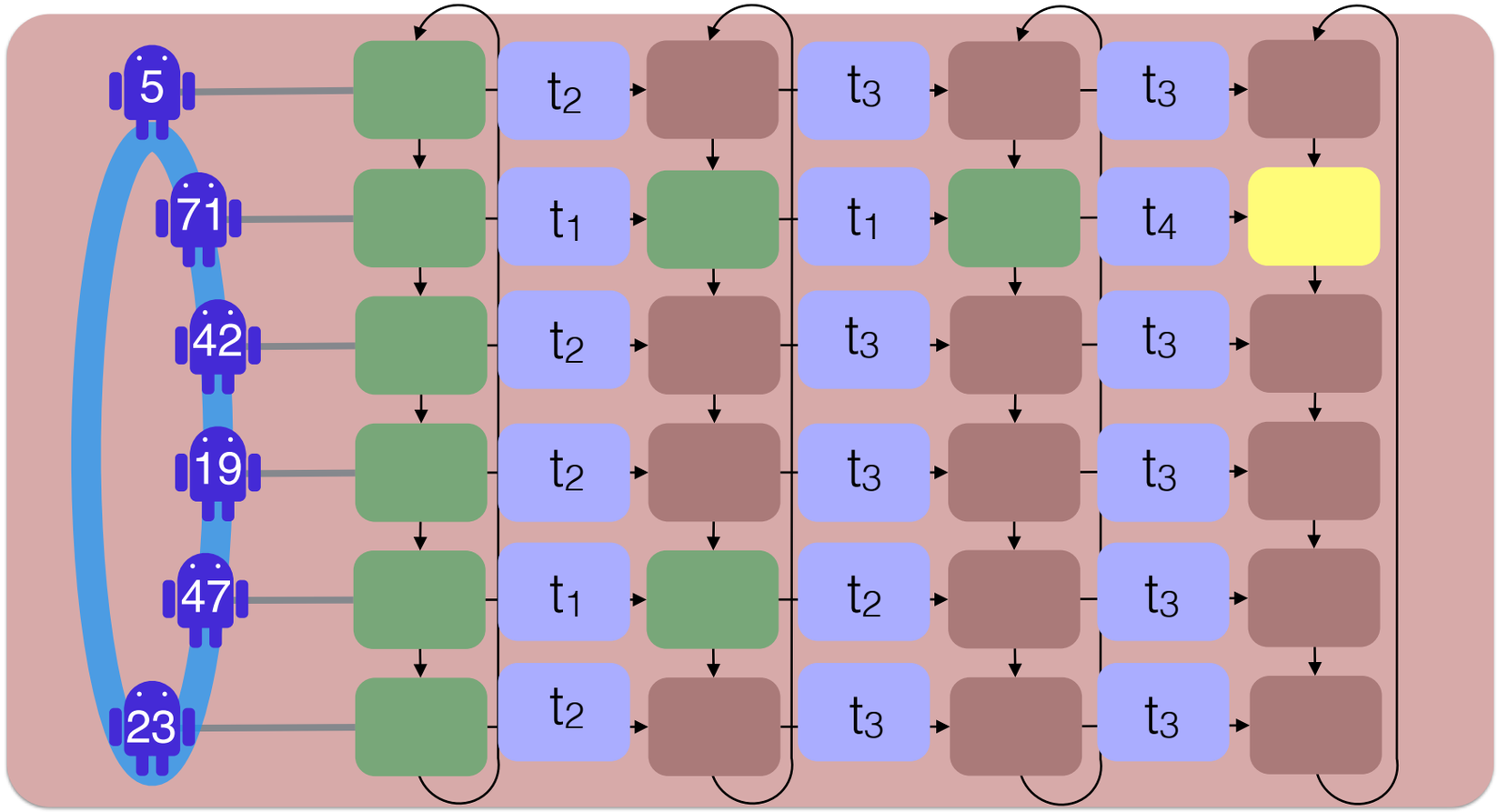
Data abstraction: symbolic runs + tracking data

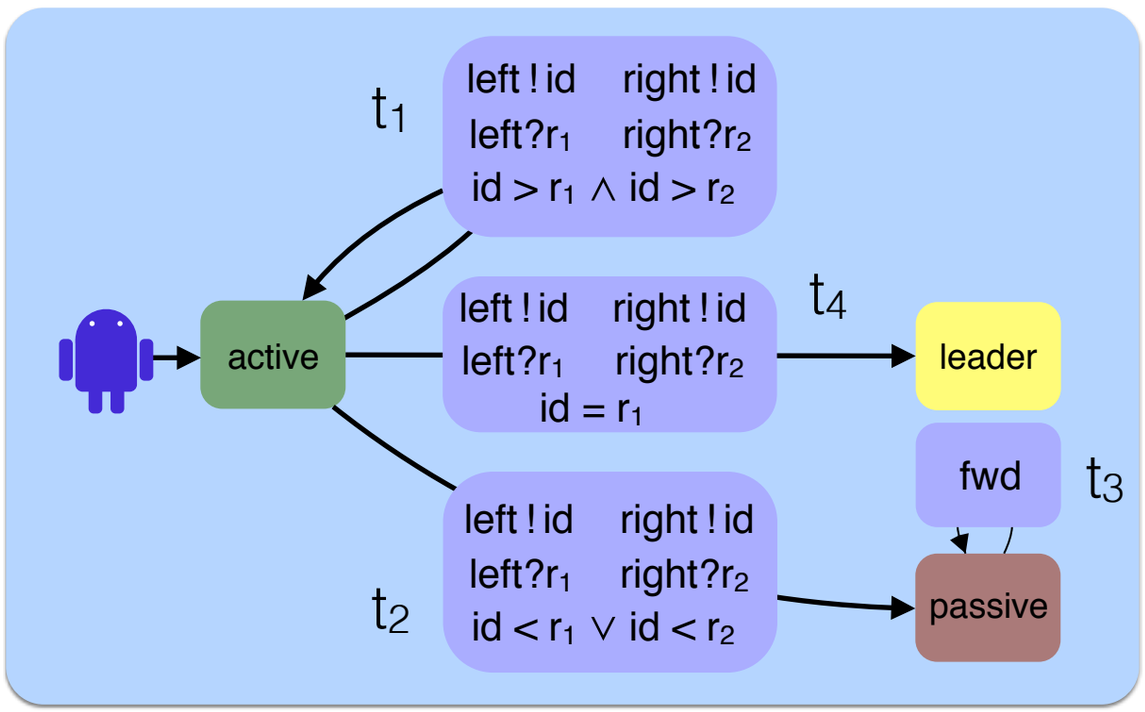




Distributed algorithm

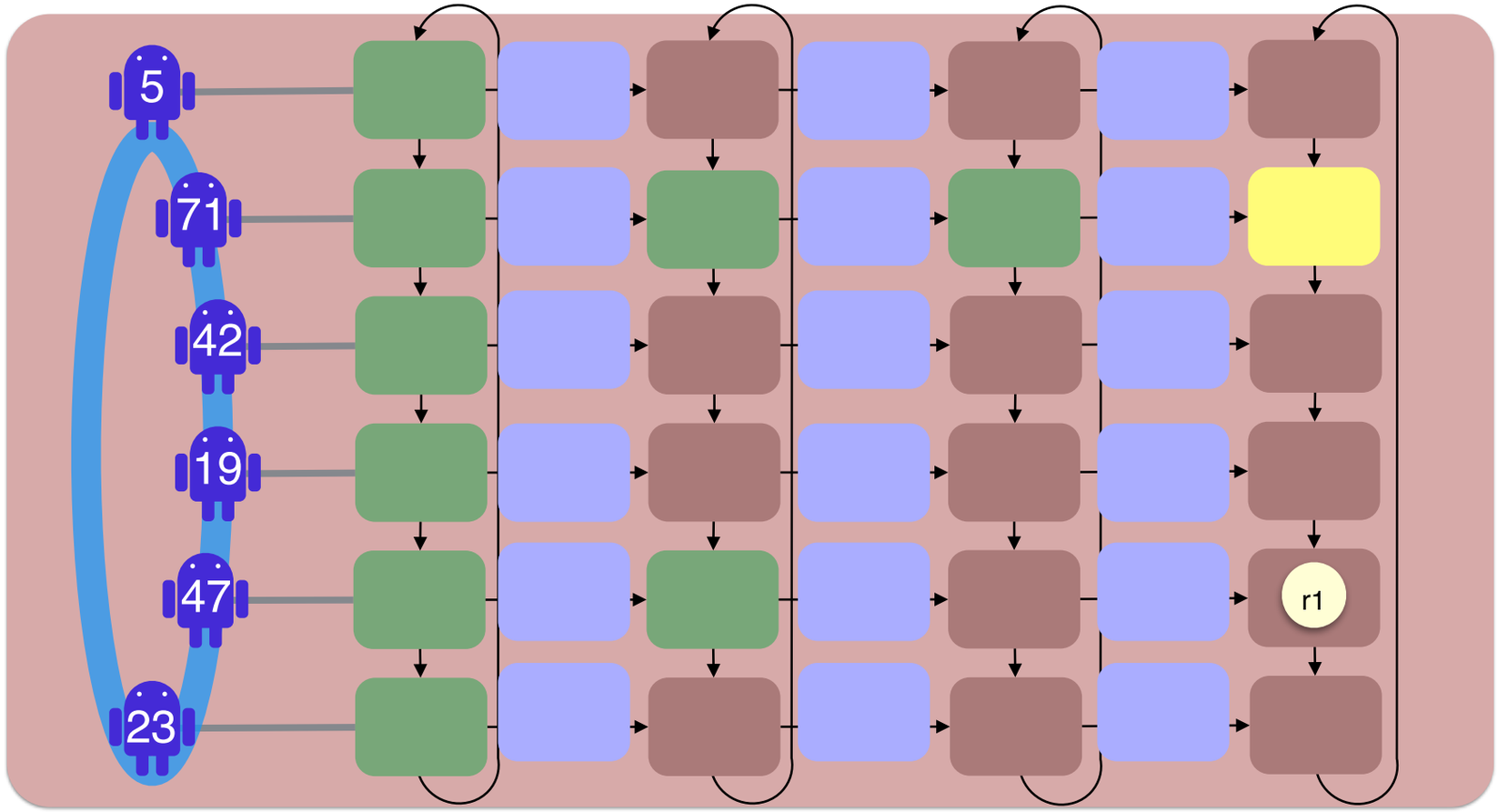
Data abstraction: symbolic runs + tracking data





Distributed algorithm

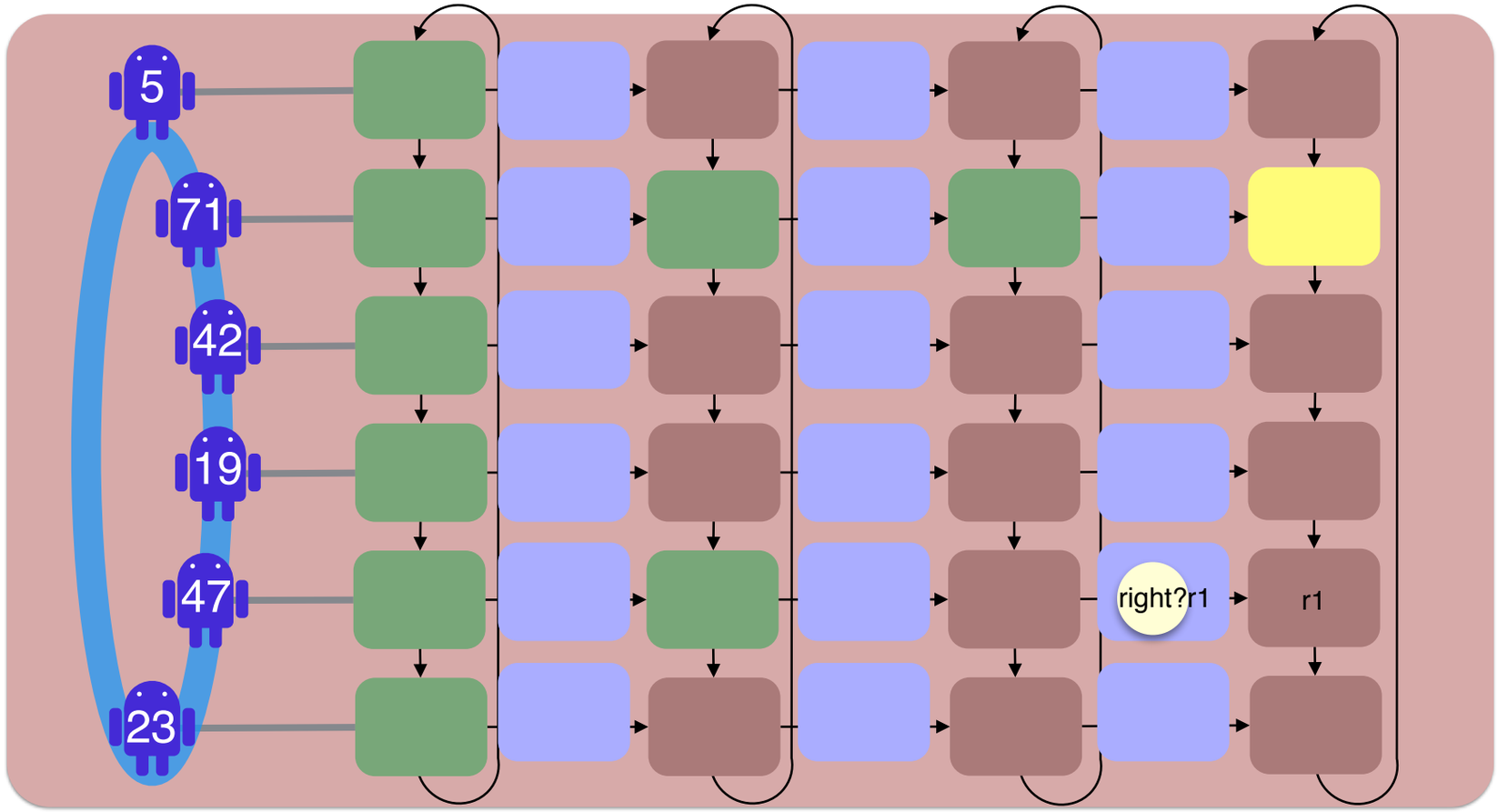
Data abstraction: symbolic runs + tracking data



- Register updates

Distributed algorithm

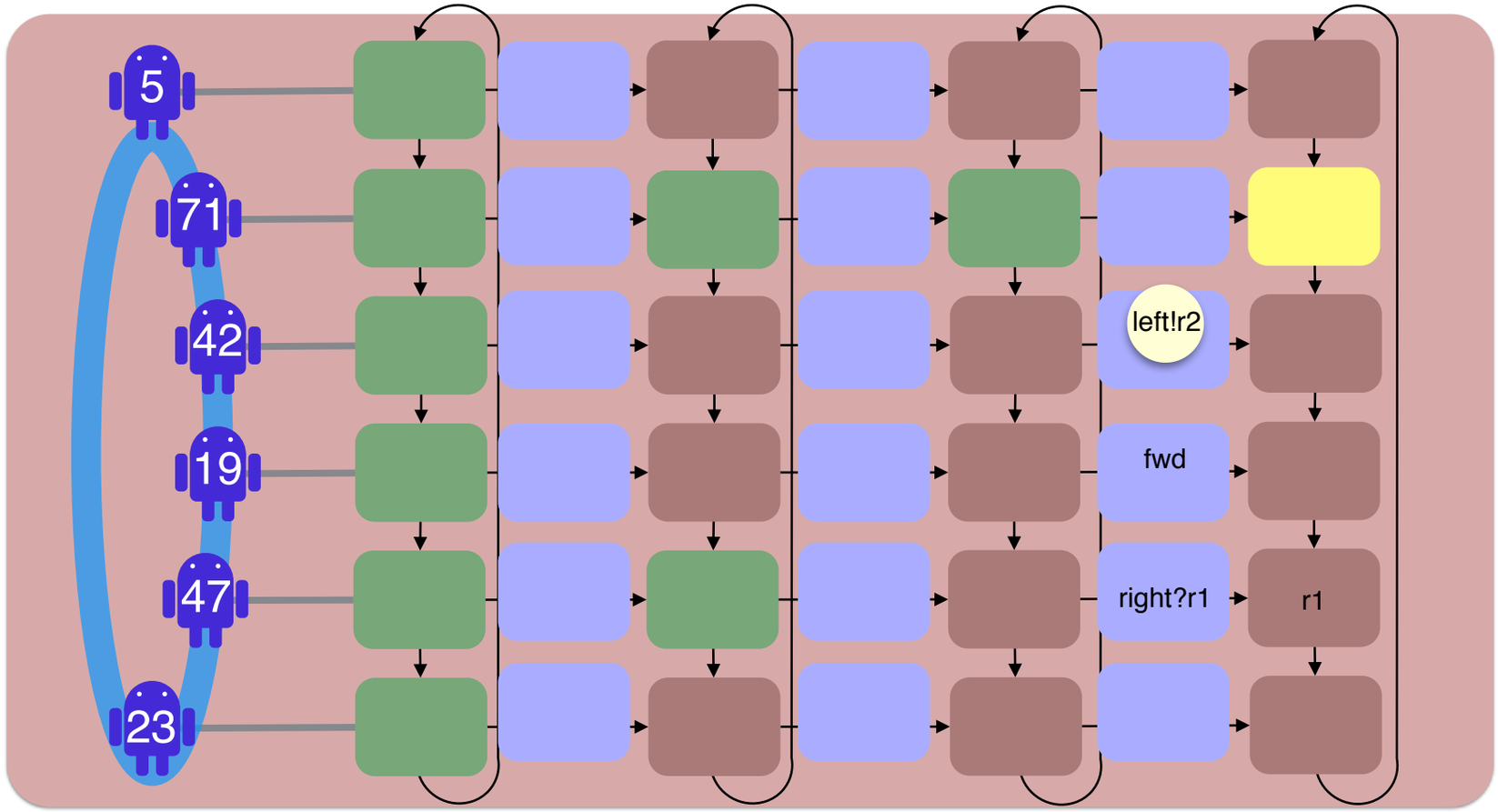
Data abstraction: symbolic runs + tracking data



- Register updates

Distributed algorithm

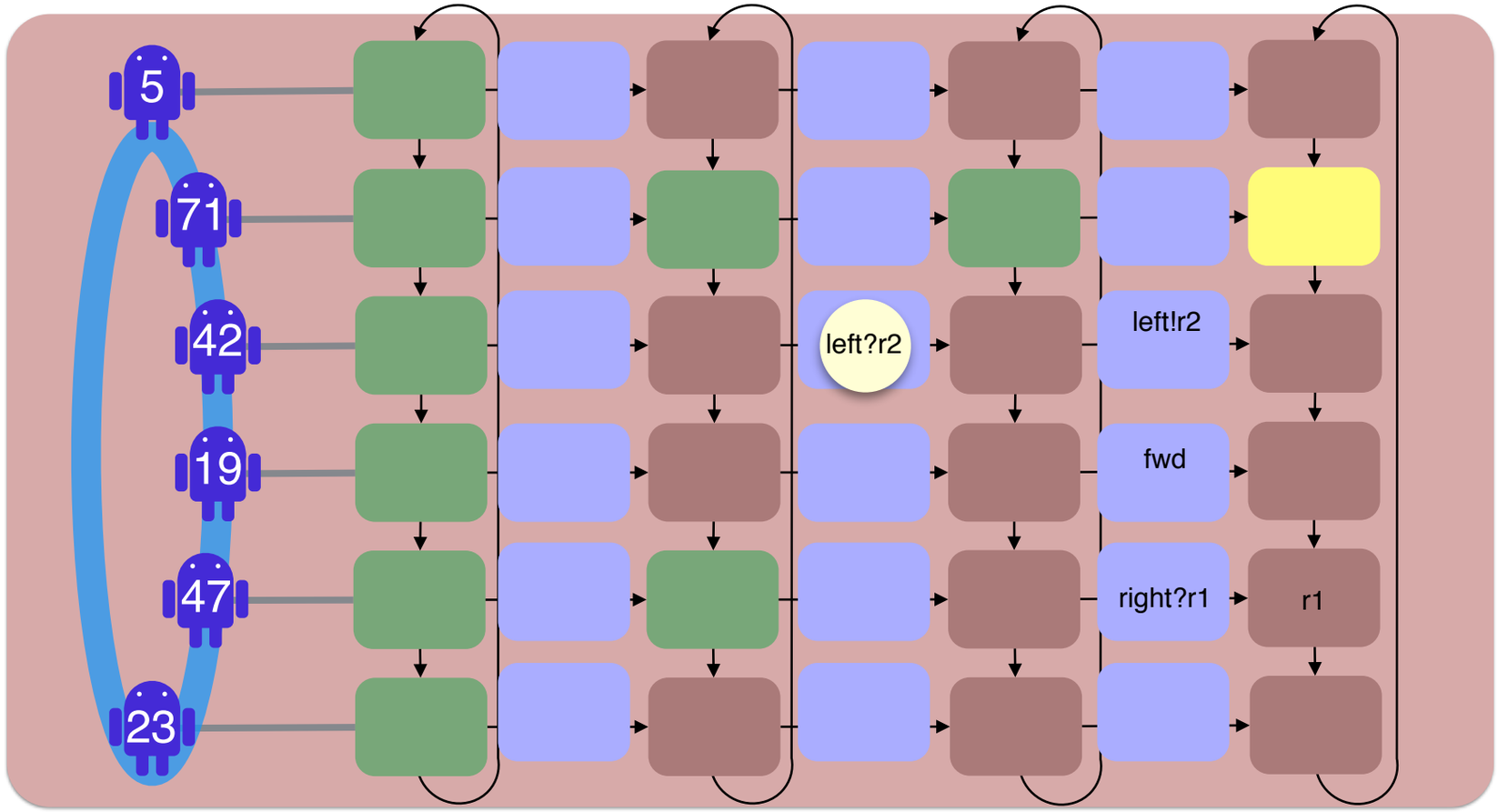
Data abstraction: symbolic runs + tracking data



- Register updates

Distributed algorithm

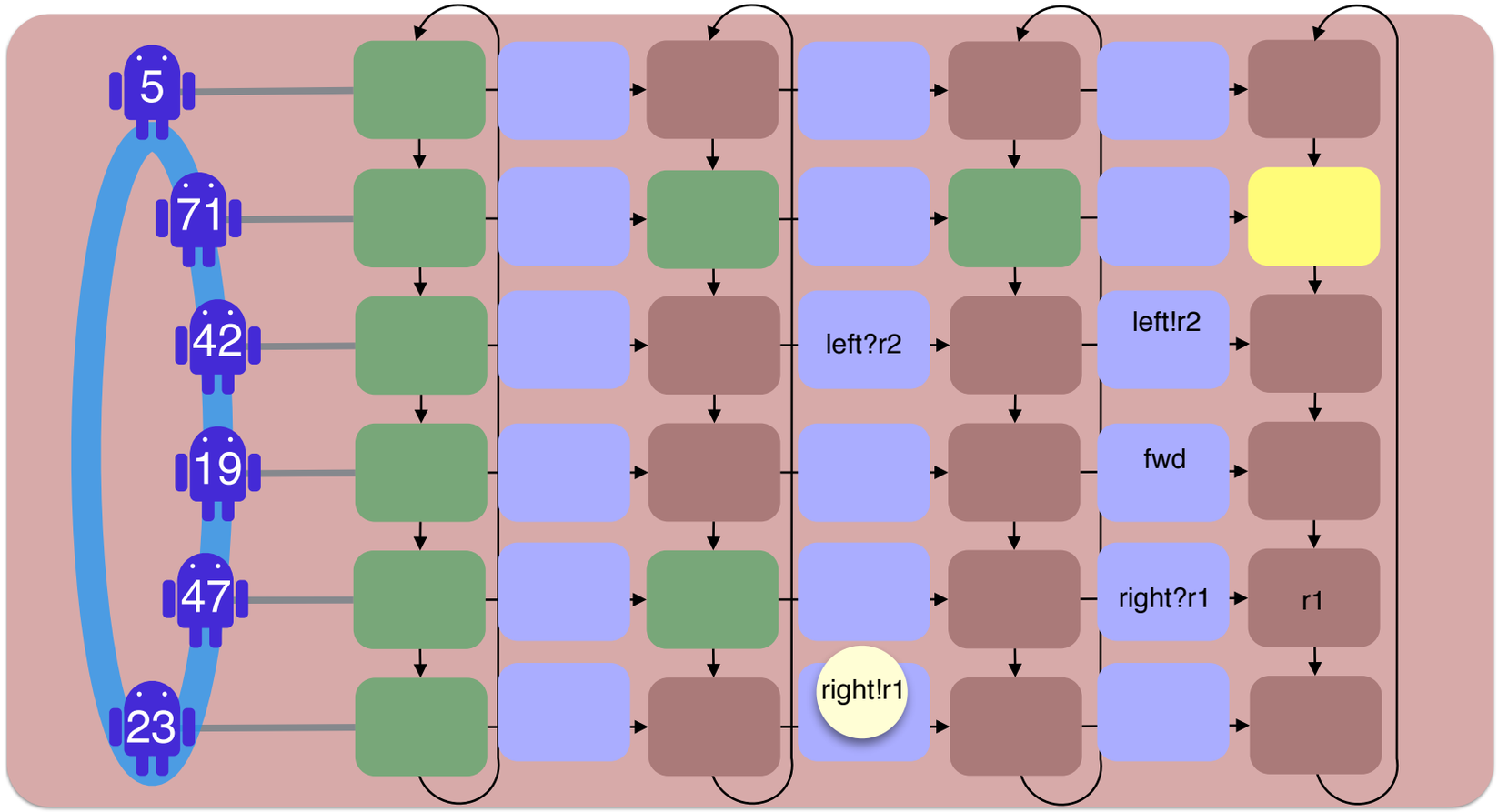
Data abstraction: symbolic runs + tracking data



- Register updates

Distributed algorithm

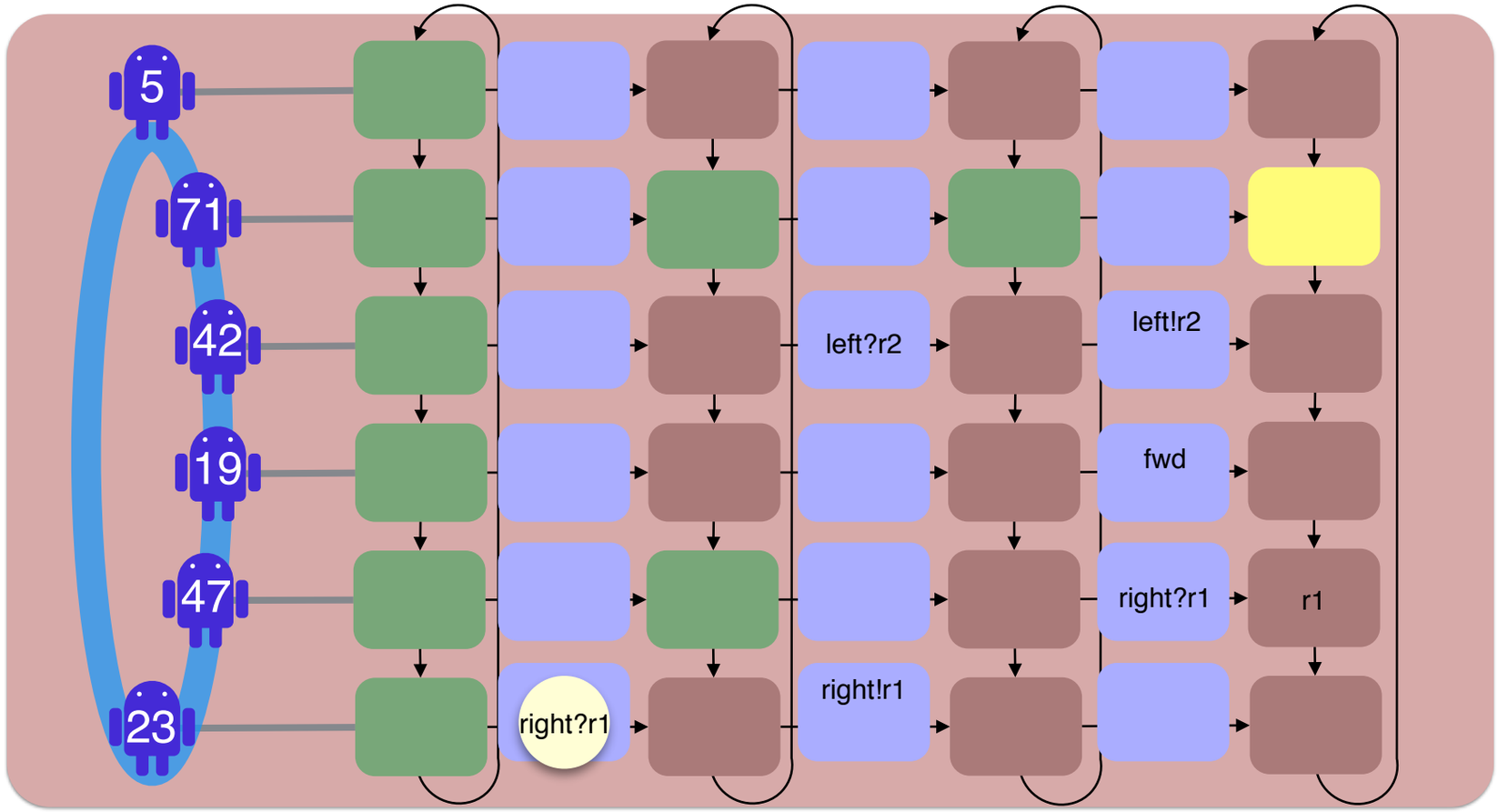
Data abstraction: symbolic runs + tracking data



- Register updates

Distributed algorithm

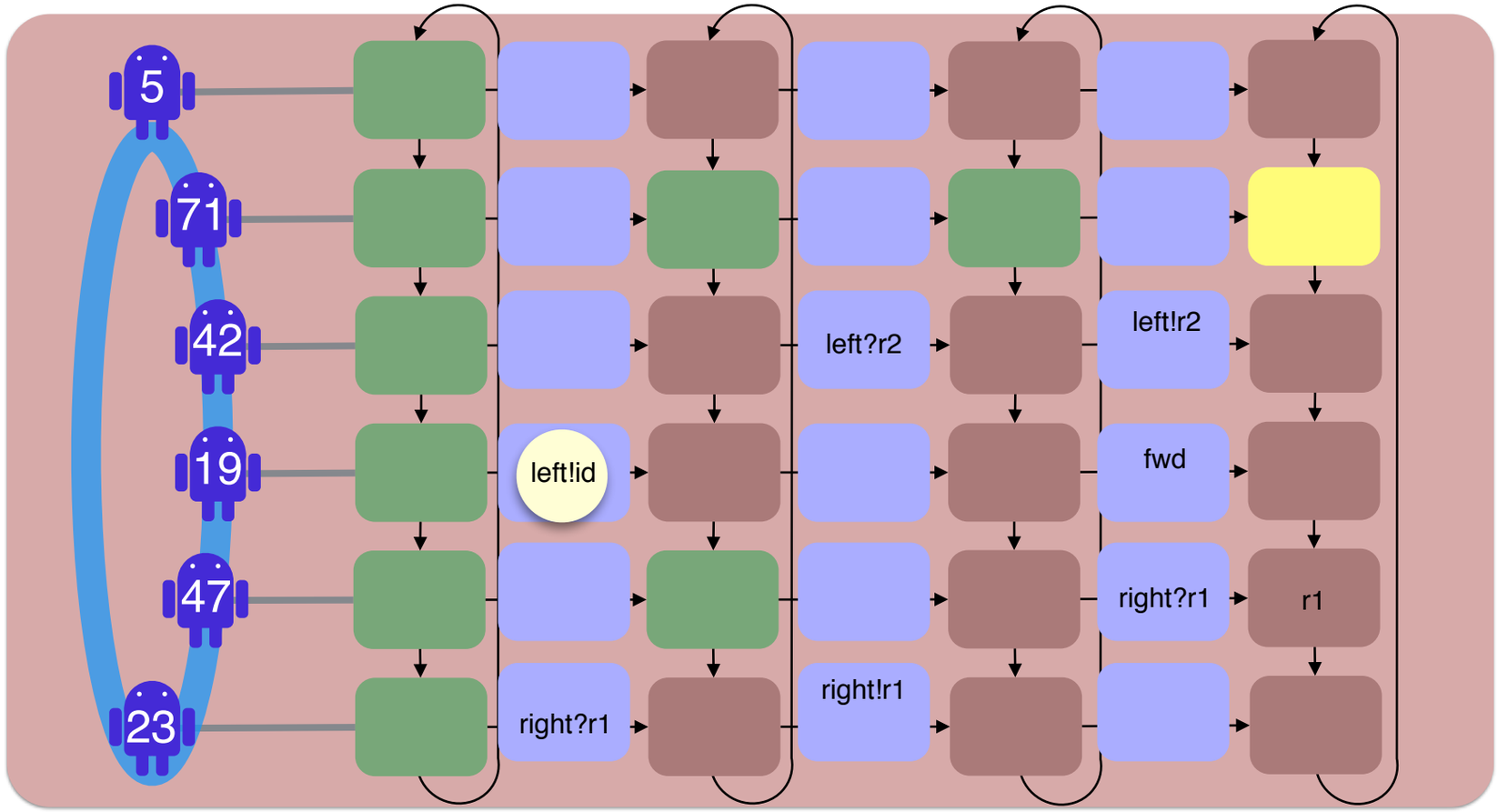
Data abstraction: symbolic runs + tracking data



- Register updates

Distributed algorithm

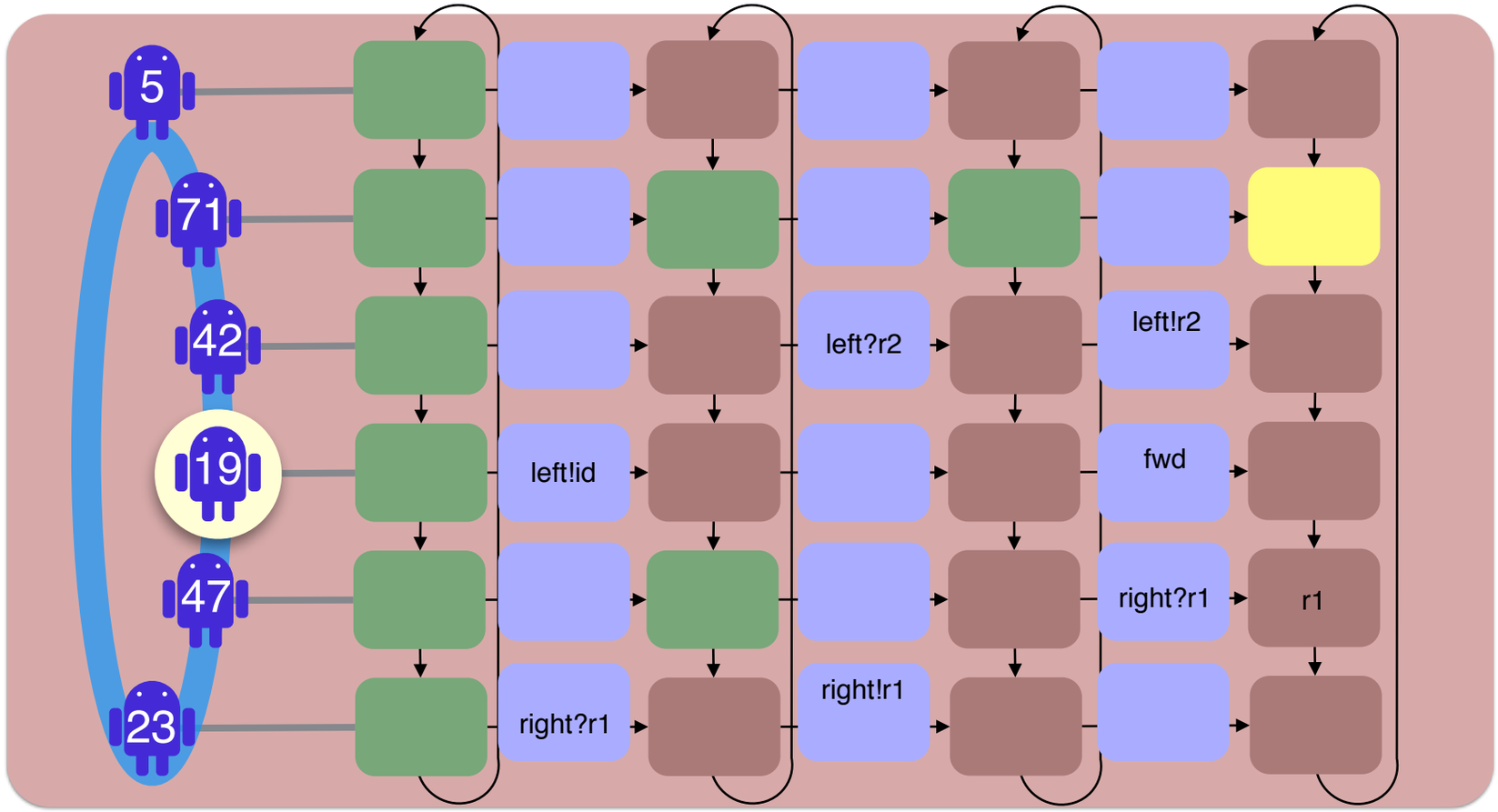
Data abstraction: symbolic runs + tracking data



- Register updates

Distributed algorithm

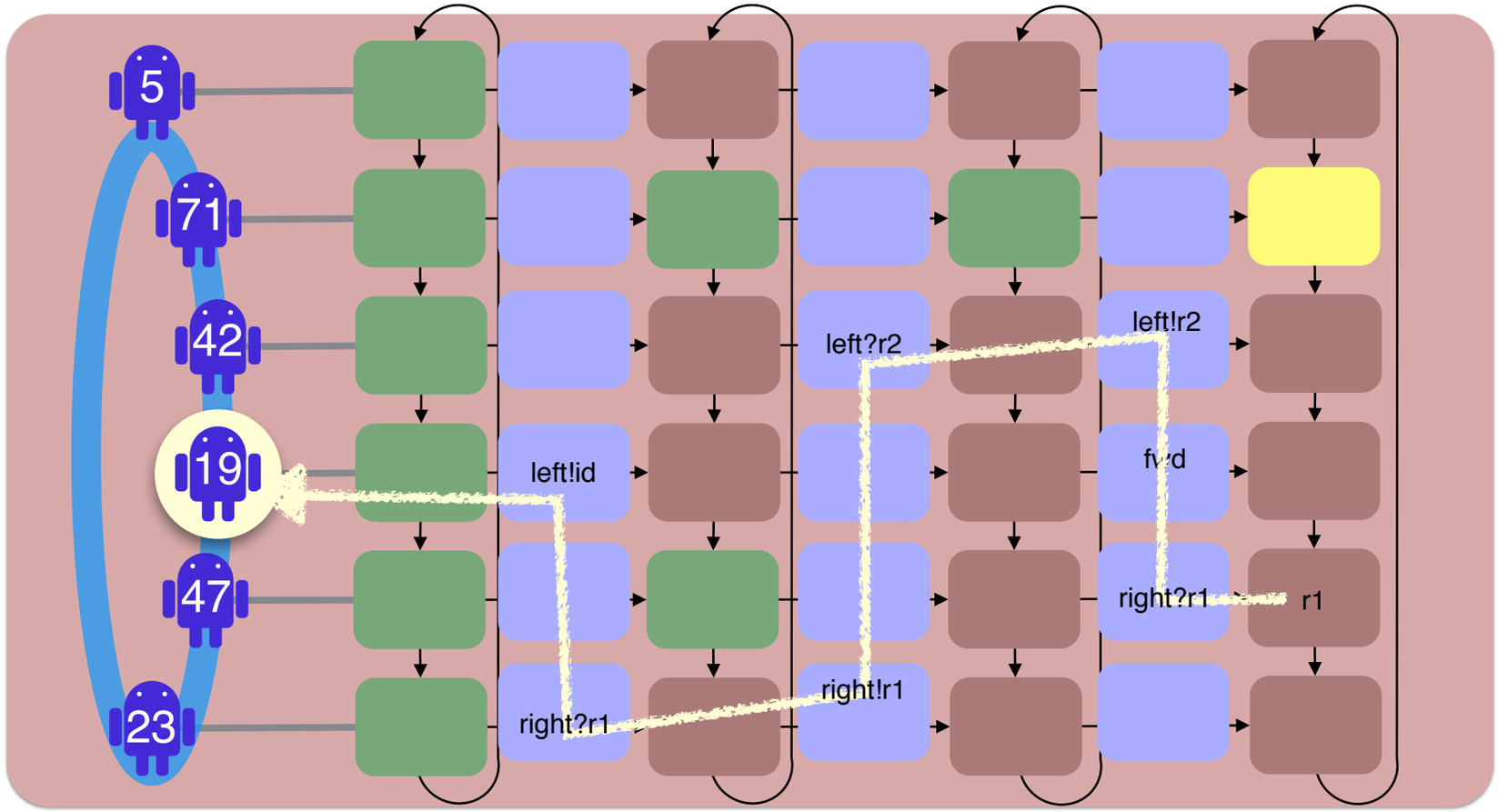
Data abstraction: symbolic runs + tracking data



- Register updates

Distributed algorithm

Data abstraction: symbolic runs + tracking data



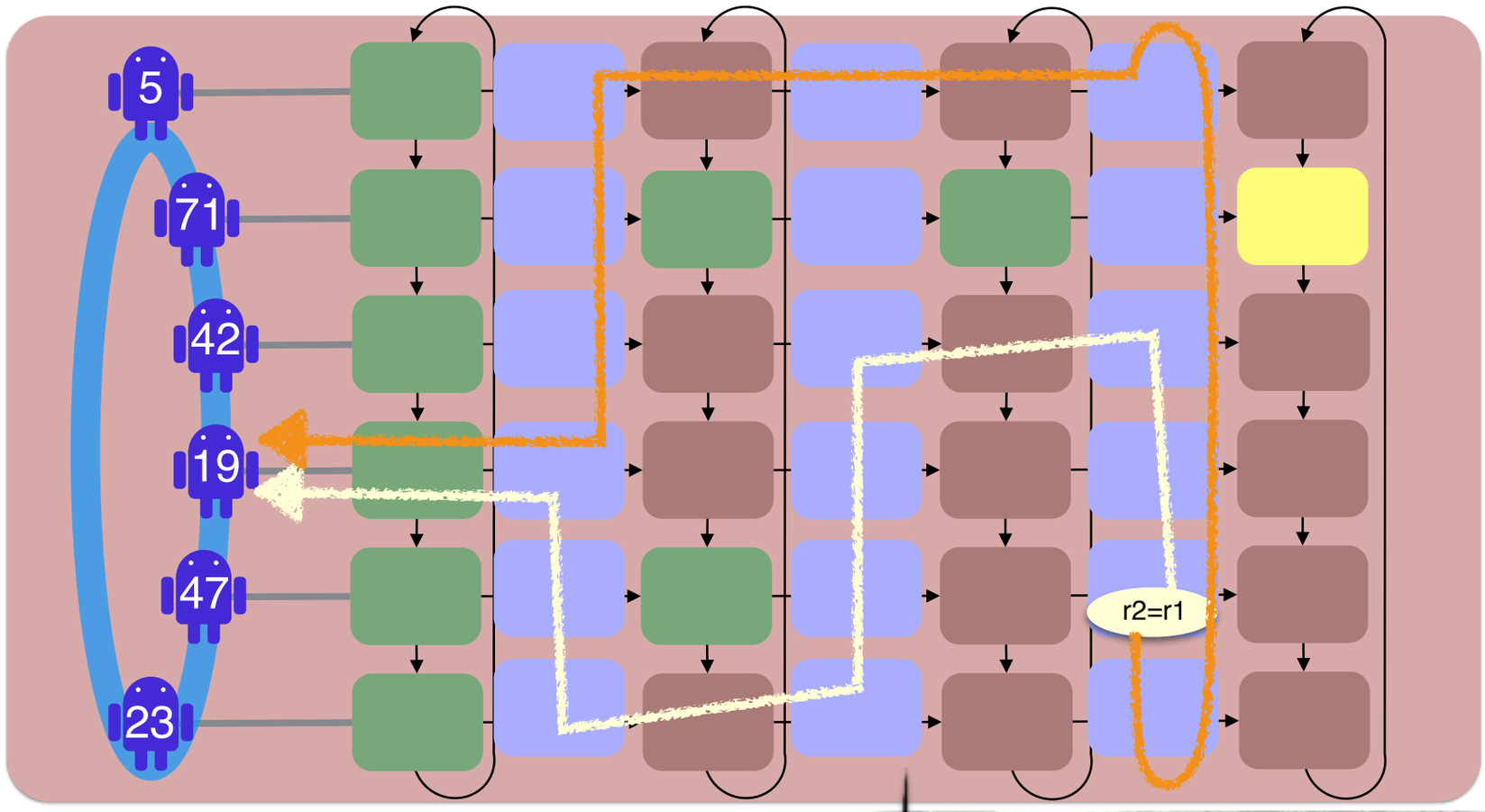
- Register updates

Distributed algorithm

(r_1,id)-path

can be expressed in CPDL
PDL with converse

Data abstraction: symbolic runs + tracking data



- Register updates
- Register equality check

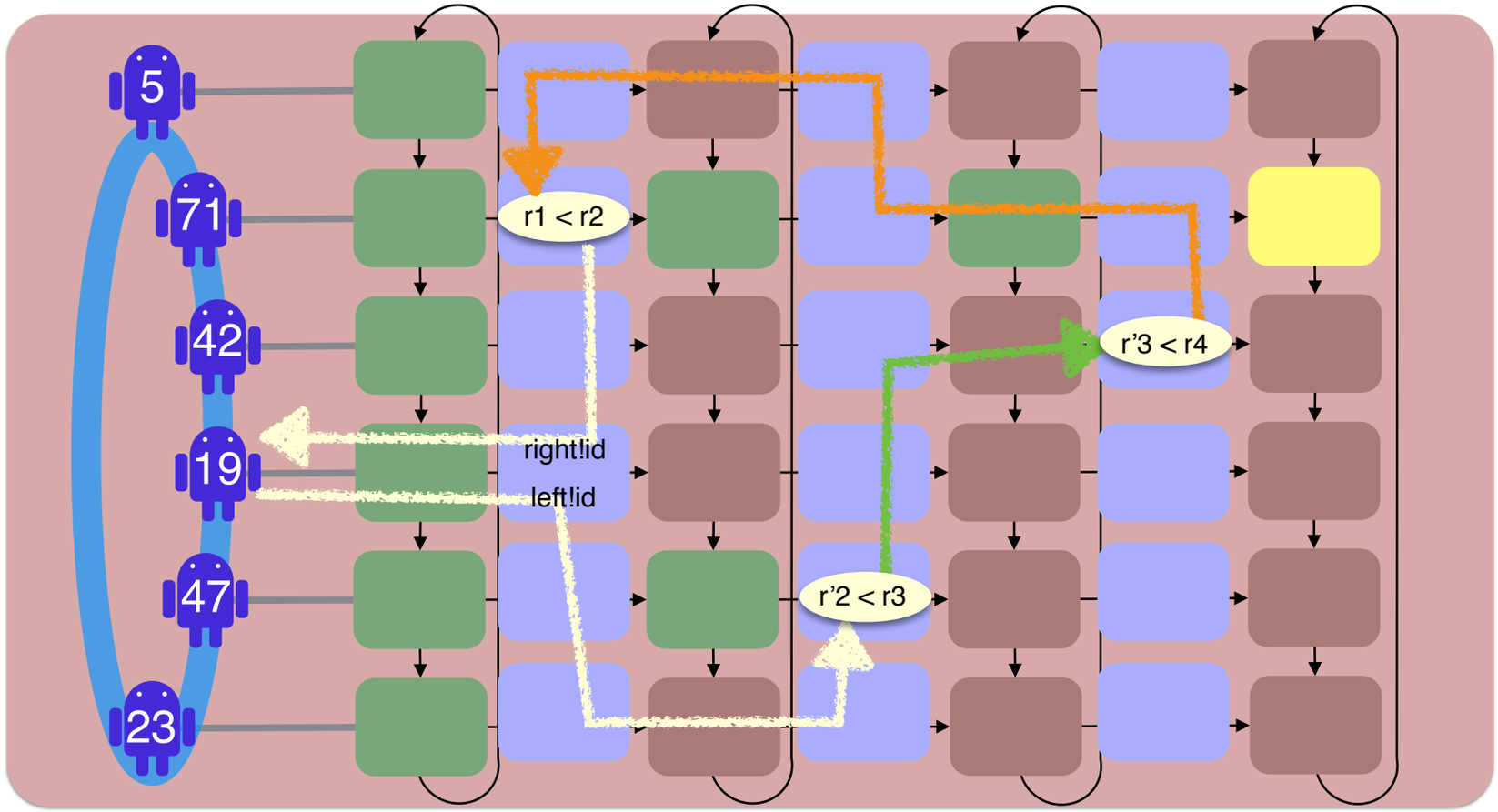
Distributed algorithm

π_1:(r_1,id)-path
π_2:(r_2,id)-path

$r_2 = r_1$ iff loop(π_1 π_2^{-1})

can be expressed in LCPDL
CPDL with loop

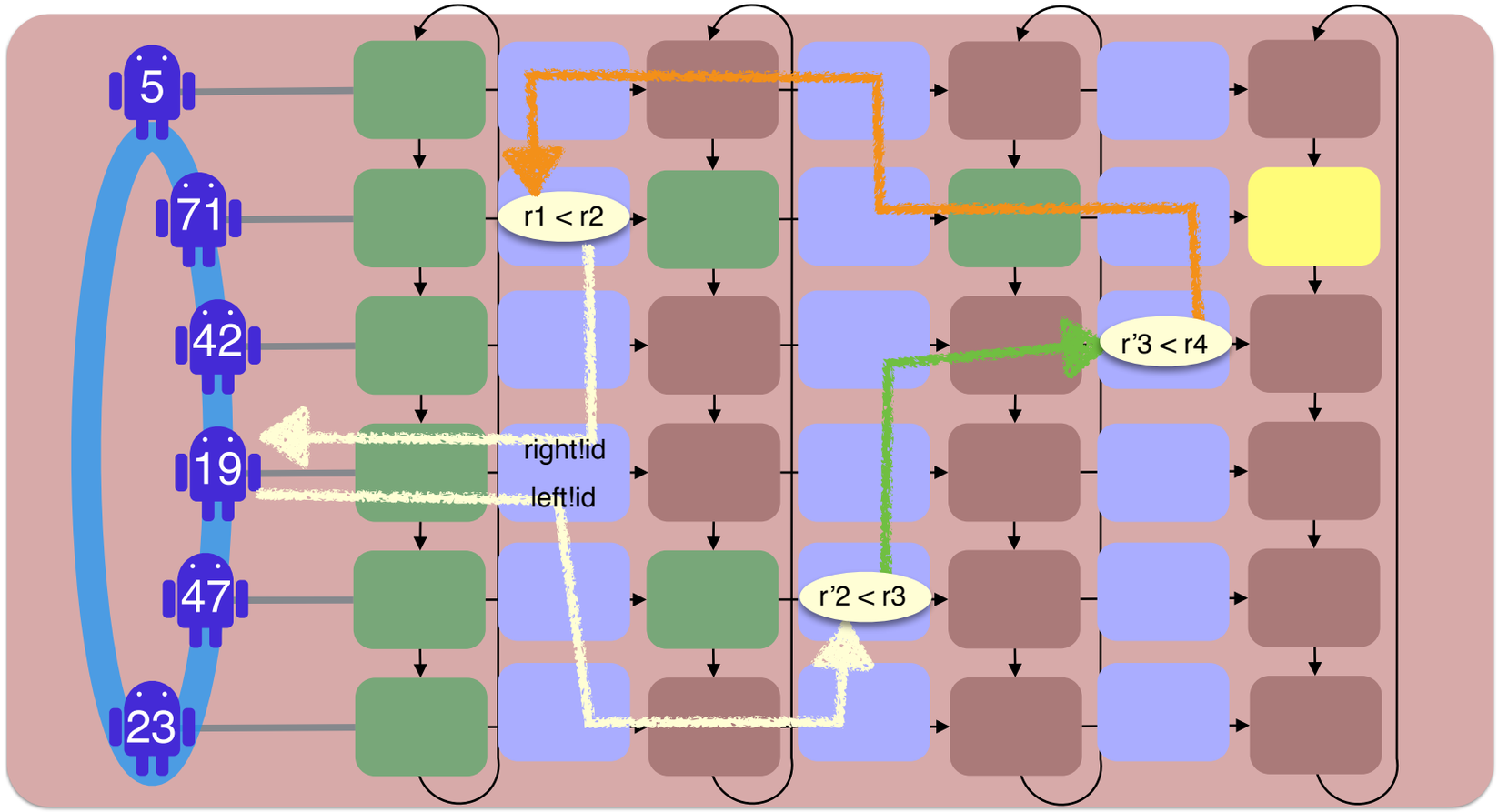
Data abstraction: symbolic runs + tracking data



- Register updates
- Register equality check
- Register comparison

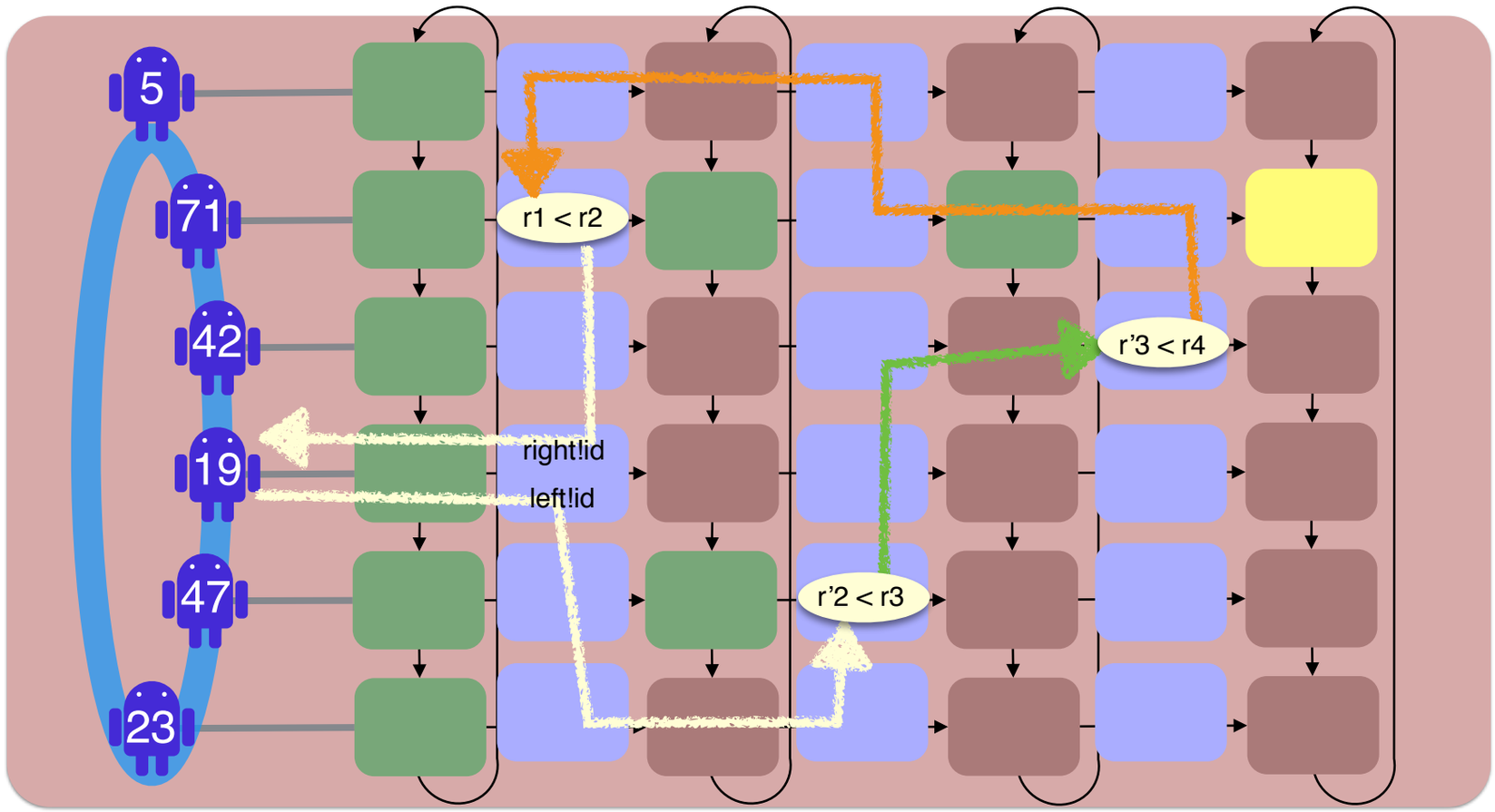
Distributed algorithm

Data abstraction: symbolic runs + tracking data



- If there is a <-loop, no pids assignments can turn the symbolic cylinder into a valid run.
- If no such loops, then there are pids that allow a valid realization of the abstract grid

Data abstraction: symbolic runs + tracking data



No loop of the form

$r_{i0} < r_{i1}$; (r_{i1}, r_{i2})-path; $r_{i2} < r_{i3}$; (r_{i3}, r_{i4})-path; … ; $r_{in} < r_{i0}$

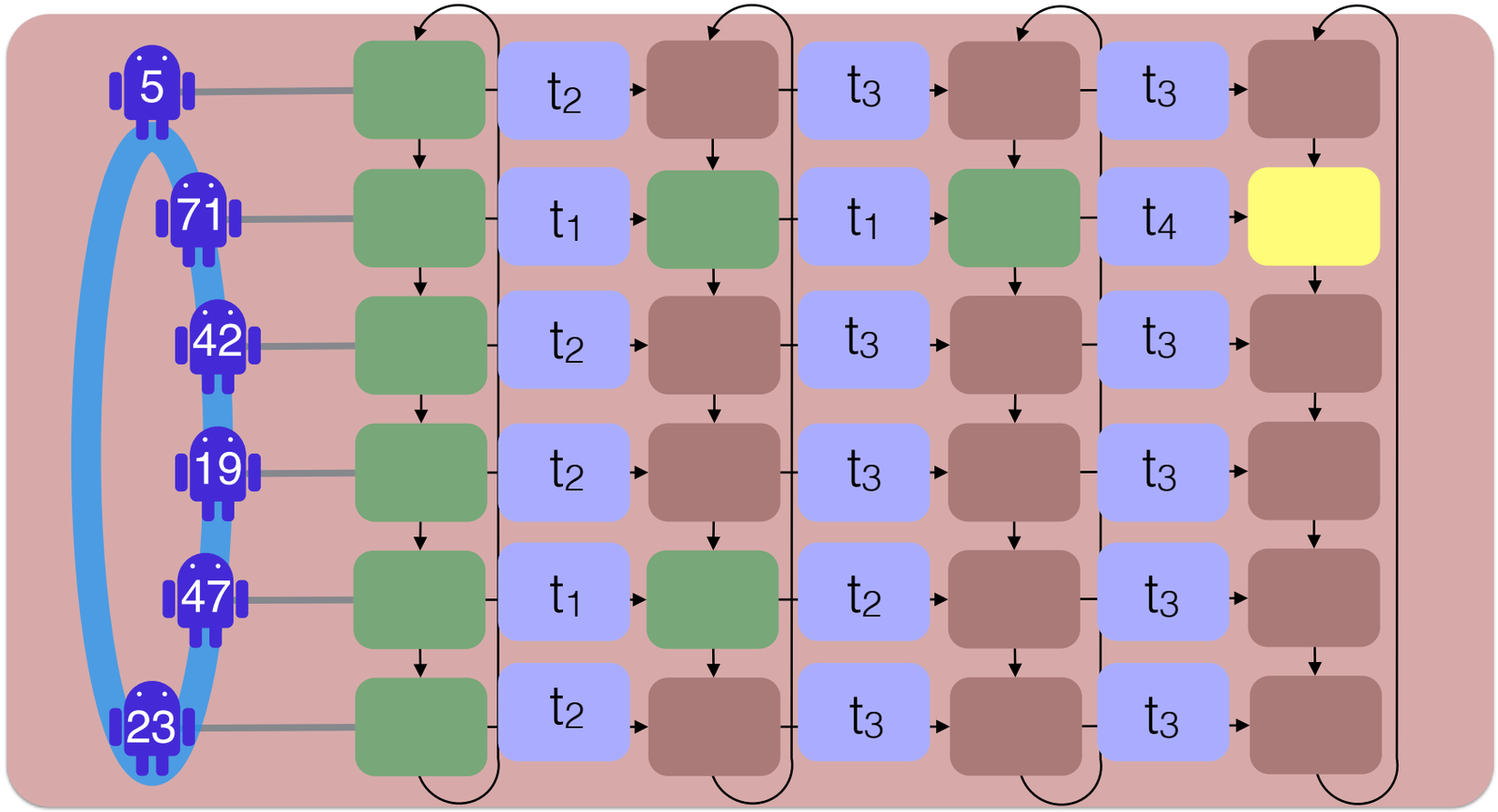
- Register updates
- Register equality check
- Register comparison

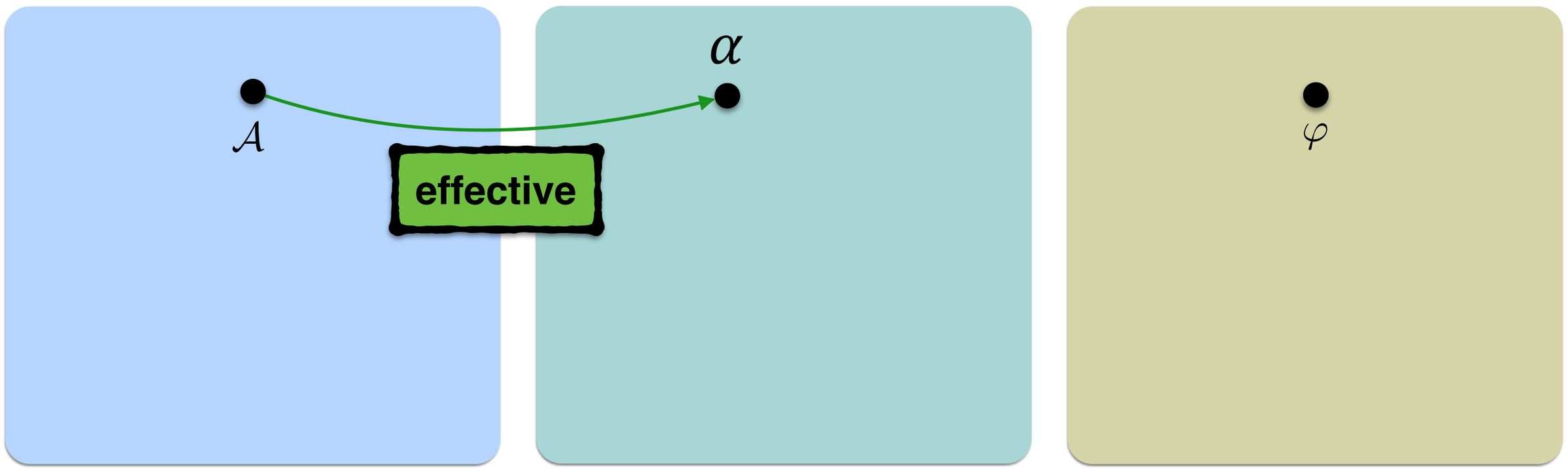
Distributed algorithm

can be expressed in LCPDL

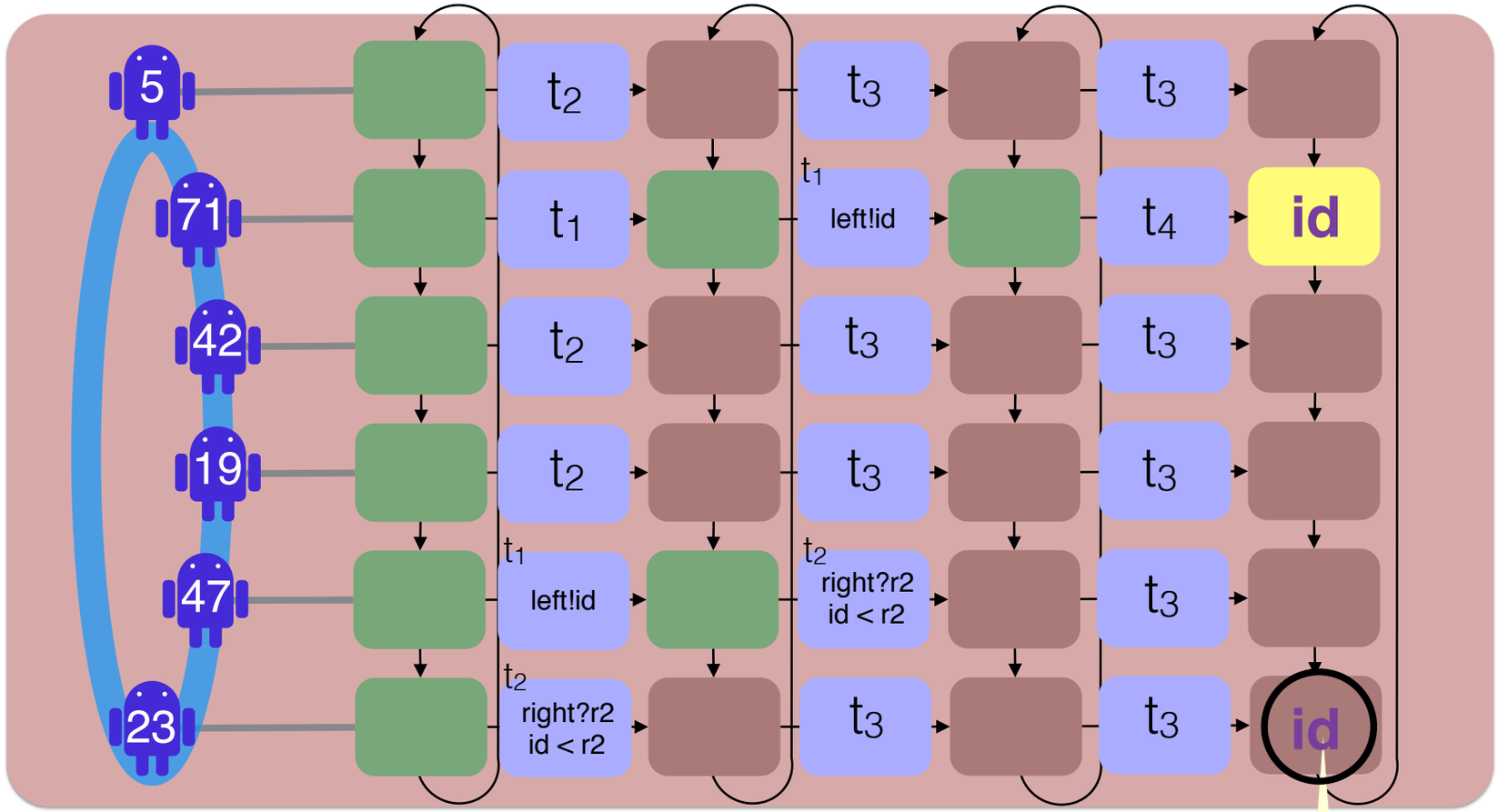
CPDL with loop

Data abstraction: symbolic runs + tracking data

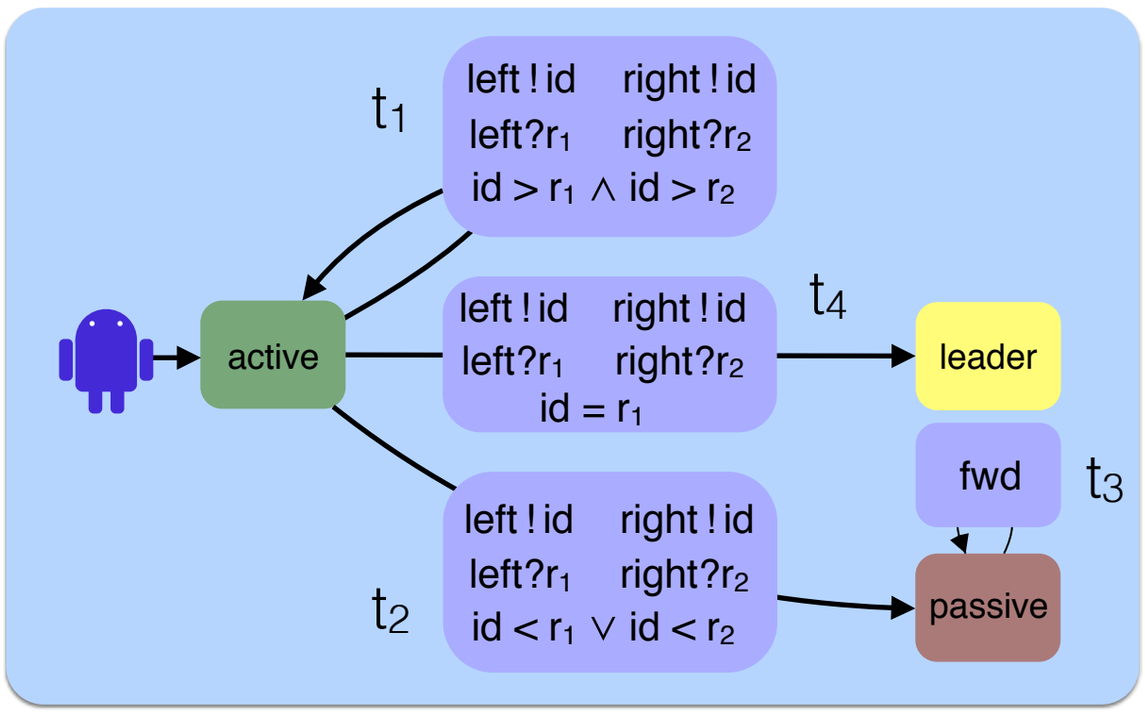




Distributed algorithms



Behavior



Distributed algorithm

«There is a leader, and the leader is the process with the maximum id.»

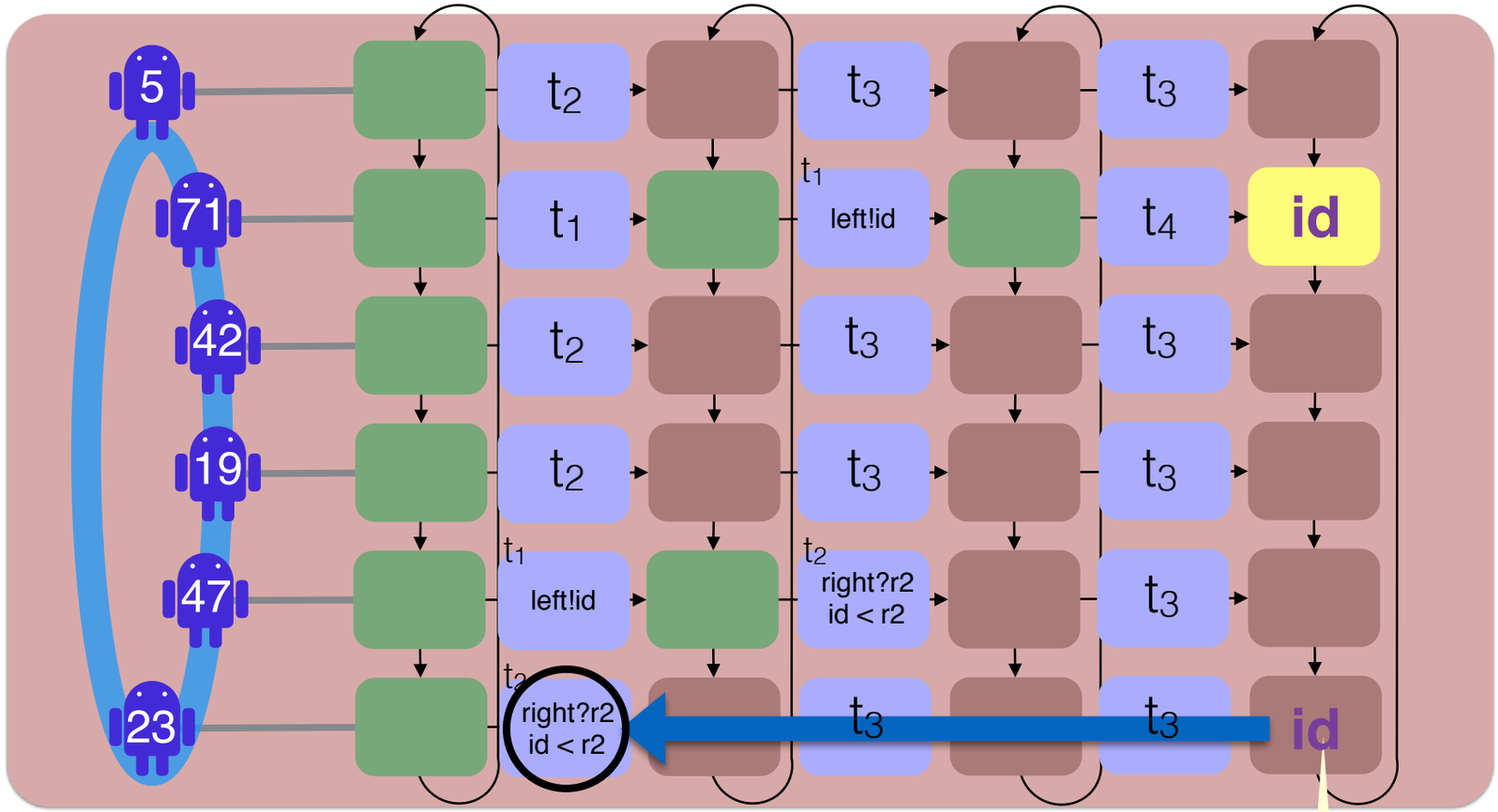
For all *n*, pid distributions, **accepting runs**, and processes:

$$\langle \rightarrow^* \rangle \Big(\neg\langle \rightarrow \rangle \wedge \langle \text{go-to-}\square \rangle \wedge [\downarrow^*] \underbrace{\big(\mathbf{id} \leq \langle \text{go-to-}\square \rangle \mathbf{id}\big)}_{\varphi} \Big)$$

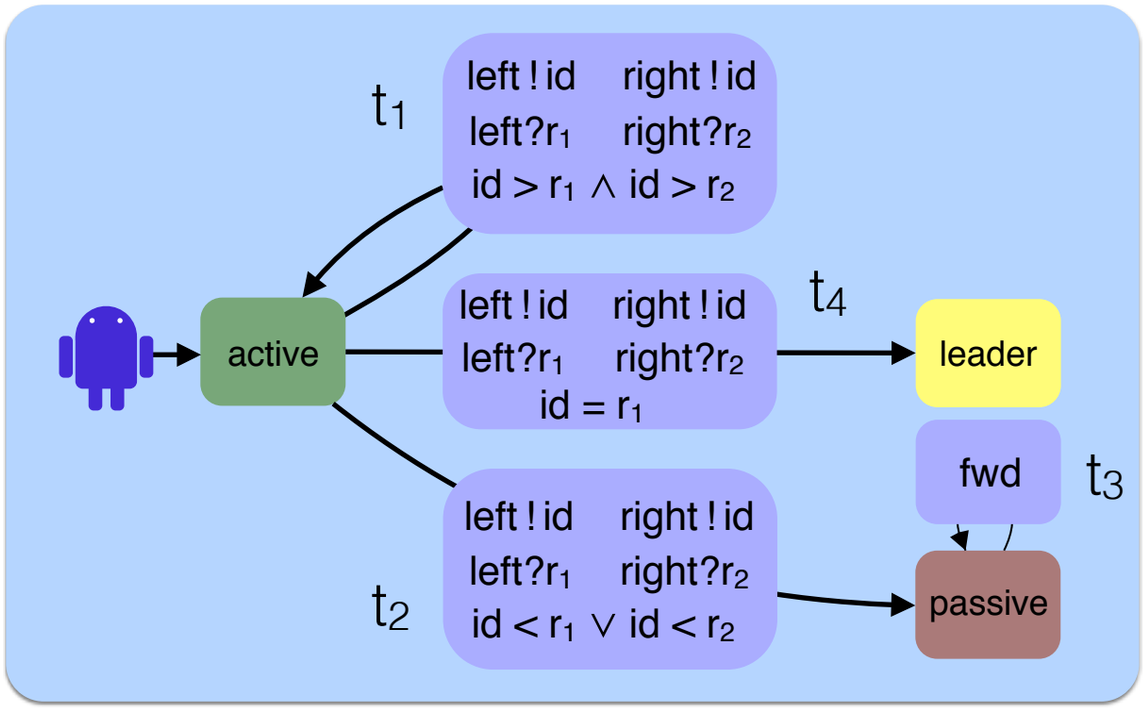
$$\text{go-to-}\square = (\neg\square\downarrow)^*\square$$

Data PDL

Distributed algorithms



Behavior



Distributed algorithm

«There is a leader, and the leader is the process with the maximum id.»

For all *n*, pid distributions, **accepting runs**, and processes:

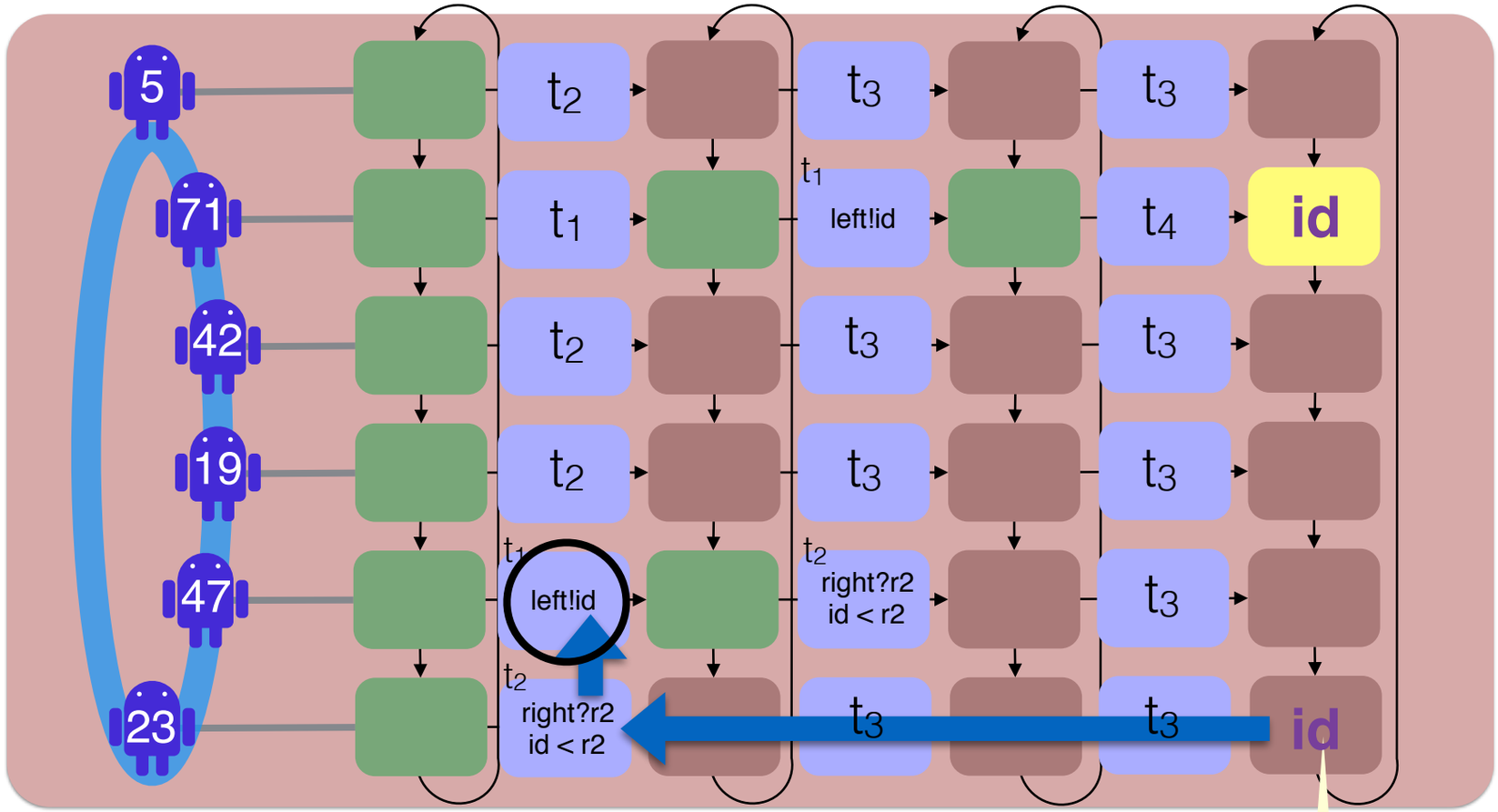
$$\langle \rightarrow^* \rangle \Big(\quad \neg \langle \rightarrow \rangle \ \wedge \ \langle \text{go-to-}\square \rangle$$

$$\wedge \ [\downarrow^*] \underbrace{\big(\mathbf{id} \leq \langle \text{go-to-}\square \rangle \mathbf{id}\big)}_{\varphi} \Big)$$

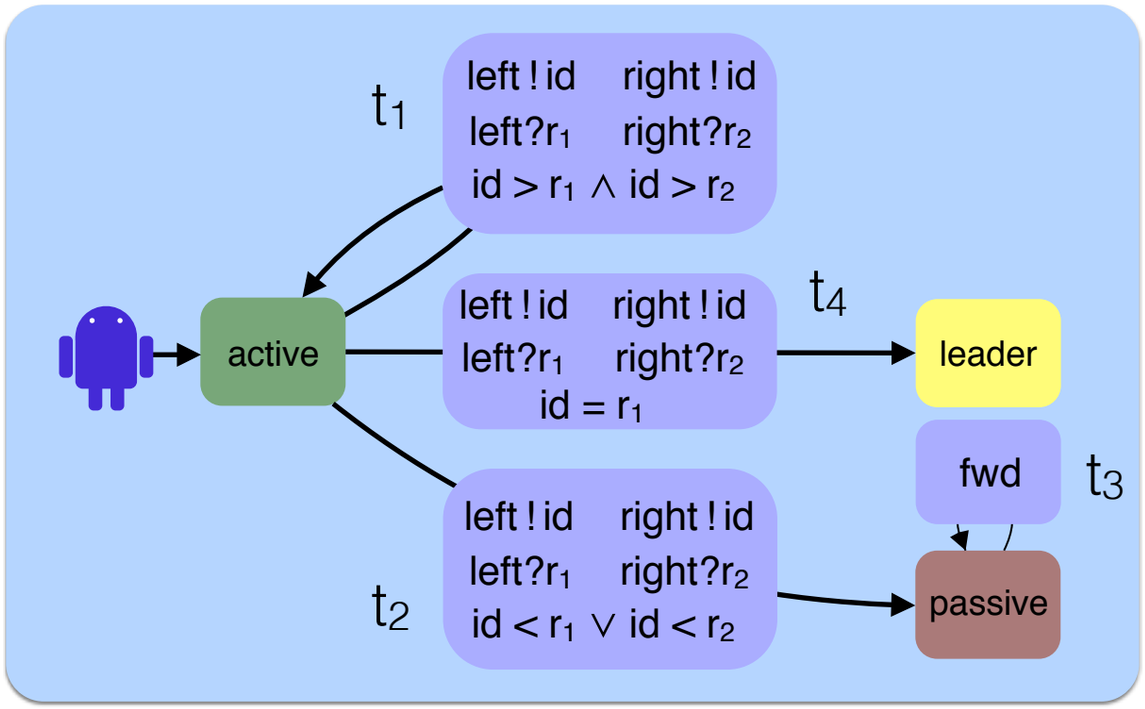
$$\text{go-to-}\square \ = \ (\neg \square \downarrow)^* \square$$

Data PDL

Distributed algorithms



Behavior



Distributed algorithm

«There is a leader, and the leader is the process with the maximum id.»

For all *n*, pid distributions, **accepting runs**, and processes:

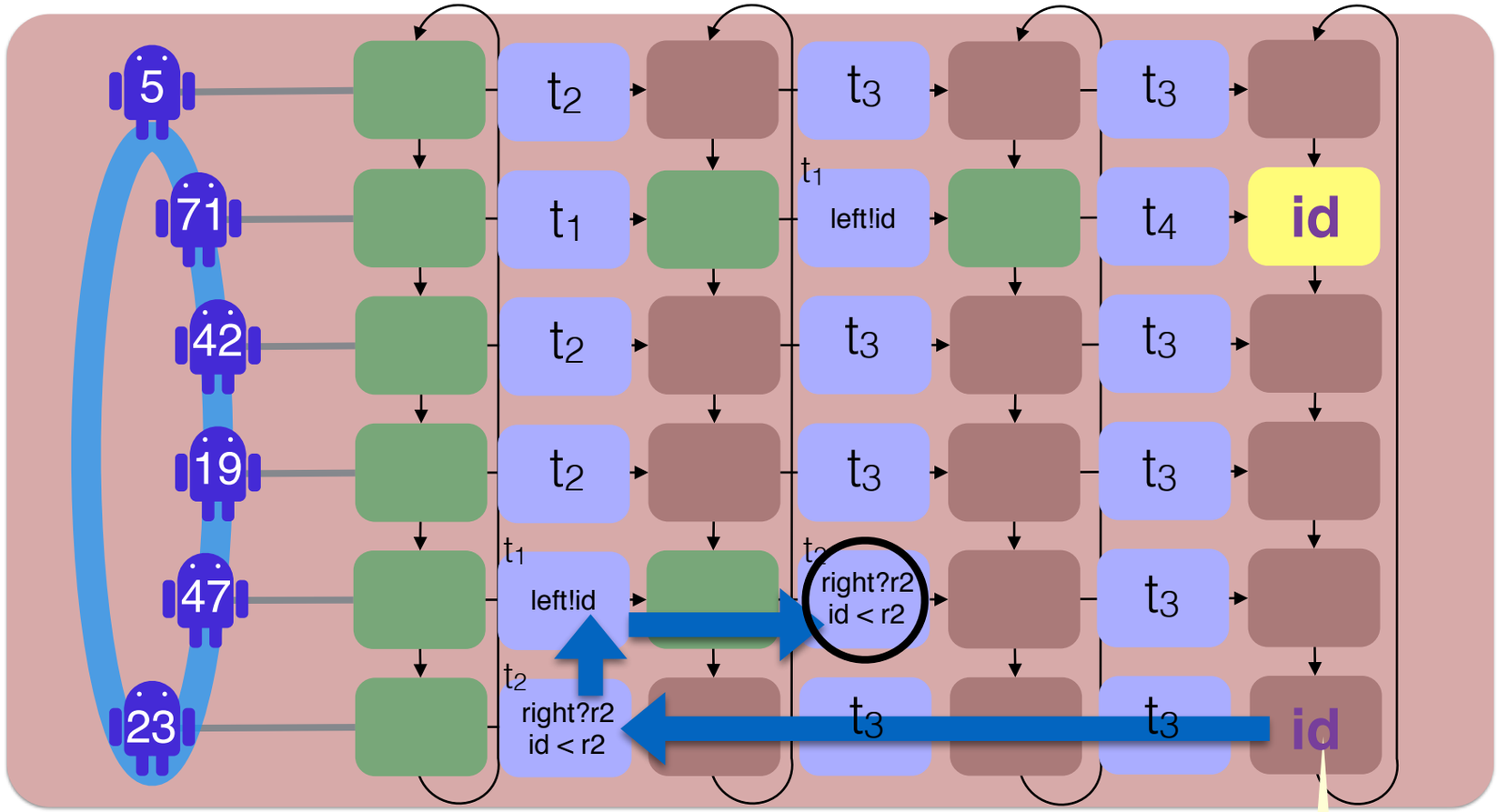
$$\langle \rightarrow^* \rangle \Big(\quad \neg \langle \rightarrow \rangle \ \wedge \ \langle \text{go-to-}\square \rangle$$

$$\wedge \ [\downarrow^*] \underbrace{\big(\mathbf{id} \leq \langle \text{go-to-}\square \rangle \mathbf{id}\big)}_{\varphi} \Big)$$

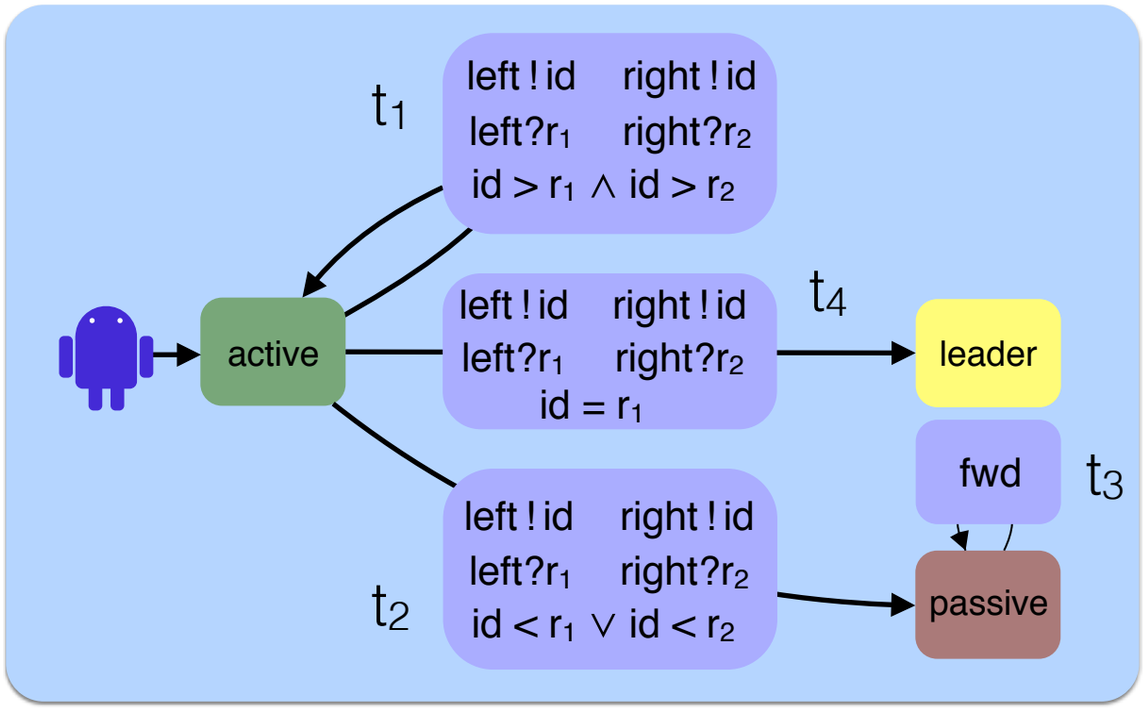
$$\text{go-to-}\square \ = \ (\neg \square \downarrow)^* \square$$

Data PDL

Distributed algorithms



Behavior



Distributed algorithm

«There is a leader, and the leader is the process with the maximum id.»

For all *n*, pid distributions, **accepting runs**, and processes:

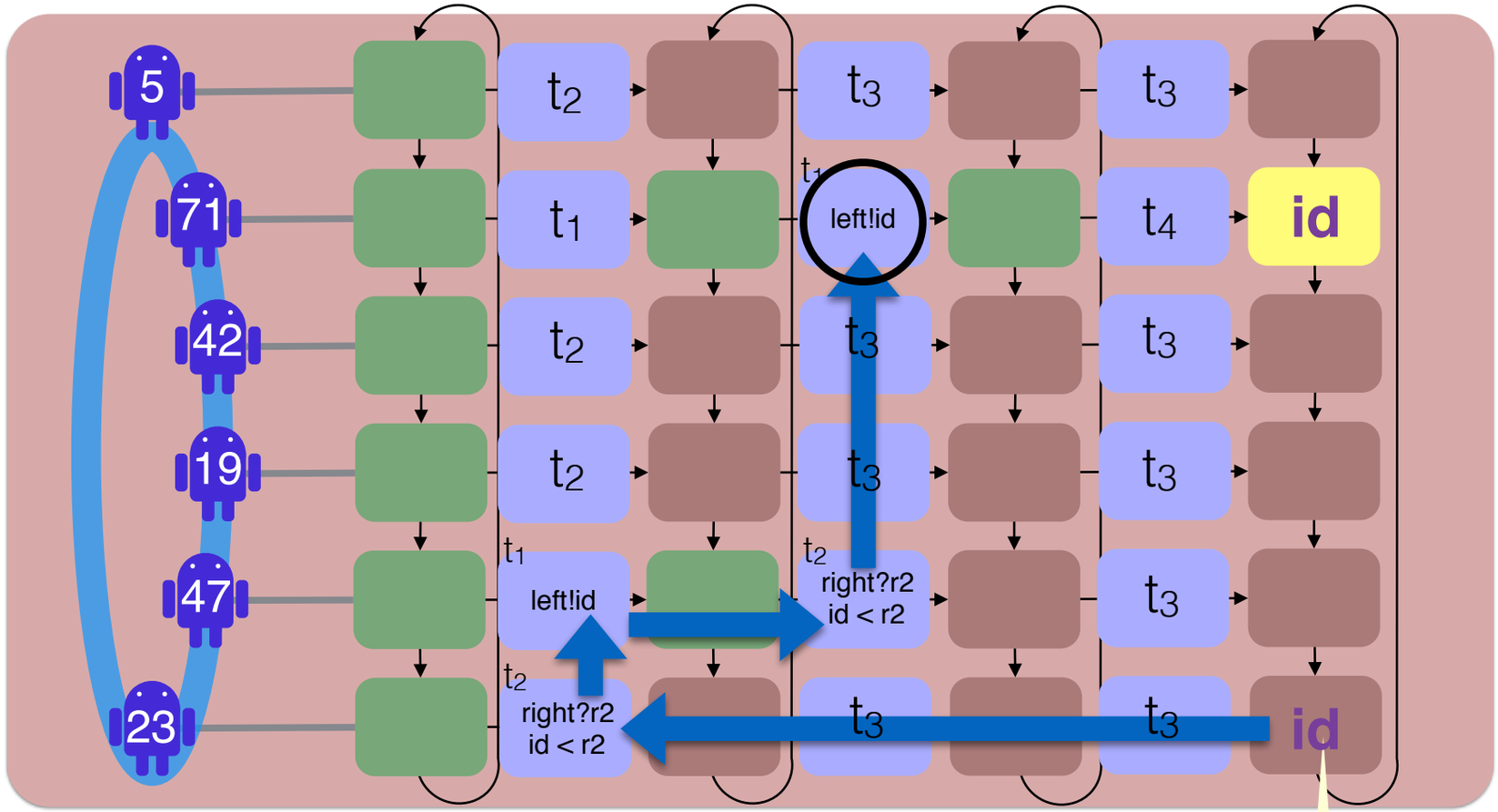
$$\langle \rightarrow^* \rangle \Big(\quad \neg \langle \rightarrow \rangle \ \wedge \ \langle \text{go-to-}\square \rangle$$

$$\wedge \ [\downarrow^*] \underbrace{\big(\mathbf{id} \leq \langle \text{go-to-}\square \rangle \mathbf{id}\big)}_{\varphi} \Big)$$

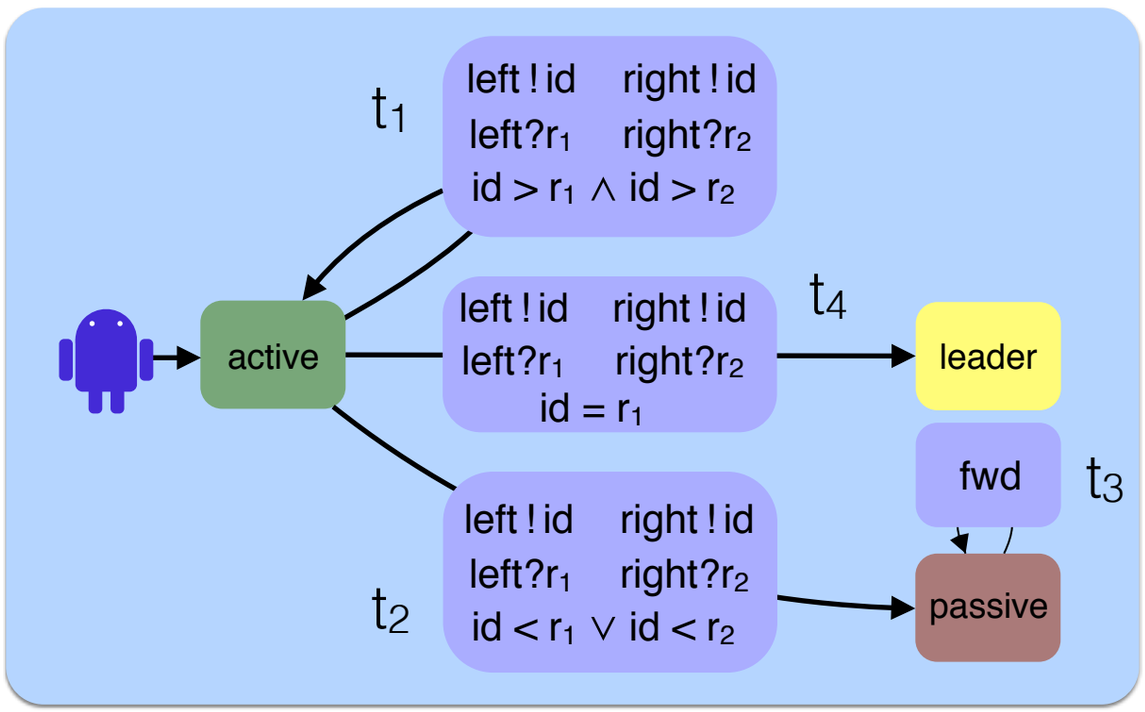
$$\text{go-to-}\square \ = \ (\neg \square \downarrow)^* \square$$

Data PDL

Distributed algorithms



Behavior



Distributed algorithm

«There is a leader, and the leader is the process with the maximum id.»

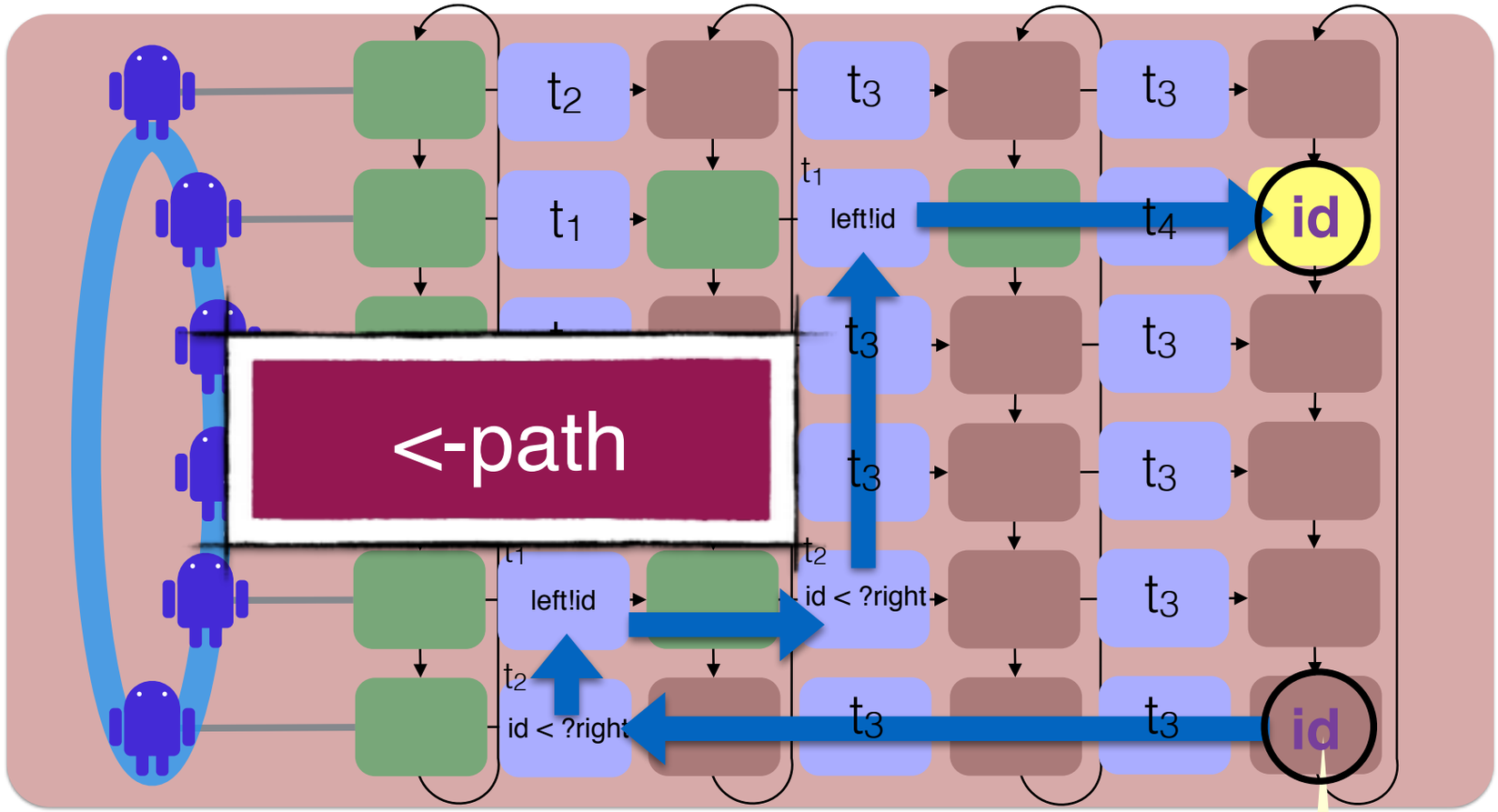
For all *n*, pid distributions, **accepting runs**, and processes:

$$\langle \rightarrow^* \rangle \Big(\neg\langle \rightarrow \rangle \wedge \langle \text{go-to-}\square \rangle \wedge [\downarrow^*] \underbrace{\big(\mathbf{id} \leq \langle \text{go-to-}\square \rangle \mathbf{id}\big)}_{\varphi} \Big)$$

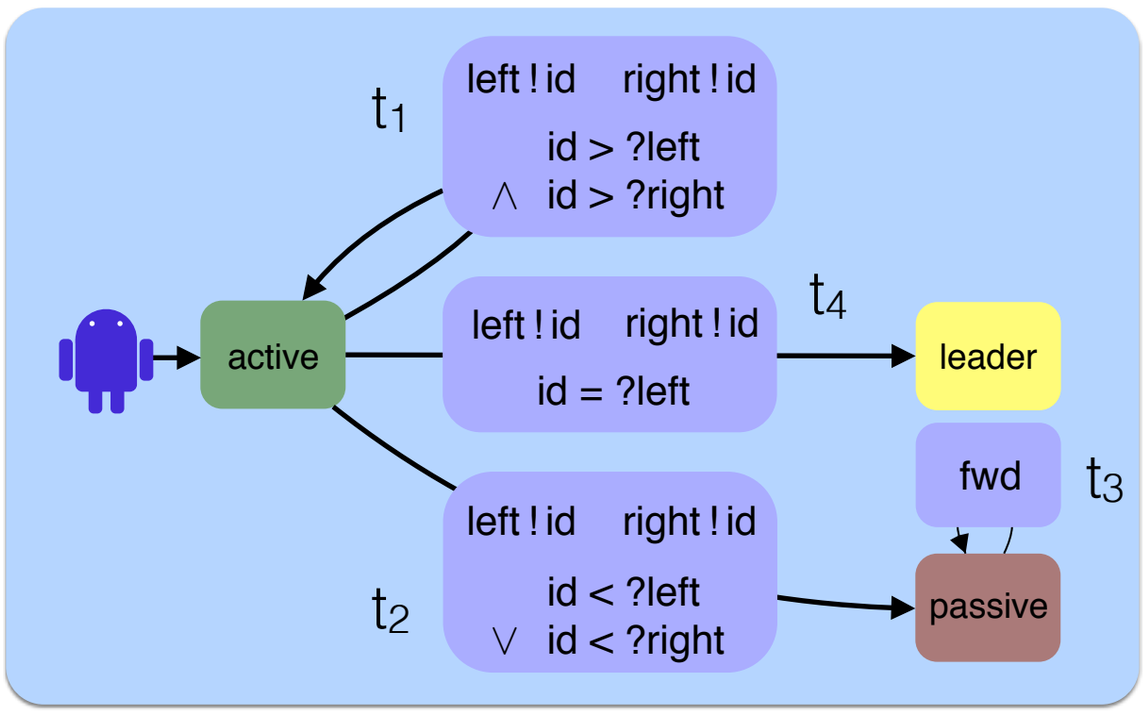
$$\text{go-to-}\square = (\neg\square\downarrow)^*\square$$

Data PDL

Distributed algorithms



Behavior



Distributed algorithm

«There is a leader, and the leader is the process with the maximum id.»

For all *n*, pid distributions, **accepting runs**, and processes:

$$\langle \longrightarrow^* \rangle \big(\quad \neg \langle \longrightarrow \rangle \ \wedge \ \langle \text{go-to-}\square \rangle$$

$$\wedge \ [\downarrow^*] \underbrace{\big(\mathbf{id} \leq \langle \text{go-to-}\square \rangle \mathbf{id} \big)}_{\varphi} \big)$$

$$\text{go-to-}\square \ = \ (\neg \square \downarrow)^* \square$$

Data PDL

Distributed algorithms

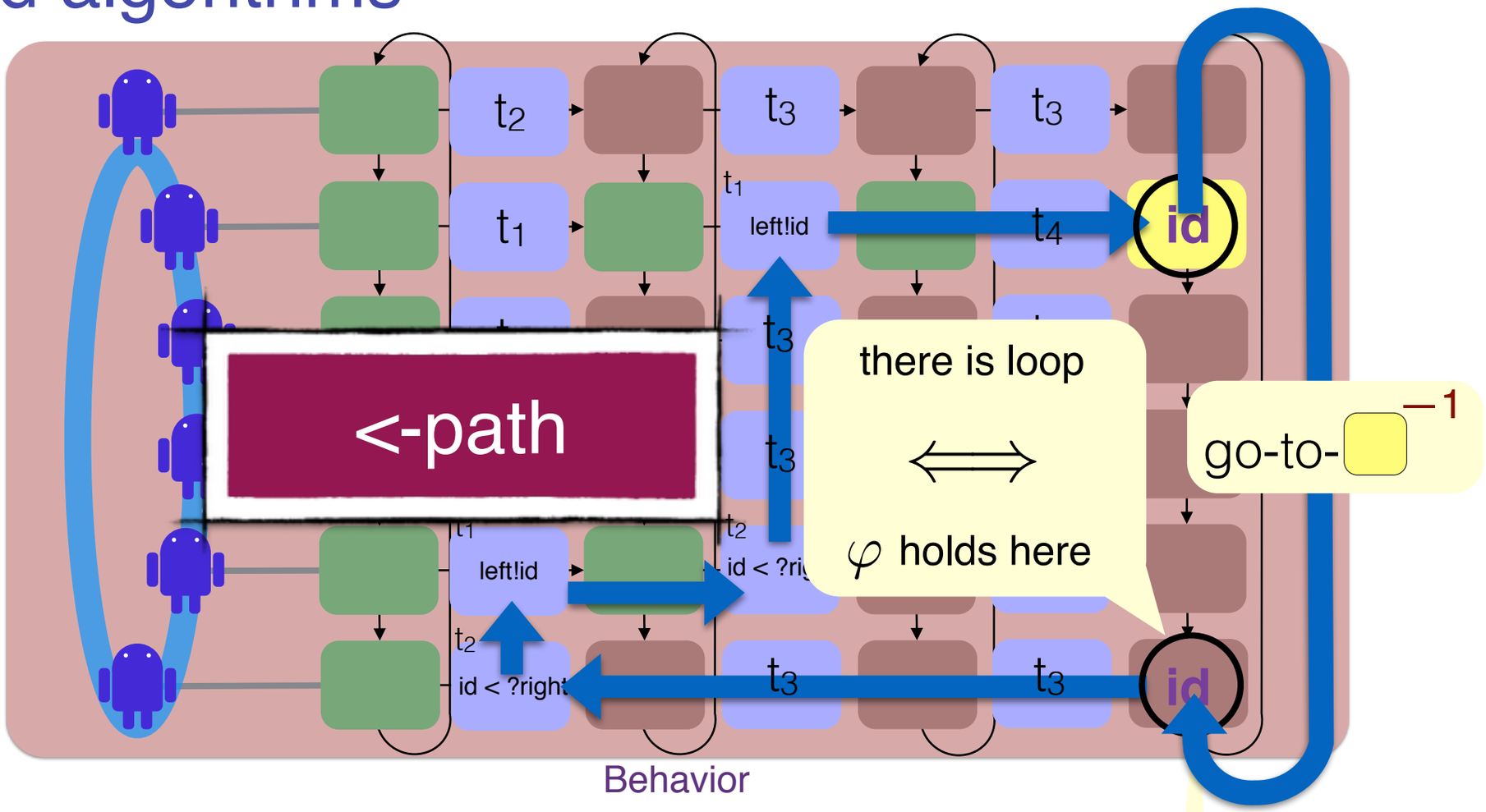
no loop

$\Rightarrow$

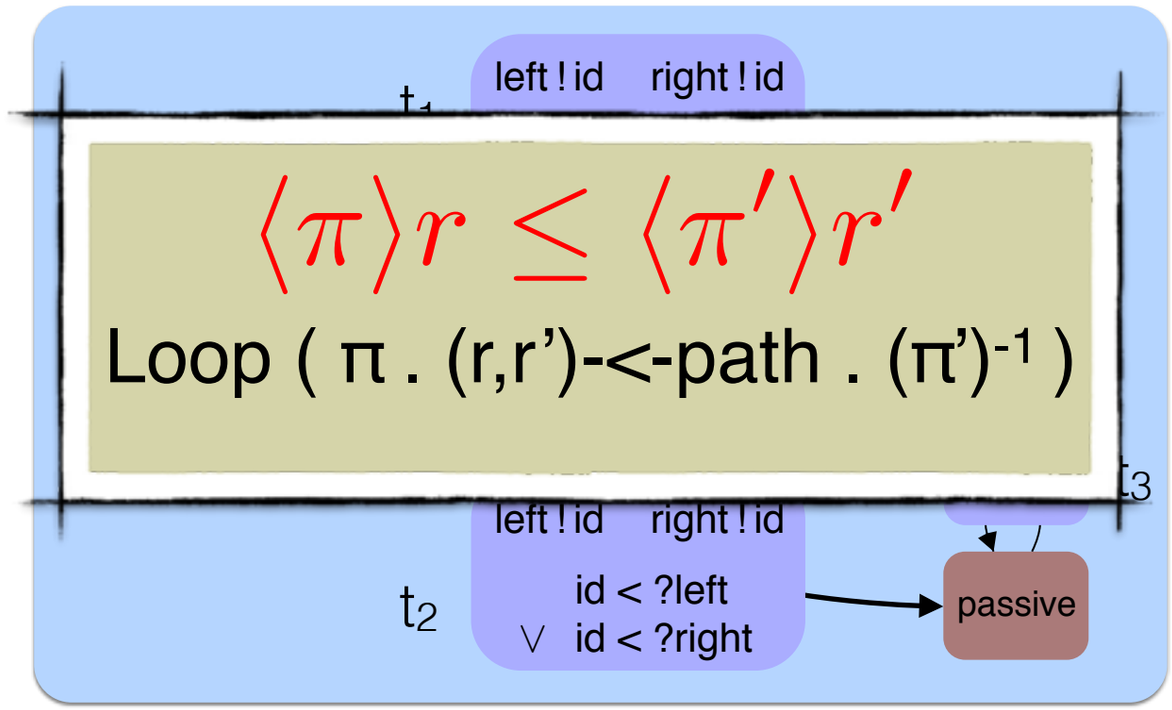
no evidence of φ

$\Rightarrow$

there are pids making φ false



Behavior



Distributed algorithm

«There is a leader, and the leader is the process with the maximum id.»

For all *n*, pid distributions, **accepting runs**, and processes:

$\langle \rightarrow^* \rangle (\quad \neg\langle \rightarrow \rangle \ \wedge \ \langle \text{go-to-}\square \rangle$

$\wedge \ [\downarrow^*] \underbrace{(\textbf{id} \leq \langle \text{go-to-}\square \rangle \textbf{id})}_{\varphi})$

$\text{go-to-}\square \ = \ (\neg\square\downarrow)^*\square$

Data PDL

Distributed algorithms

no loop

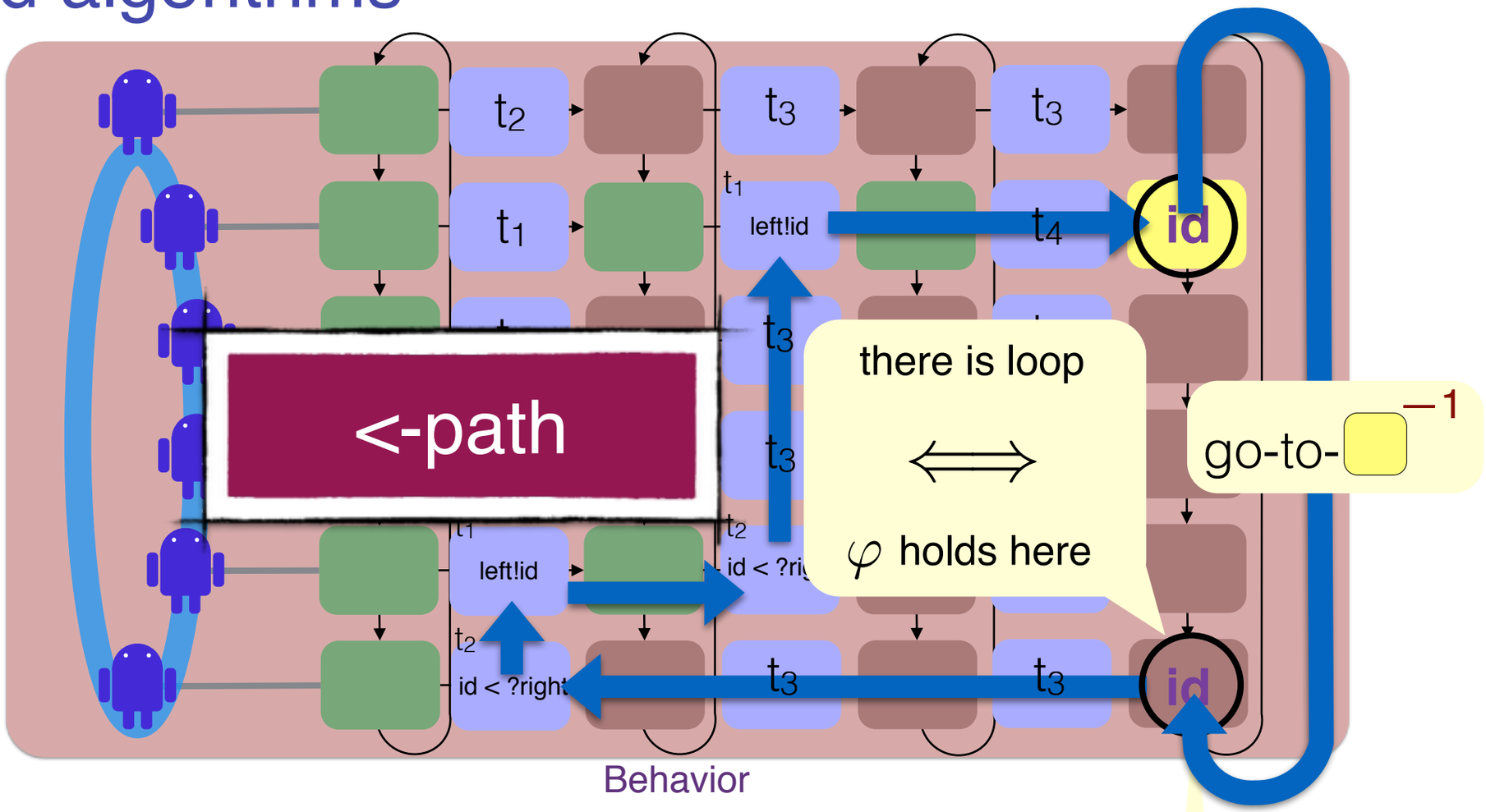
$\Rightarrow$

no evidence of φ

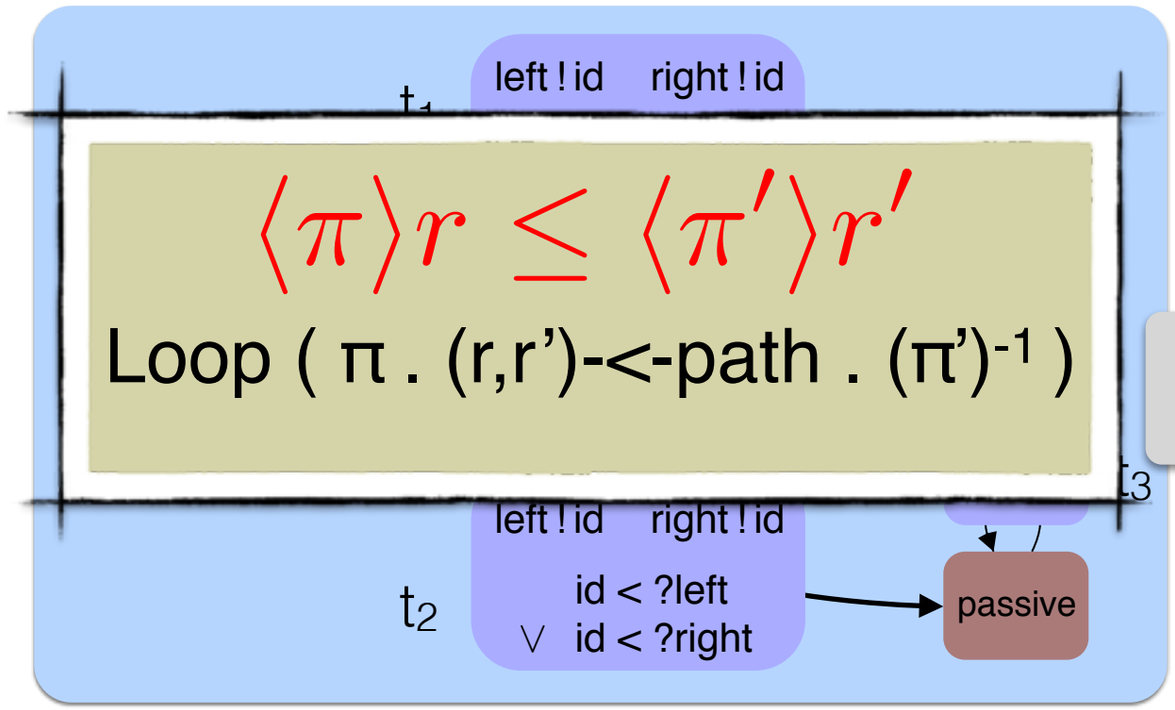
$\Rightarrow$

there are pids making φ false

$\not\mathsf{id} \leq \langle\downarrow\rangle \mathsf{id} \quad \vee \quad \not\mathsf{id} > \langle\downarrow\rangle \mathsf{id}$



Behavior



Distributed algorithm

deterministic

«There is a leader, and the leader is the process with the maximum id.»

For all *n*, pid distributions, **accepting runs**, and processes:

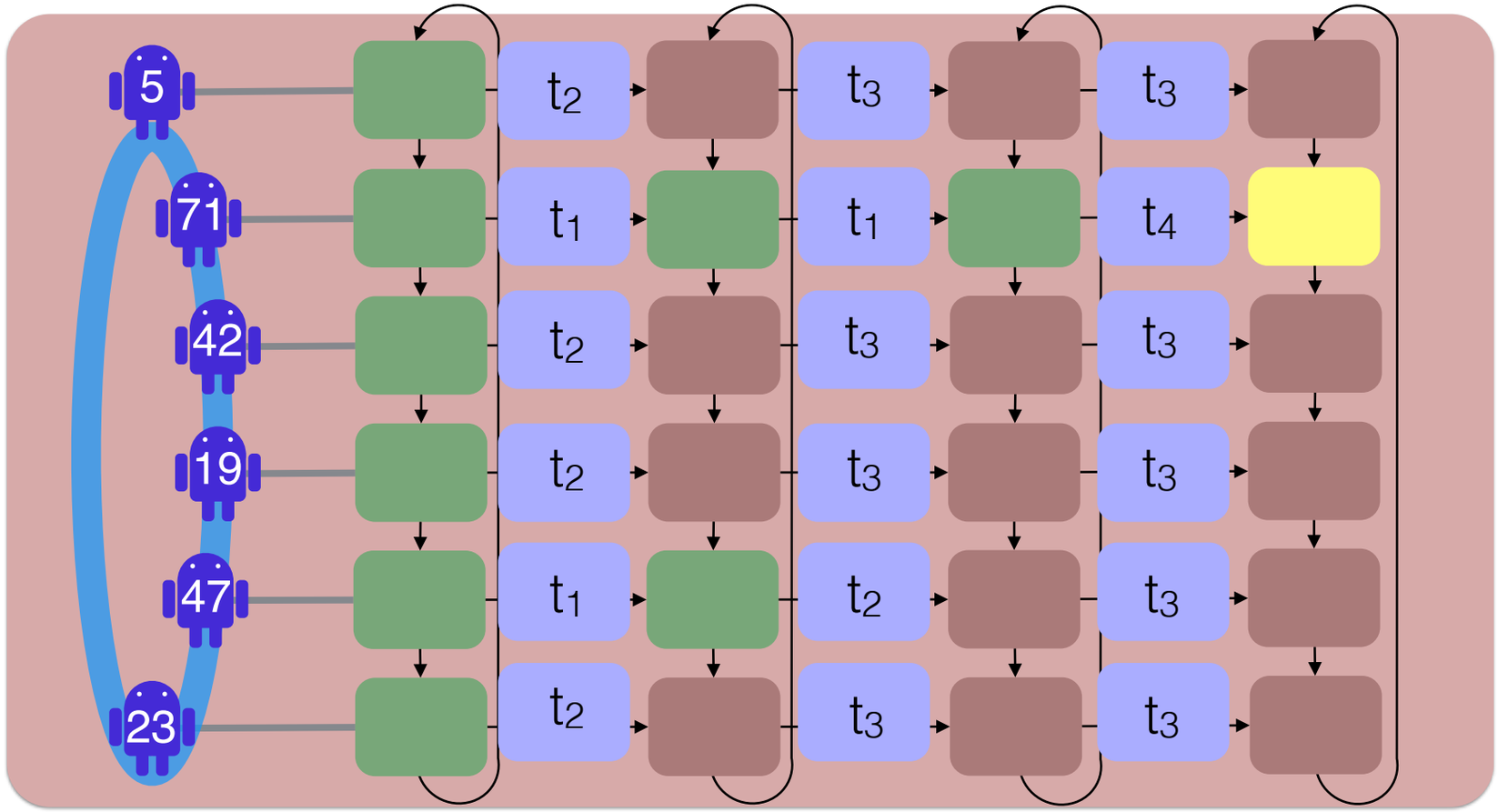
$\langle \rightarrow^* \rangle (\quad \neg\langle\rightarrow\rangle \wedge \langle \text{go-to-}\square \rangle$

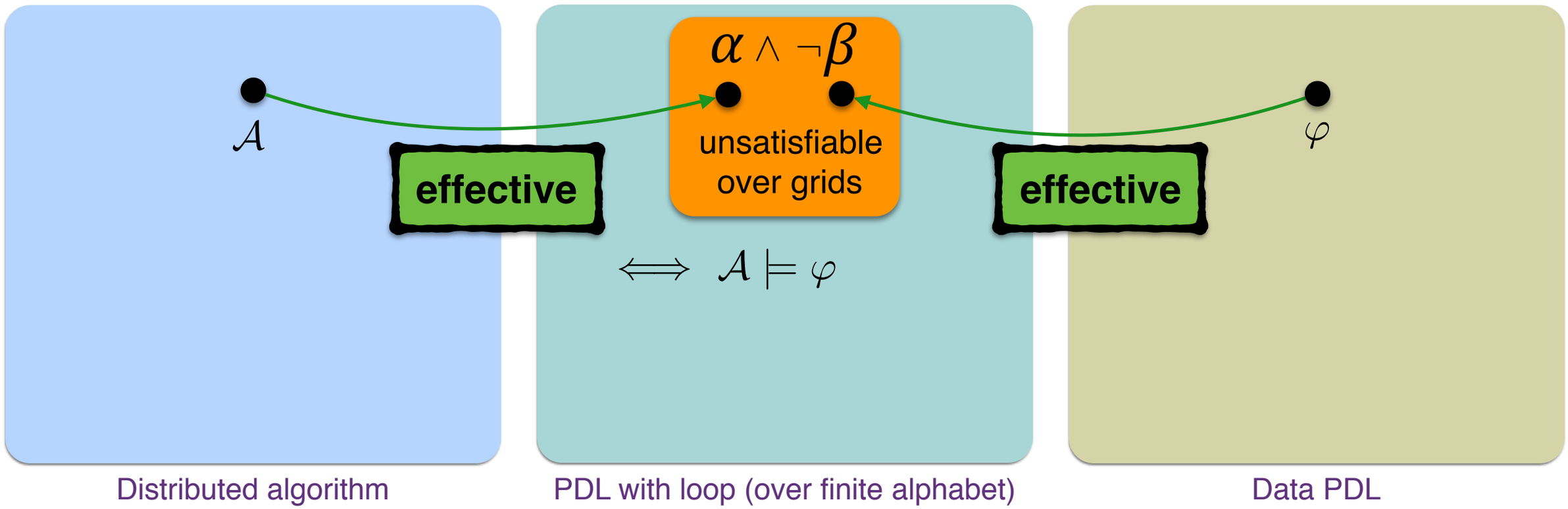
$[\downarrow^*] \underbrace{(\mathsf{id} \leq \langle \text{go-to-}\square \rangle \mathsf{id})}_{\varphi})$

$\text{go-to-}\square = (\neg\square\downarrow)^*\square$

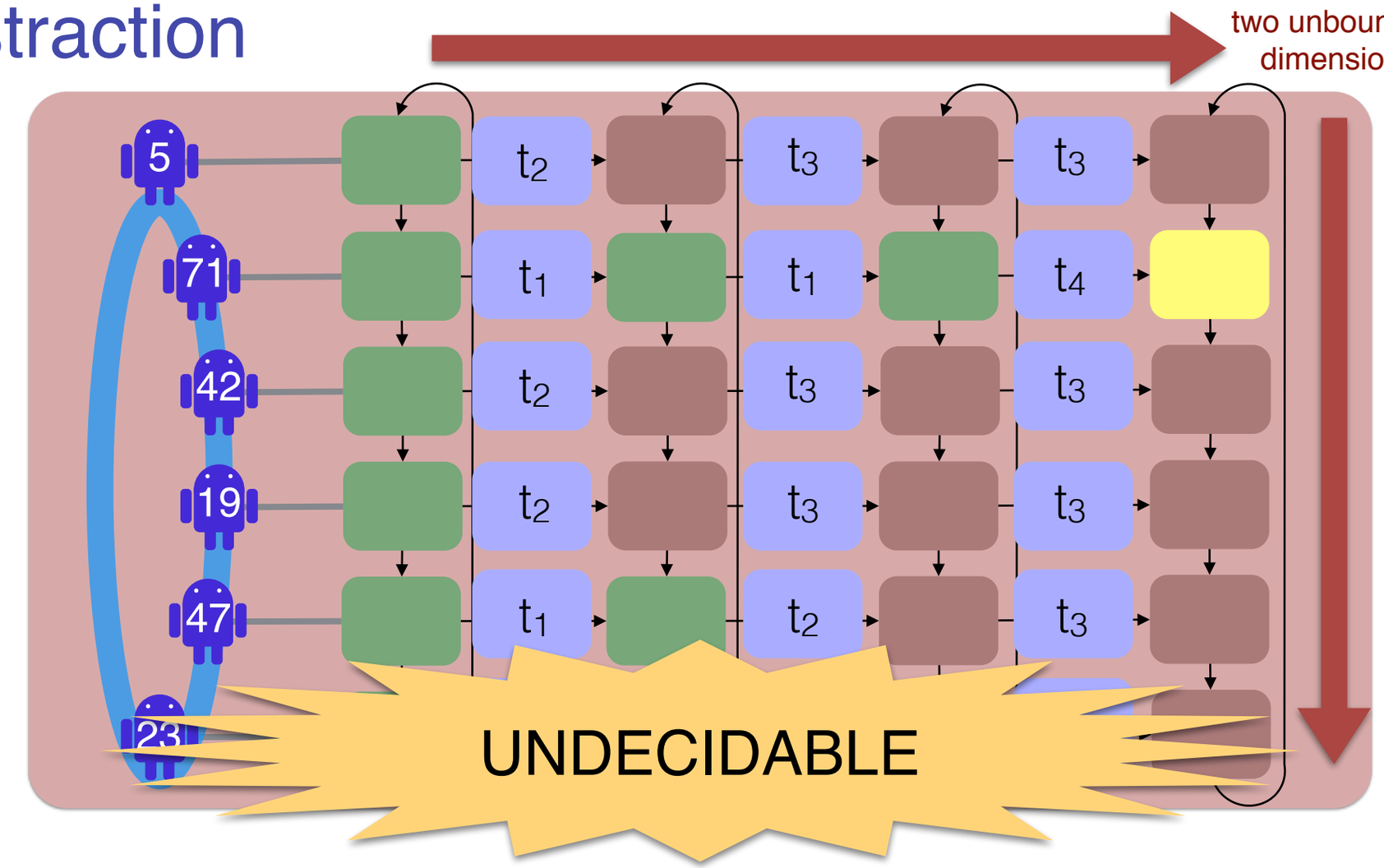
Data PDL

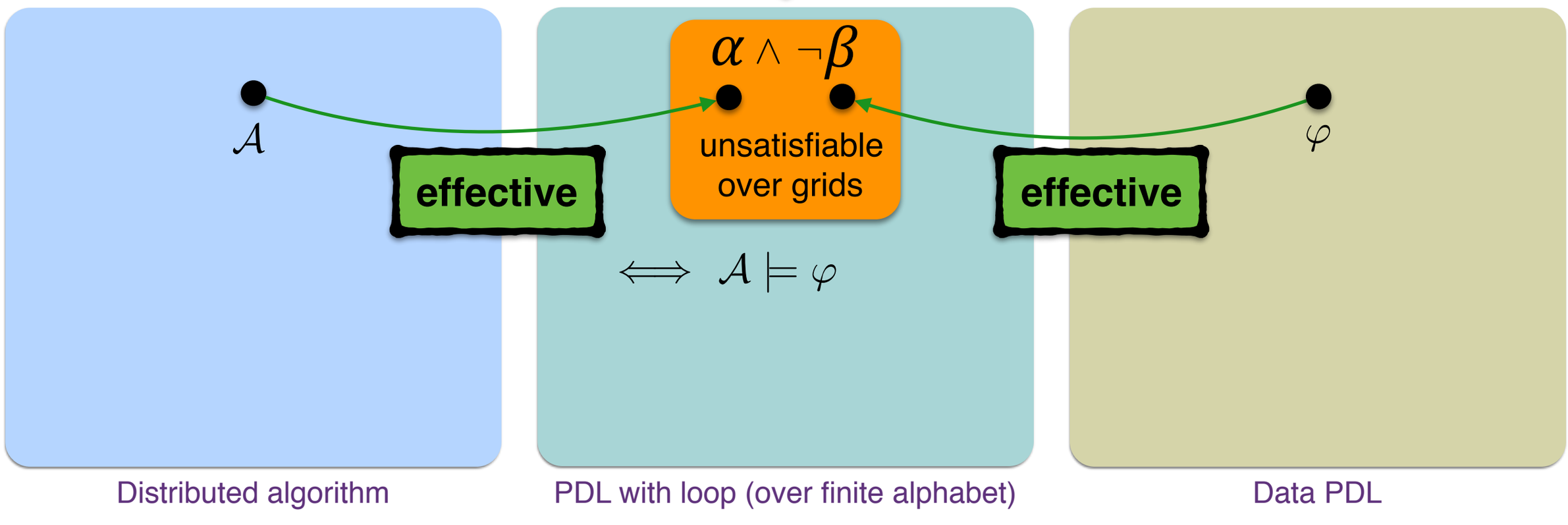
Data abstraction



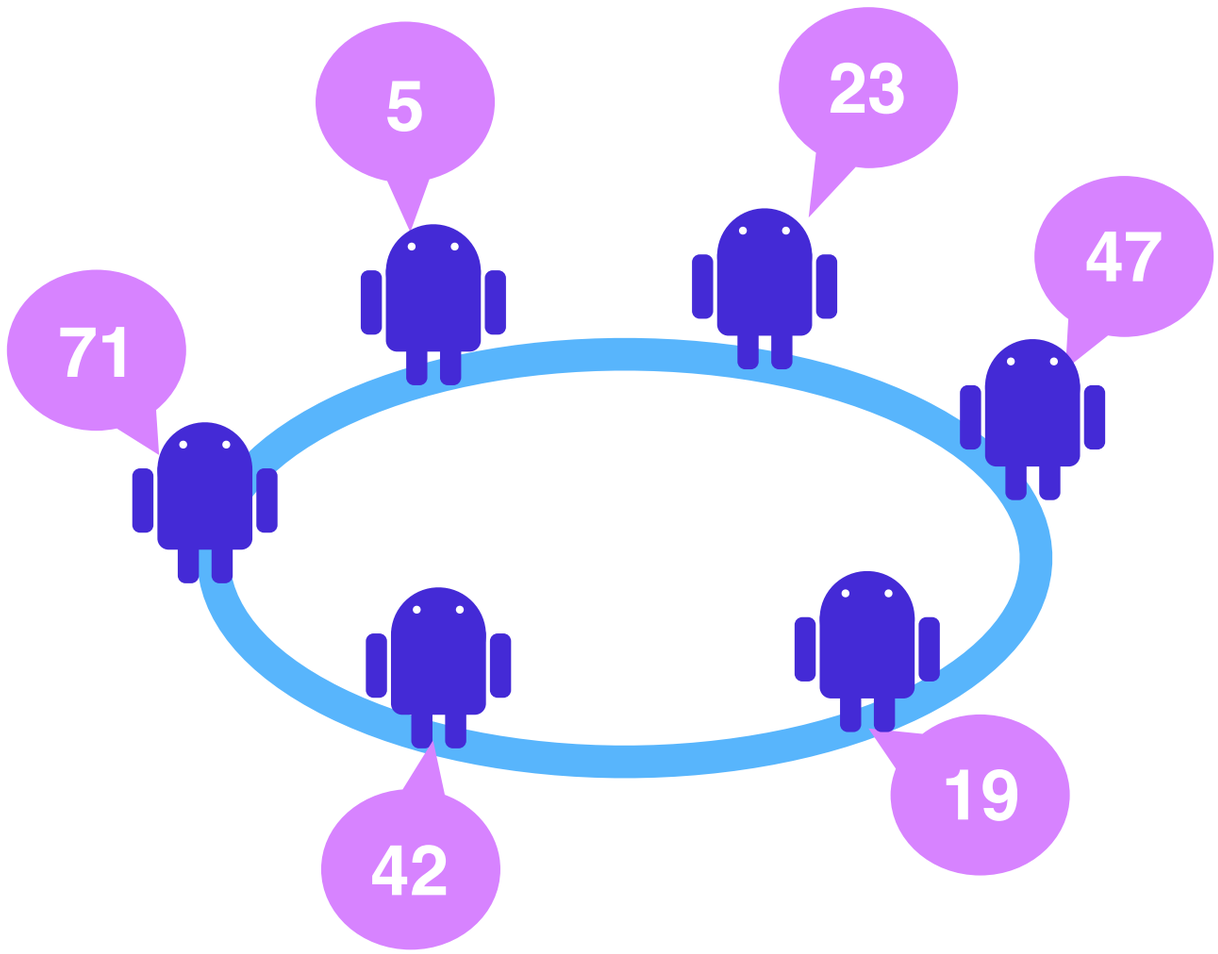


Data abstraction

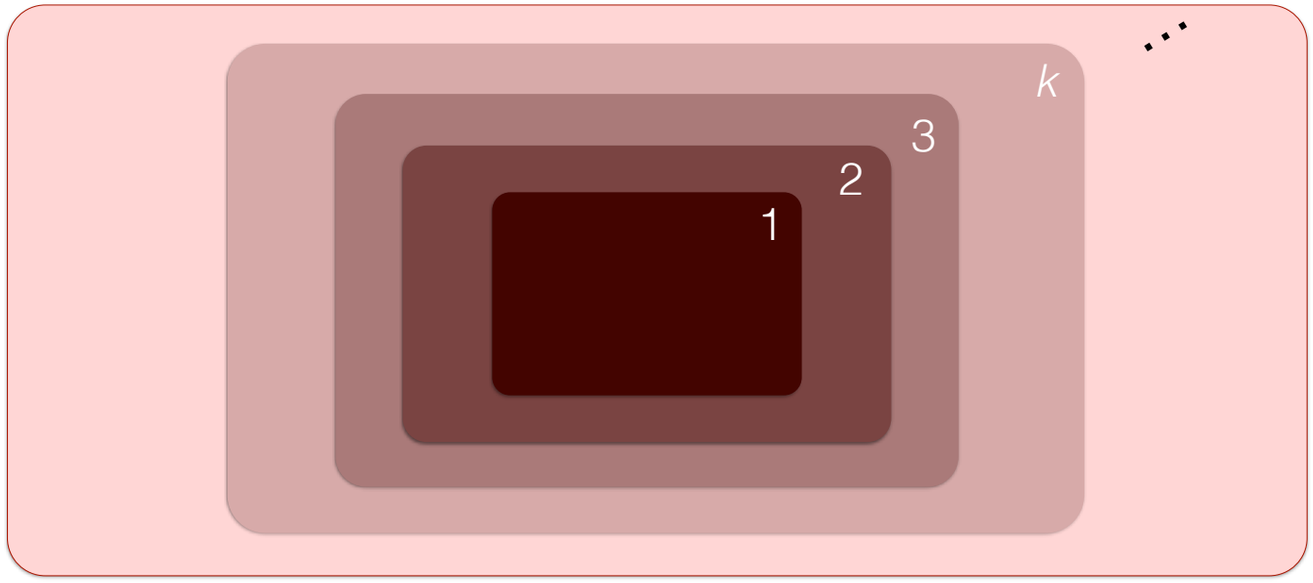




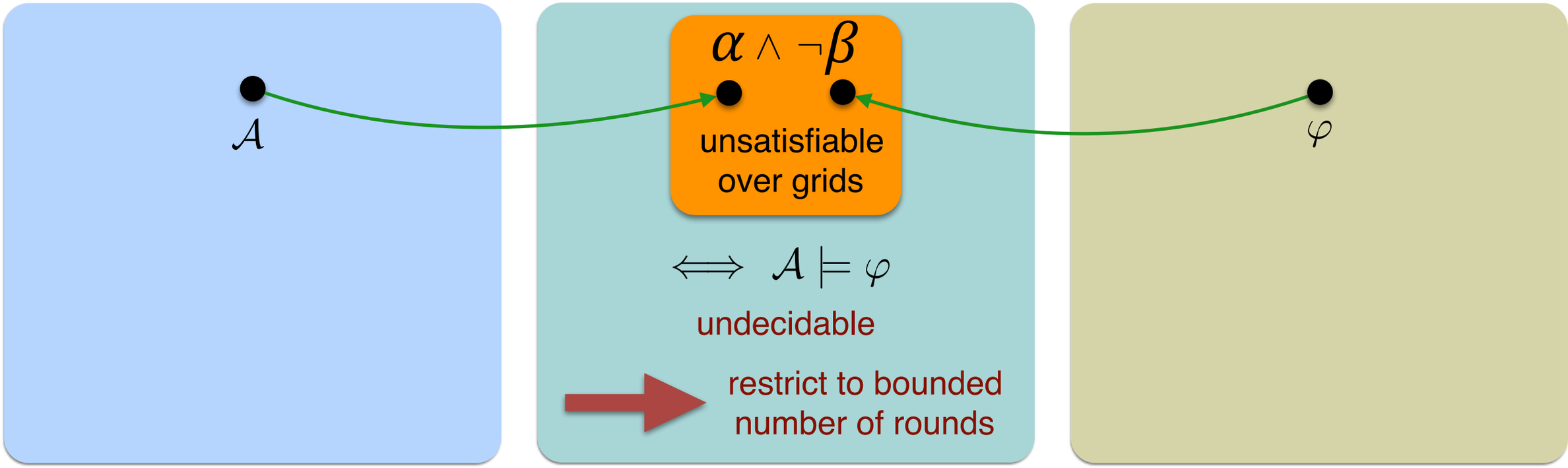
Model Checking 2



Under approximate verification



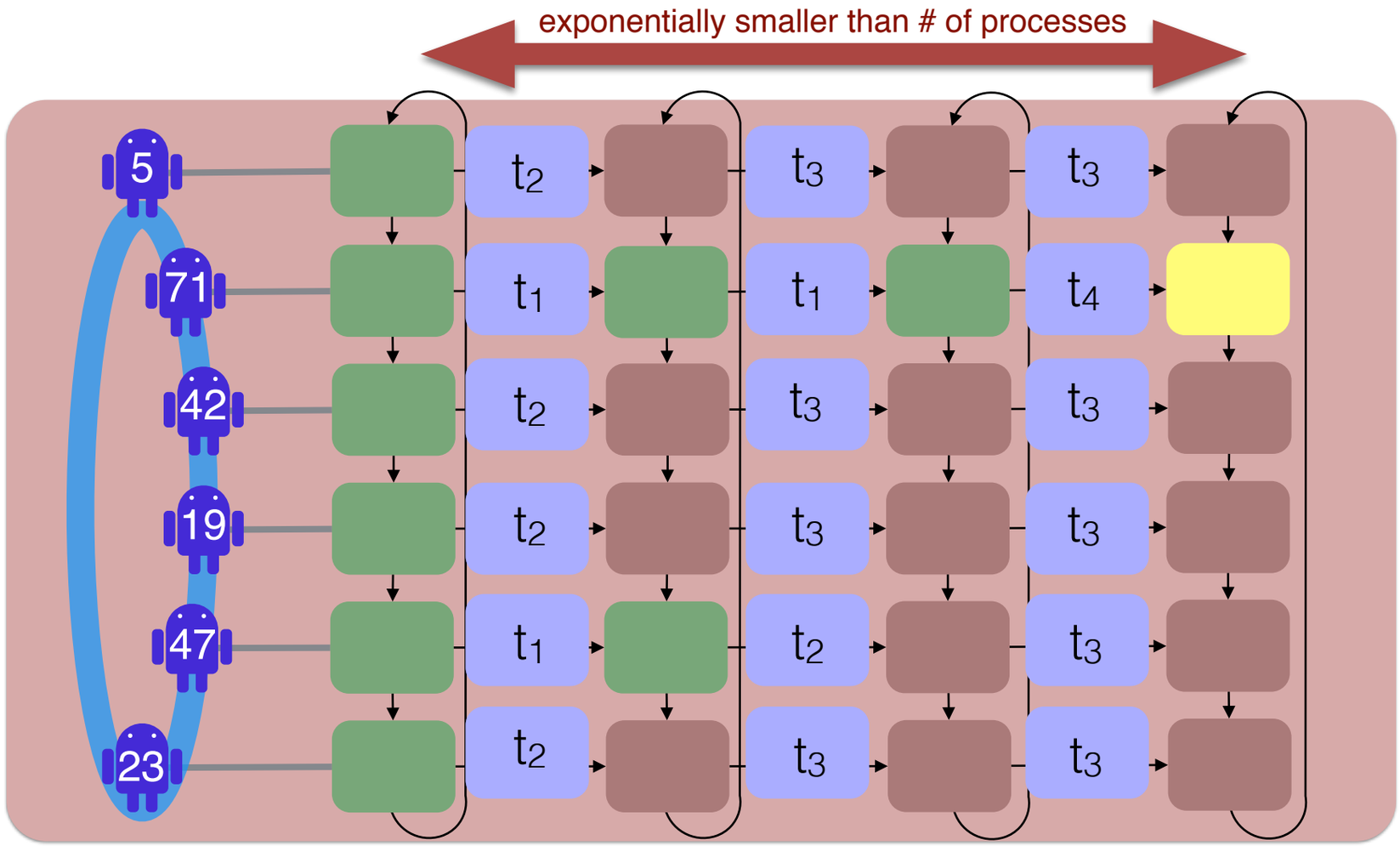
Behavior

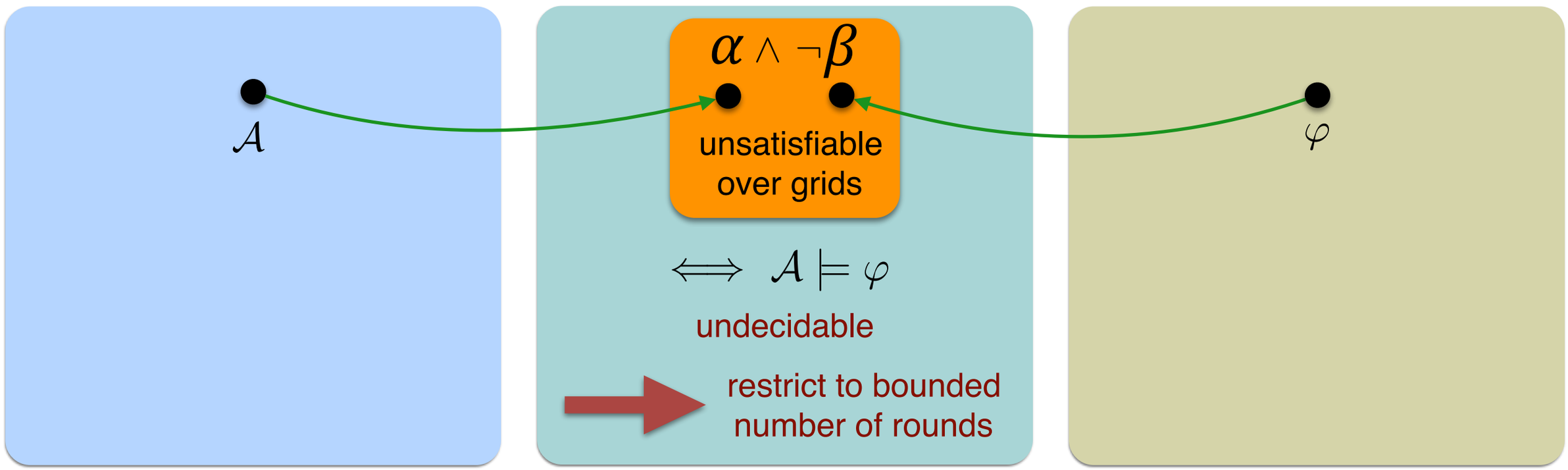


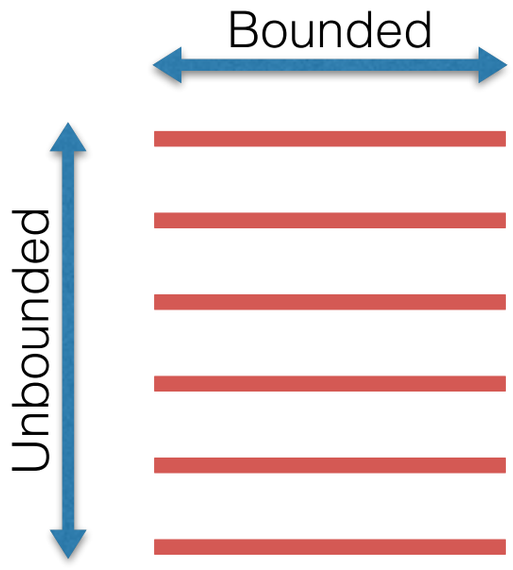
Distributed algorithm

PDL with loop (over finite alphabet)

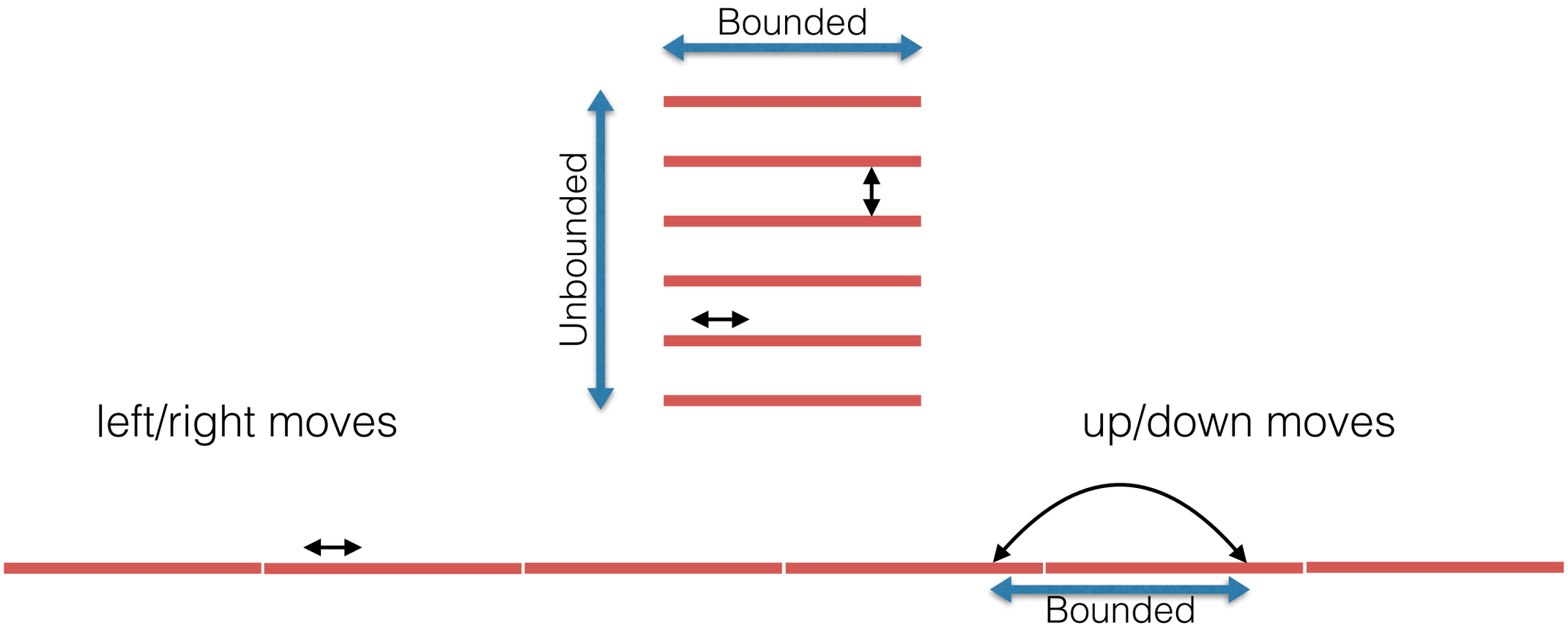
Data PDL


exponentially smaller than # of processes
5
71
42
19
47
23
t2
t3
t1
t4


$\mathcal{A}$
$\alpha \wedge \neg\beta$
unsatisfiable over grids
φ
$\iff \mathcal{A} \models \varphi$
undecidable
restrict to bounded number of rounds
Distributed algorithm
PDL with loop (over finite alphabet)
Data PDL



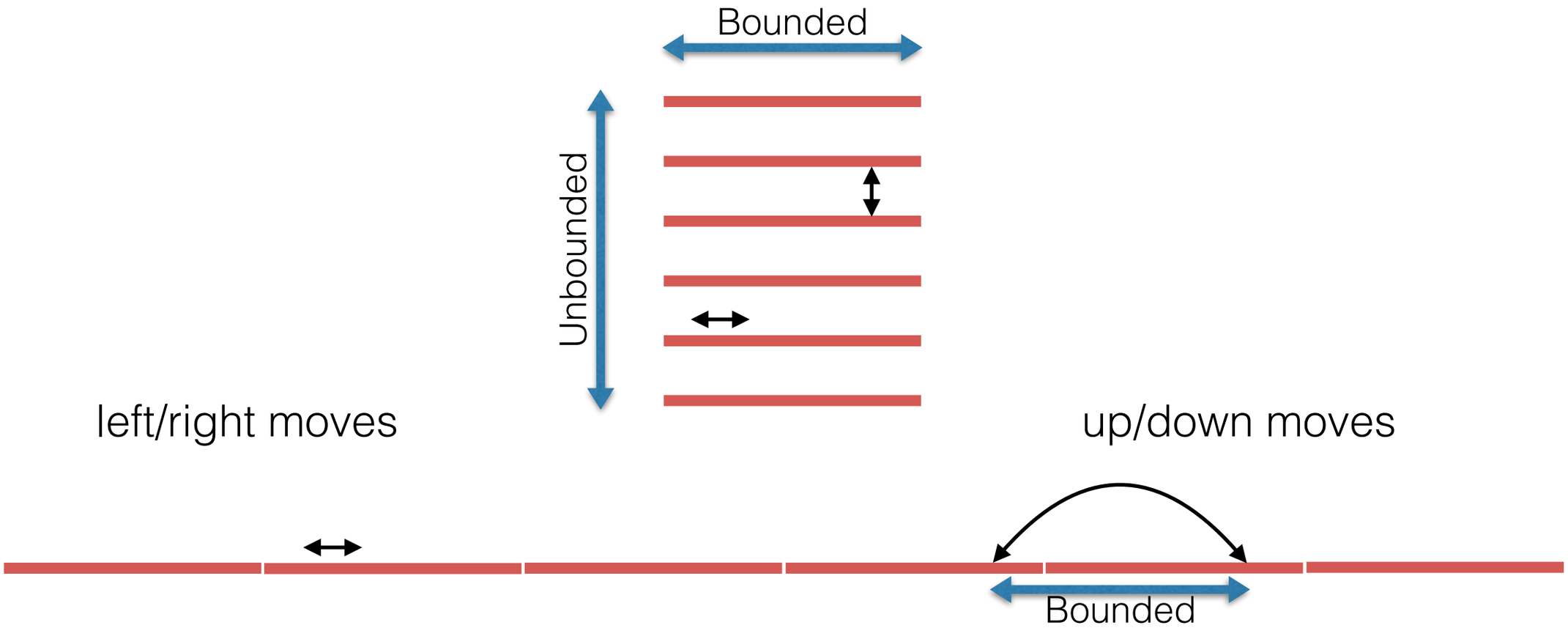
PDL with loop over bounded grids



PDL with loop over bounded grids

⇨

PDL with loop over words



PDL with loop over bounded grids

⇨

PDL with loop over words

⇨

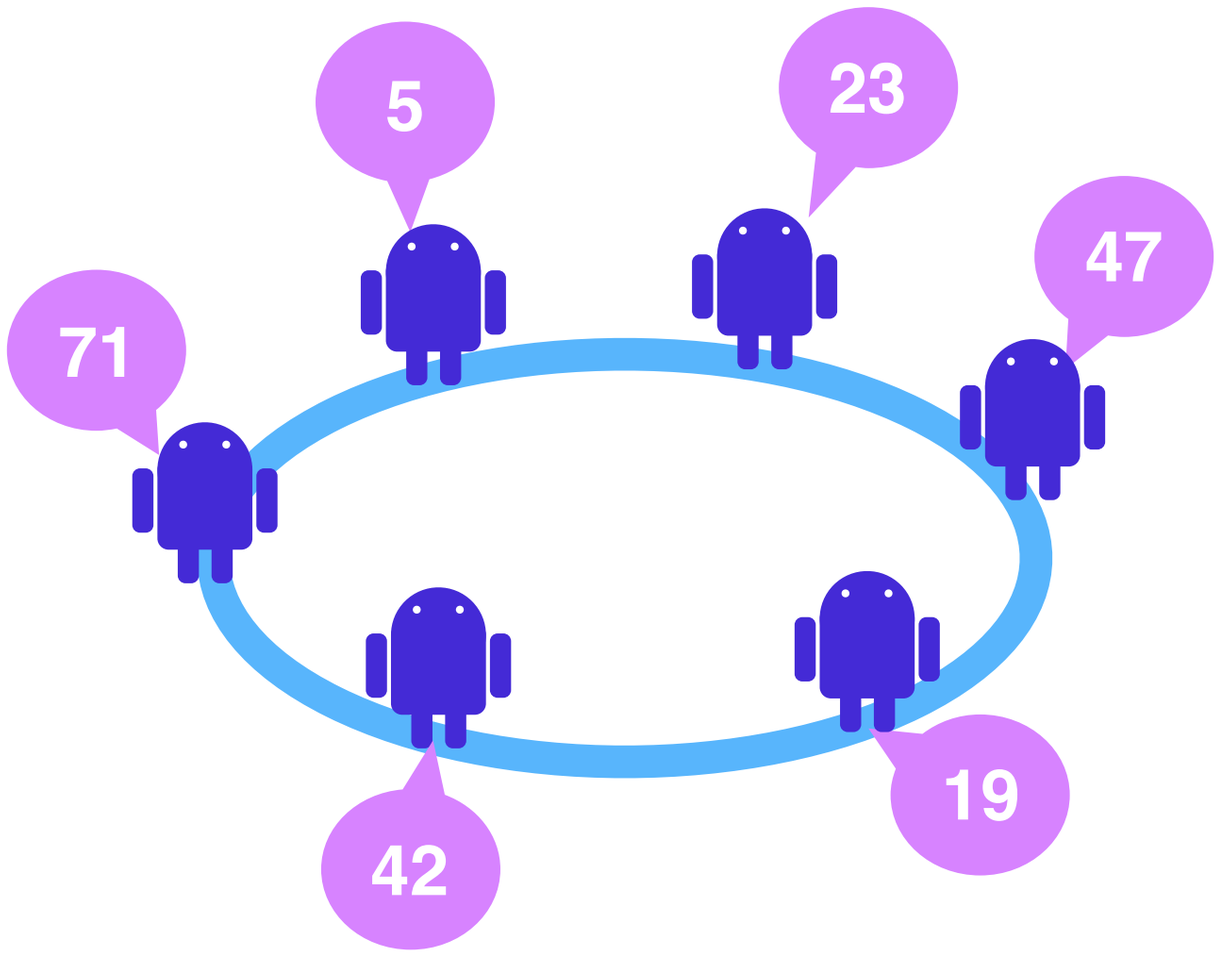
Alternating 2-way Automata

⇨

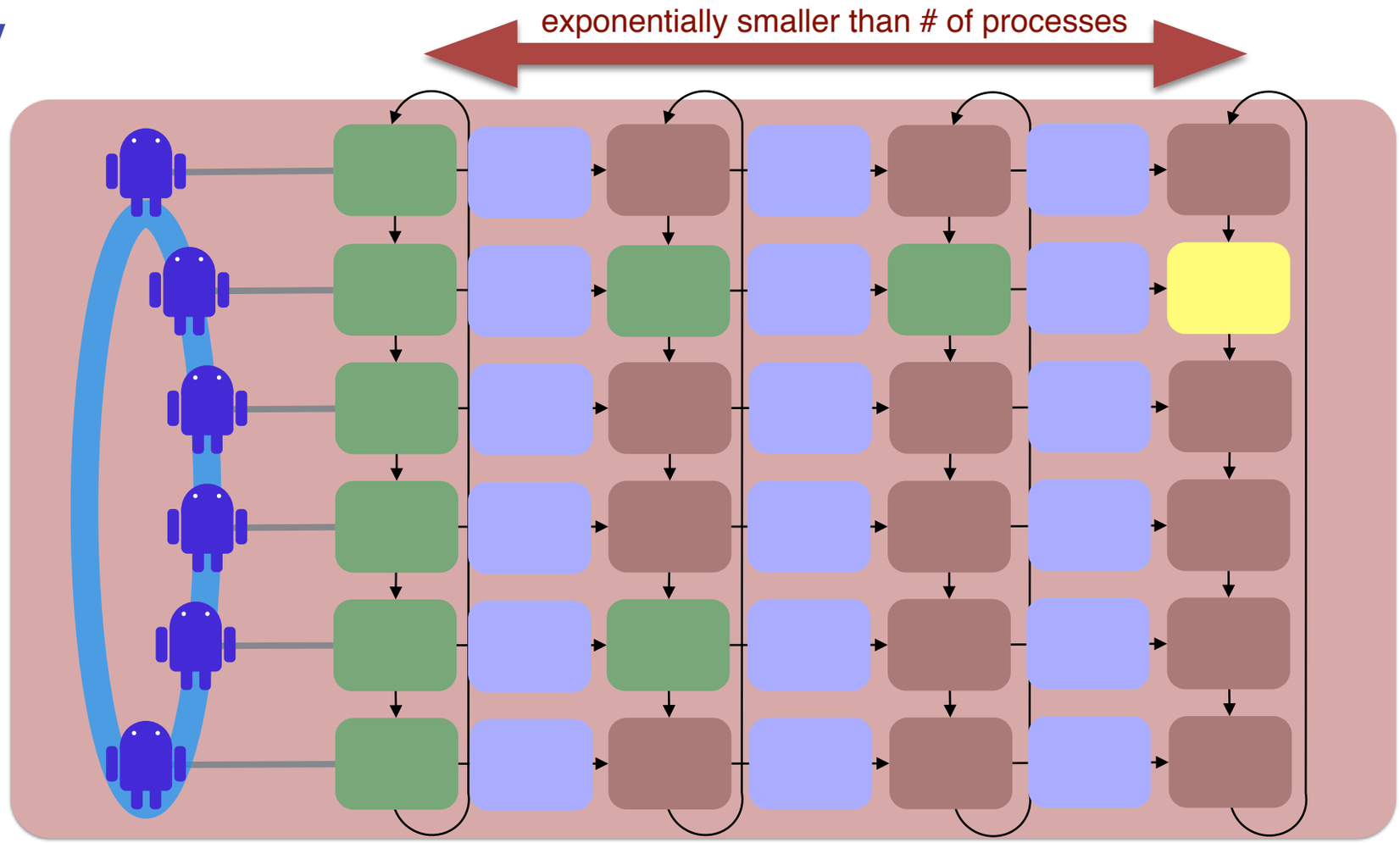
PSPACE

[Göller-Lohrey-Lutz ’08] [Serre ’08]

Summary & Conclusion



Summary



Theorem (Aiswarya-Bollig-Gastin; CONCUR '15).

Round-bounded model checking distributed algorithms* against Data PDL is PSPACE-complete**.

* with registers, register guards, and register updates

** unary encoding of # of rounds

Conclusion

- What is the right temporal logic? Use generic Data PDL.
- How to deal with data? Use symbolic technique.

Independent of
- **number of rounds**
- **restriction to rings**

Conclusion

- What is the right temporal logic? Use generic Data PDL.
- How to deal with data? Use symbolic technique.
- How to deal with undecidability? Under-approximation.

Independent of
- **number of rounds**
- **restriction to rings**

Future work …

- Other operations?
- Other topologies?
- Other restrictions?
- Other communications?


Thank you!