Formal Methods for the Verification of Distributed Algorithms

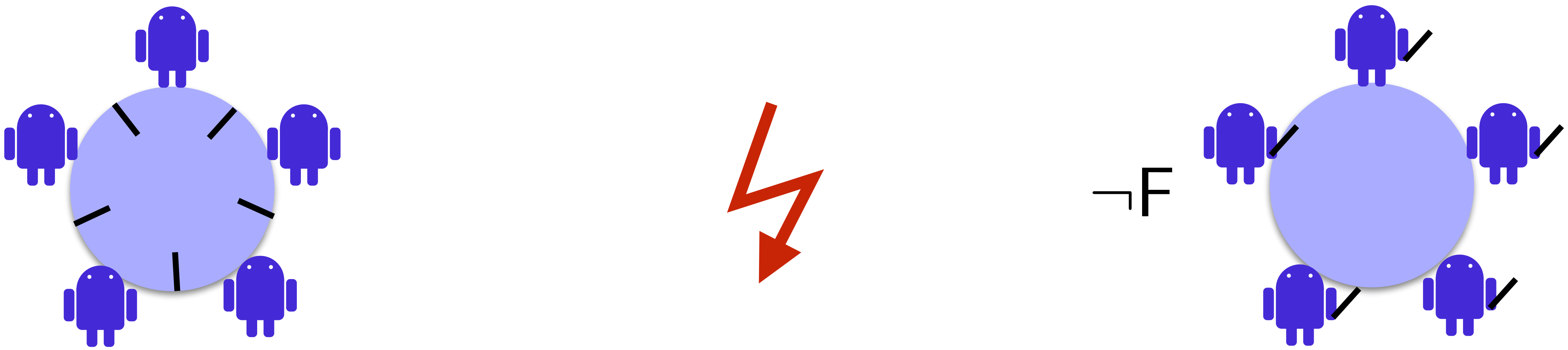
Paul Gastin

Laboratoire Spécification et Vérification
ENS Cachan, CNRS & Inria

Joint work with C. Aiswarya & Benedikt Bollig

Infinity
December 15, 2015

Landscape & Objectives

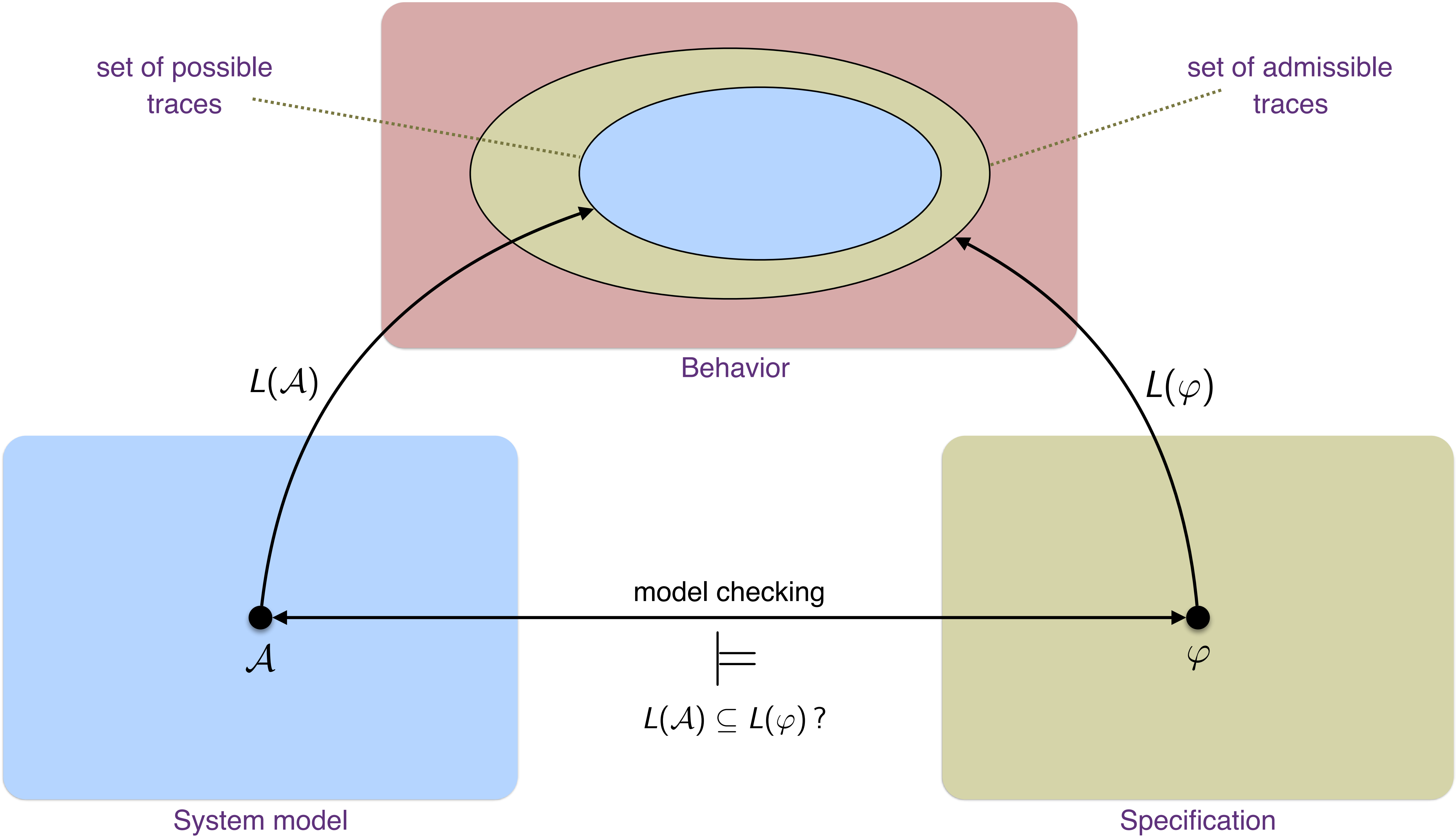


distributed system

$\neg F$

specification

Landscape & Objectives



set of possible traces

set of admissible traces

Behavior

$L(\mathcal{A})$

$L(\varphi)$

model checking

$\mathcal{A}$

$\models$

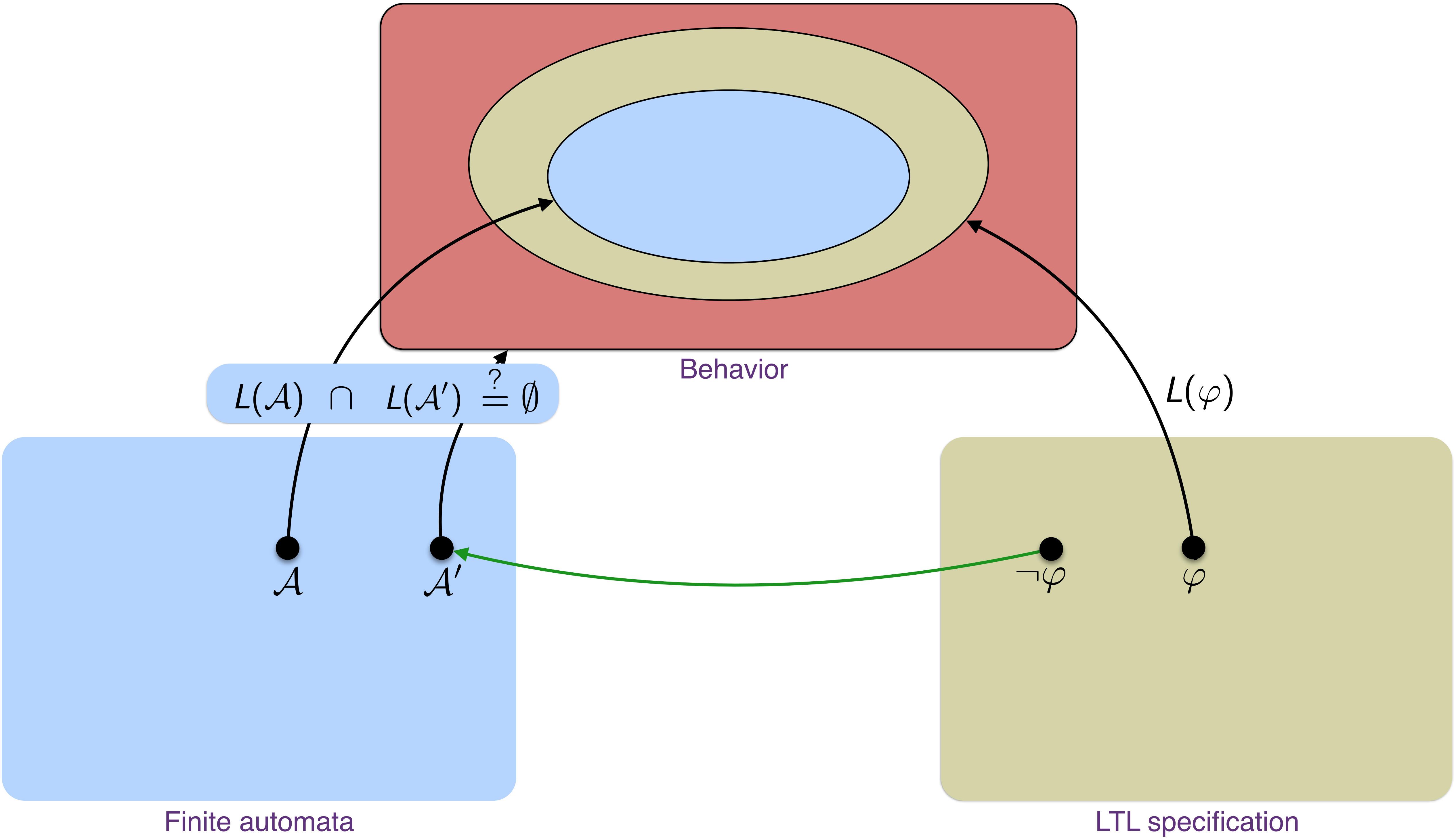
φ

$L(\mathcal{A}) \subseteq L(\varphi)$?

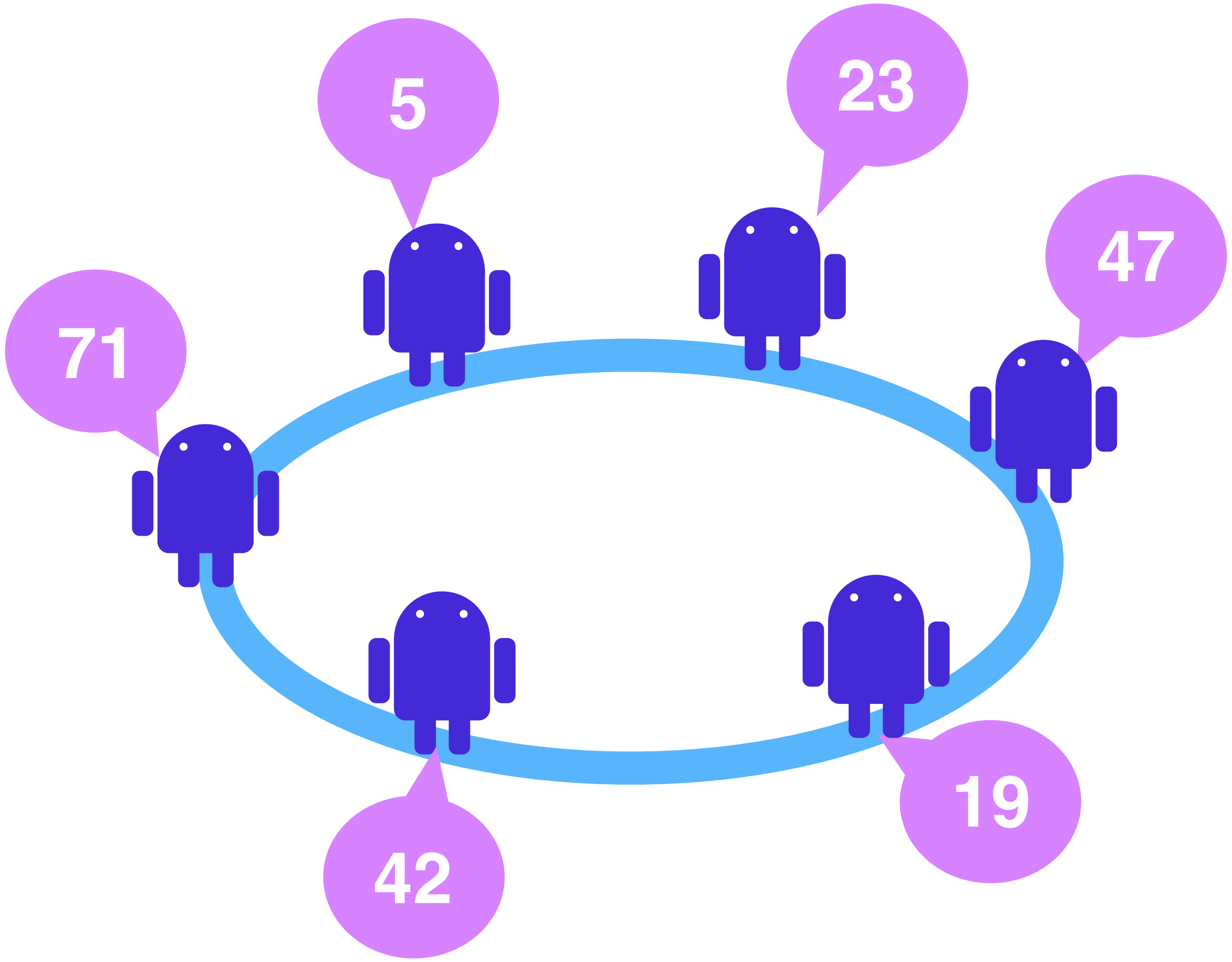
System model

Specification

Landscape & Objectives



Models of Distributed Systems



Landscape & Objectives

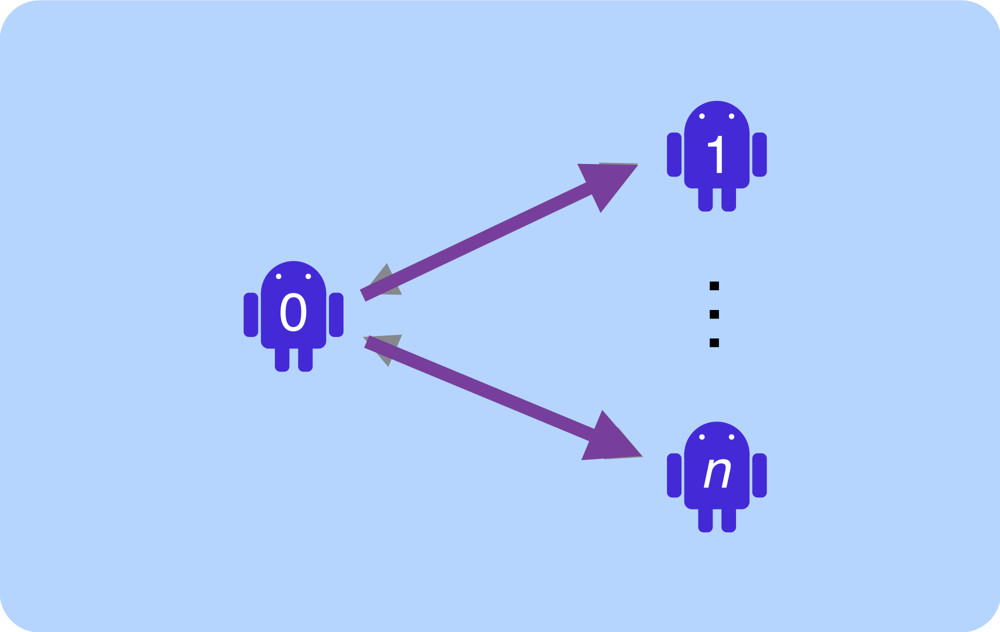
Topology

- tree, ring, star, …

Number of processes

- fixed & static
- non-fixed & unbounded

  static (parameterized)



System model

$\models$

for all n

φ

Specification

Landscape & Objectives

Topology

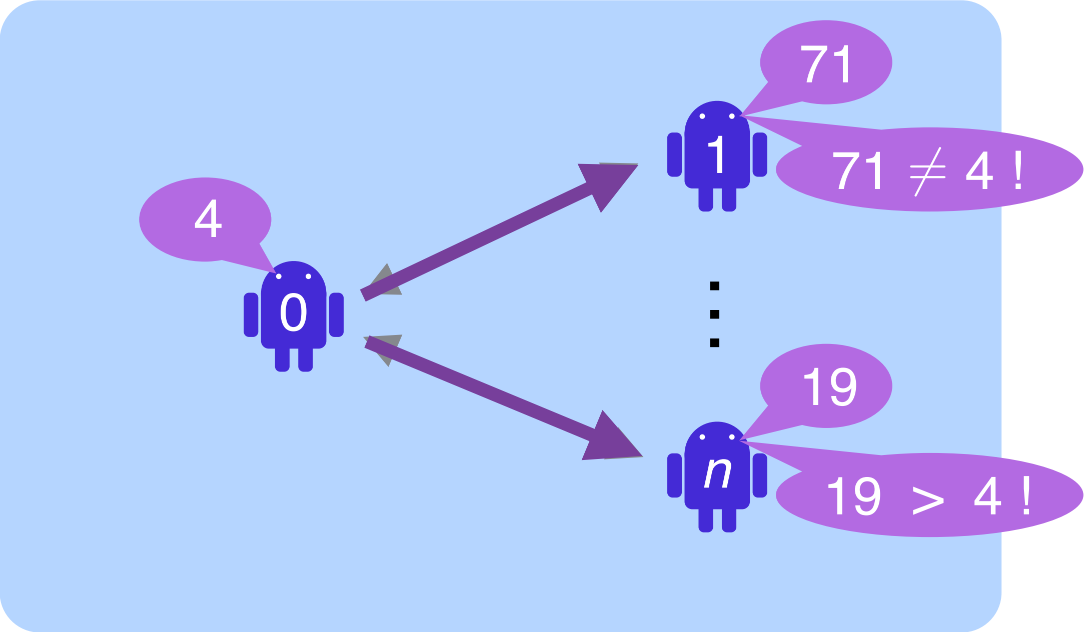
- tree, ring, star, …

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



System model

$$\models$$

$$\varphi$$

Specification

Landscape & Objectives

Topology

- tree, ring, star, …

Number of processes

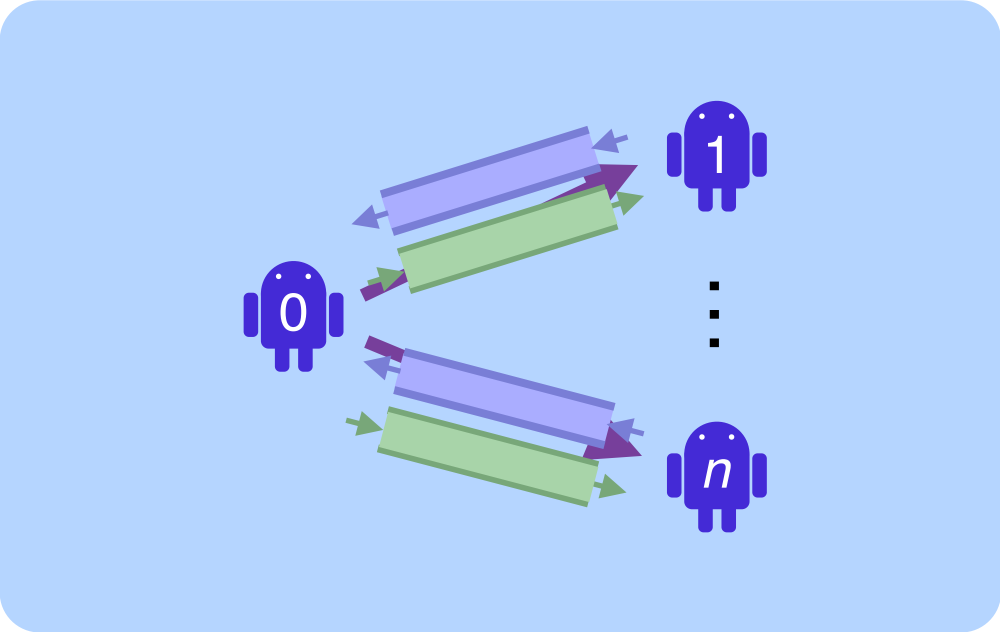
- fixed & static
- non-fixed & unbounded
    - static (parameterized)
    - dynamic

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



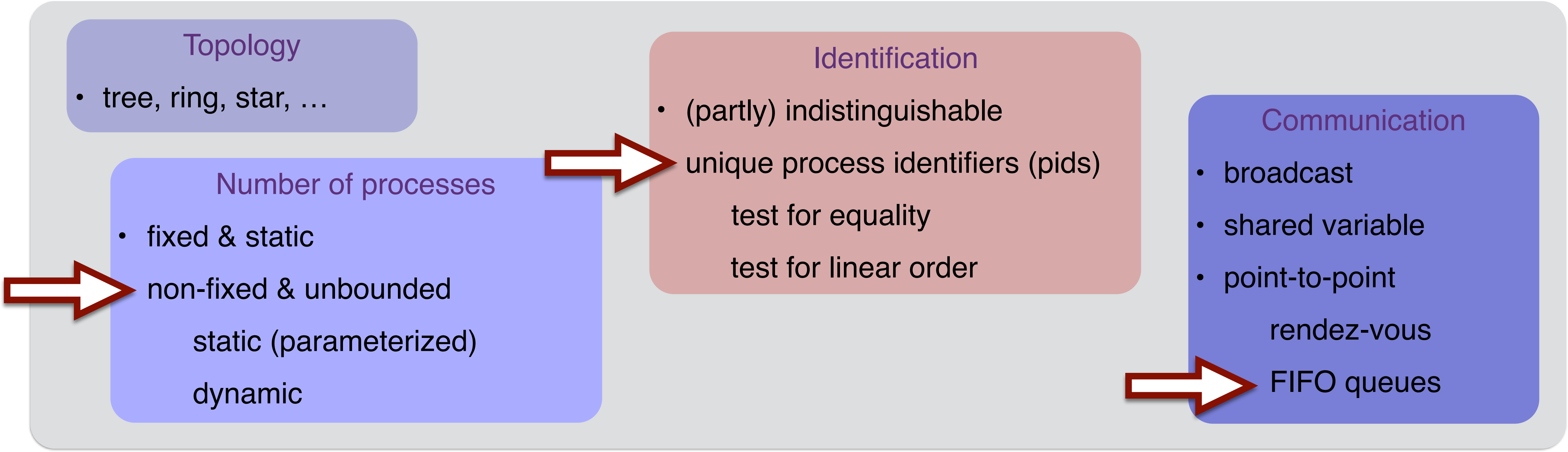
System model

$\models$

φ

Specification

Landscape & Objectives



Topology

- tree, ring, star, …

Number of processes

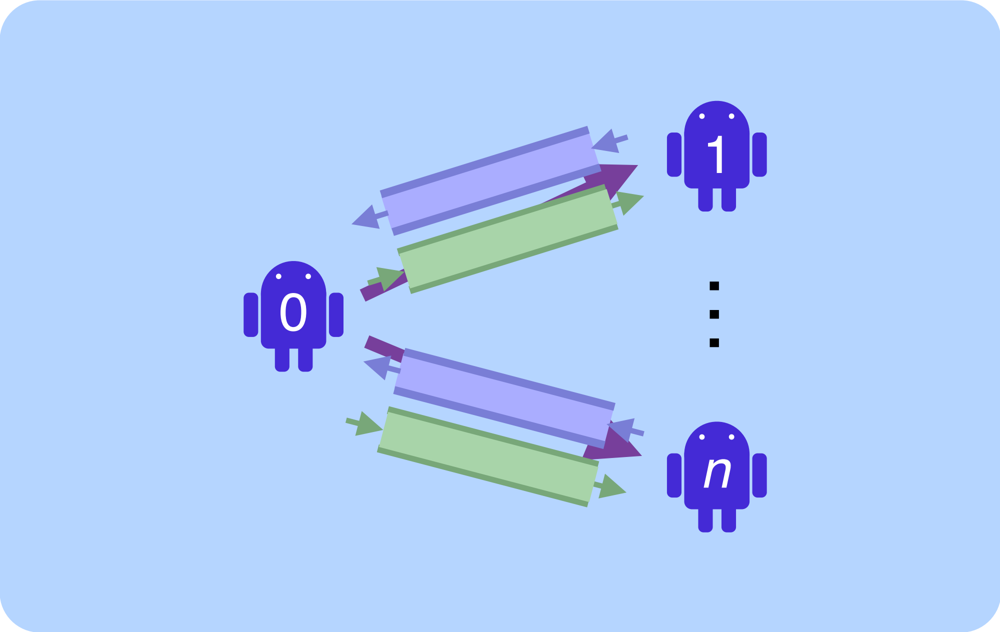
- fixed & static
- non-fixed & unbounded
    - static (parameterized)
    - dynamic

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

$$\models$$

$$\varphi$$

Specification

Several sources of infinity / unboundedness

Landscape & Objectives

Topology

- tree, ring, star, …

Number of processes

- fixed & static
- non-fixed & unbounded
  - static (parameterized)
  - dynamic

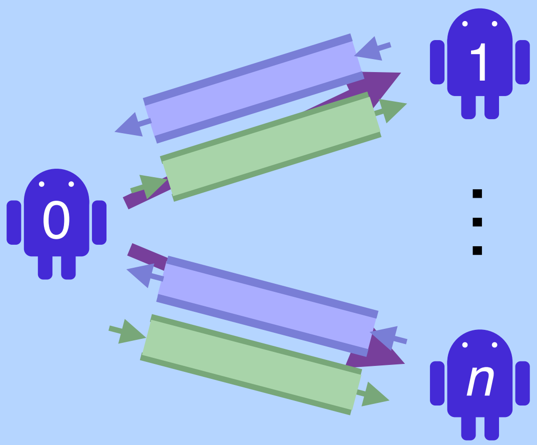
Identification

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

Communicating automata
[Brand-Zafiropulo ’83]

Communication

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



System model

$$\models$$

$$\varphi$$

Specification

Several sources of infinity / unboundedness

Landscape & Objectives

Topology

- tree, ring, star, …

Number of processes

- fixed & static
- non-fixed & unbounded
  - static (parameterized)
  - dynamic

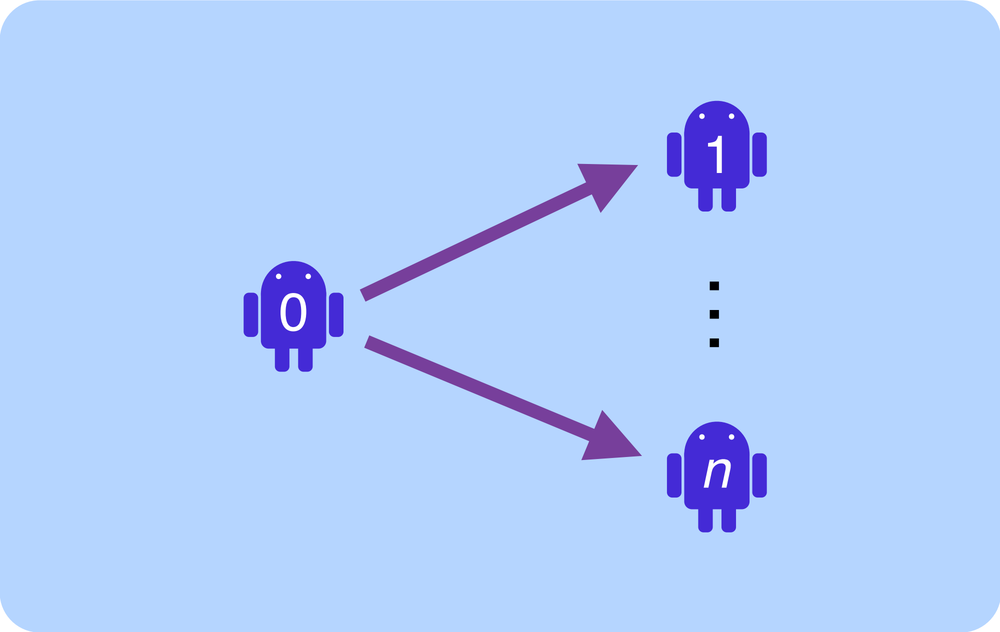
Identification

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

Data automata
[Bojanczyk et al. ’06]
[Björklund-Schwentick ’07]

Communication

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



System model

$$\models$$

$$\varphi$$

Specification

Several sources of infinity / unboundedness

Landscape & Objectives

Topology

- tree, ring, star, …

Number of processes

- fixed & static
- non-fixed & unbounded
  - static (parameterized)
  - dynamic

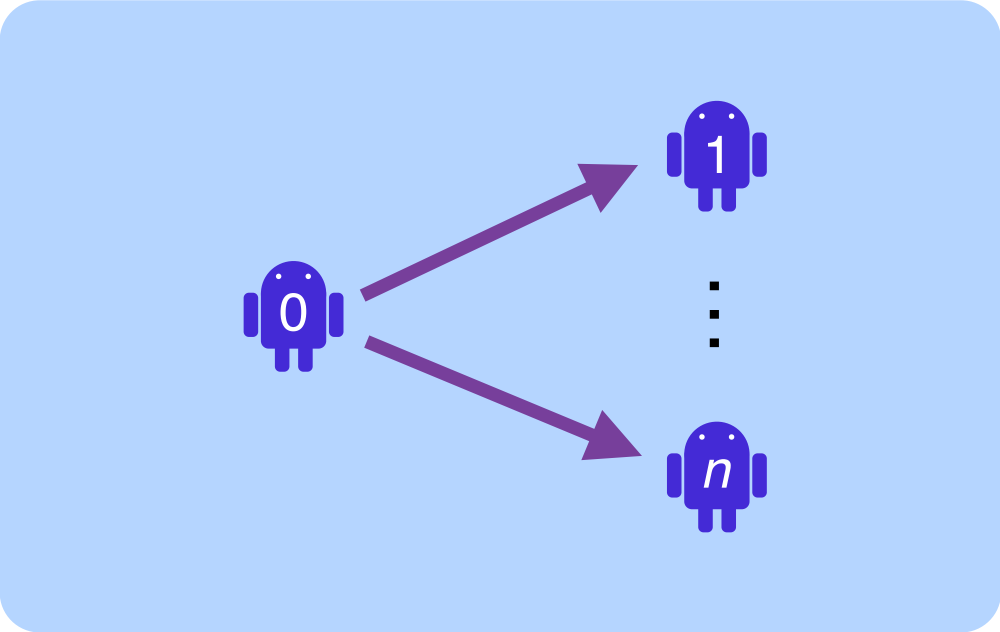
Identification

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

Communication

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

Distributed algorithms



System model

$$\models$$

$$\varphi$$

Specification

Several sources of infinity / unboundedness

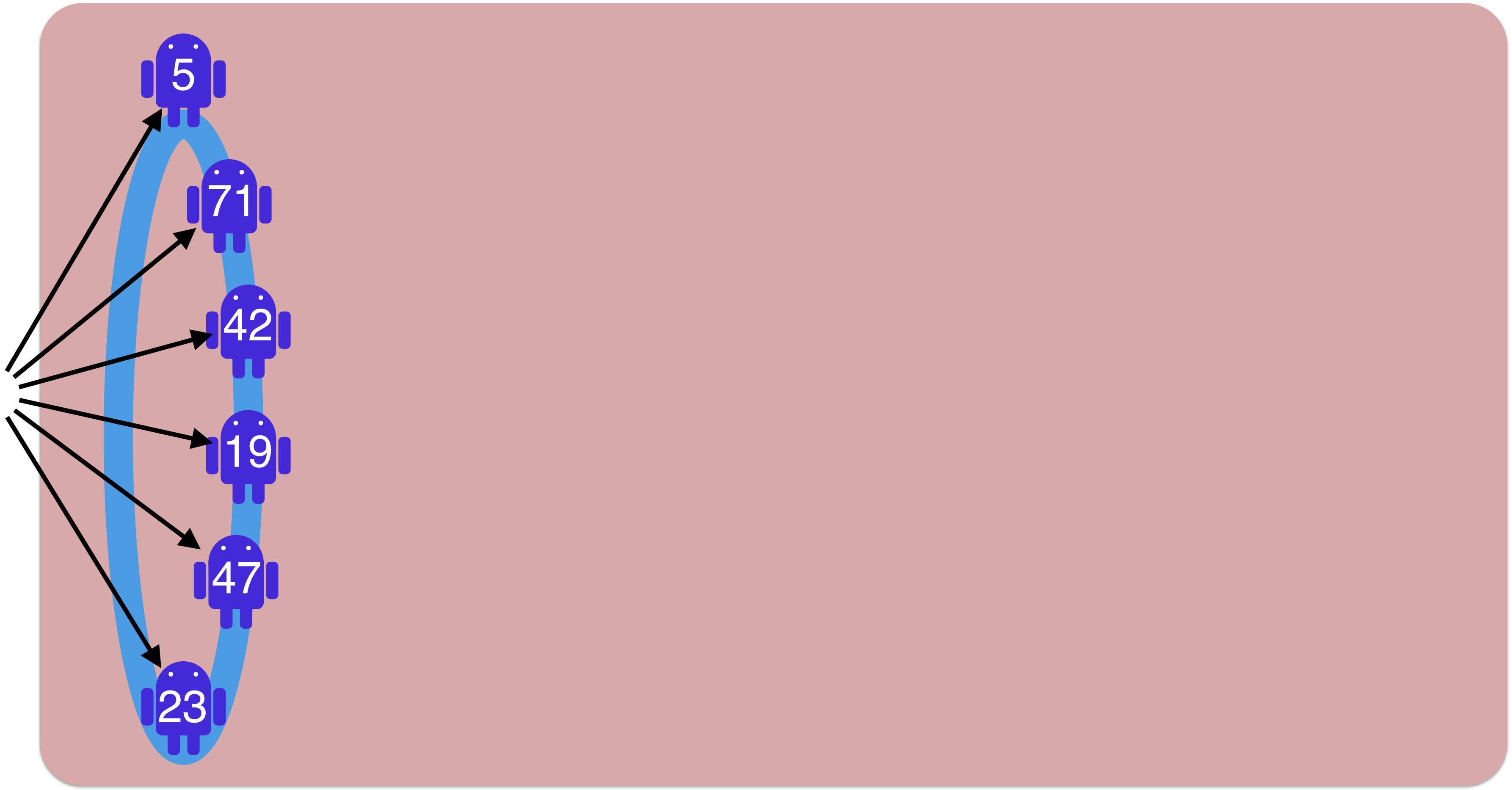
Distributed algorithms

Leader election [Franklin ’82]

arbitrary distribution
of process identifiers

5 < 19 < 23 < …

elect process
with maximum id

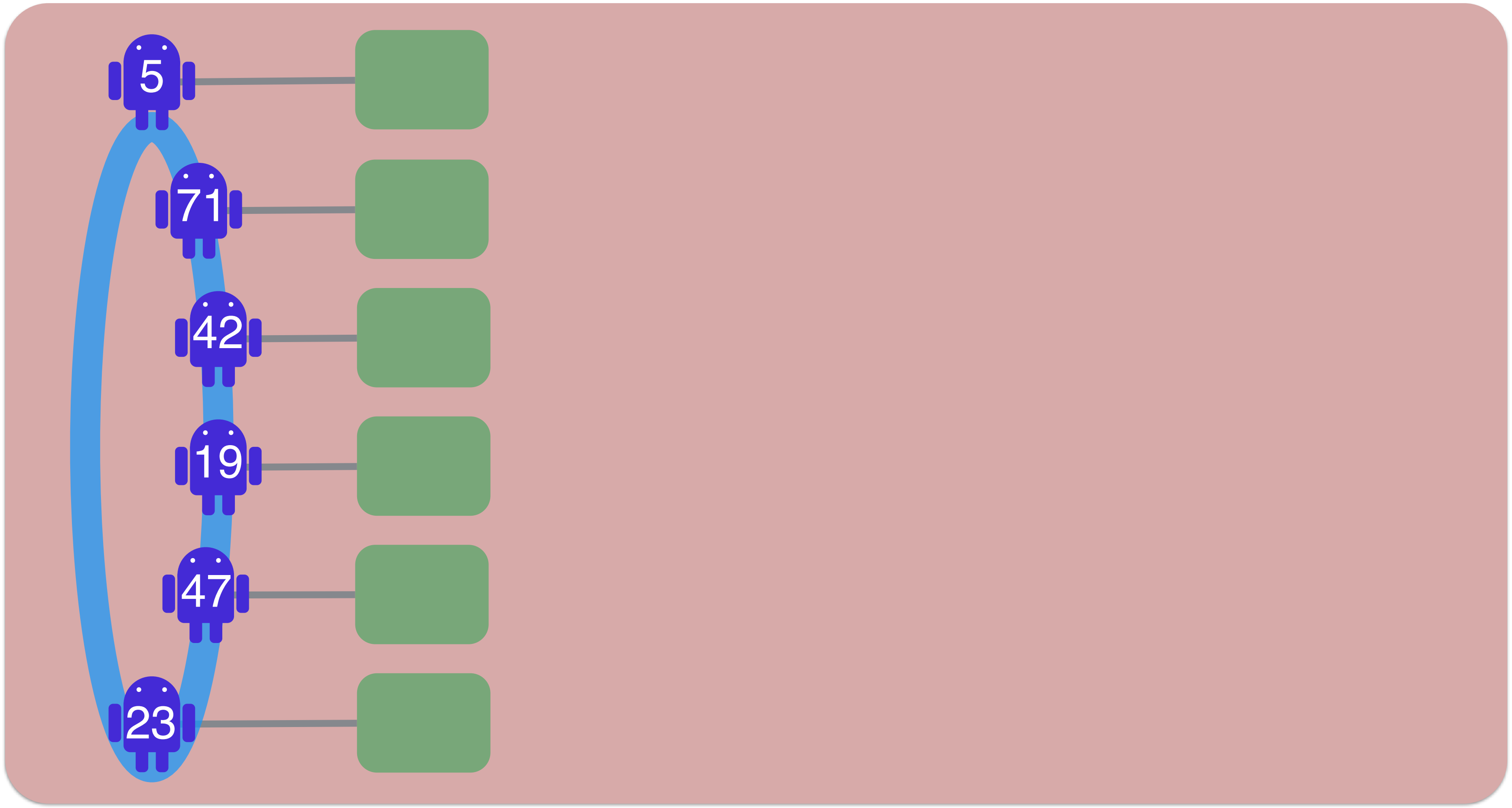


Behavior

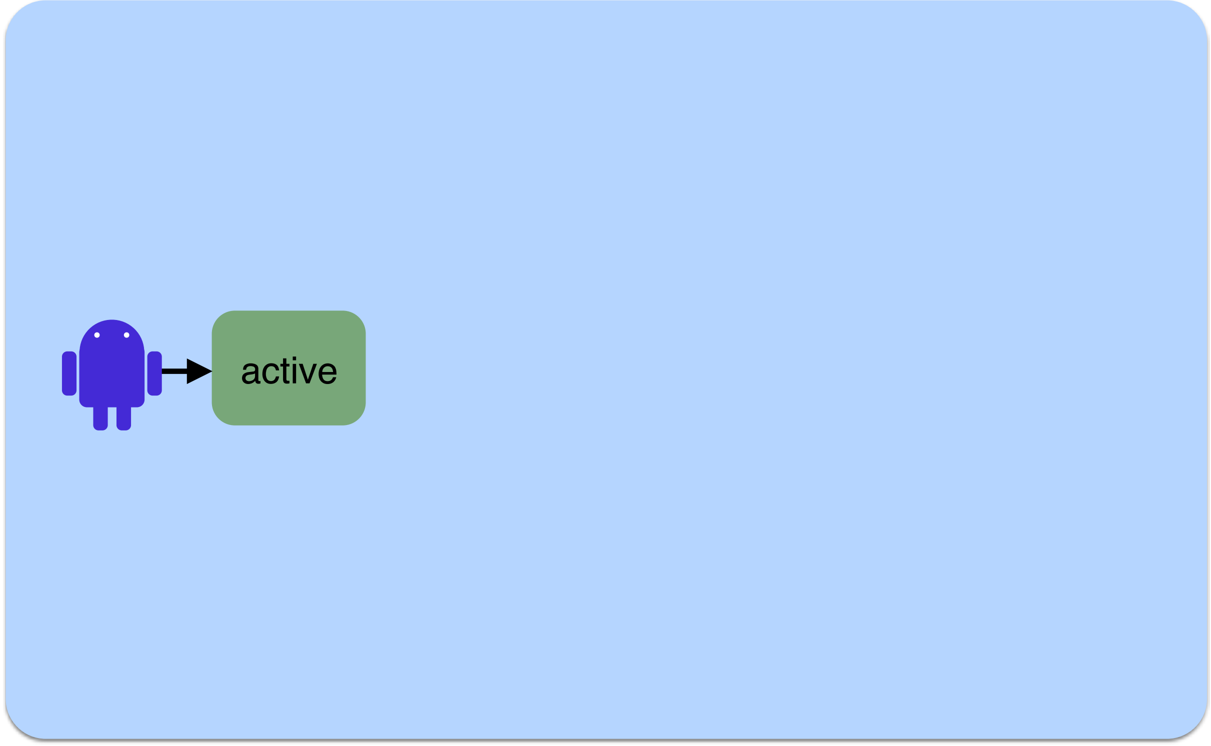
Distributed algorithm

Distributed algorithms

Leader election [Franklin ’82]



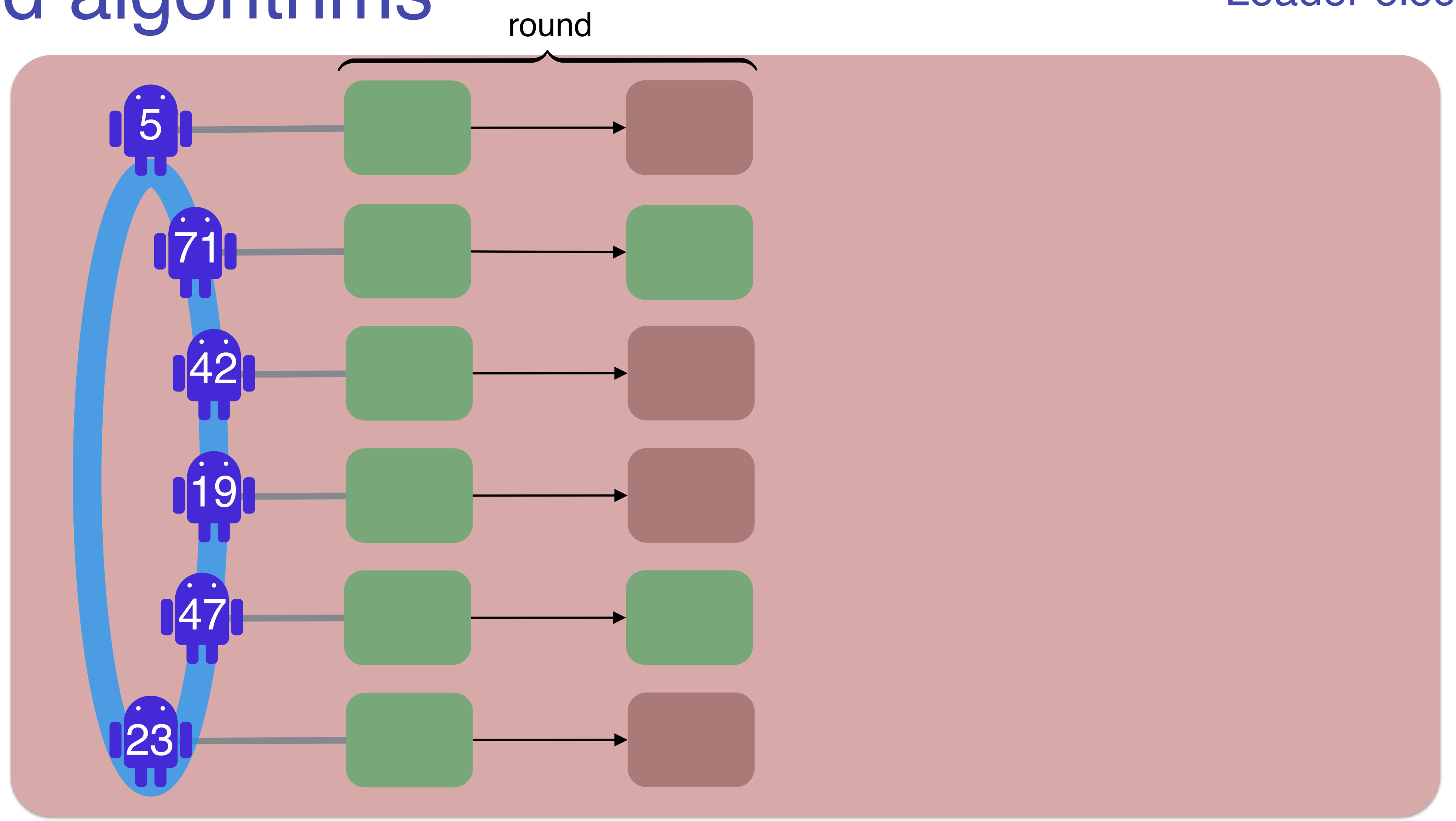
Behavior



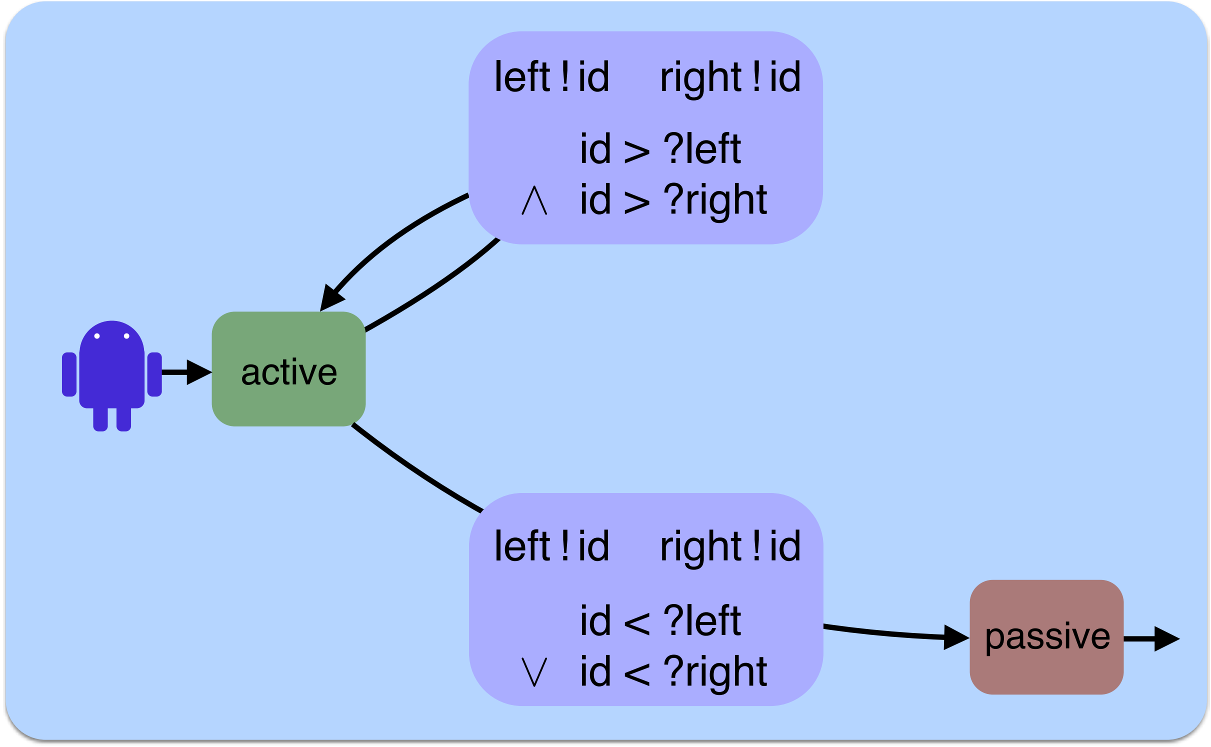
Distributed algorithm

Distributed algorithms

Leader election [Franklin ’82]



Behavior



left ! id   right ! id

id > ?left
∧ id > ?right

active

left ! id   right ! id

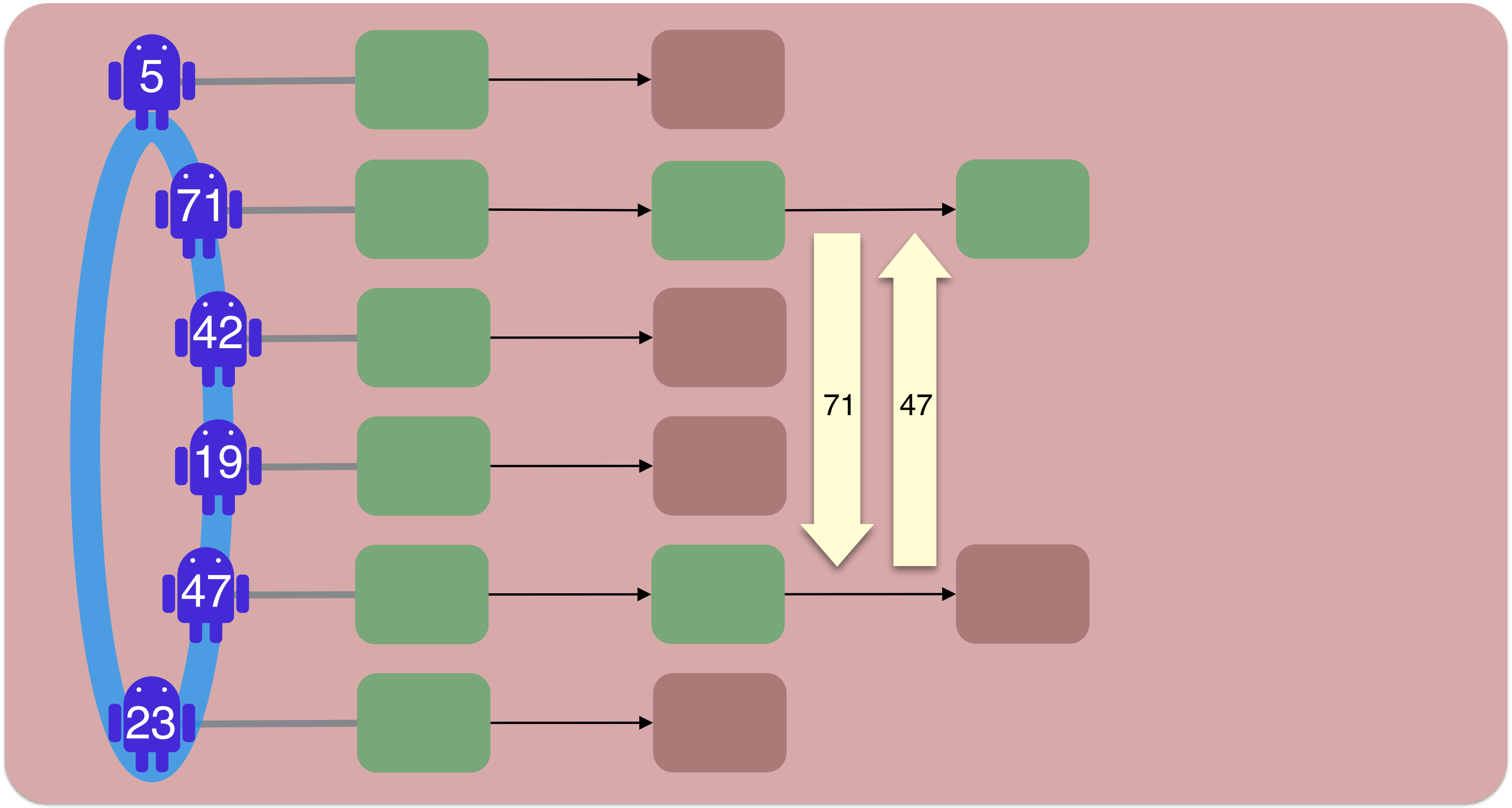
id < ?left
∨ id < ?right

passive

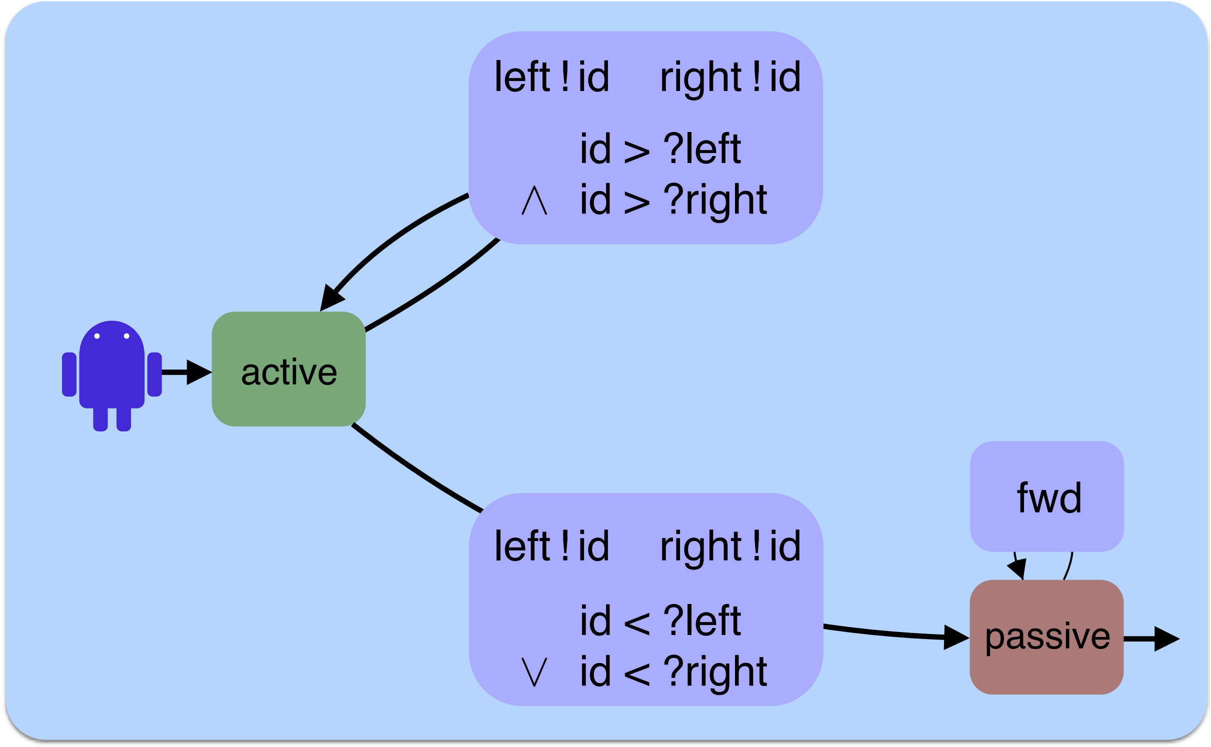
Distributed algorithm

Distributed algorithms

Leader election [Franklin ’82]



Behavior



left ! id right ! id

id > ?left
∧ id > ?right

active

left ! id right ! id

id < ?left
∨ id < ?right

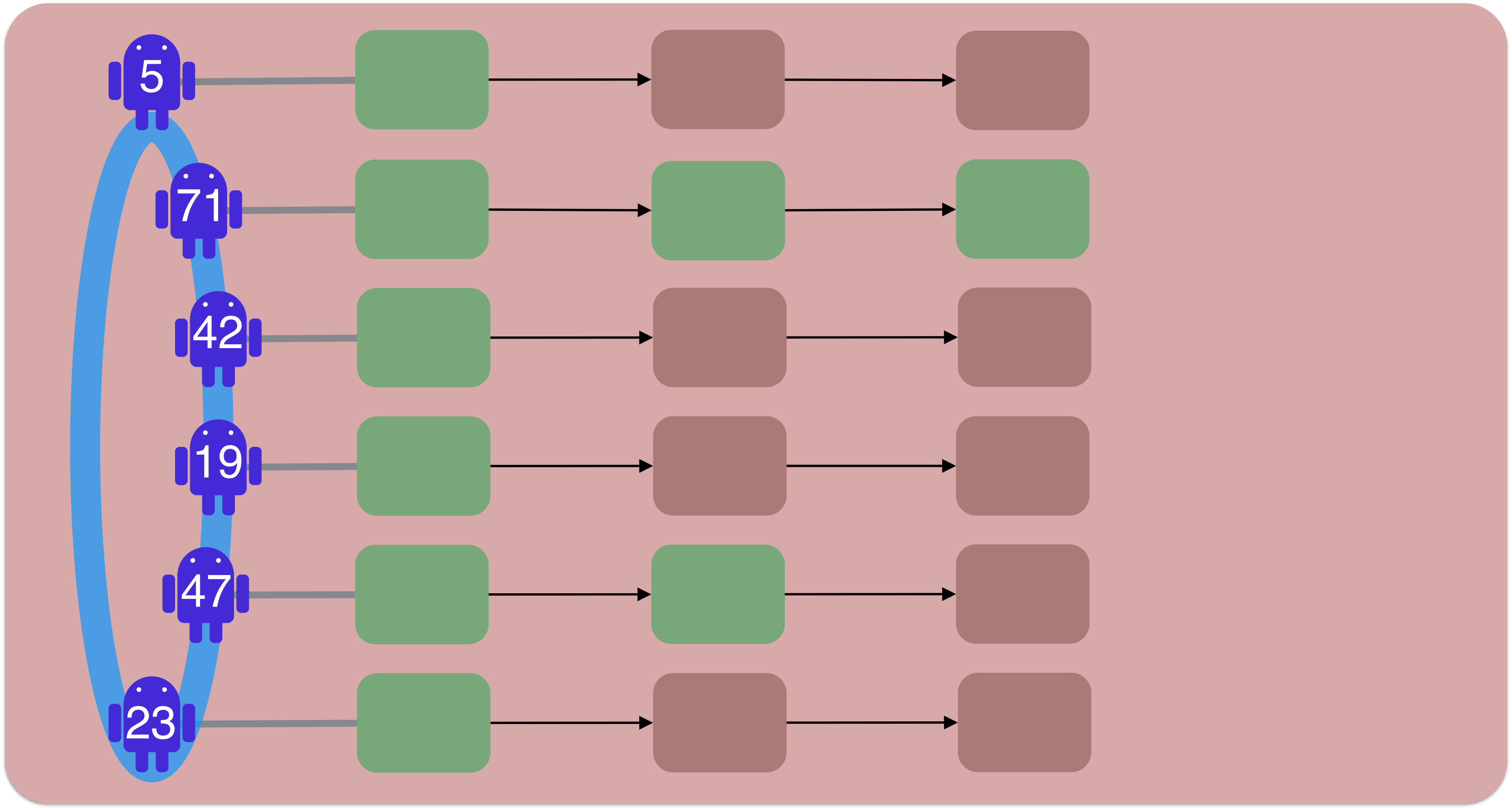
fwd

passive

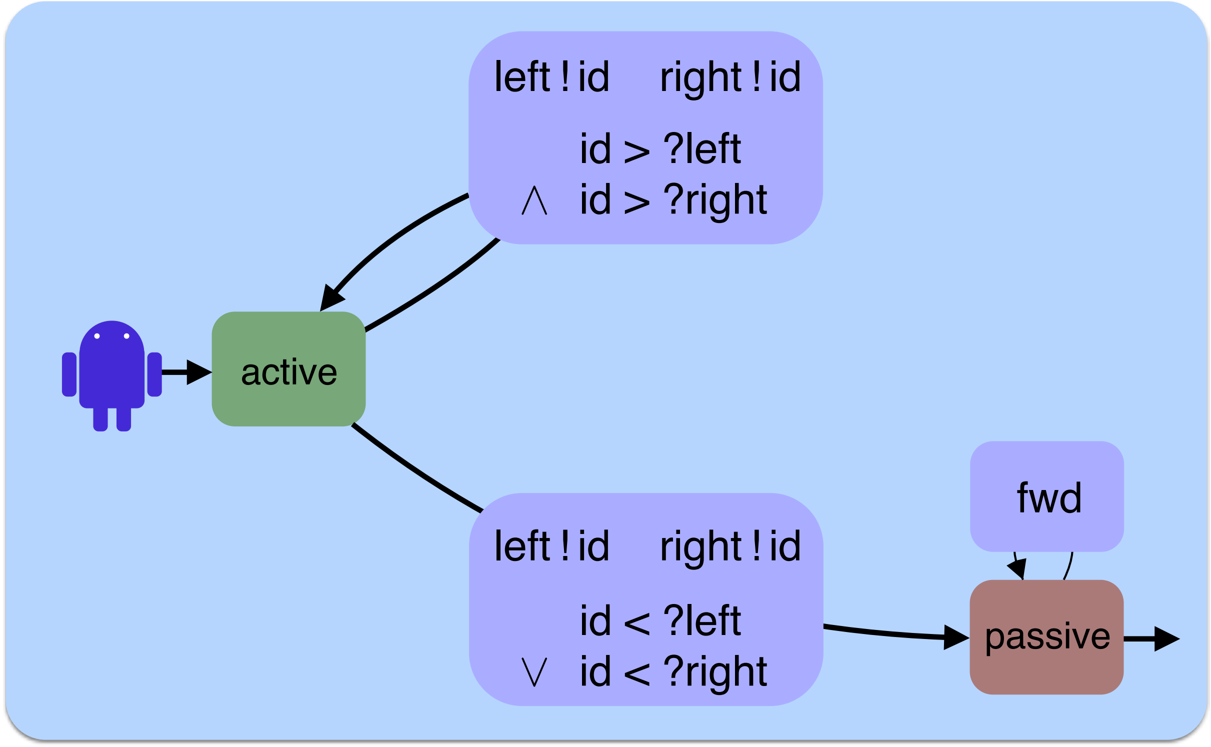
Distributed algorithm

Distributed algorithms

Leader election [Franklin ’82]



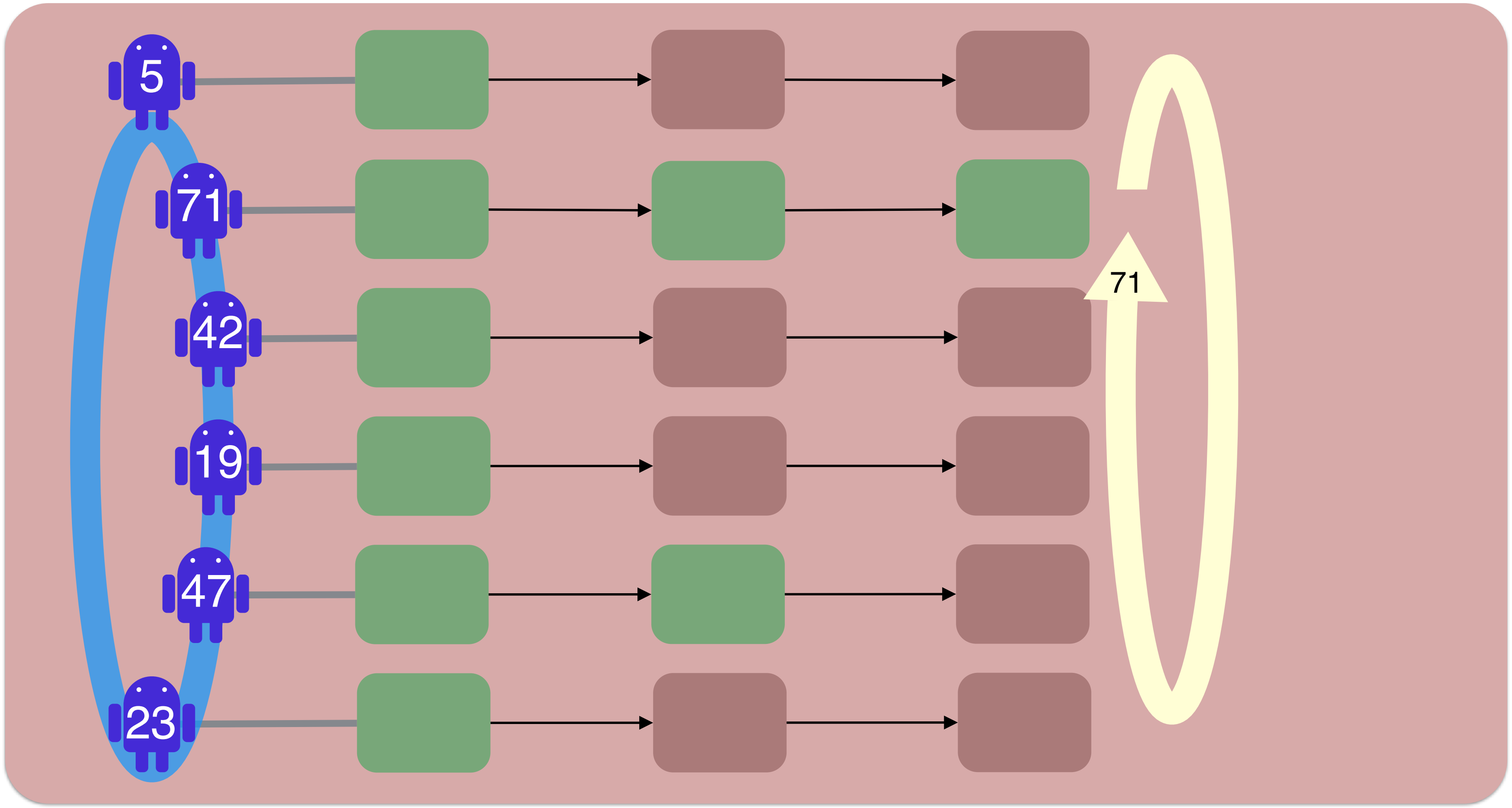
Behavior



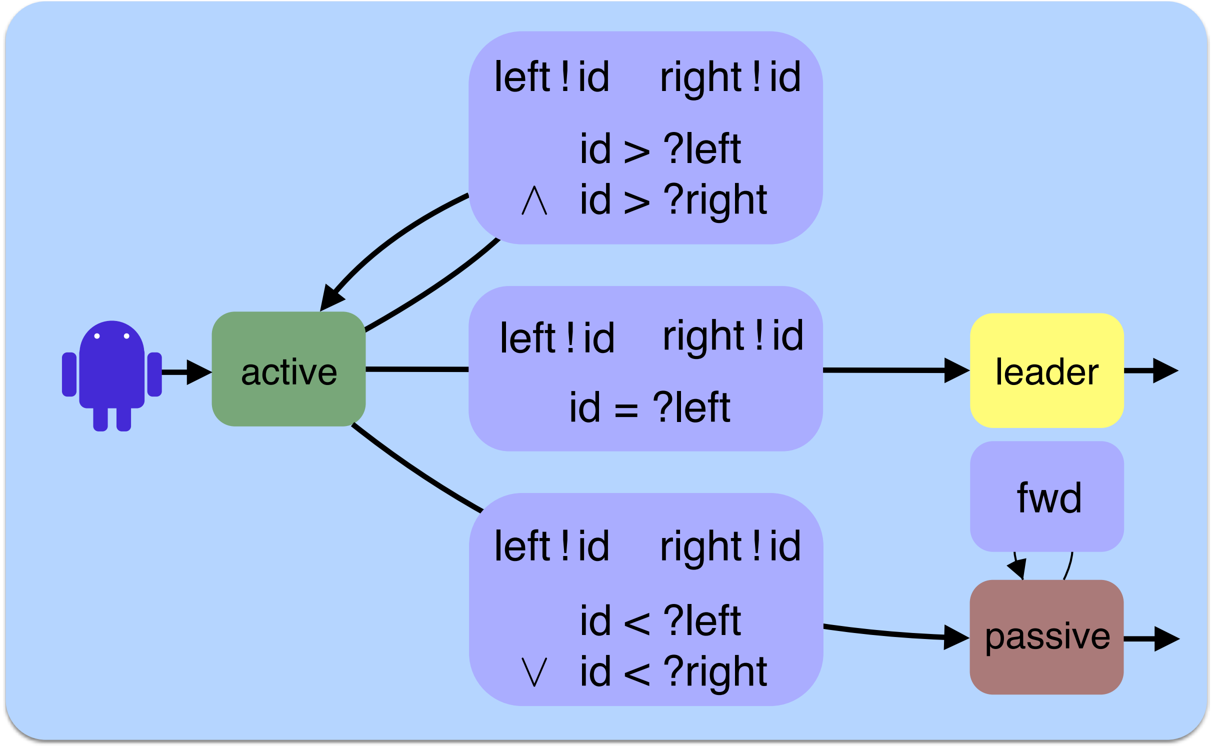
Distributed algorithm

Distributed algorithms

Leader election [Franklin ’82]



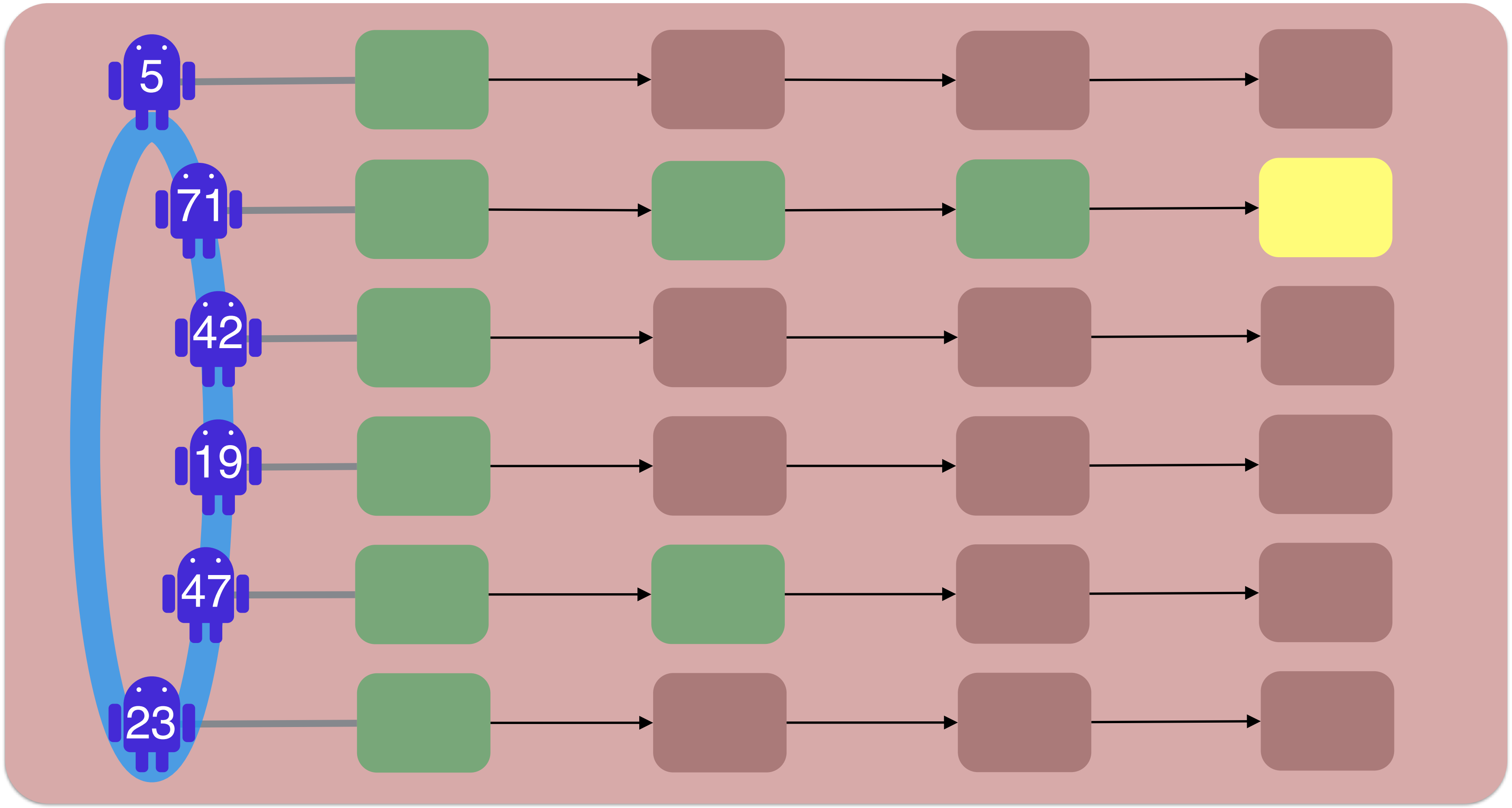
Behavior



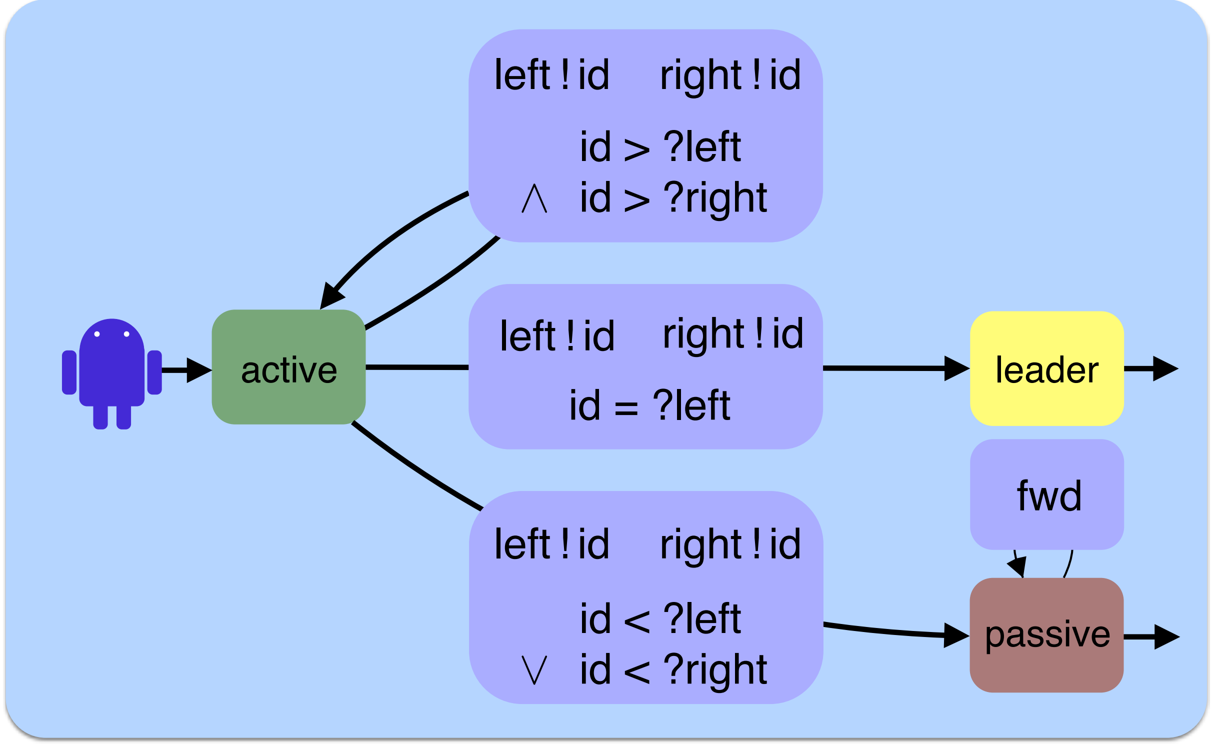
Distributed algorithm

Distributed algorithms

Leader election [Franklin '82]



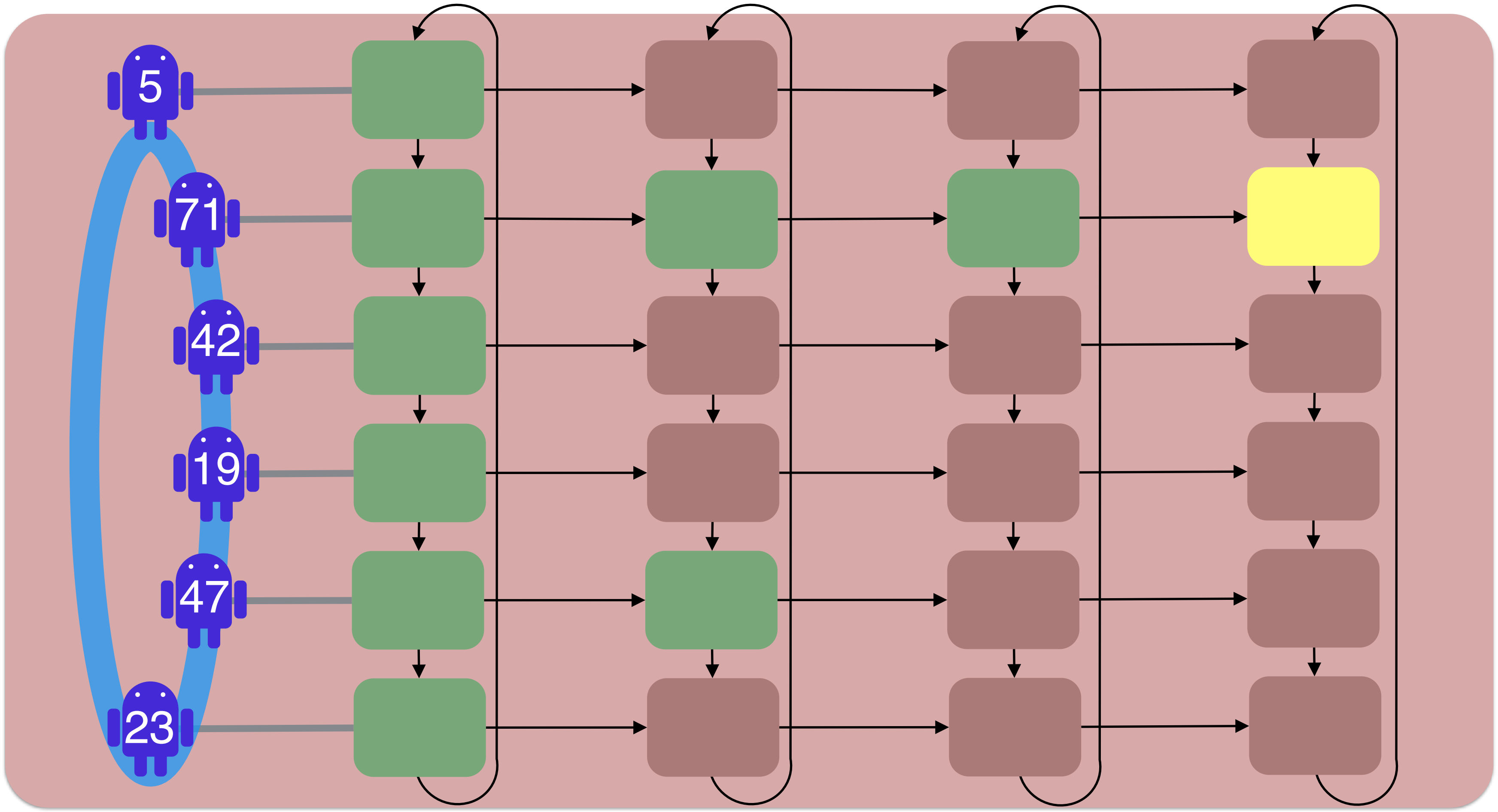
Behavior



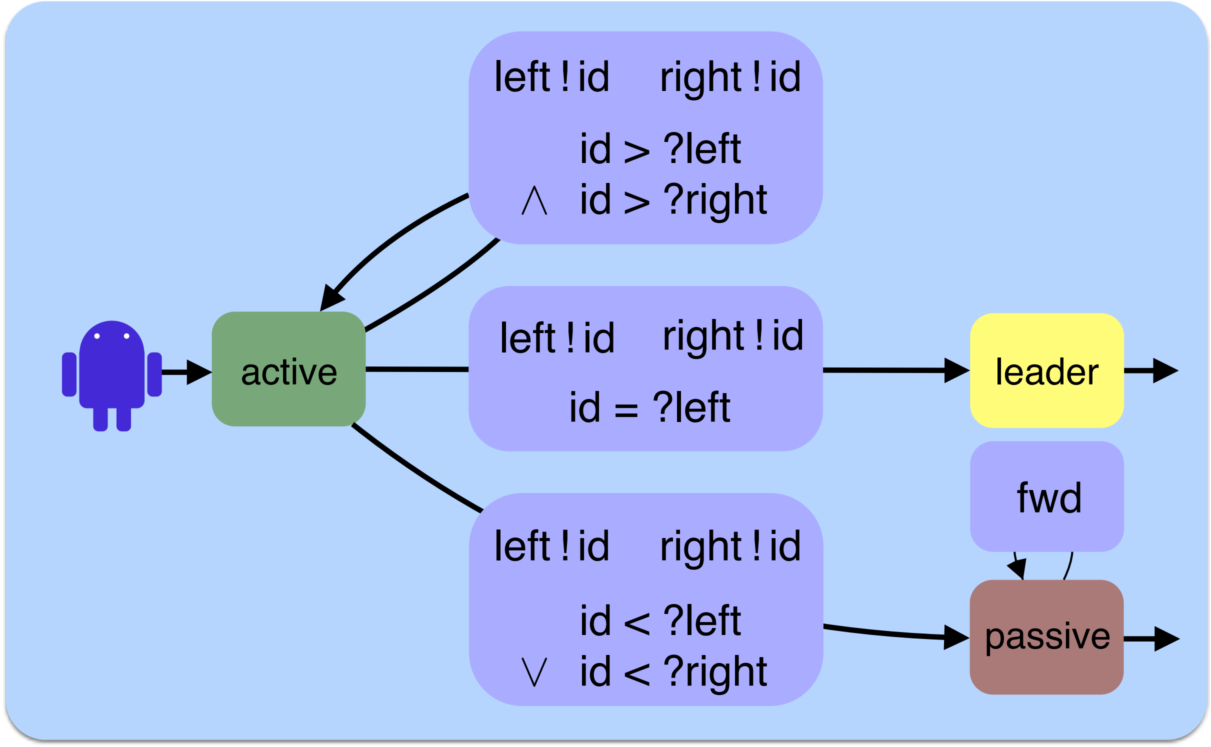
Distributed algorithm

Distributed algorithms

Leader election [Franklin ’82]

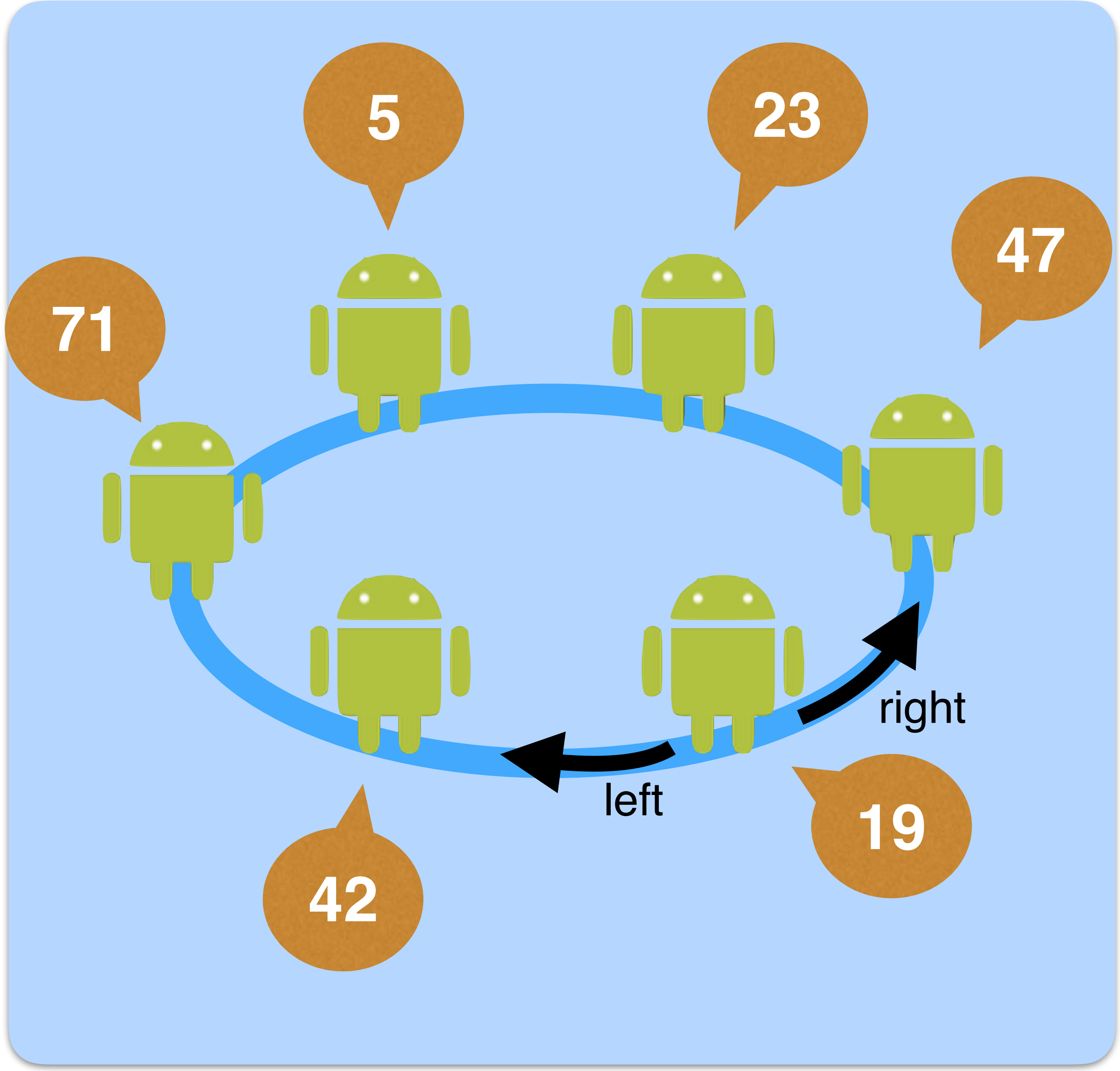


Behavior



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

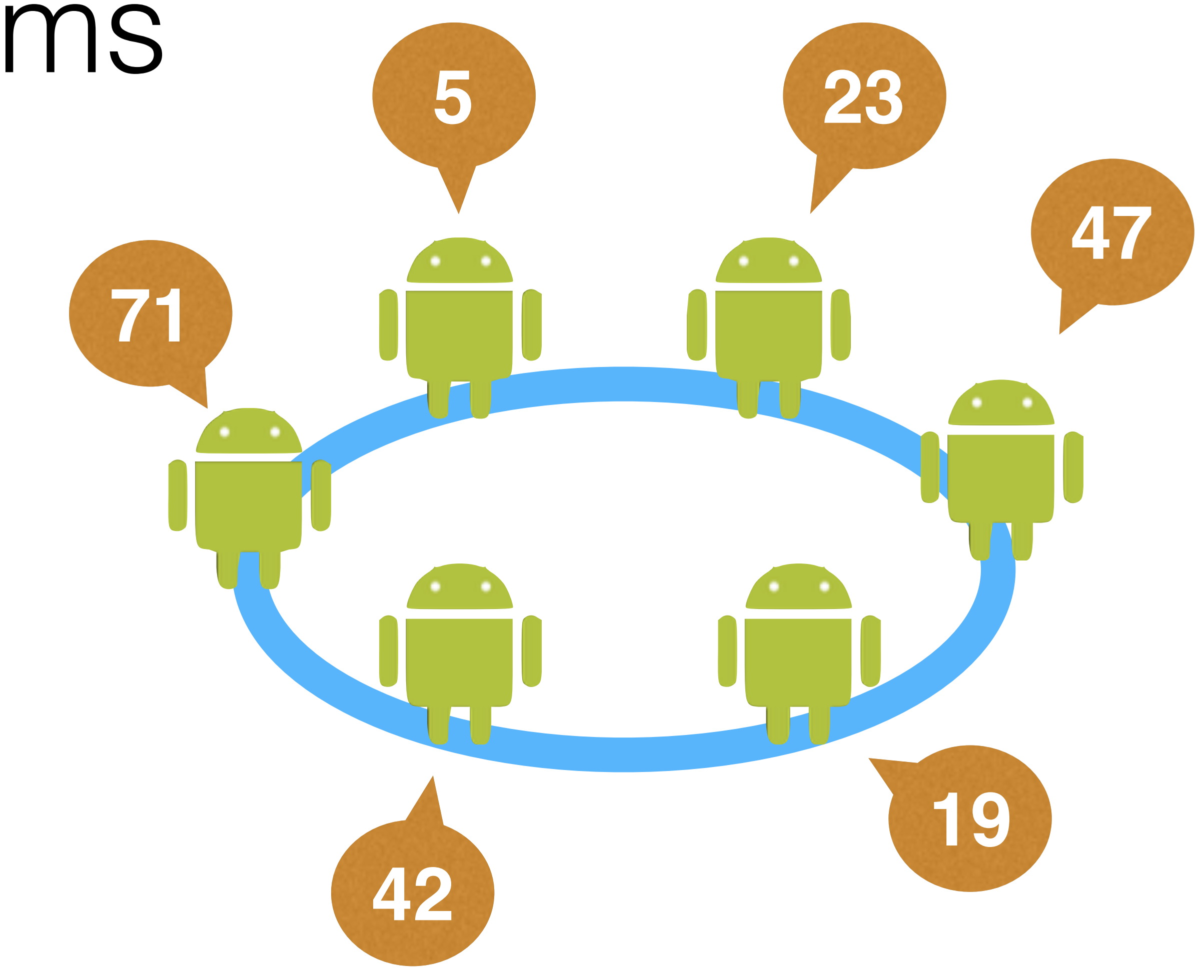
Every process 🤖 can be described by:

- 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

Franklin82



states: $\mathit{active}, \mathit{passive}$

found

initial state: active

registers: id, r, r_1, r_2

$t_1 = \langle \mathit{active}\colon \textbf{left}!id\,;\textbf{right}!id\,;\textbf{left}?r_1\,;\textbf{right}?r_2\,;r_1 < id\,;r_2 < id\,;\textbf{goto}\ \mathit{active}\rangle$

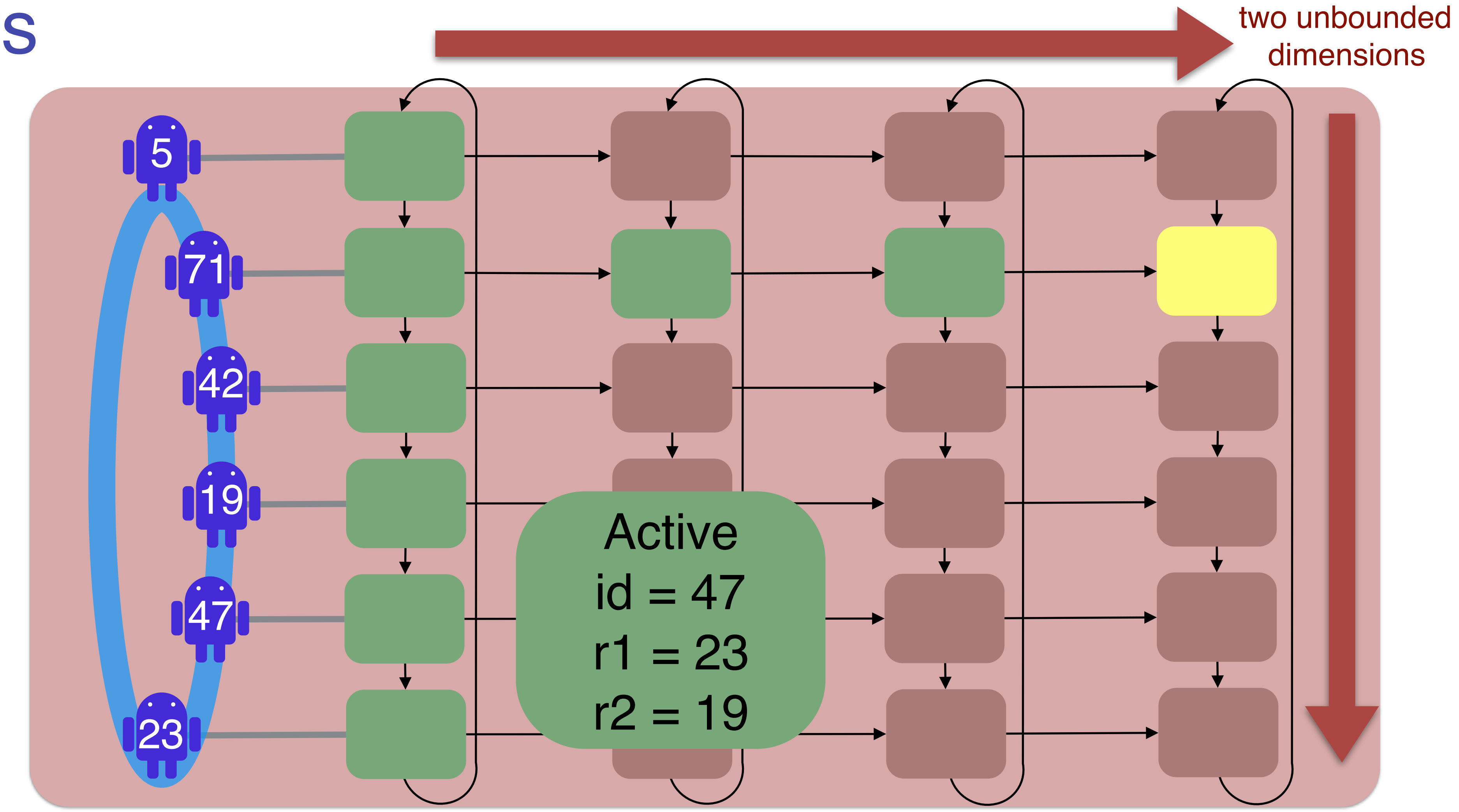
$t_2 = \langle \mathit{active}\colon \underline{\hspace{8cm}}\,;id < r_1\,;\textbf{goto}\ \mathit{passive}\rangle$

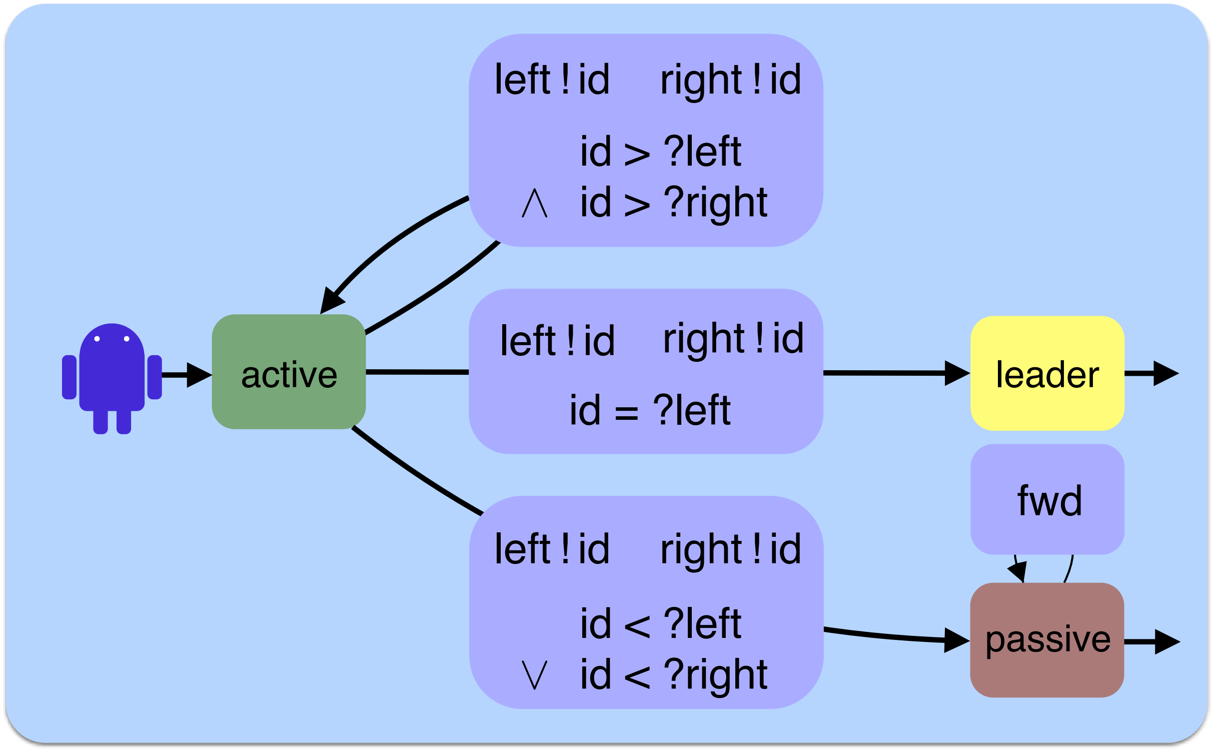
$t_3 = \langle \mathit{active}\colon \underline{\hspace{8cm}}\,;id < r_2\,;\textbf{goto}\ \mathit{passive}\rangle$

$t_4 = \langle \mathit{active}\colon \underline{\hspace{8cm}}\,;id = r_1\,;r := id\,;\textbf{goto}\ \mathit{found}\rangle$

$t_5 = \langle \mathit{passive}\colon \textbf{fwd}\,;\textbf{left}?r\,;\textbf{goto}\ \mathit{passive}\rangle$

Behaviors

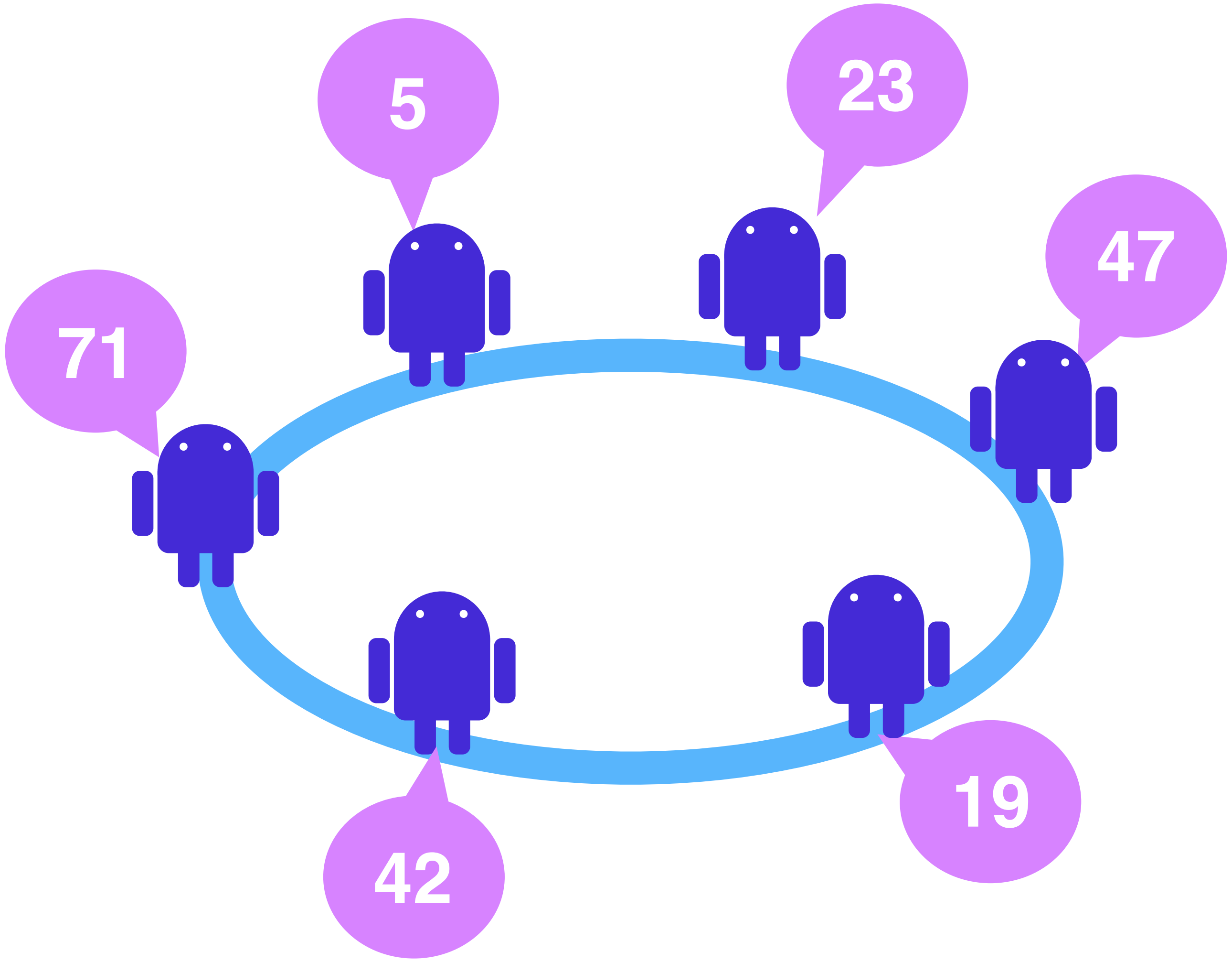




Distributed algorithm

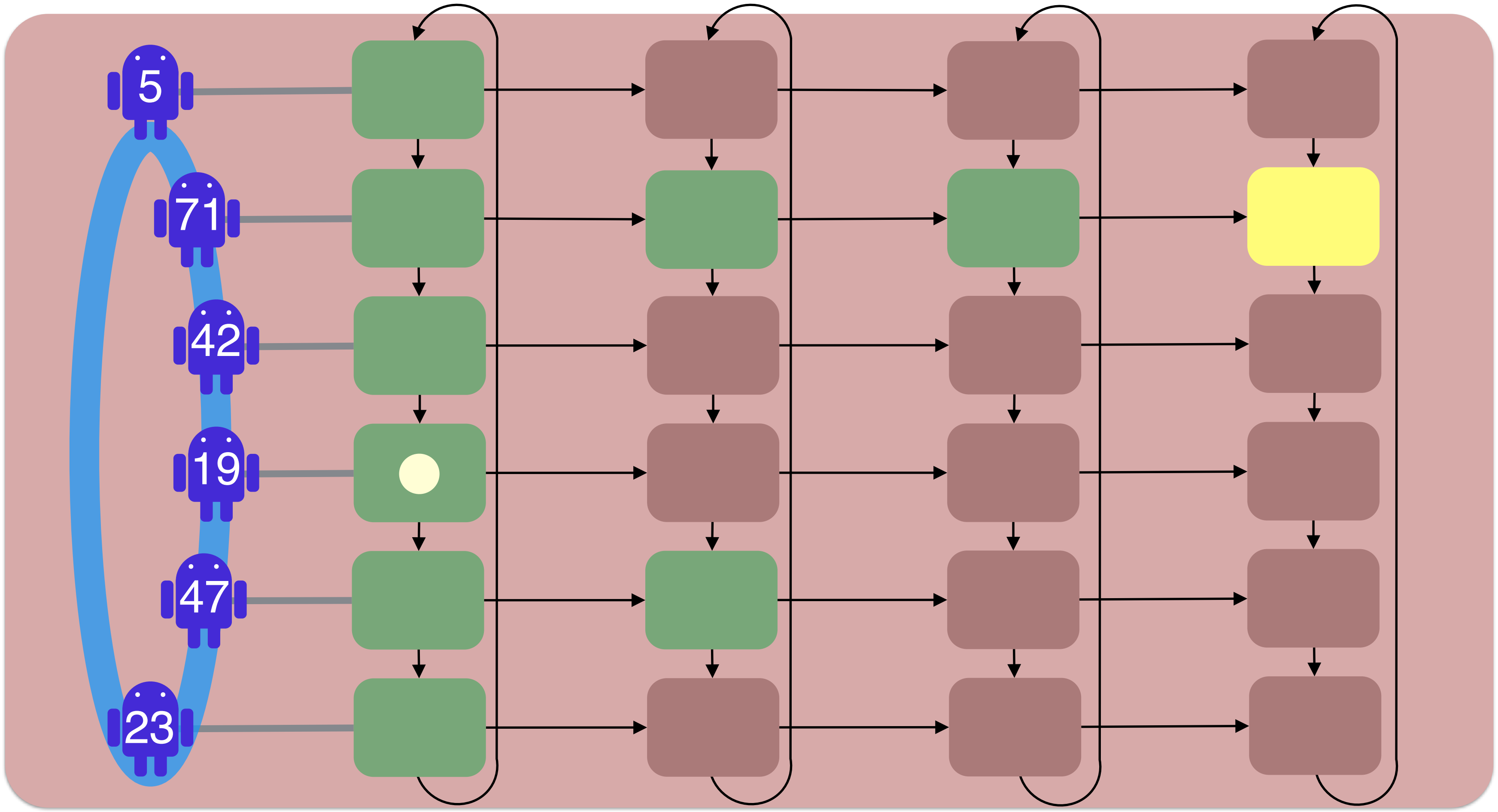
Cylinders
Arbitrary length and width
Labelled with data
from an infinite domain

Specification language

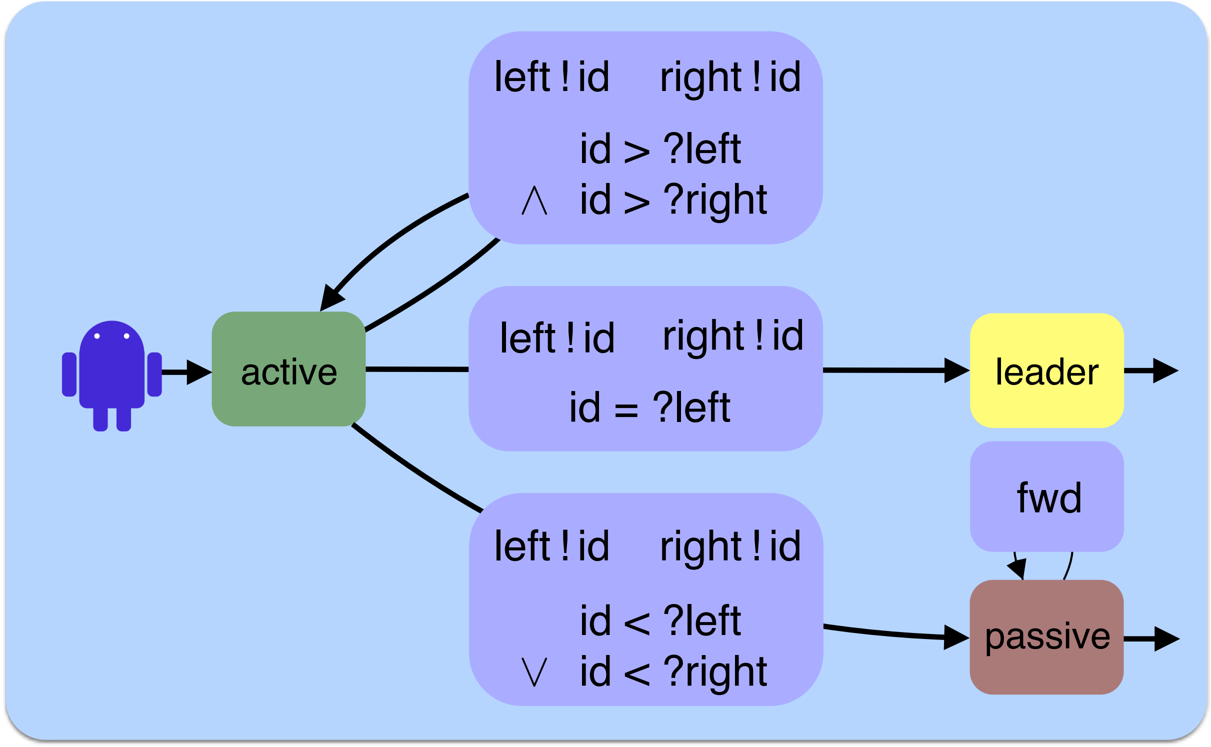


Distributed algorithms

Leader election [Franklin '82]



Behavior



left ! id right ! id

id > ?left

∧ id > ?right

active

left ! id right ! id

id = ?left

leader

fwd

left ! id right ! id

id < ?left

∨ id < ?right

passive

Distributed algorithm

⊨

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

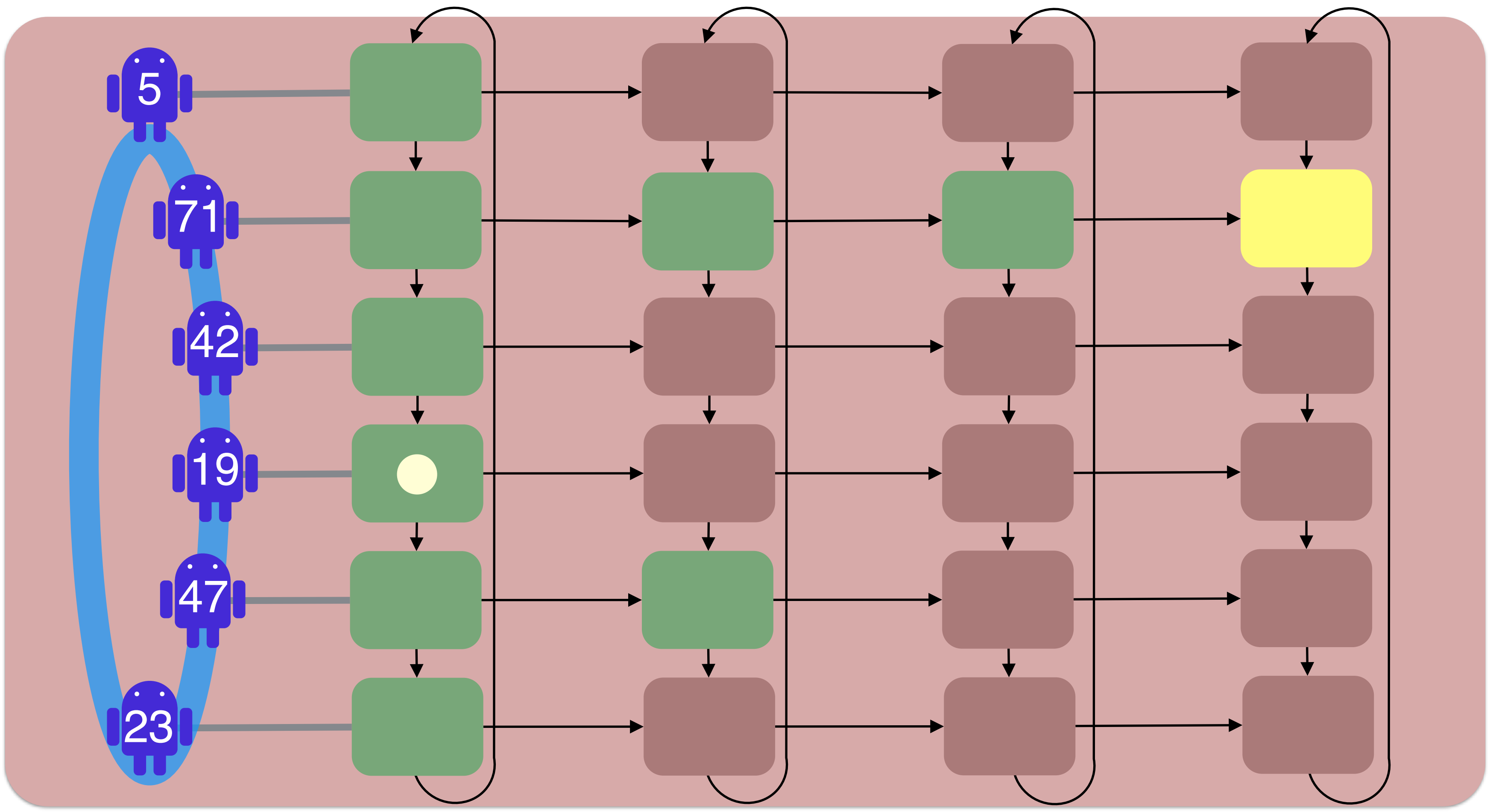
For all *n*, pid distributions, **accepting runs**, and processes:

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

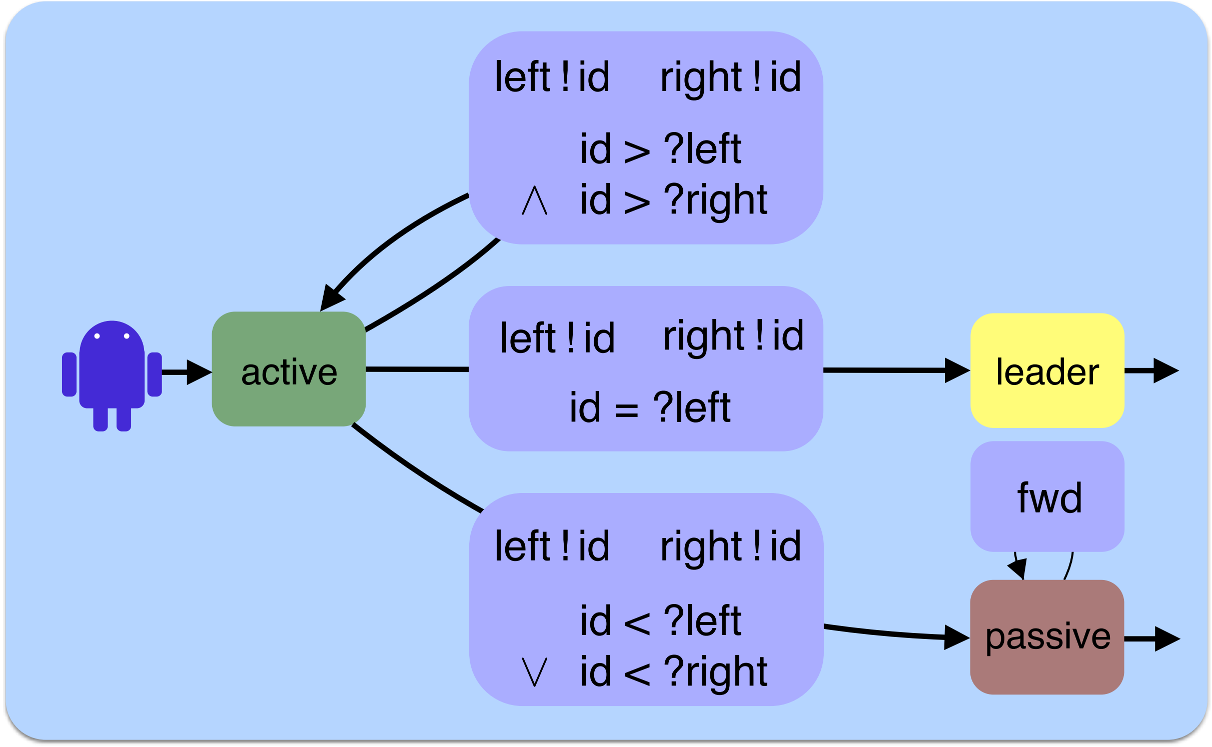
Data Propositional Dynamic Logic
[Bojanczyk et al. '09; Figueira-Segoufin '11]

Distributed algorithms

Leader election [Franklin '82]



Behavior



active

left ! id right ! id

id > ?left

∧ id > ?right

left ! id right ! id

id = ?left

left ! id right ! id

id < ?left

∨ id < ?right

leader

fwd

passive

Distributed algorithm

⊨

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

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

$\langle \longrightarrow^* \rangle \; (\; \neg \langle \longrightarrow \rangle \; \wedge \; \langle$go-to-🟨$\rangle$

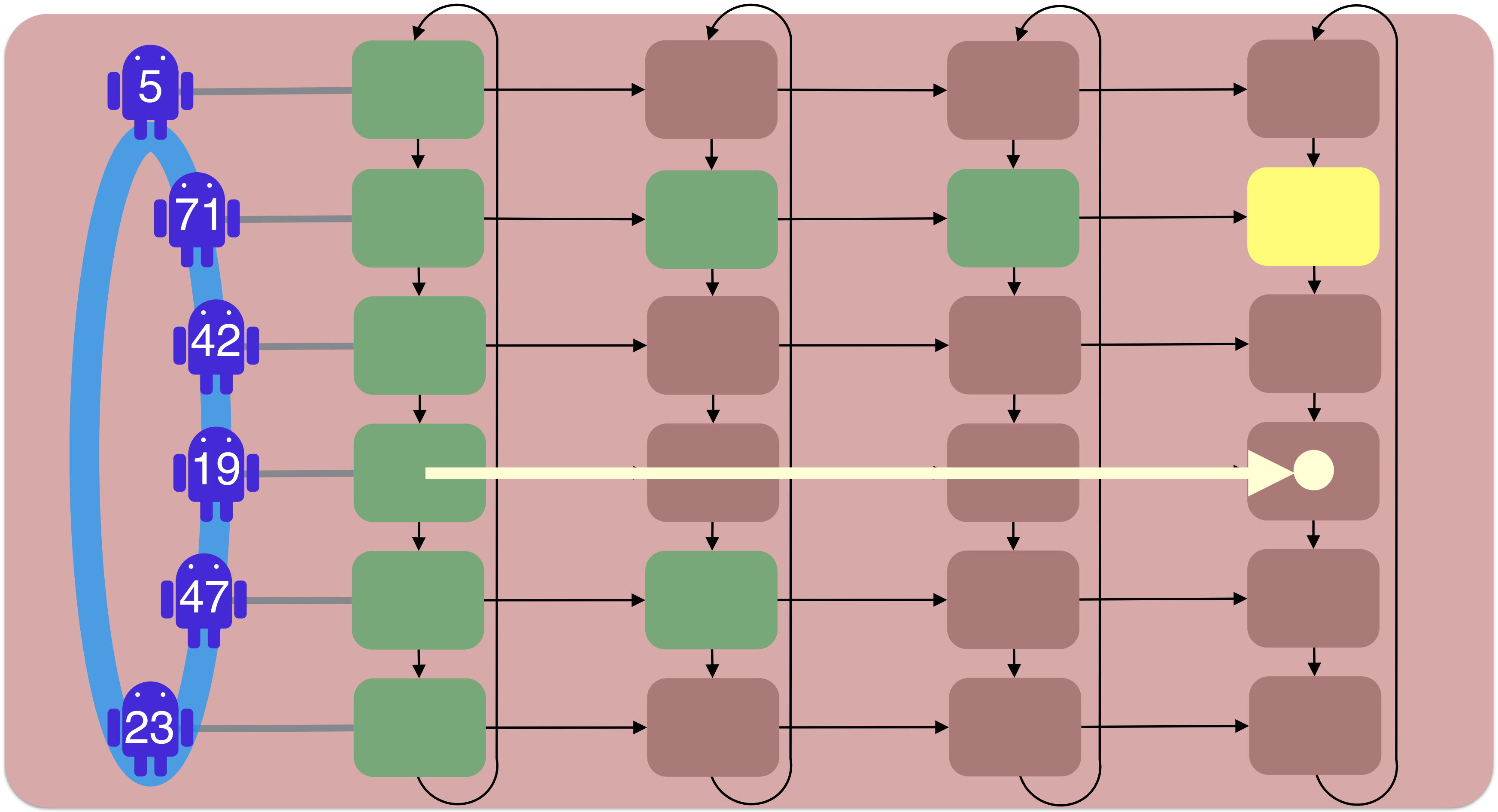
$\wedge \; [\downarrow^*] \; ($id $\leq \langle$go-to-🟨$\rangle$ id$))$

Data Propositional Dynamic Logic

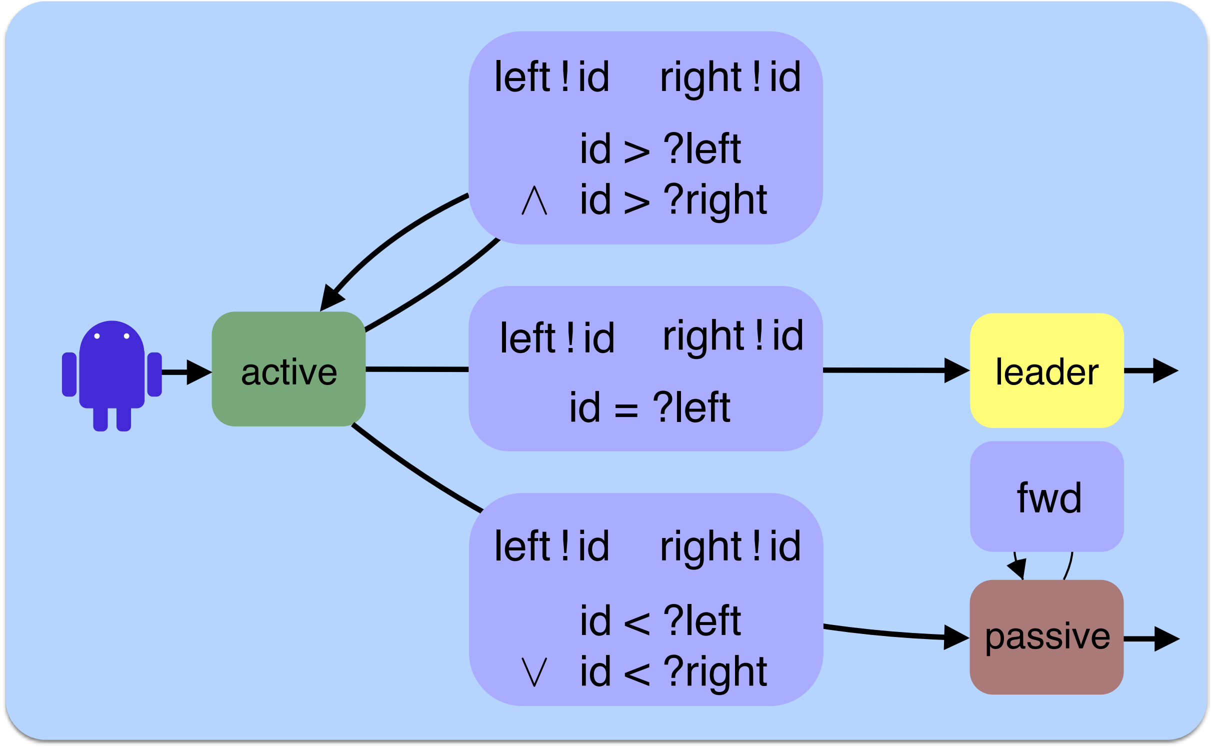
[Bojanczyk et al. '09; Figueira-Segoufin '11]

Distributed algorithms

Leader election [Franklin '82]



Behavior



Distributed algorithm

⊨

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

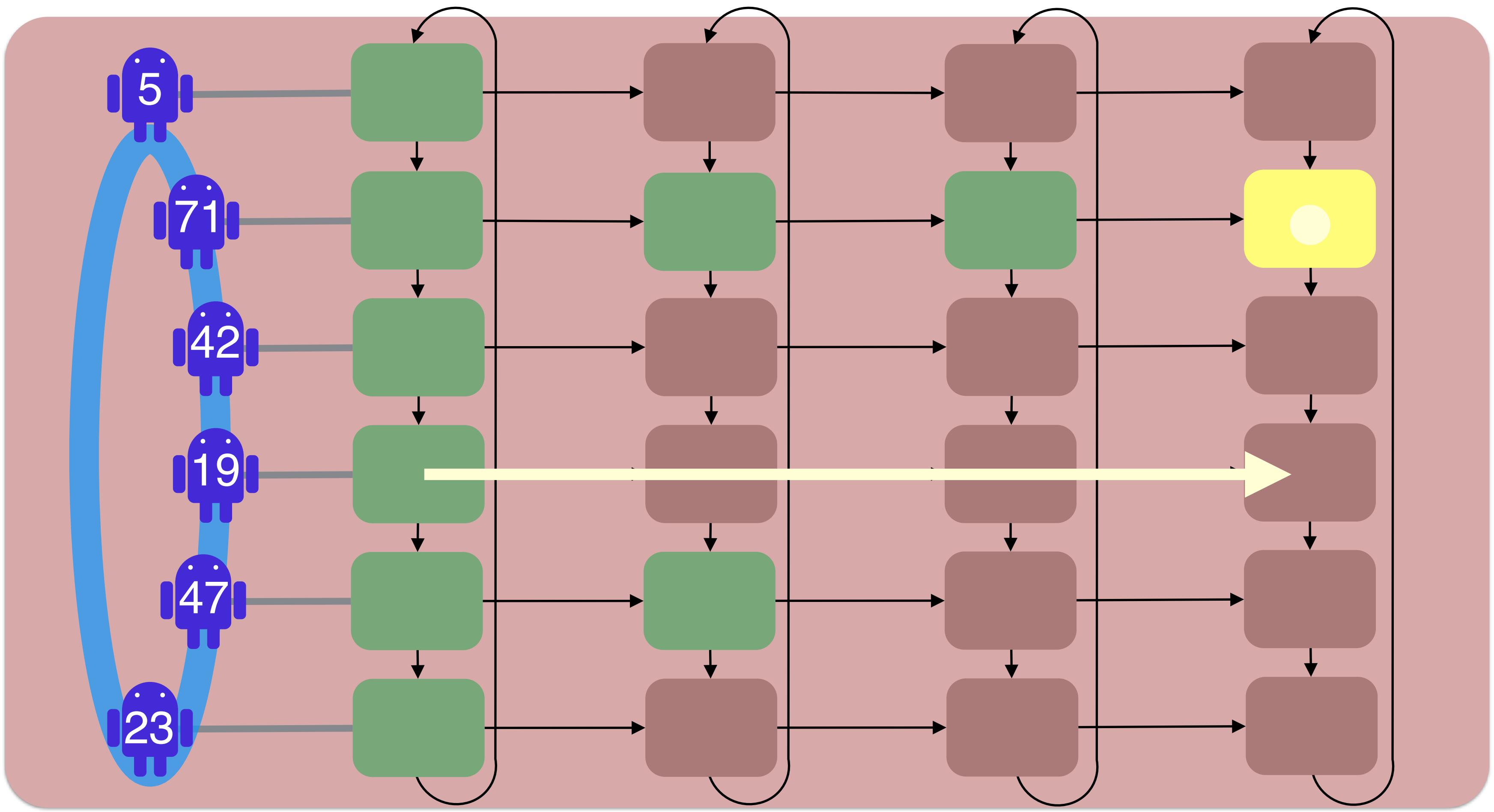
For all *n*, pid distributions, **accepting runs**, and processes:

$$\langle \longrightarrow^{*} \rangle \big(\neg\langle \longrightarrow \rangle \wedge \langle \text{go-to-}\text{🟨} \rangle \wedge [\downarrow^{*}] (\mathit{id} \leq \langle \text{go-to-}\text{🟨} \rangle\, \mathit{id}) \big)$$

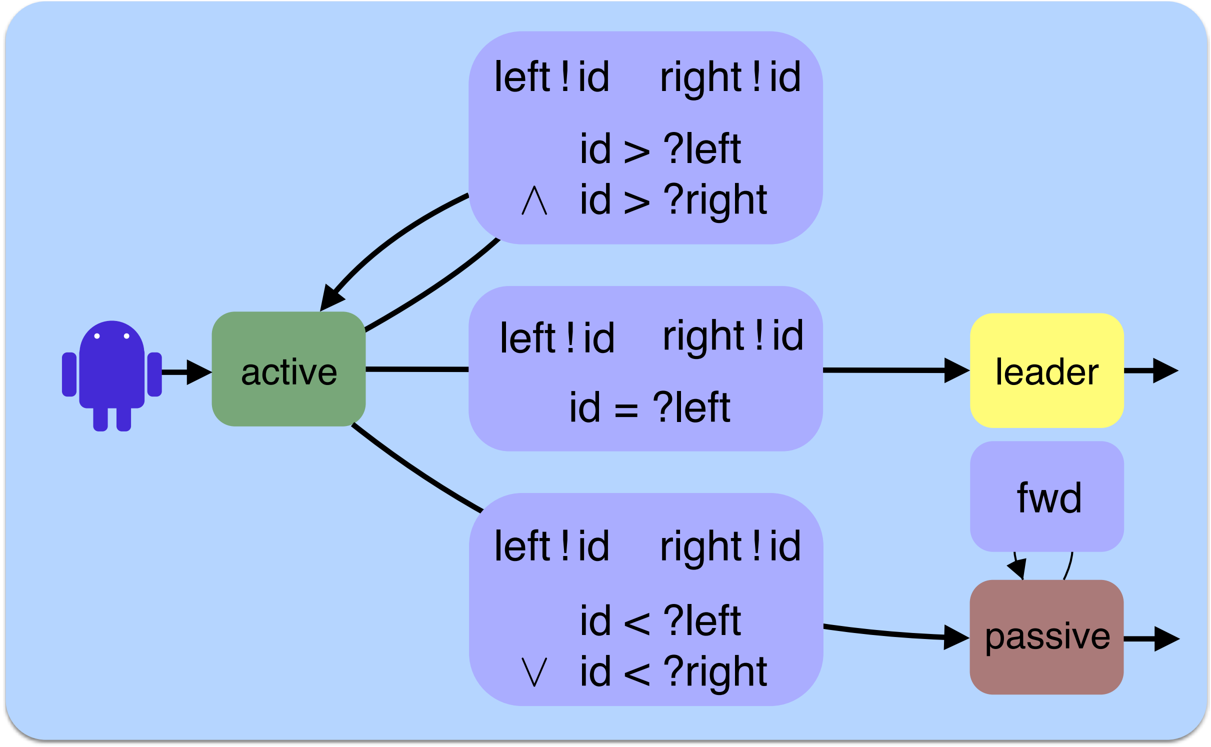
Data Propositional Dynamic Logic
[Bojanczyk et al. '09; Figueira-Segoufin '11]

Distributed algorithms

Leader election [Franklin ’82]



Behavior



Distributed algorithm

$\models$

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

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

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

$$\wedge\ [\downarrow^*]\ (\mathit{id} \leq \langle \text{go-to-}\square \rangle\, \mathit{id})\big)$$

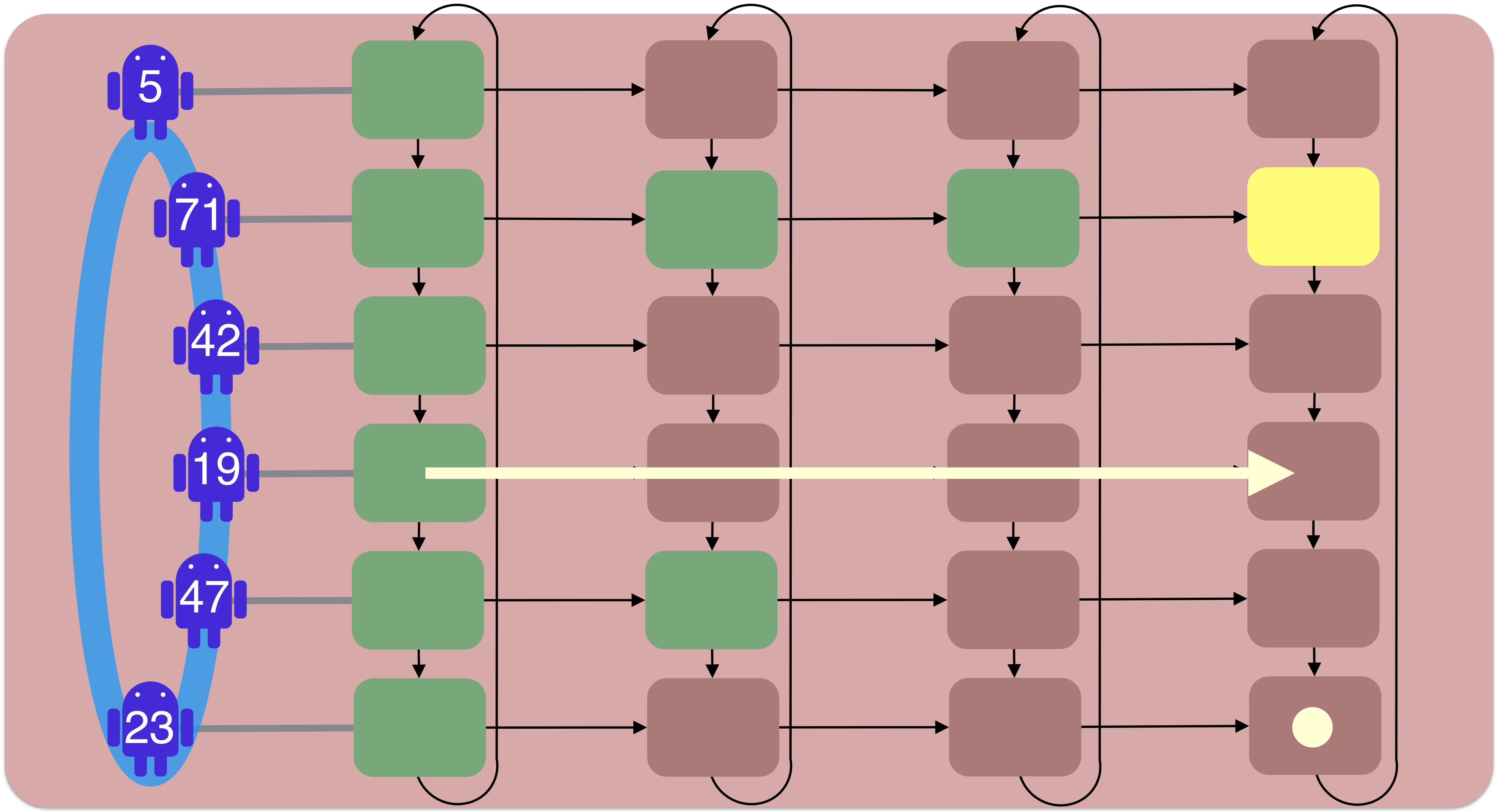
$$\text{go-to-}\square \ = \ (\neg \square \downarrow)^* \ \square$$

Data Propositional Dynamic Logic

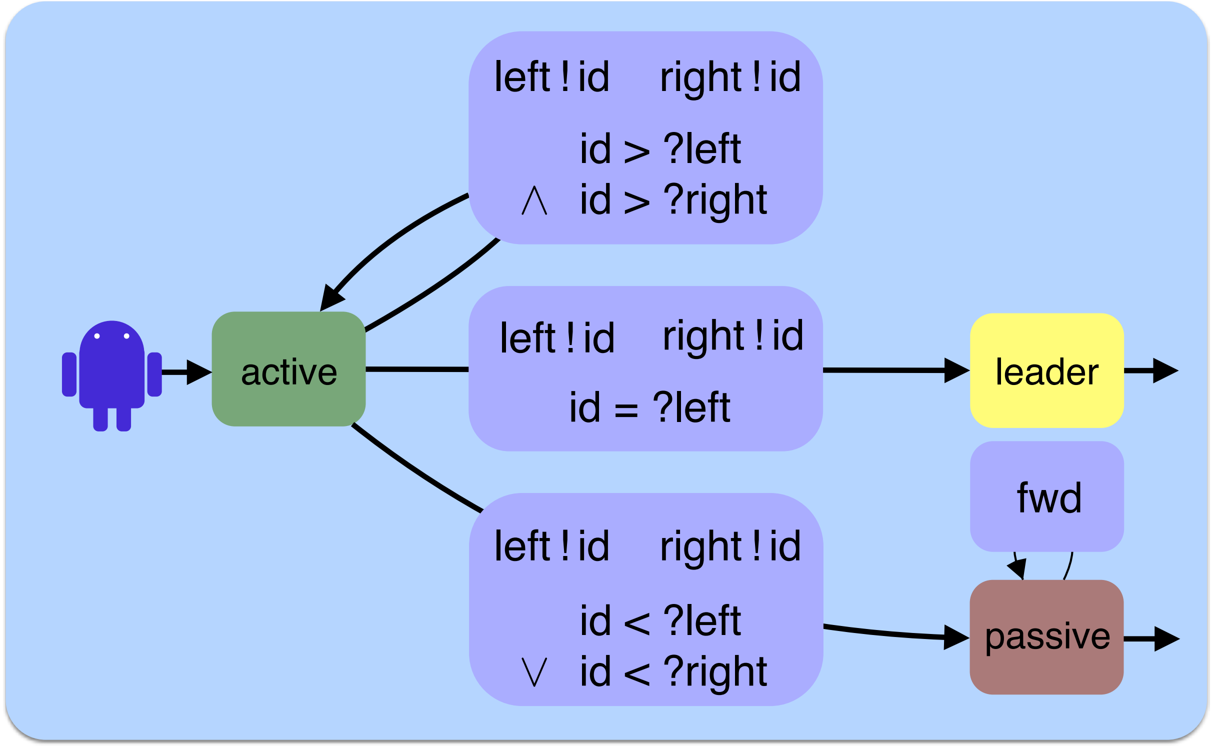
[Bojanczyk et al. ’09; Figueira-Segoufin ‘11]

Distributed algorithms

Leader election [Franklin '82]



Behavior



left ! id right ! id

id > ?left

∧ id > ?right

active

left ! id right ! id

id = ?left

leader

fwd

left ! id right ! id

id < ?left

∨ id < ?right

passive

Distributed algorithm

$\models$

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

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

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

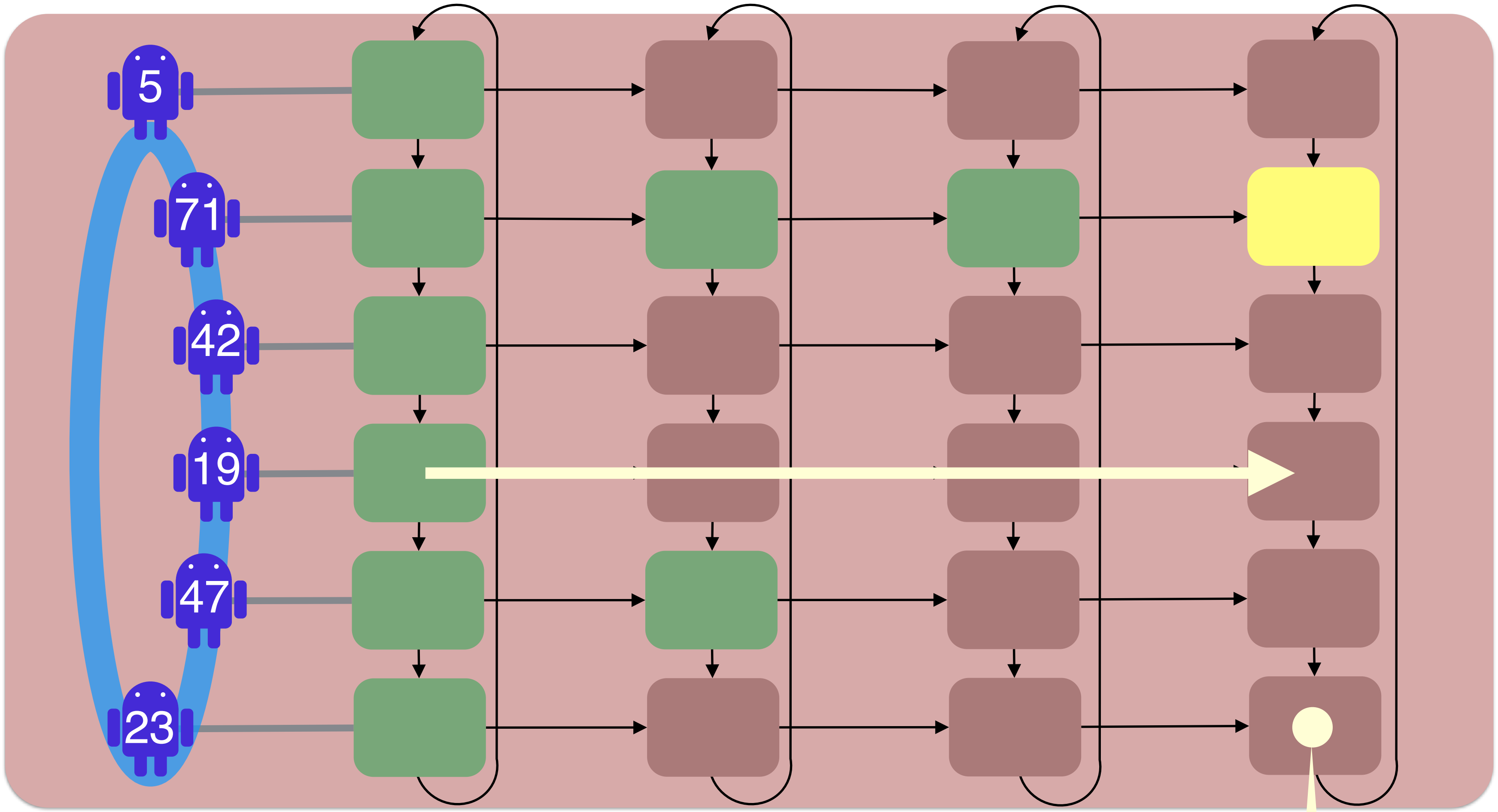
$$\wedge \; [\downarrow^*] \; (id \leq \langle \text{go-to-🟨} \rangle \, id))$$

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

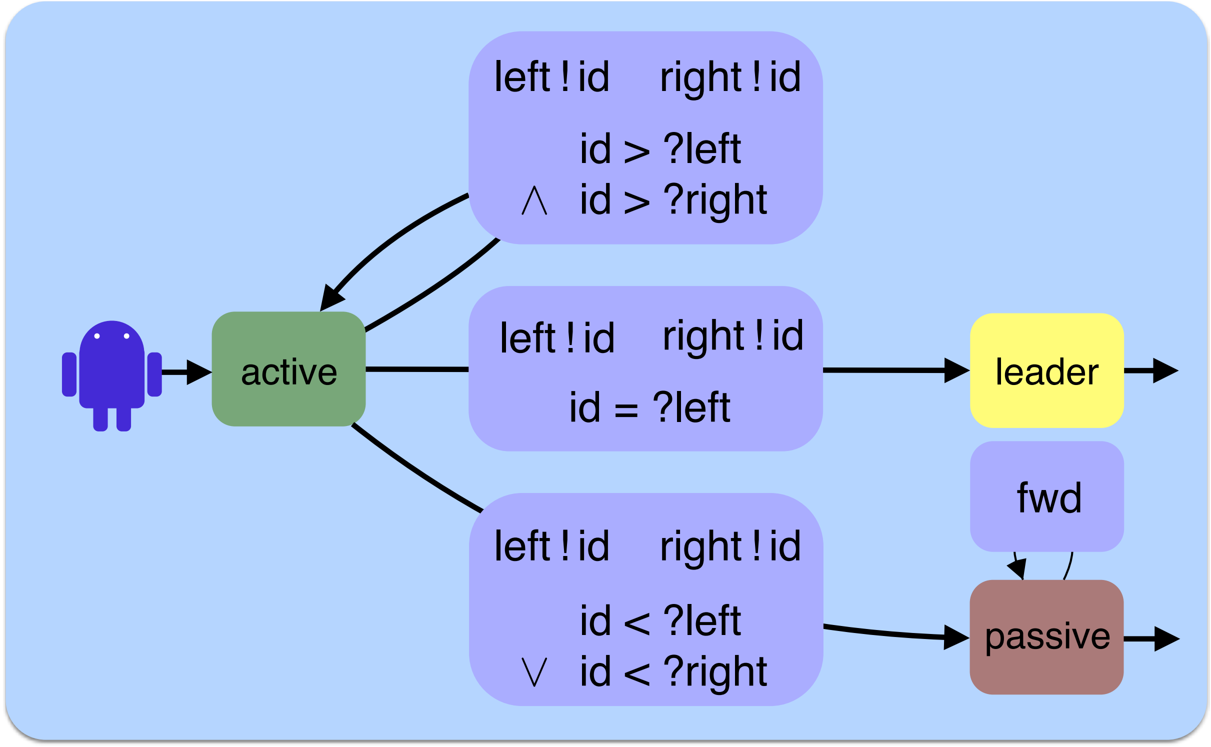
Data Propositional Dynamic Logic
[Bojanczyk et al. '09; Figueira-Segoufin '11]

Distributed algorithms

Leader election [Franklin '82]



Behavior



active

left ! id right ! id
id > ?left
∧ id > ?right

left ! id right ! id
id = ?left

left ! id right ! id
id < ?left
∨ id < ?right

leader

fwd

passive

Distributed algorithm

$\models$

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

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

$\langle \rightarrow^{*} \rangle \; (\quad \neg \langle \rightarrow \rangle \qquad \langle \text{go-to-}🟨 \rangle$

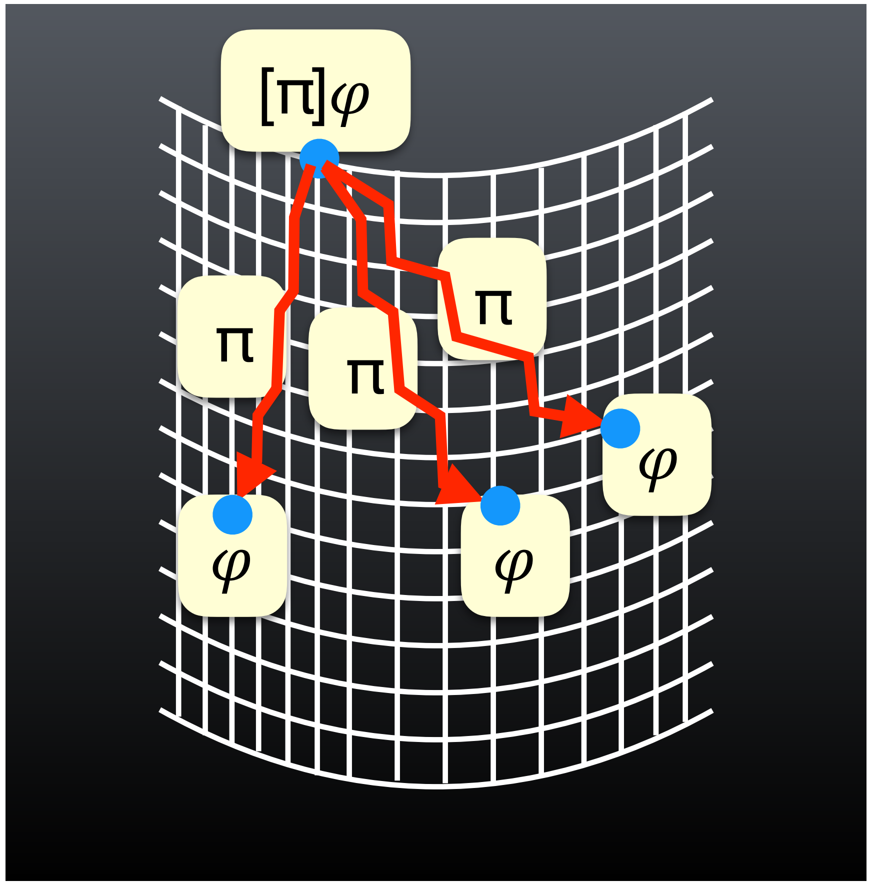
$\wedge \; [\downarrow^{*}] \; (\textbf{id} \leq \langle \text{go-to-}🟨 \rangle \, \textbf{id}))$

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

Data Propositional Dynamic Logic

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

Specifications
Data PDL



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

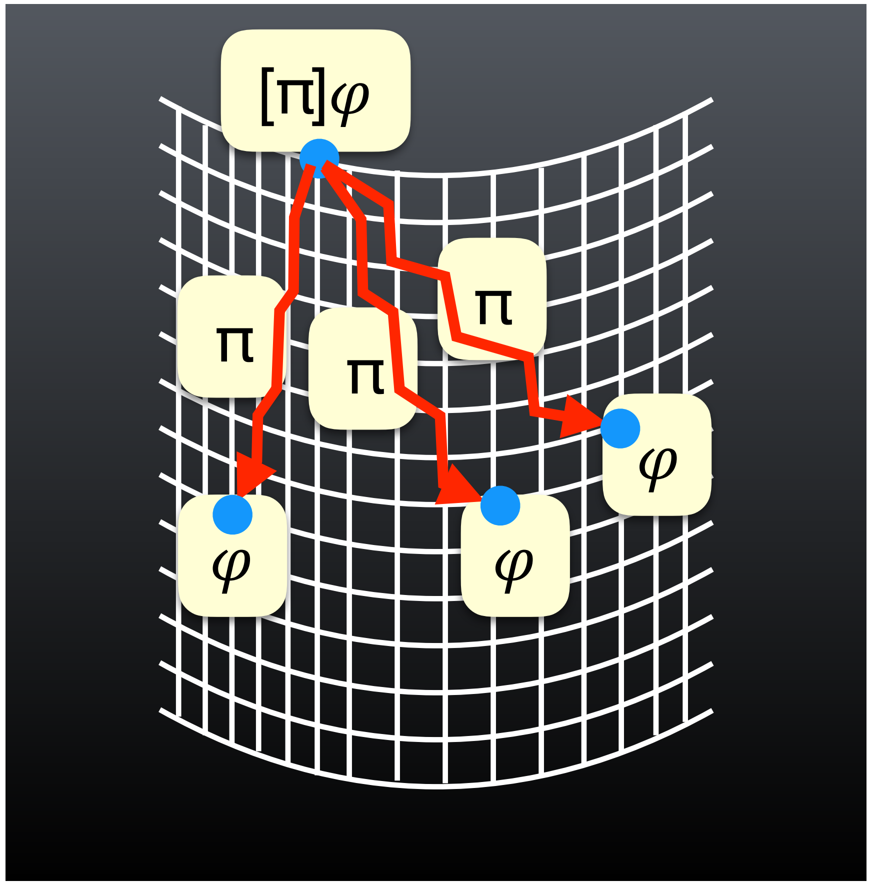
$$\pi, \pi' ::= \{\varphi\}? \mid d \mid \pi + \pi' \mid \pi \cdot \pi' \mid \pi^*$$

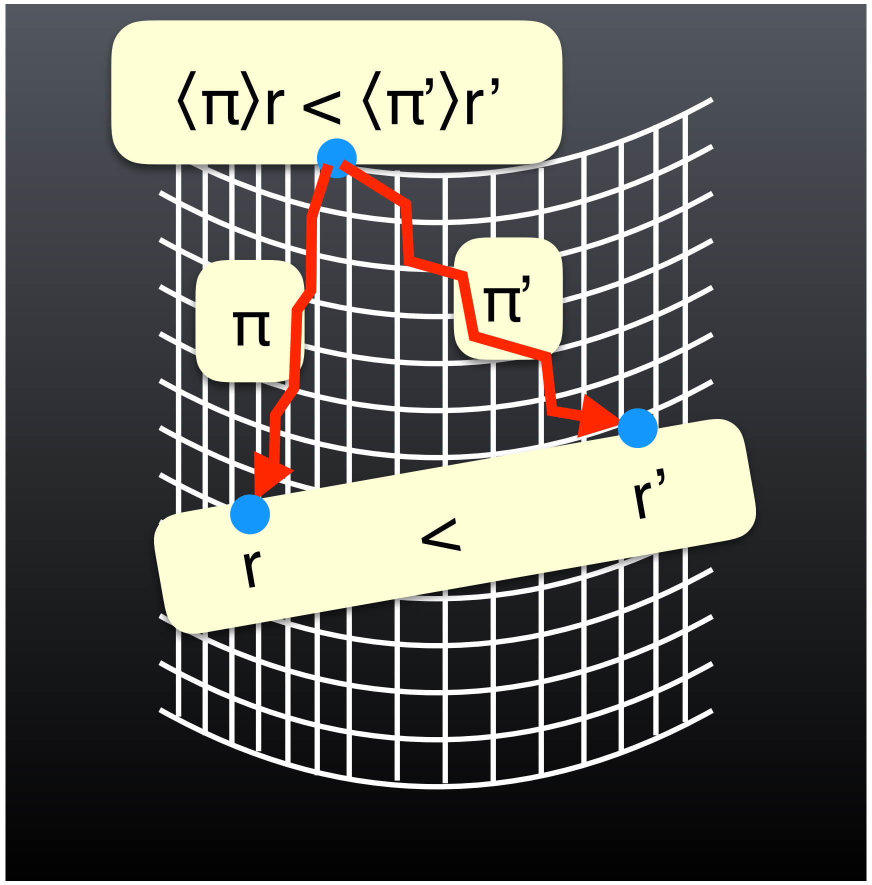
$$s \in S,\ r, r' \in \mathit{Reg},\ \bowtie\, \in \{=, \neq, <, \leq\}, \text{ and } d \in \{\epsilon, \leftarrow, \rightarrow, \uparrow, \downarrow\}.$$

M. Bojanczyk, A. Muscholl, T. Schwentick, and L. Segoufin. Two-variable logic on data trees and XML reasoning. *J. ACM*, 56(3), 2009.

D. Figueira and L. Segoufin. Bottom-up automata on data trees and vertical XPath. In STACS'11, volume 9 of LIPIcs, pages 93–104, 2011.

Specifications Data PDL





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

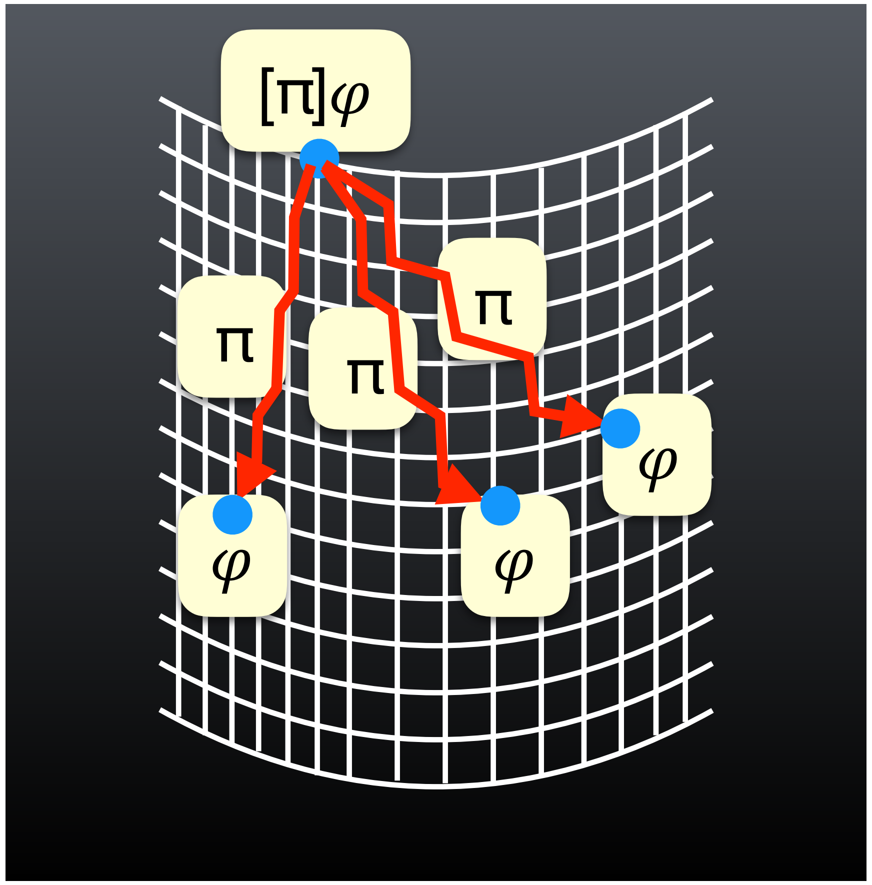
$$\pi, \pi' ::= \{\varphi\}? \mid d \mid \pi + \pi' \mid \pi \cdot \pi' \mid \pi^*$$

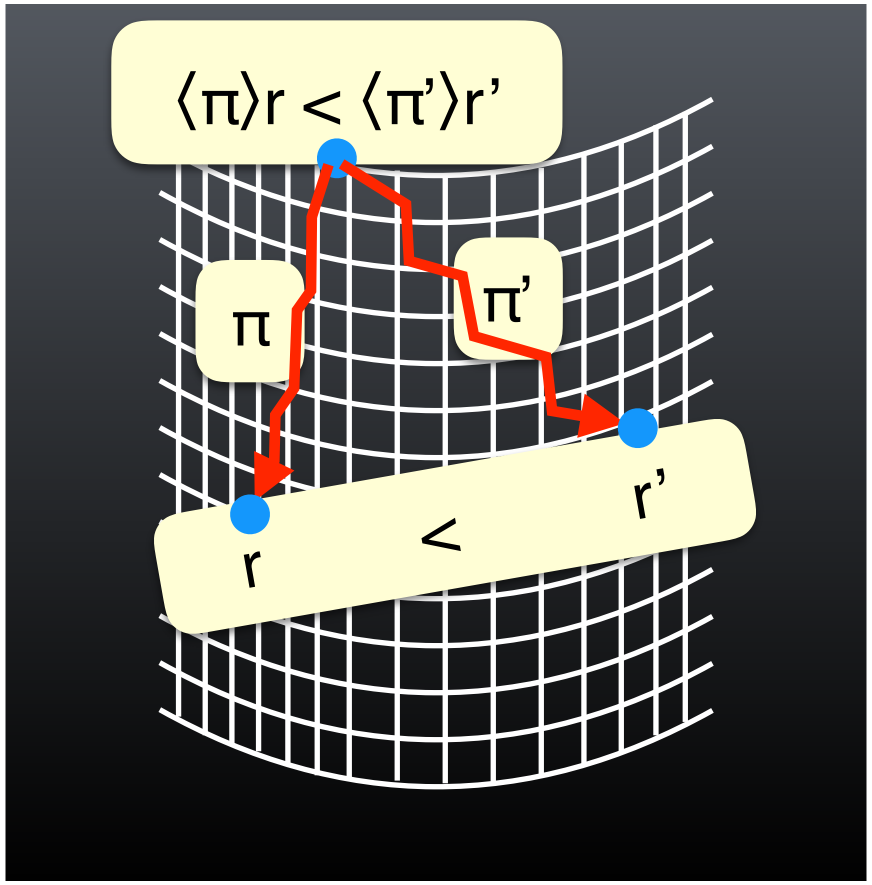
$$s \in S,\ r, r' \in Reg,\ \bowtie\ \in \{=, \neq, <, \leq\}, \text{ and } d \in \{\epsilon, \leftarrow, \rightarrow, \uparrow, \downarrow\}.$$

M. Bojanczyk, A. Muscholl, T. Schwentick, and L. Segoufin. Two-variable logic on data trees and XML reasoning. *J. ACM*, 56(3), 2009.

D. Figueira and L. Segoufin. Bottom-up automata on data trees and vertical XPath. In STACS'11, volume 9 of LIPIcs, pages 93–104, 2011.

Specifications Data PDL





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

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

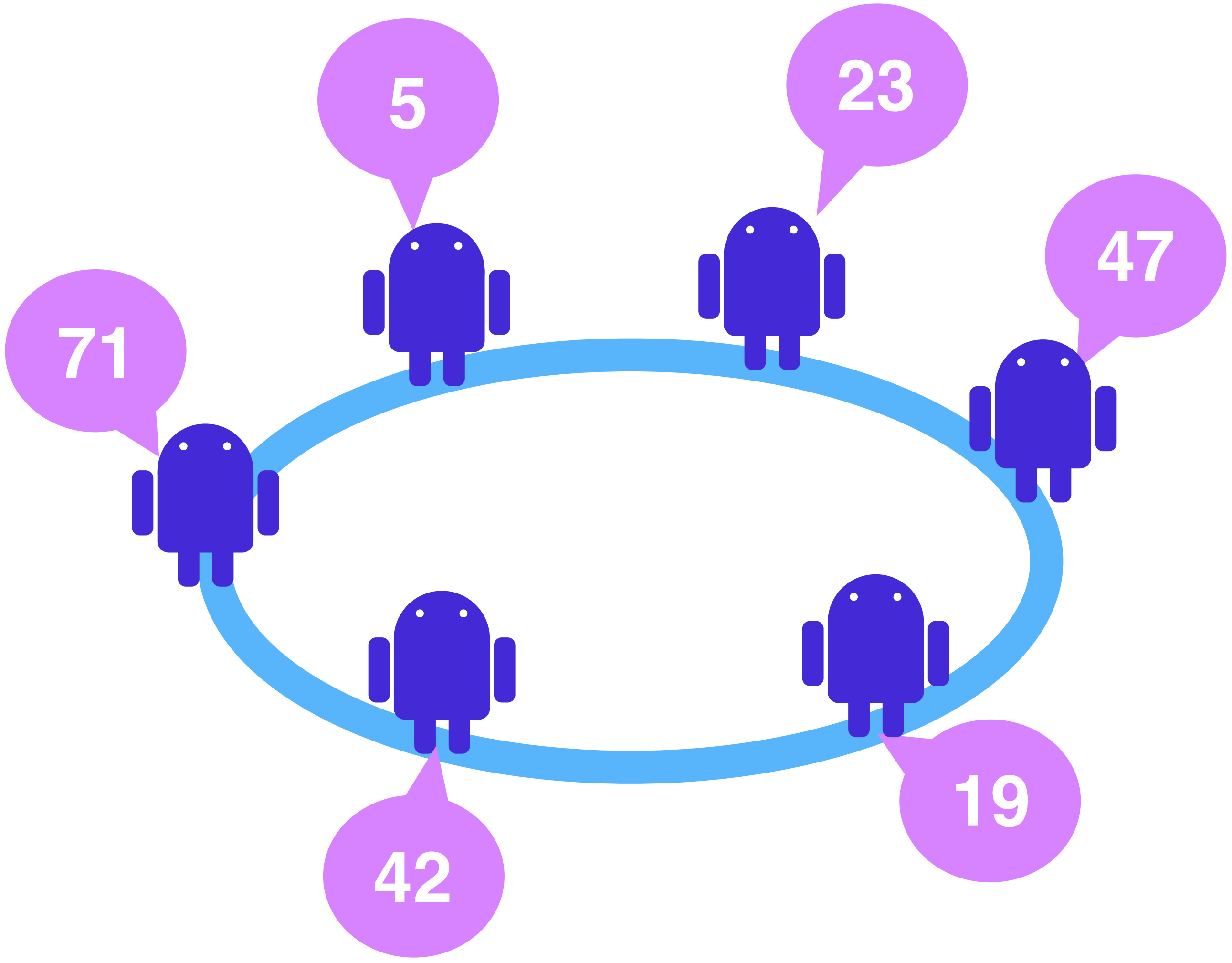
$$\pi, \pi' ::= \{\varphi\}? \mid d \mid \pi + \pi' \mid \pi \cdot \pi' \mid \pi^*$$

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

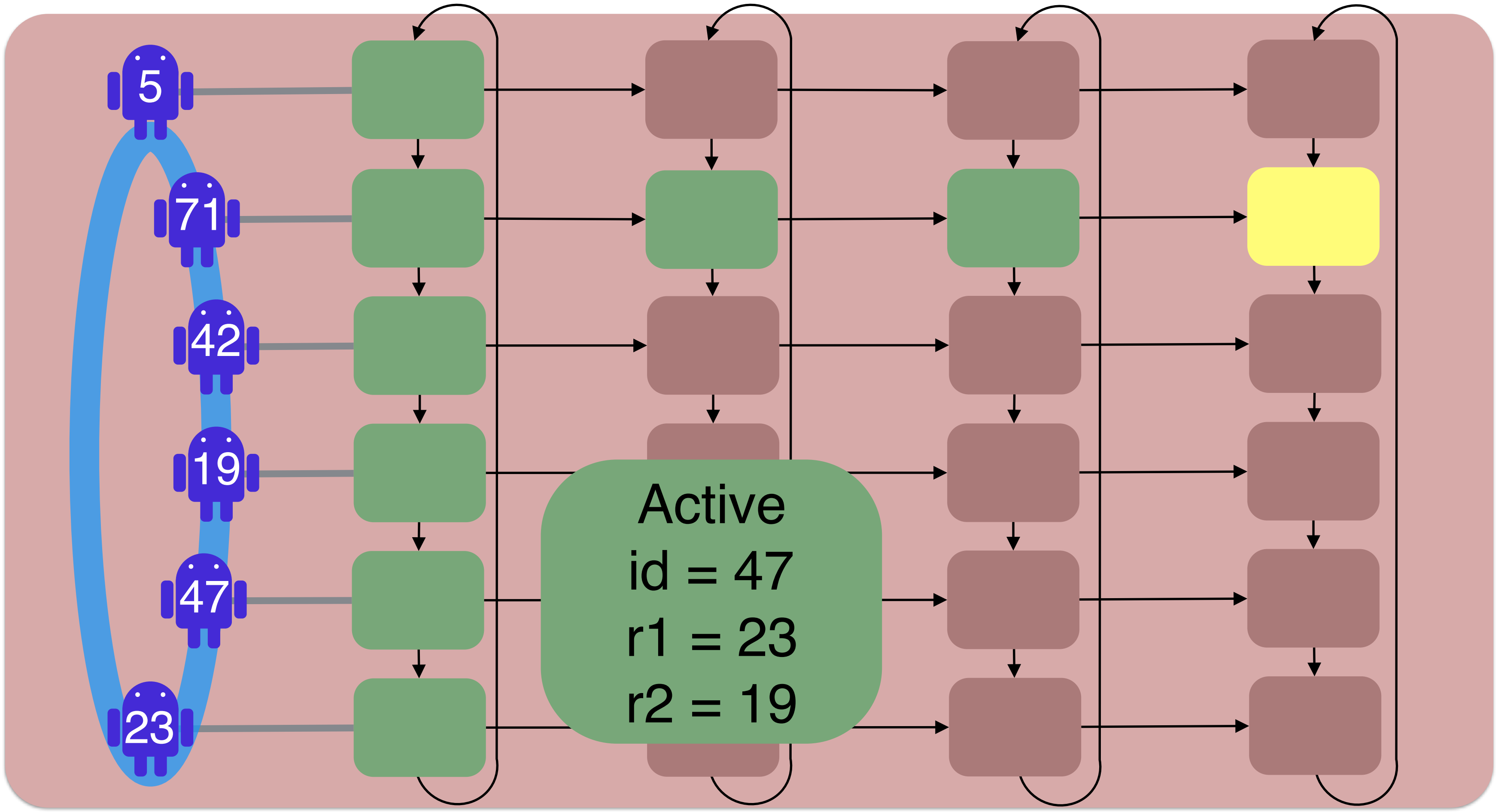
M. Bojanczyk, A. Muscholl, T. Schwentick, and L. Segoufin. Two-variable logic on data trees and XML reasoning. *J. ACM*, 56(3), 2009.

D. Figueira and L. Segoufin. Bottom-up automata on data trees and vertical XPath. In STACS'11, volume 9 of LIPIcs, pages 93–104, 2011.

Model Checking



Model Checking Distributed algorithms

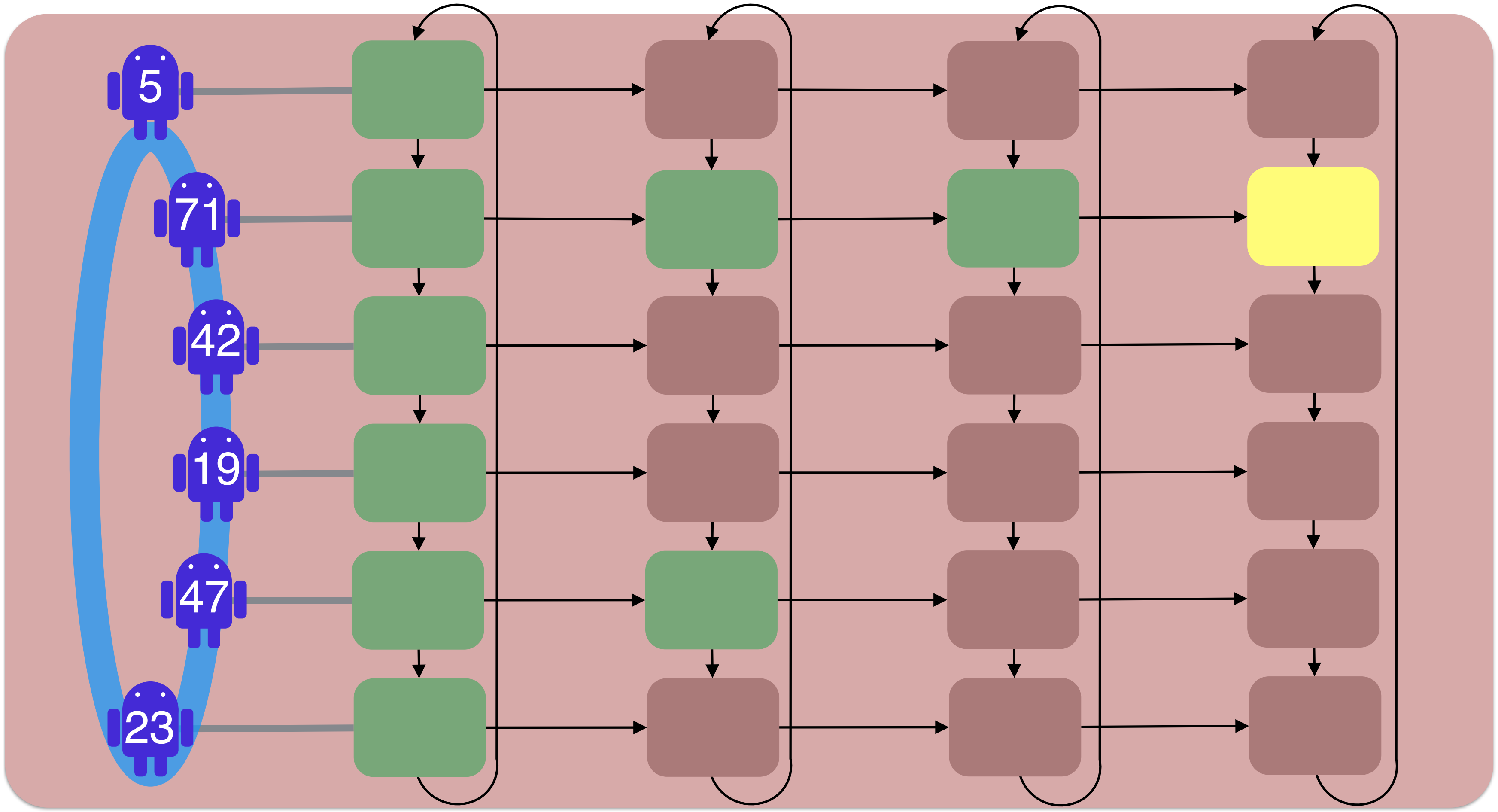


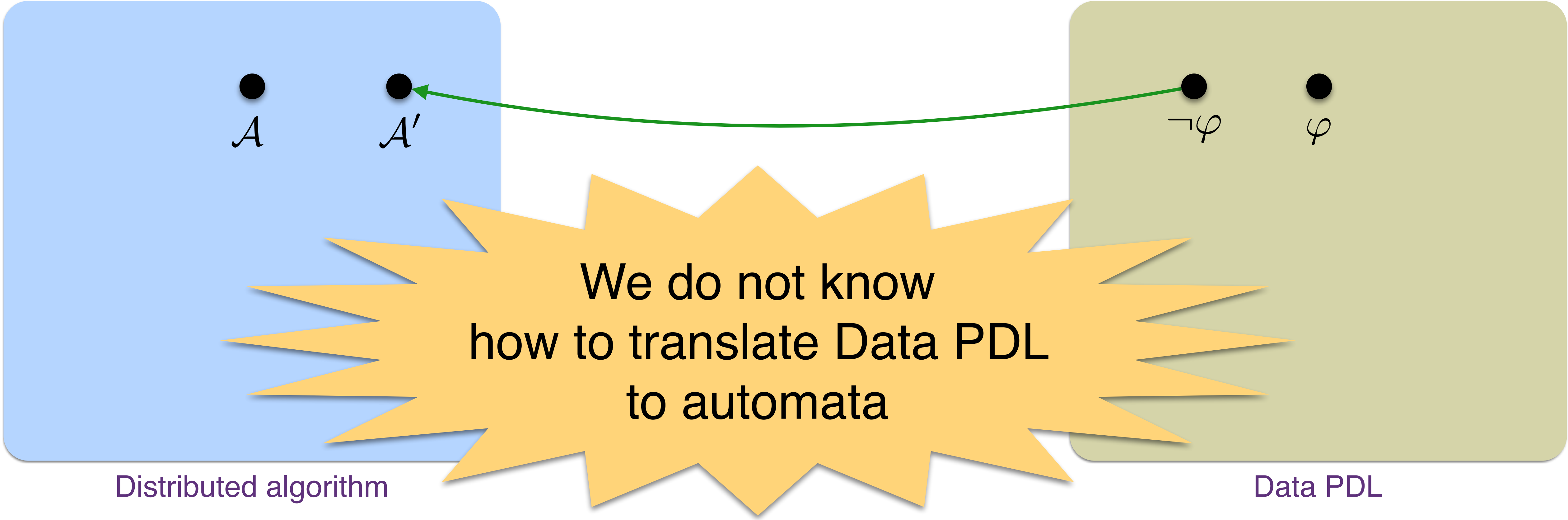
Active
id = 47
r1 = 23
r2 = 19

UNDECIDABLE

Cylinders of arbitrary width and length
Data from an infinite domain
Register automata with data comparisons
Data PDL with data comparisons

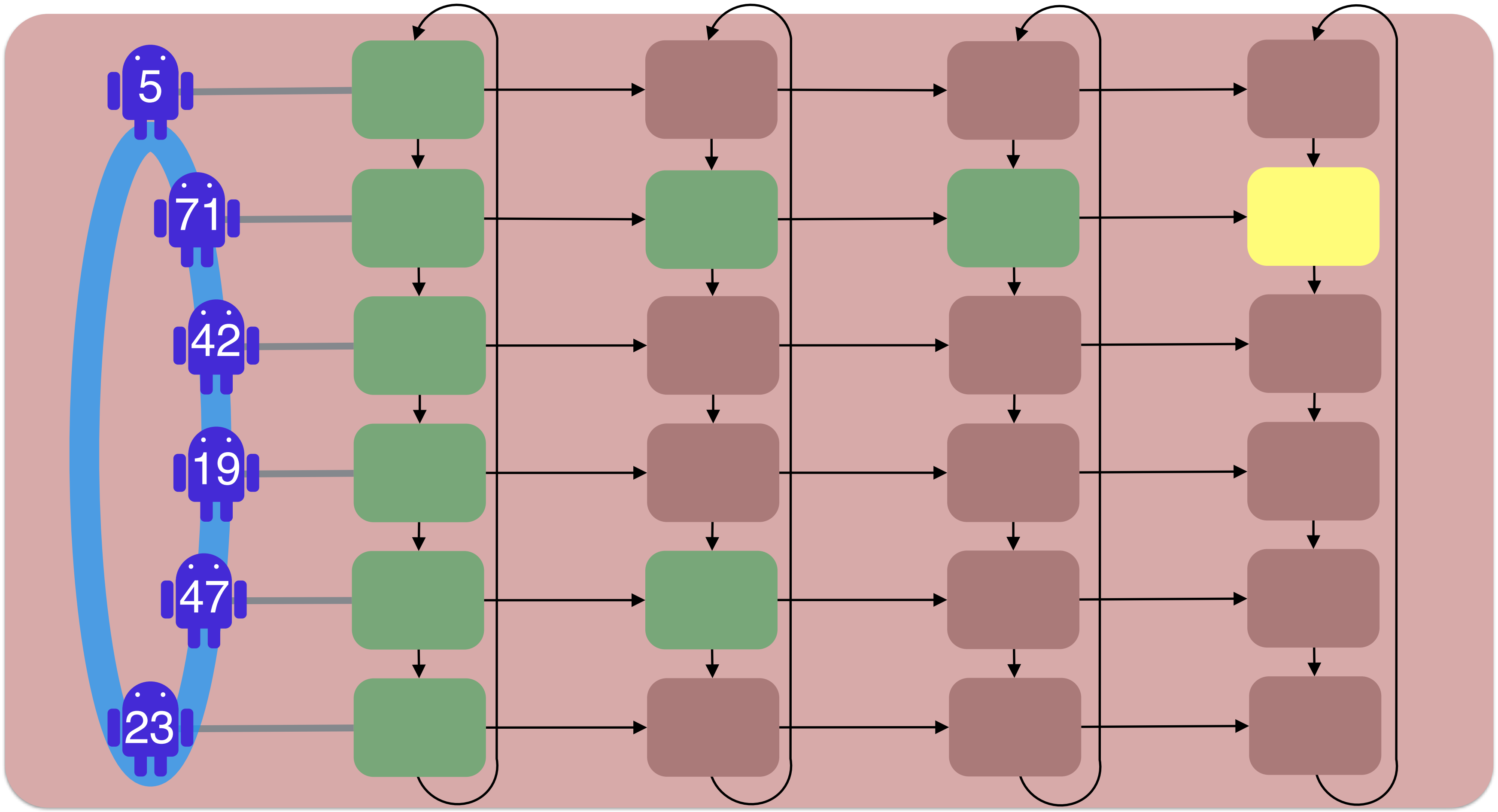
Reduction to automata?

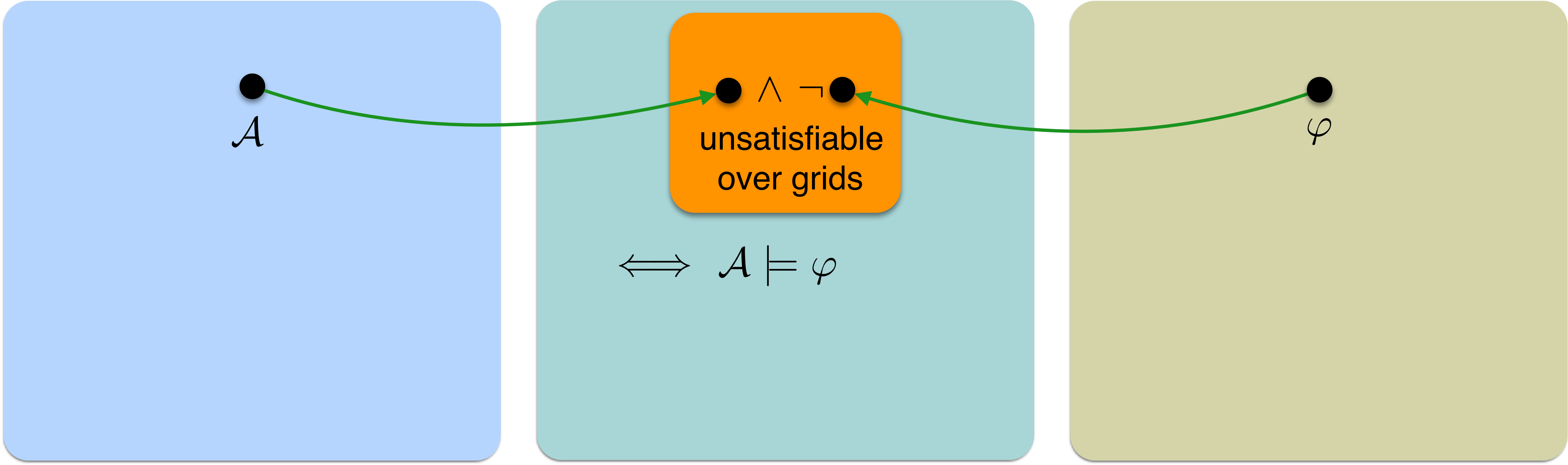




We do not know how to translate Data PDL to automata

Data abstraction





$\mathcal{A}$

● $\land$ $\neg$ ●

unsatisfiable over grids

$\iff \mathcal{A} \models \varphi$

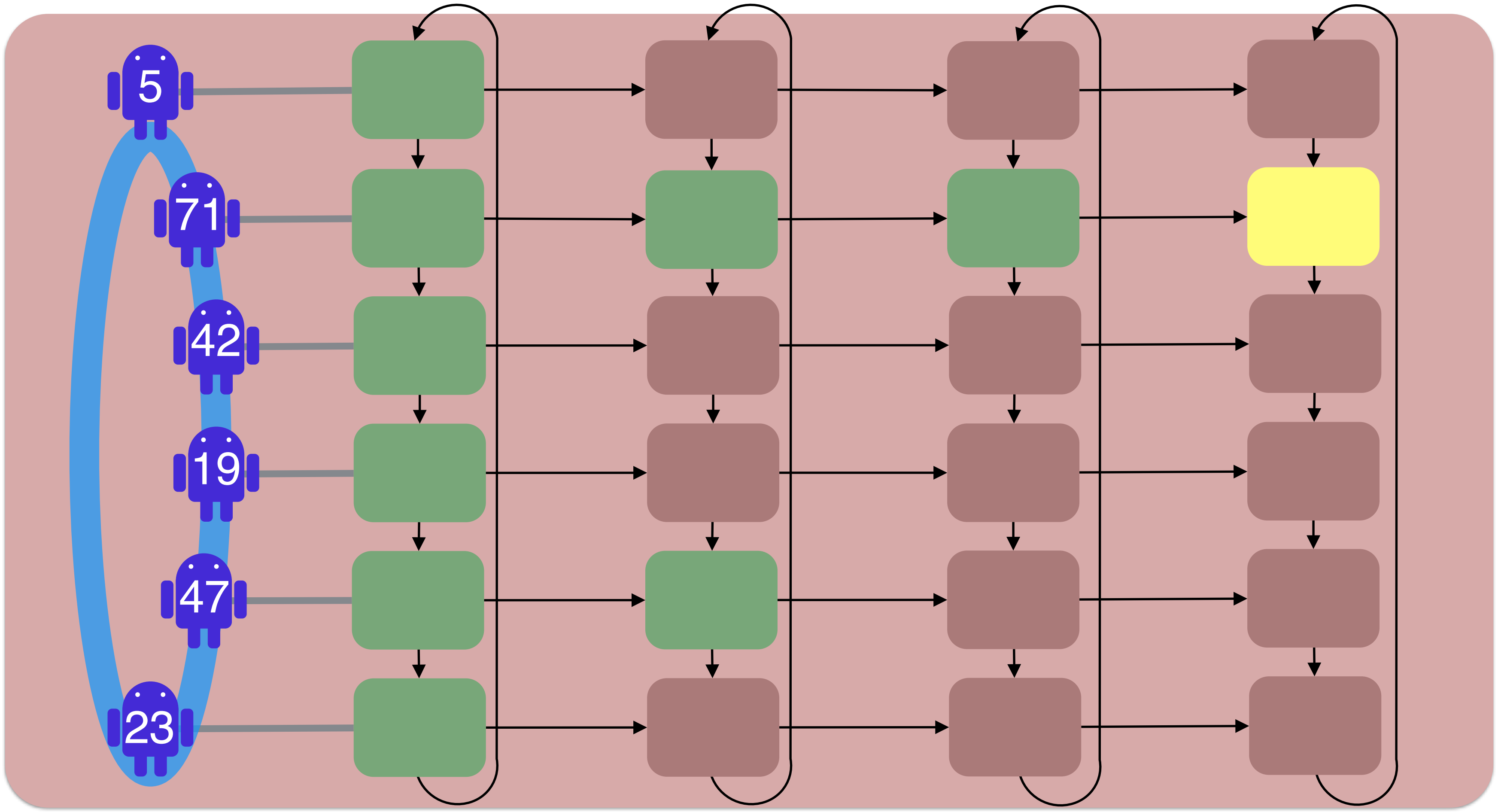
φ

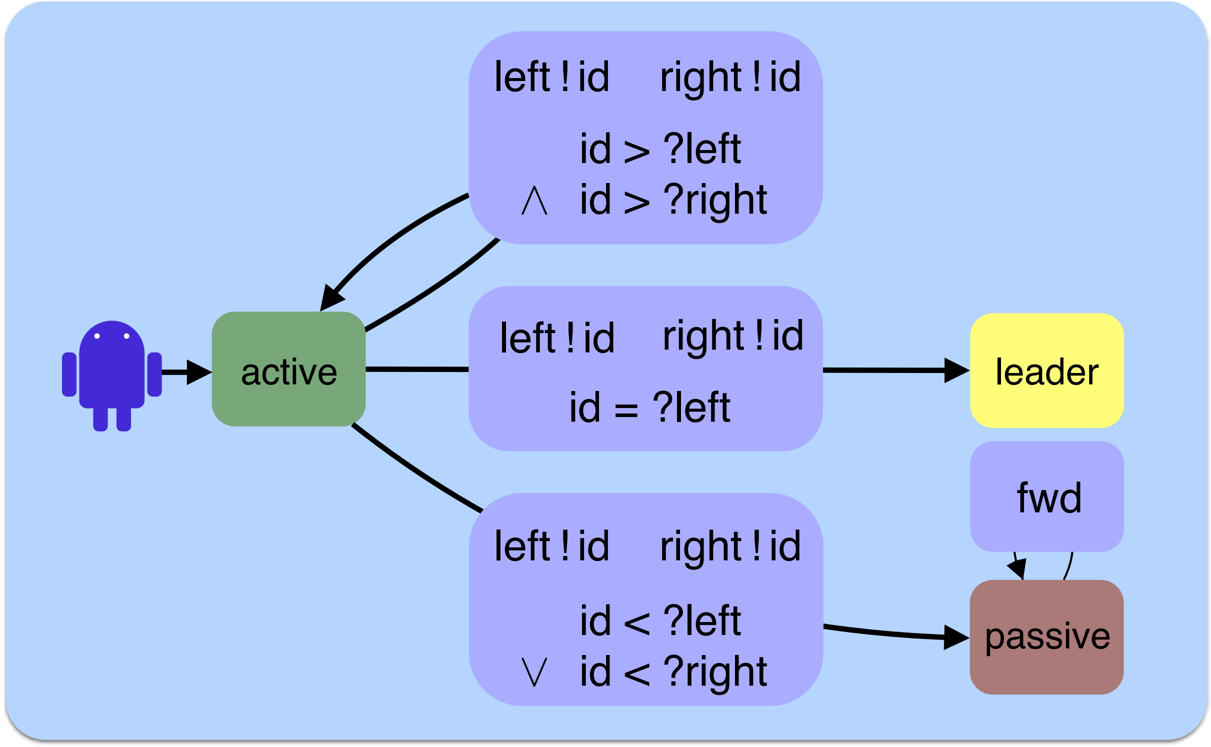
Distributed algorithm

PDL with loop (over finite alphabet)

Data PDL

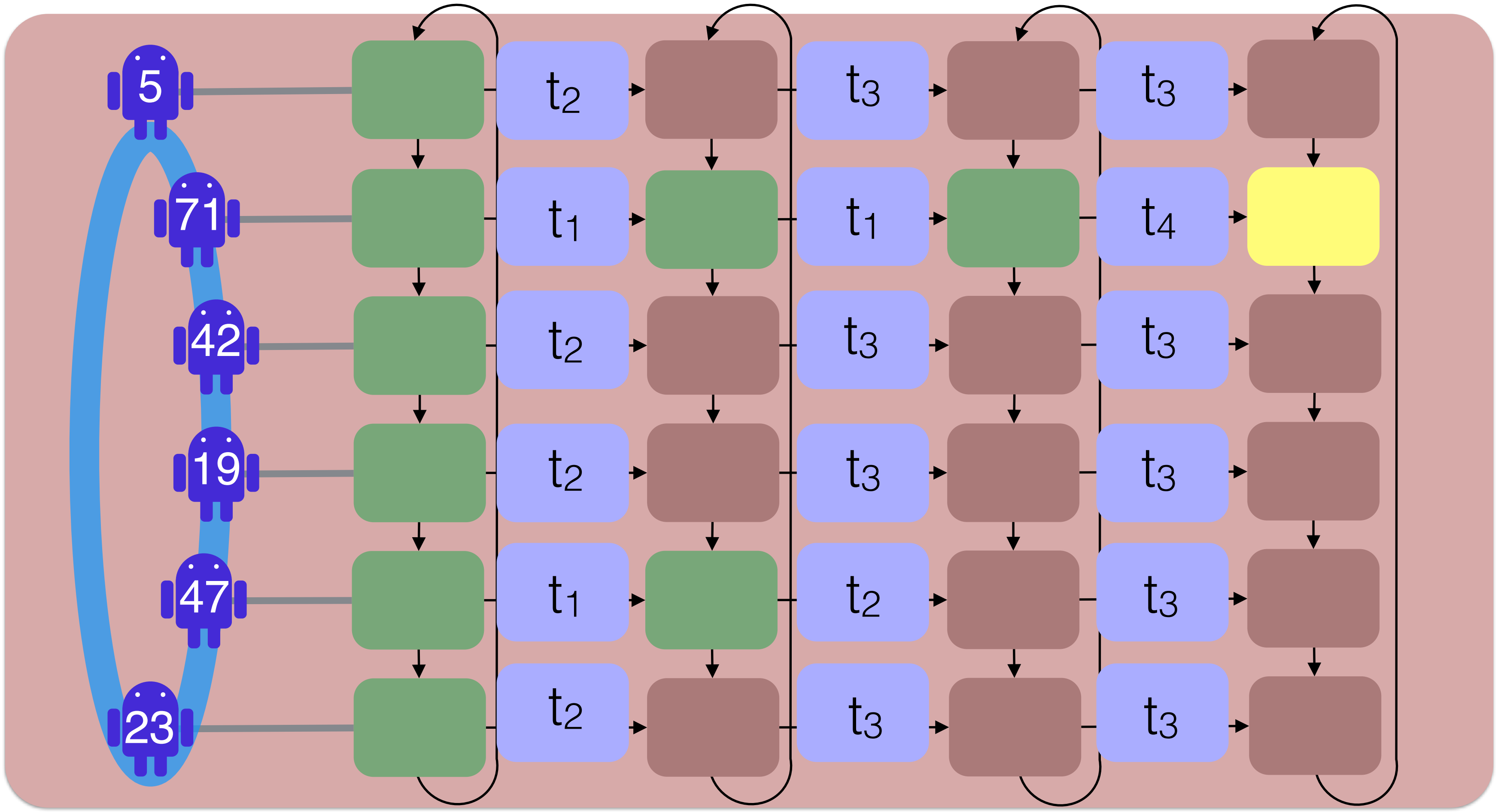
Data abstraction: symbolic runs + tracking data

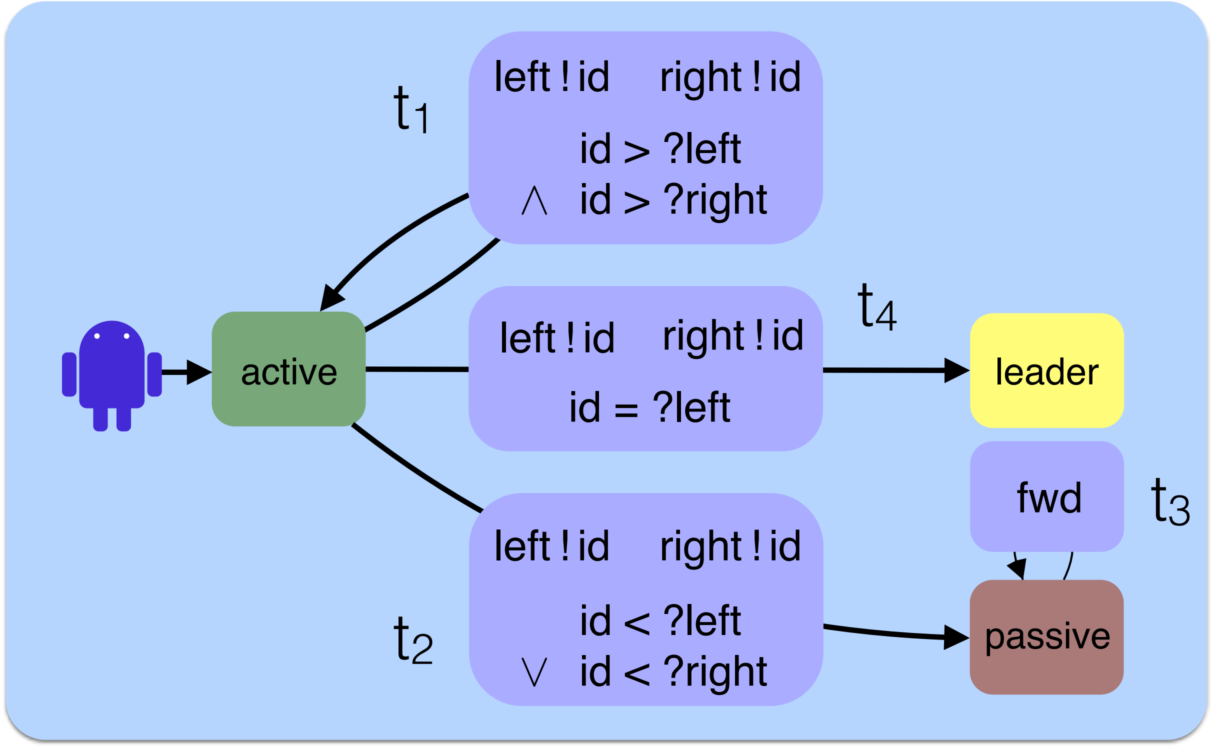




Distributed algorithm

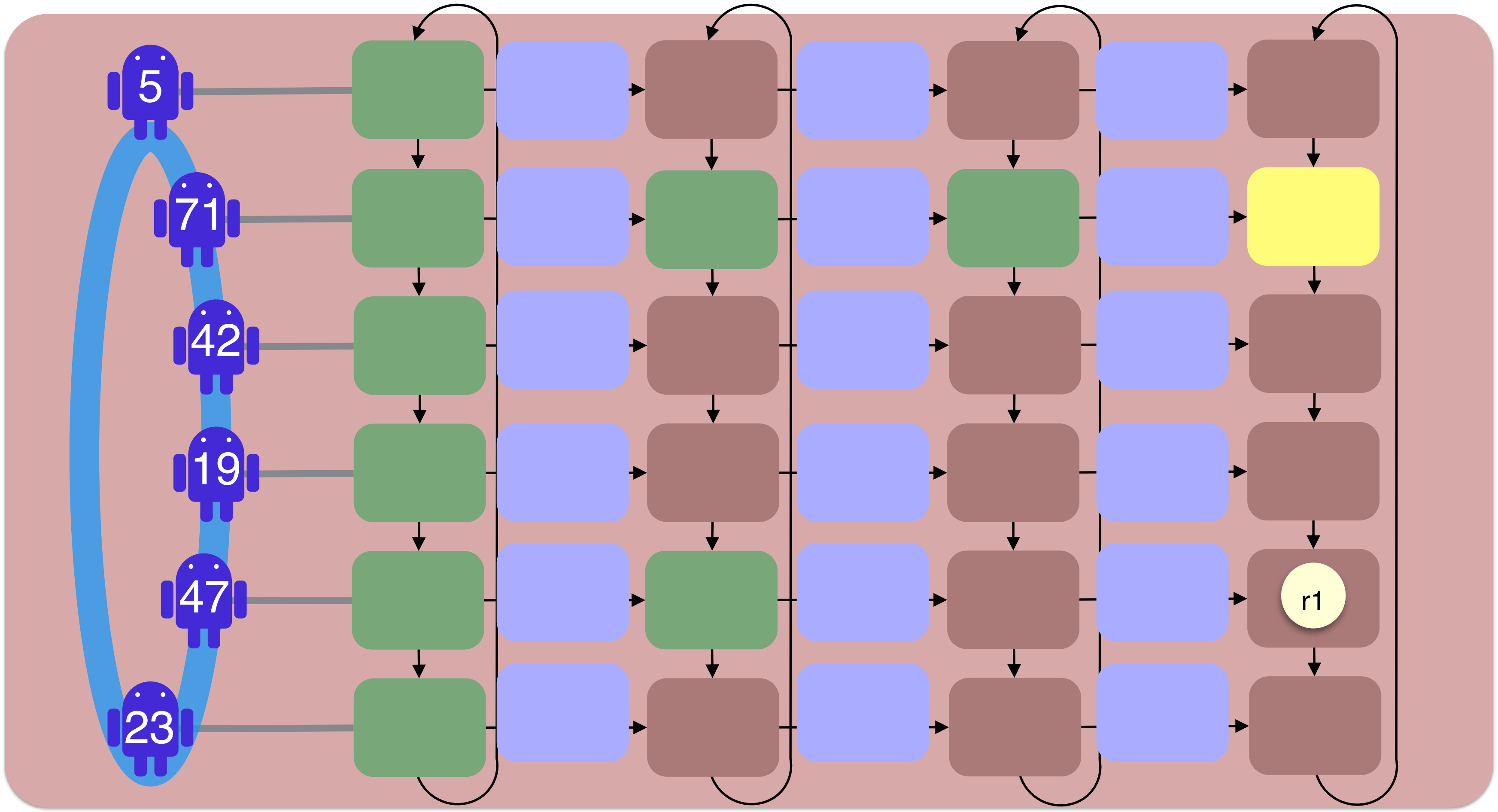
Data abstraction: symbolic runs + tracking data





Distributed algorithm

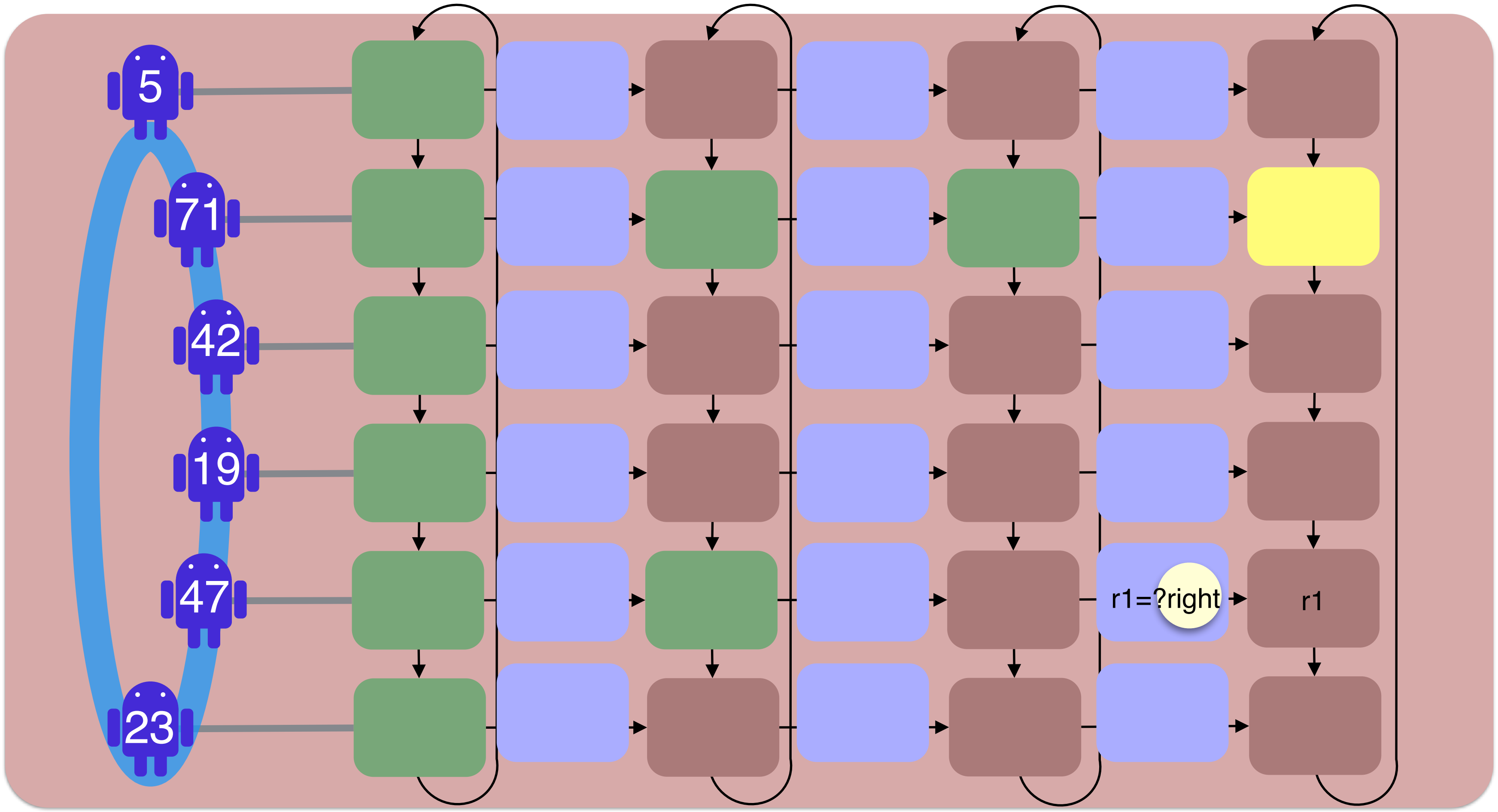
Data abstraction: symbolic runs + tracking data



- Register updates

Distributed algorithm

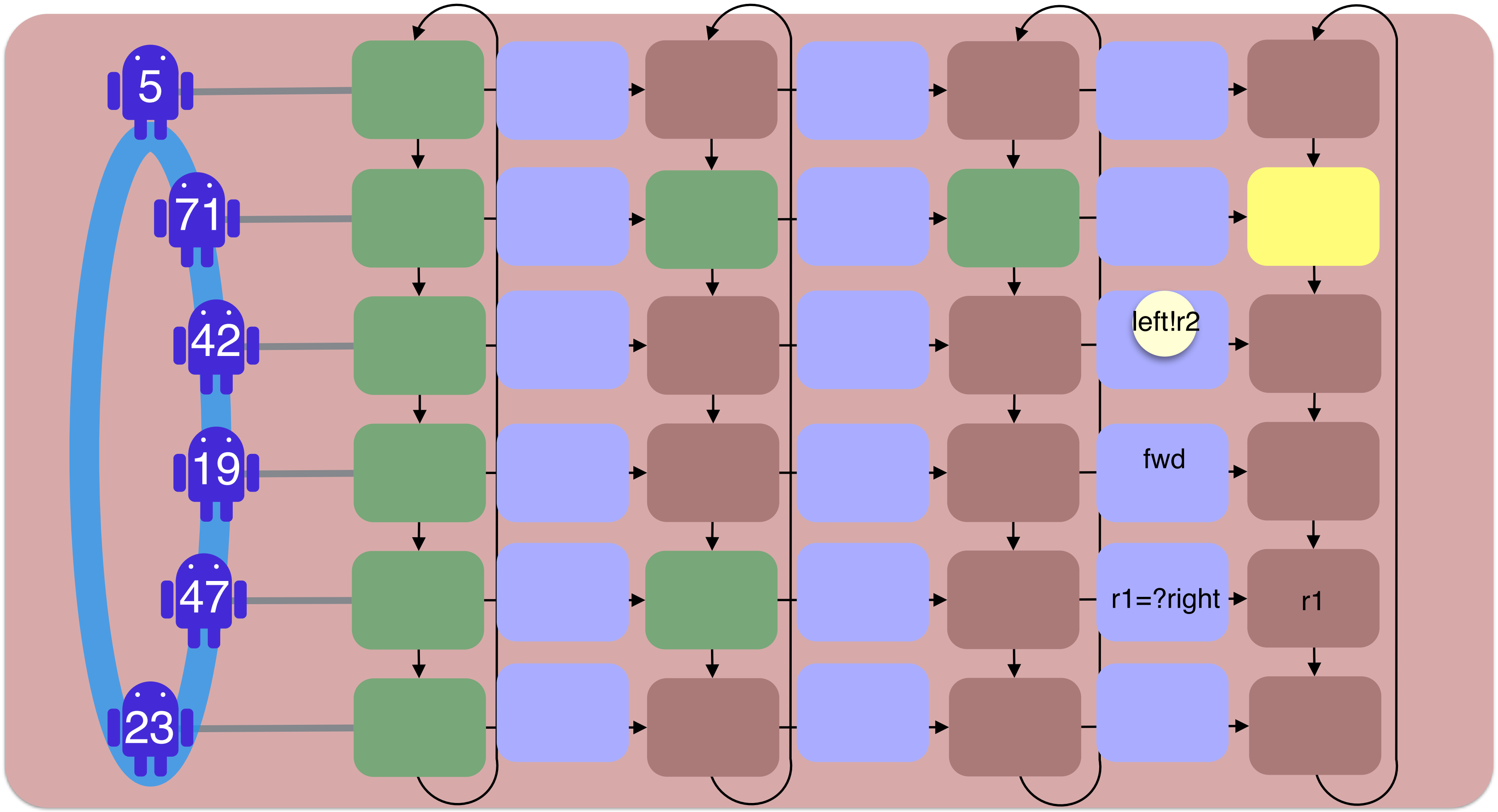
Data abstraction: symbolic runs + tracking data



- Register updates

Distributed algorithm

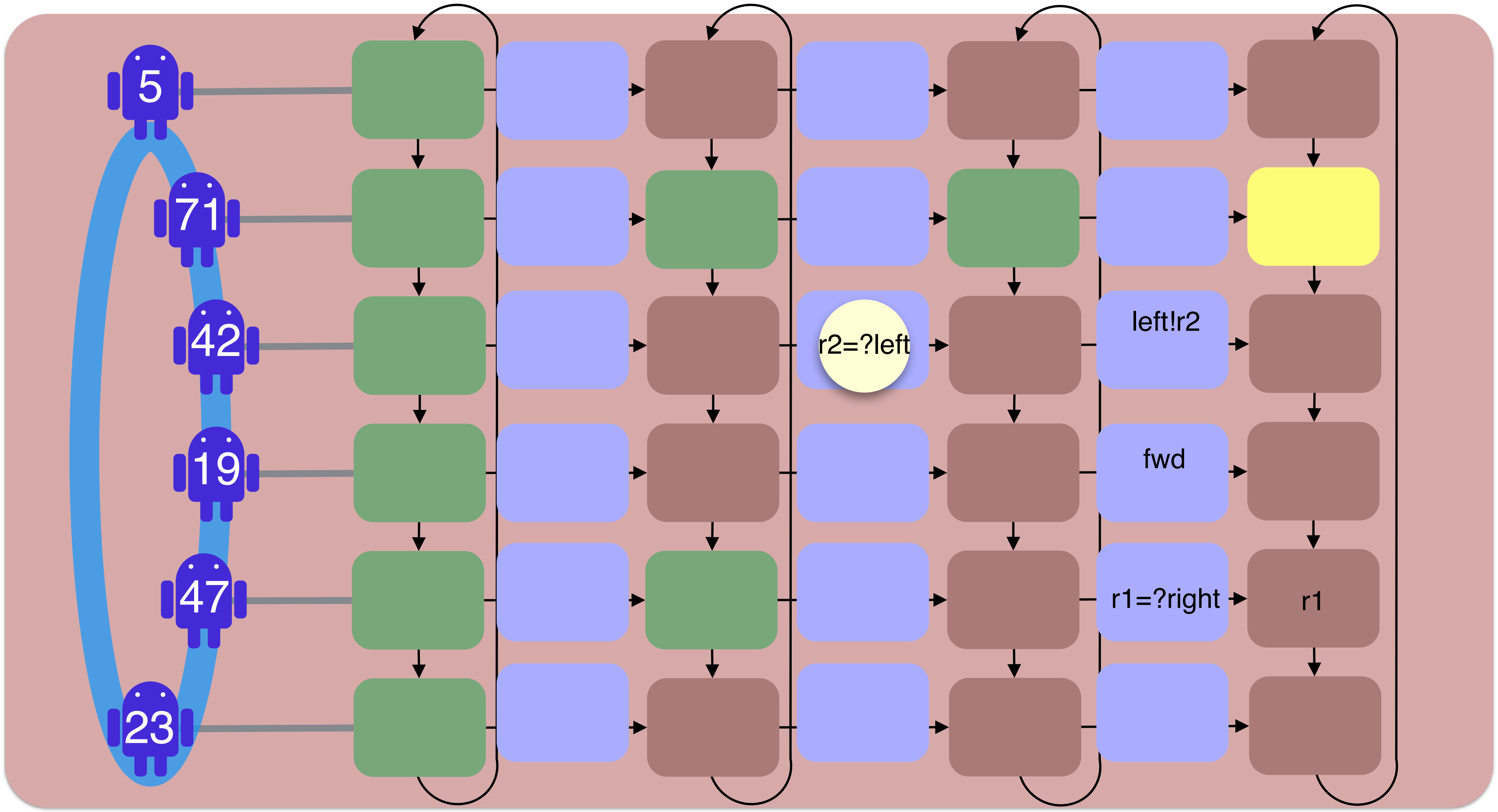
Data abstraction: symbolic runs + tracking data



- Register updates

Distributed algorithm

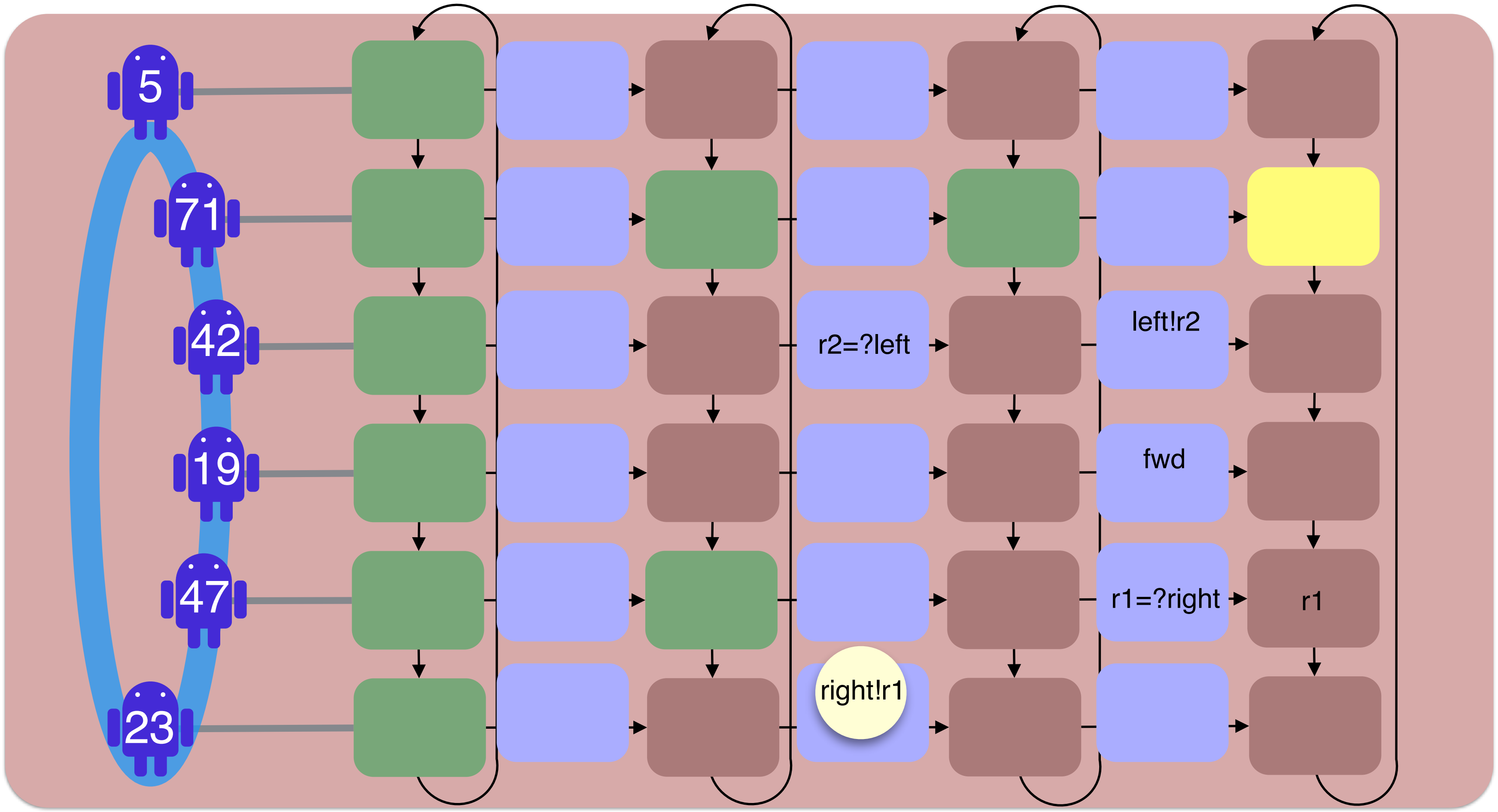
Data abstraction: symbolic runs + tracking data



- Register updates

Distributed algorithm

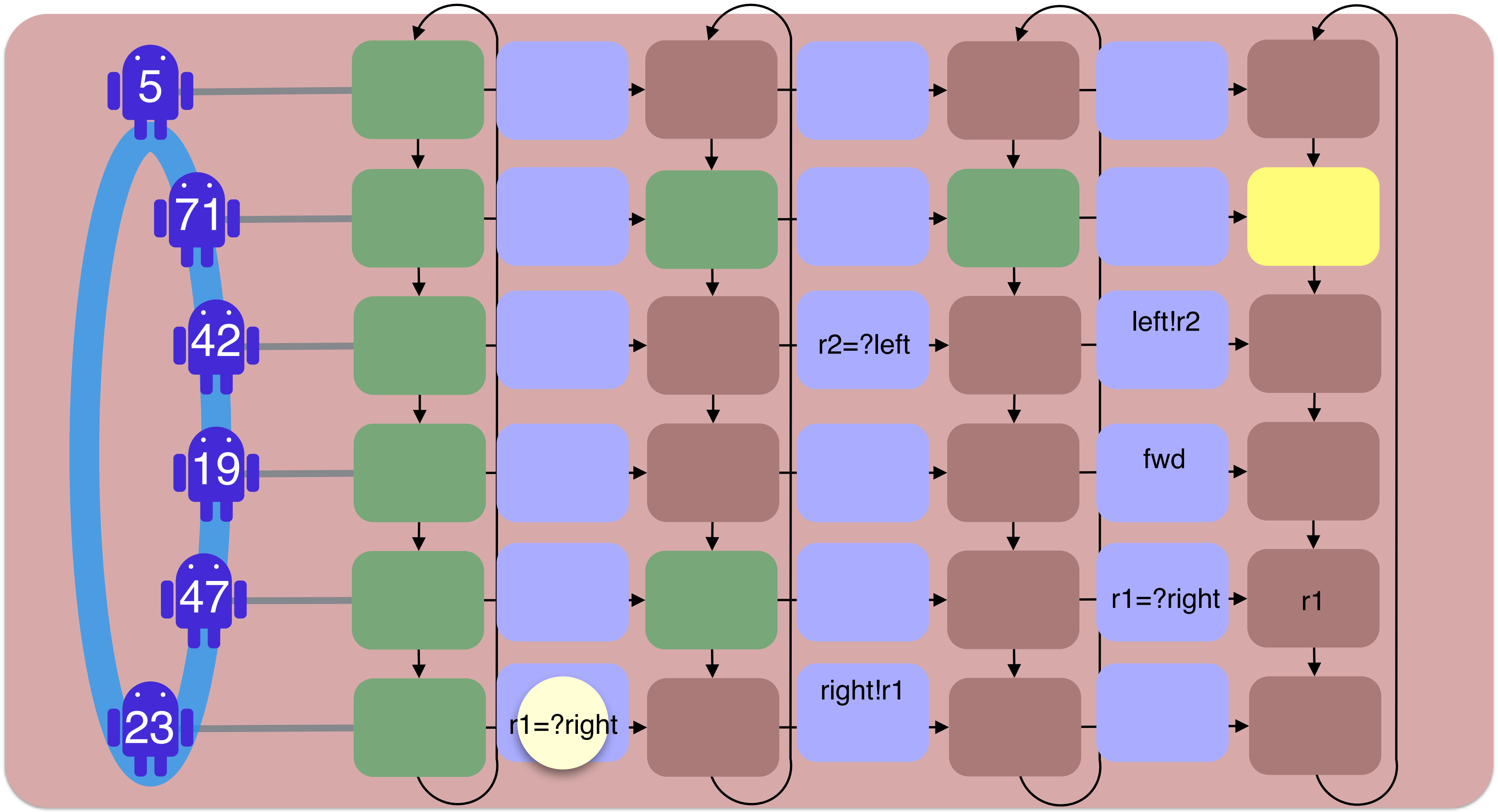
Data abstraction: symbolic runs + tracking data



- Register updates

Distributed algorithm

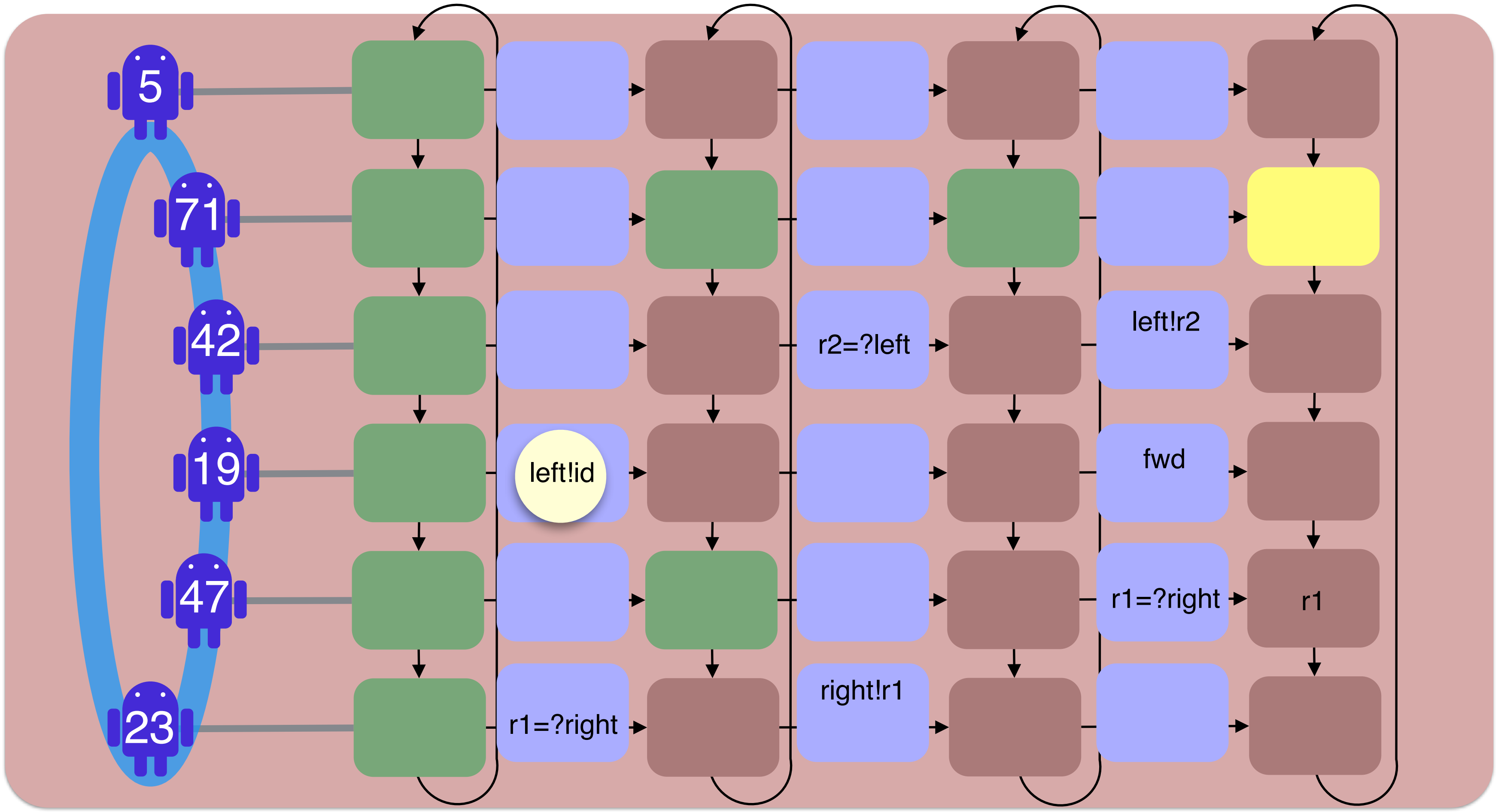
Data abstraction: symbolic runs + tracking data



- Register updates

Distributed algorithm

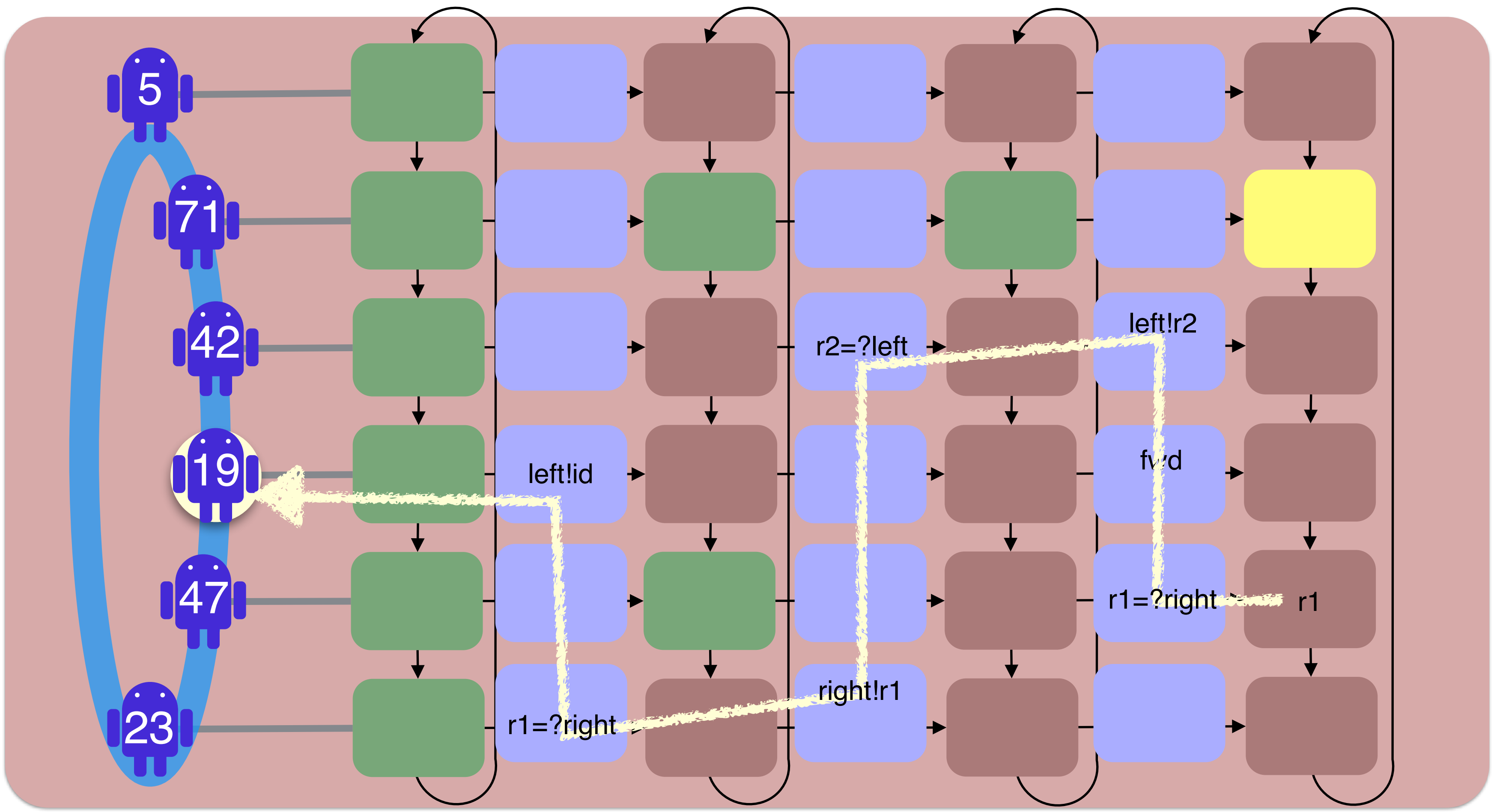
Data abstraction: symbolic runs + tracking data



- Register updates

Distributed algorithm

Data abstraction: symbolic runs + tracking data



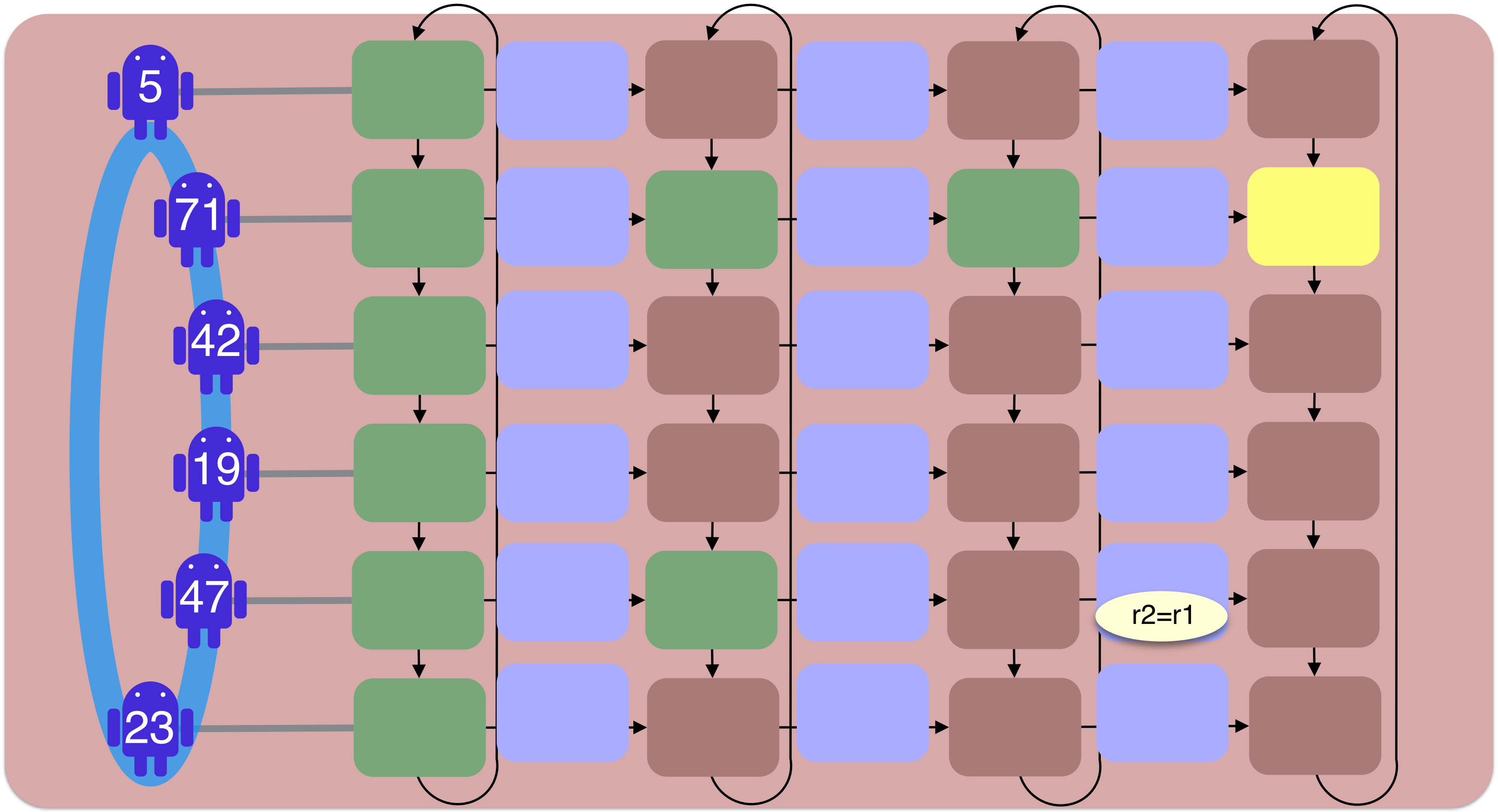
- Register updates

Distributed algorithm

(r_1,id)-path

can be expressed in PDL

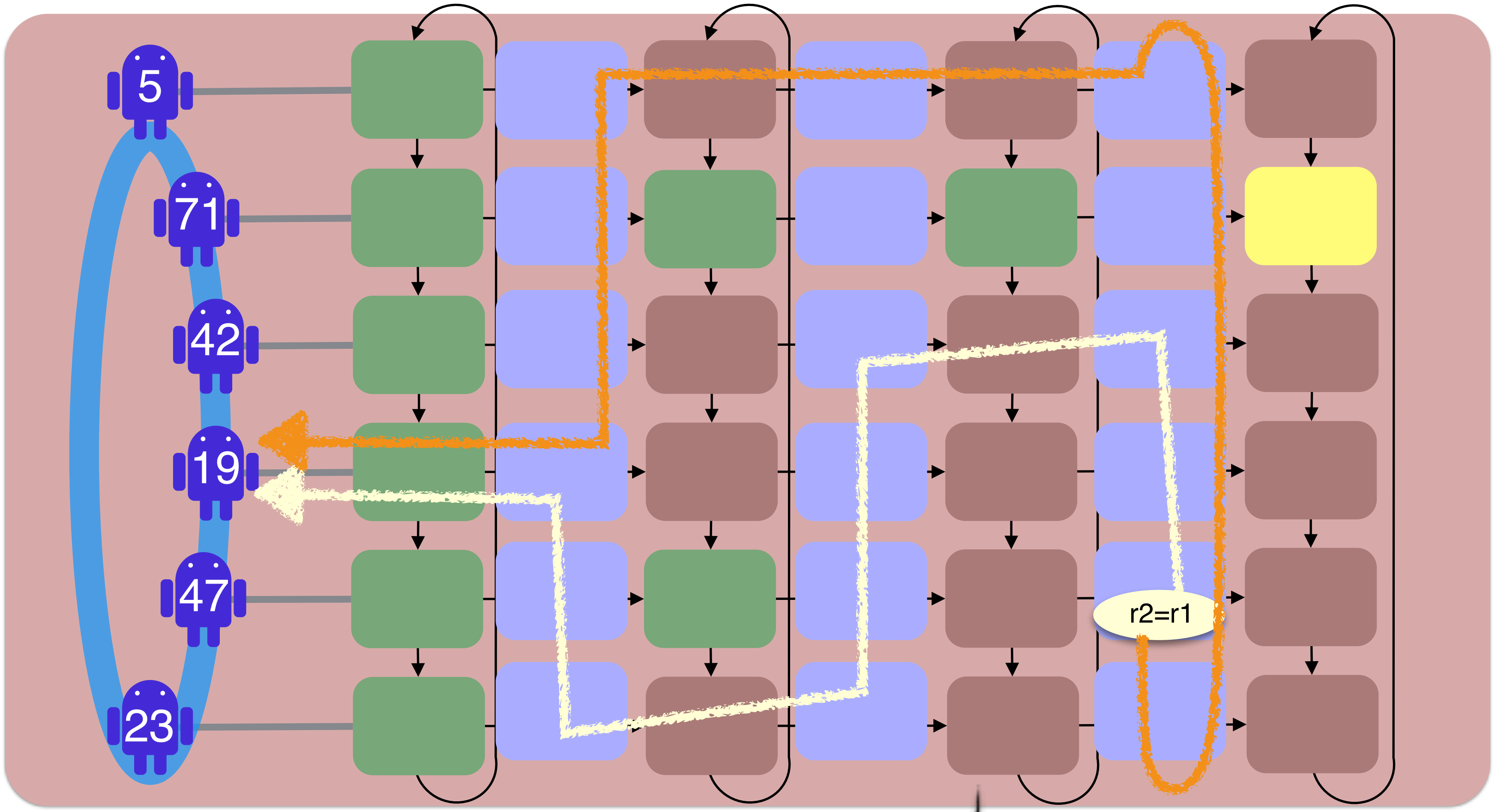
Data abstraction: symbolic runs + tracking data



- Register updates
- Register equality check

Distributed algorithm

Data abstraction: symbolic runs + tracking data



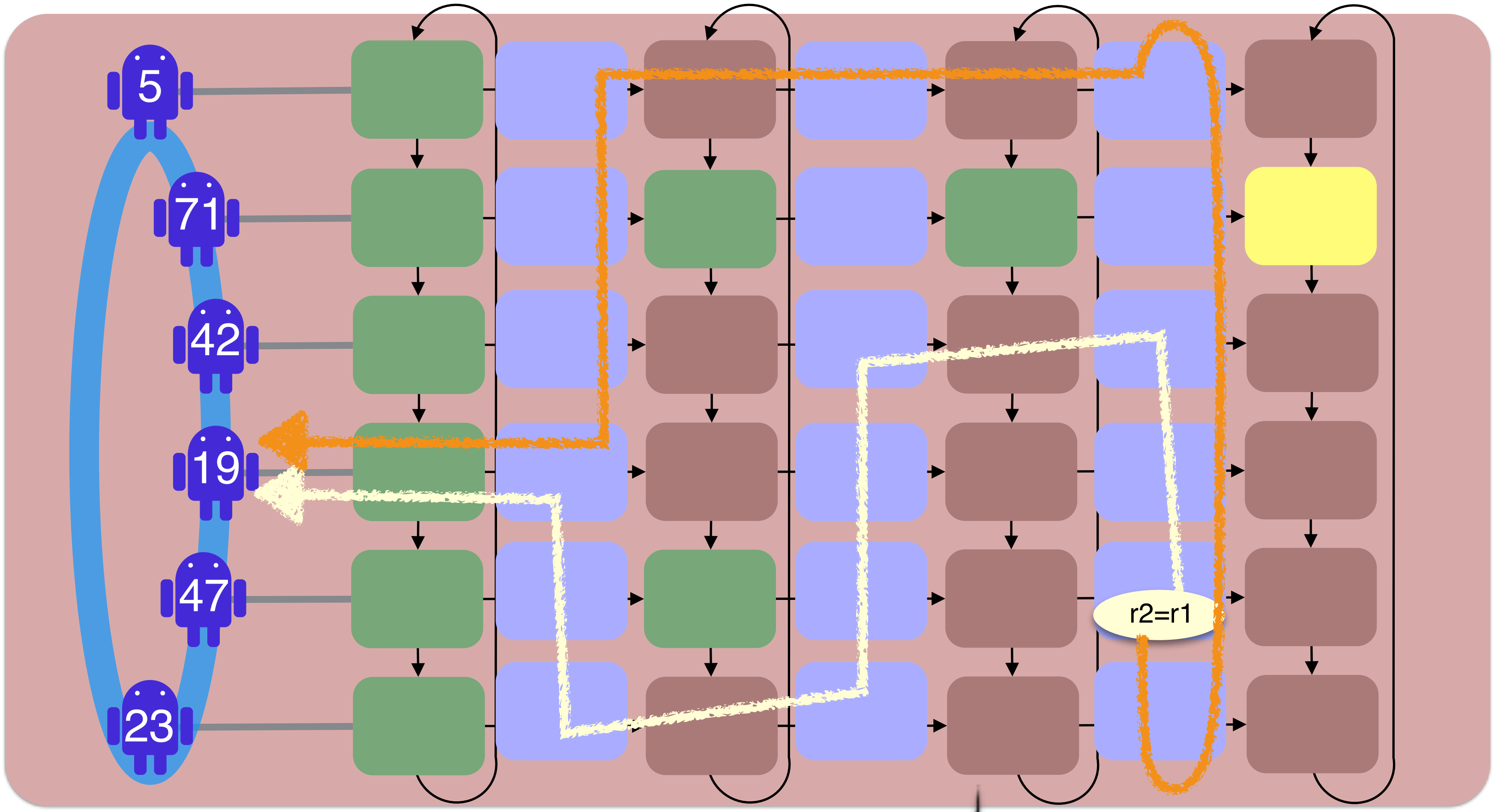
- Register updates
- Register equality check

Distributed algorithm

π_1:(r_1,id)-path

π_2:(r_2,id)-path

Data abstraction: symbolic runs + tracking data



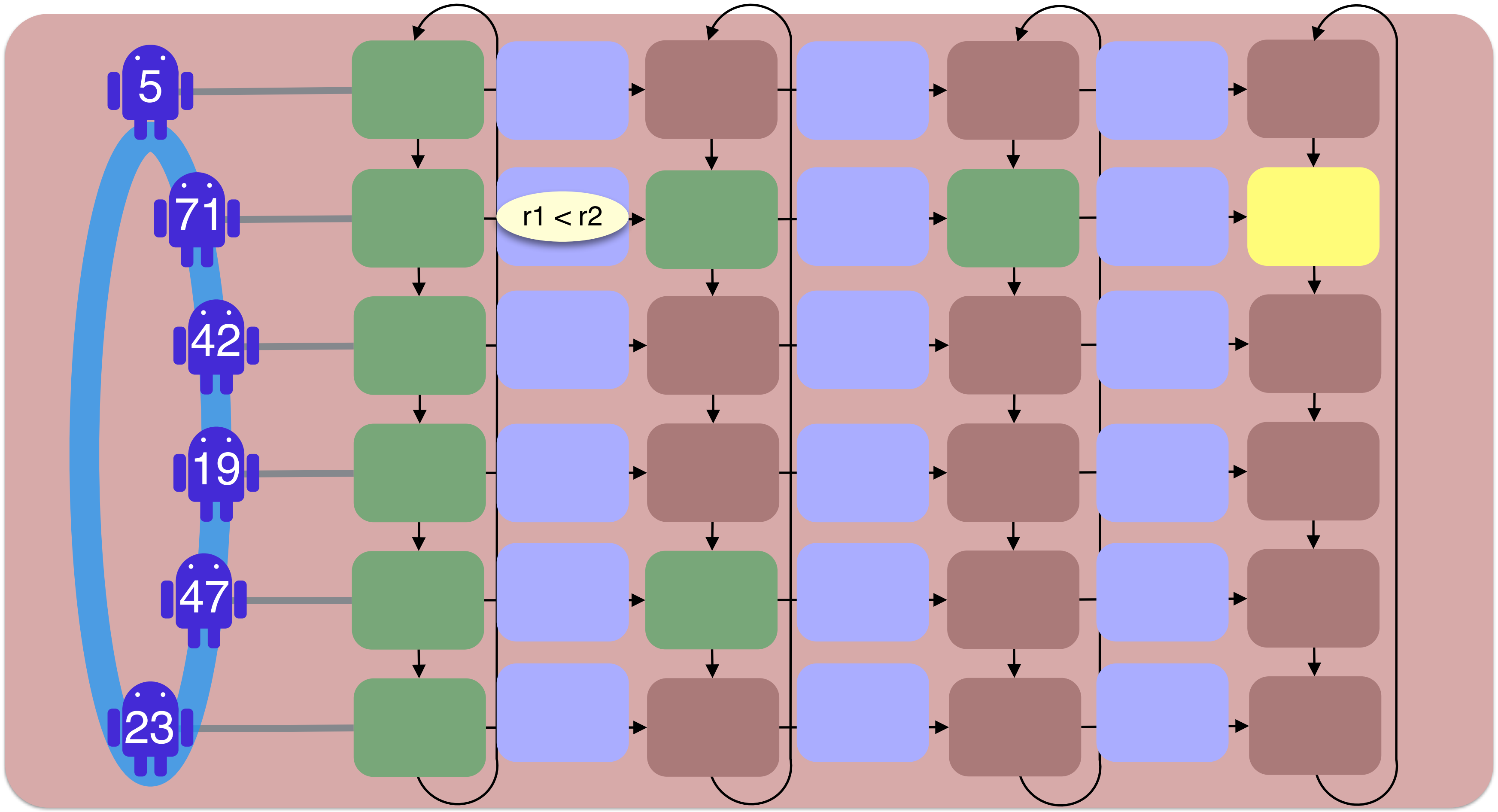
- Register updates
- Register equality check

Distributed algorithm

π_1:(r_1,id)-path
π_2:(r_2,id)-path
loop(π_1 π_2^{-1})

can be expressed
in PDL with loop

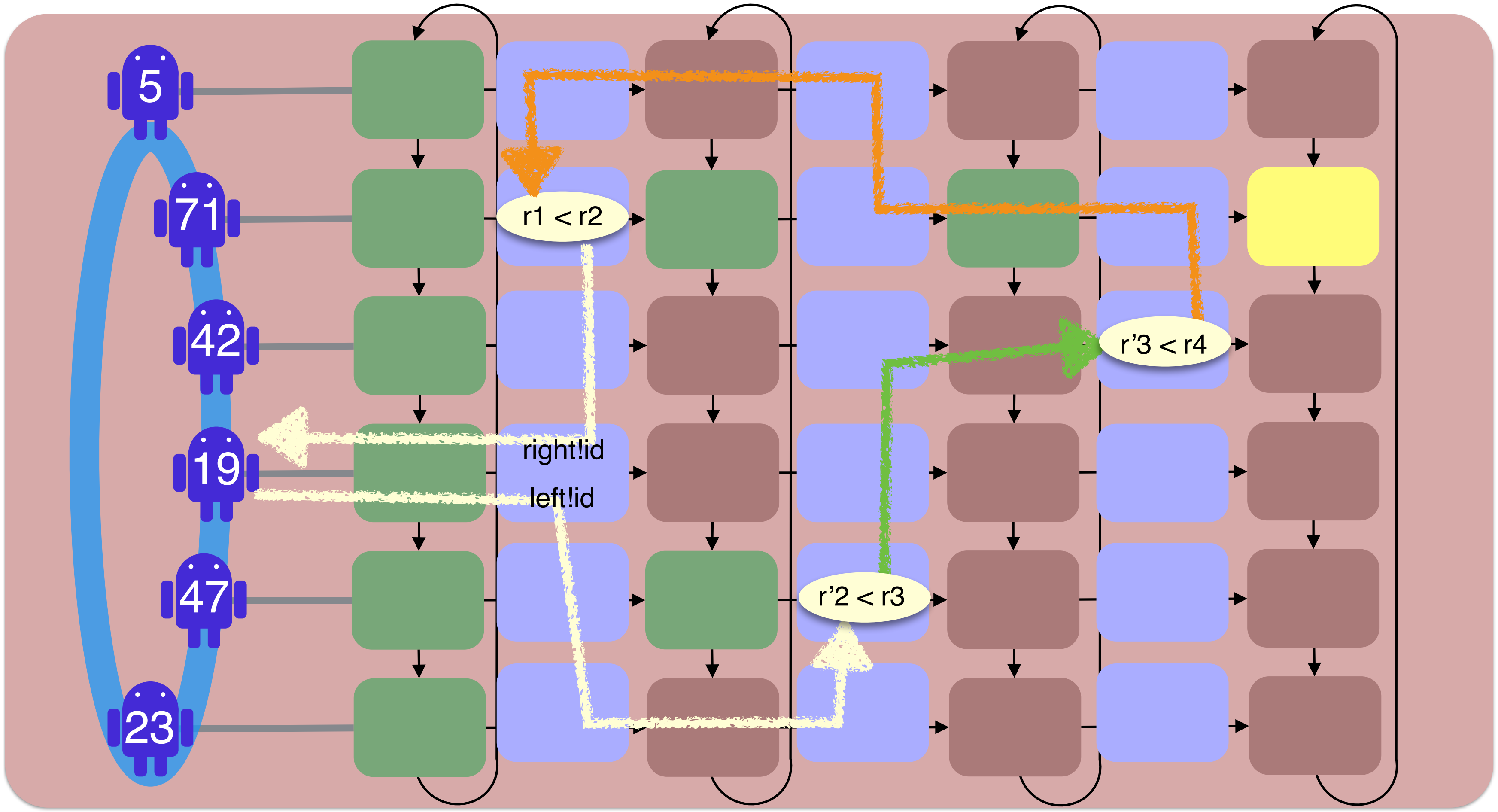
Data abstraction: symbolic runs + tracking data



- Register updates
- Register equality check
- Register comparison

Distributed algorithm

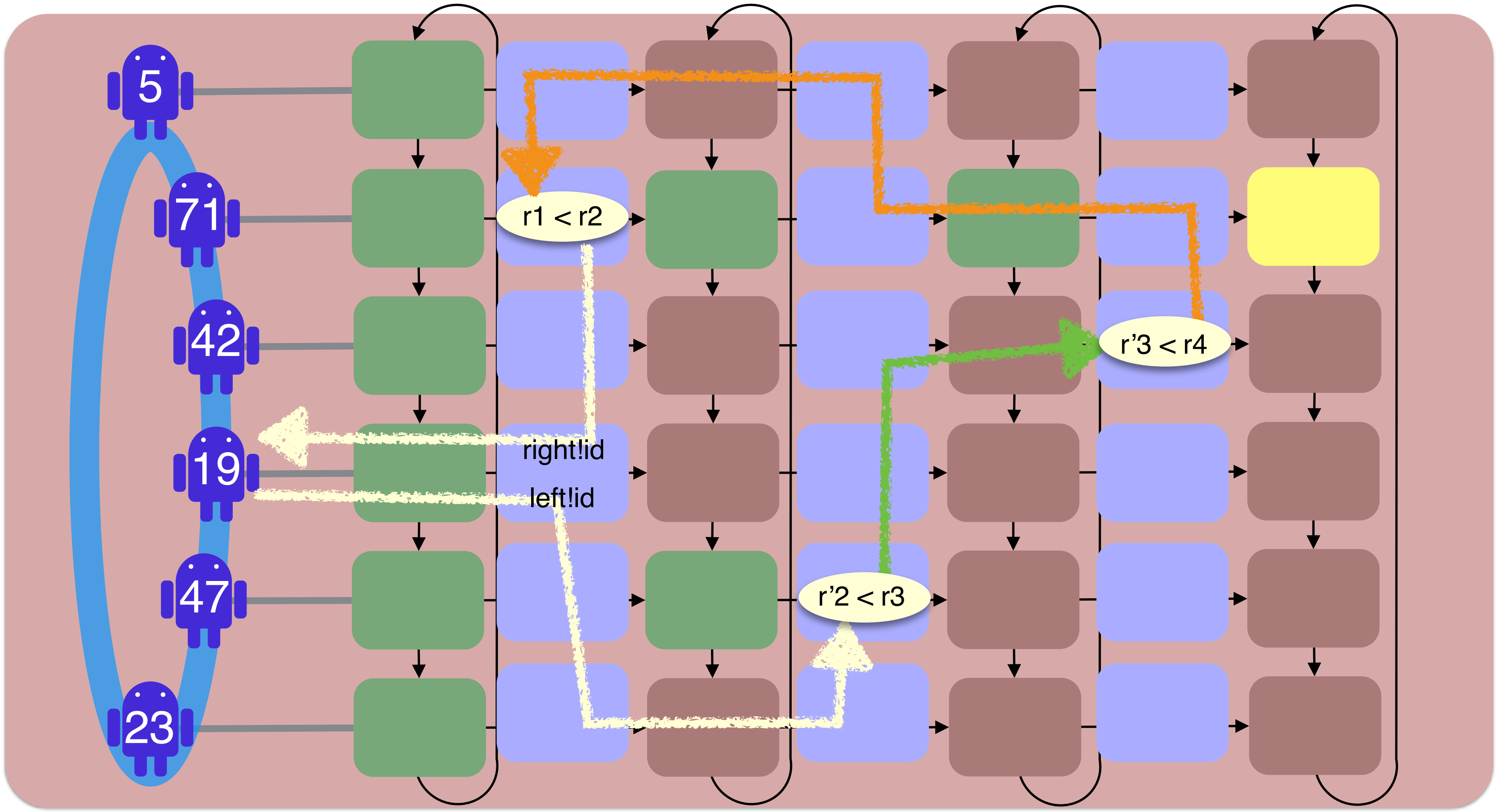
Data abstraction: symbolic runs + tracking data



- Register updates
- Register equality check
- Register comparison

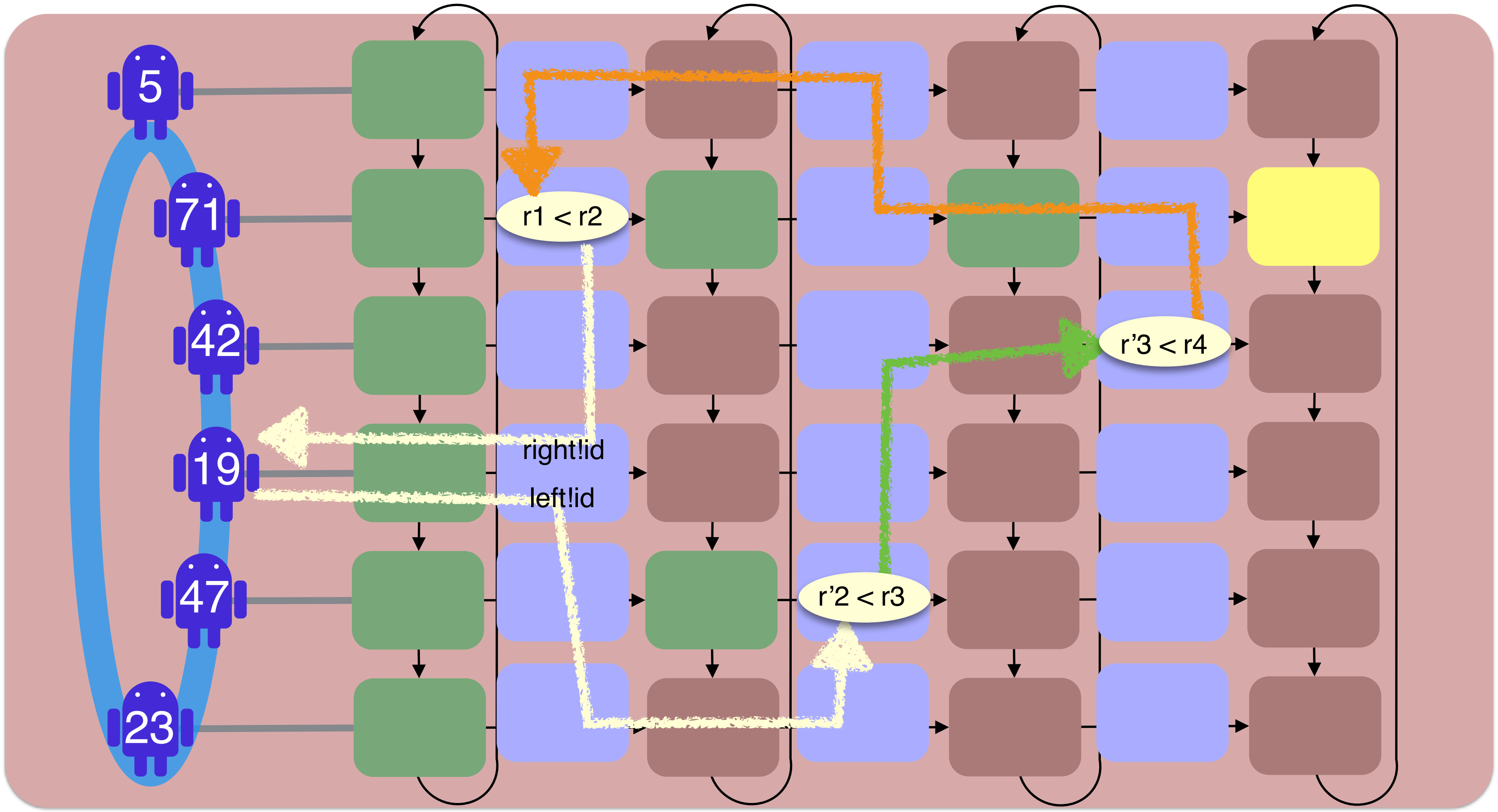
Distributed algorithm

Data abstraction: symbolic runs + tracking data



- If there is a loop, no pids assignment 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

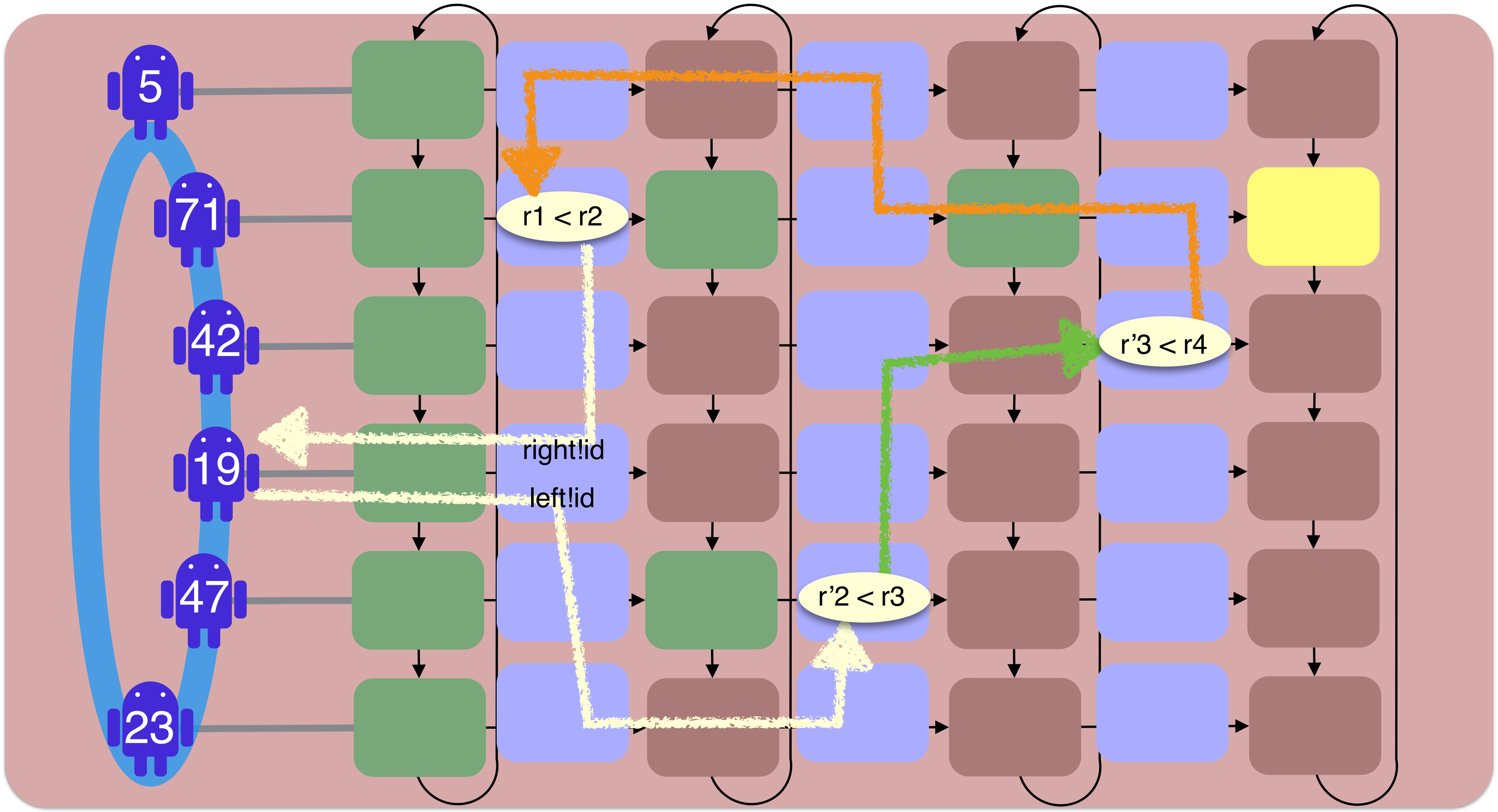


No loop of the form

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

- If there is a loop, no pids assignment 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

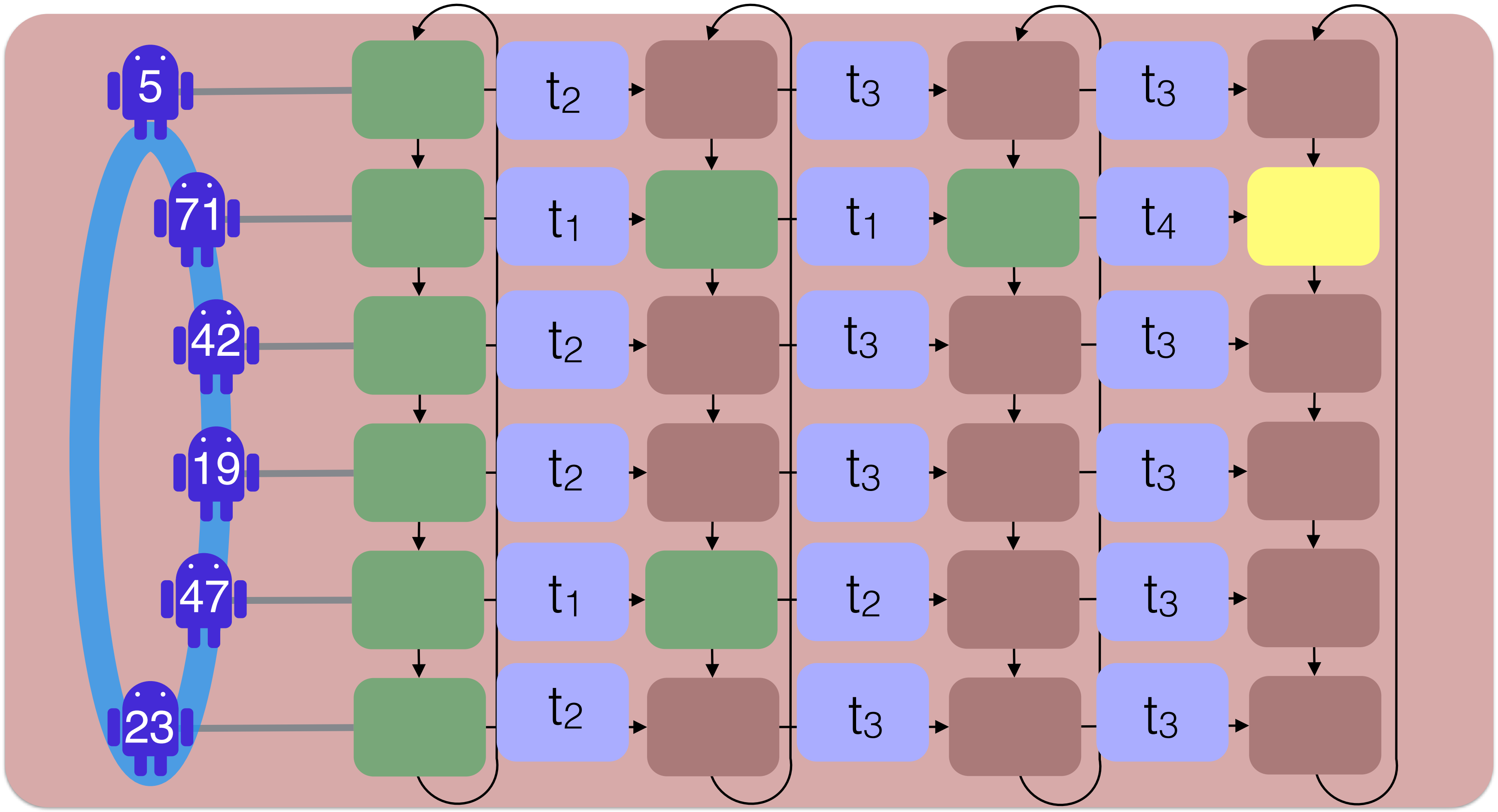
$r_{i0}<r_{i1}$; (r_{i1},r_{i2})-path; $r_{i2}<r_{i3}$; (r_{i3},r_{i4})-path; … ; $r_{in}<r_{i0}$

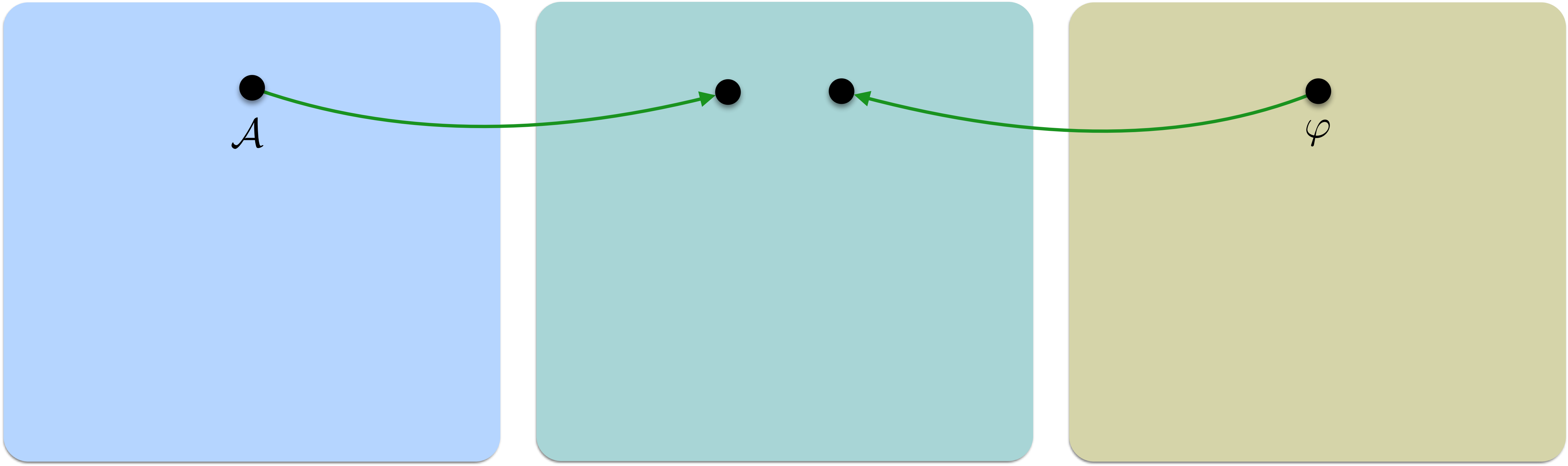
- Register updates
- Register equality check
- Register comparison

Distributed algorithm

can be expressed in PDL with loop

Data abstraction: symbolic runs + tracking data



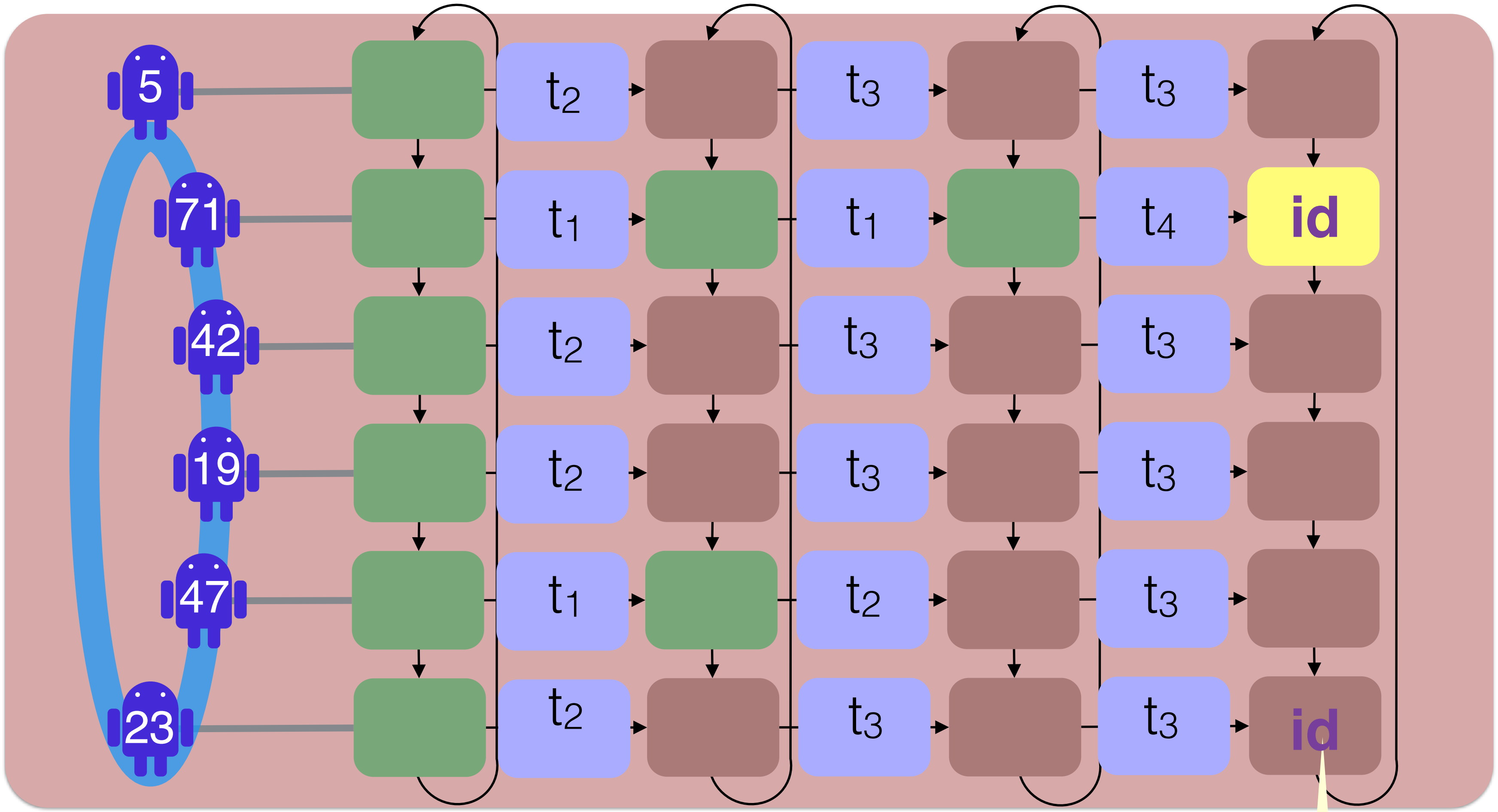


Distributed algorithm

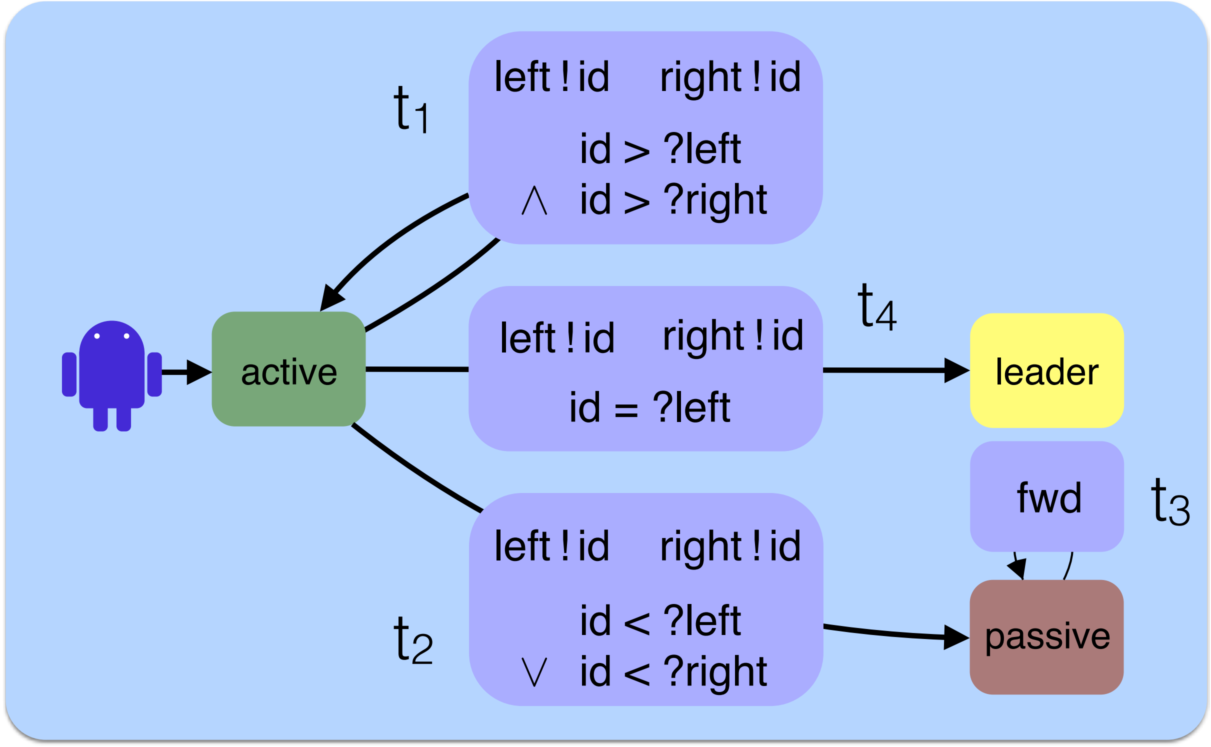
PDL with loop (over finite alphabet)

Data PDL

Distributed algorithms



Behavior


t1
left ! id right ! id
id > ?left
∧ id > ?right

active

left ! id right ! id
id = ?left
t4
leader

fwd
t3

left ! id right ! id
id < ?left
∨ id < ?right
t2
passive

Distributed algorithm

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

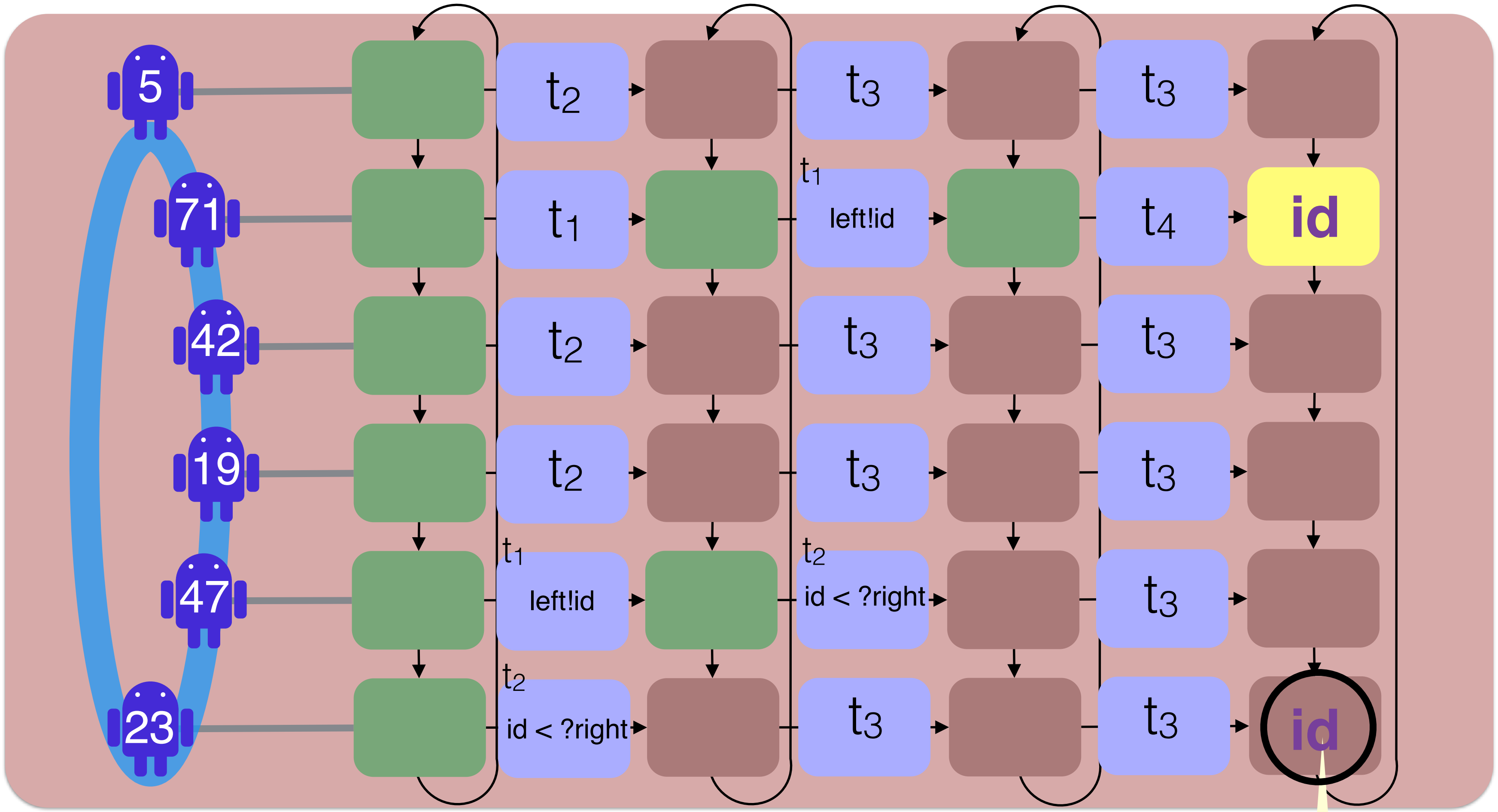
For all *n*, pid distributions, **accepting runs**, and processes:

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

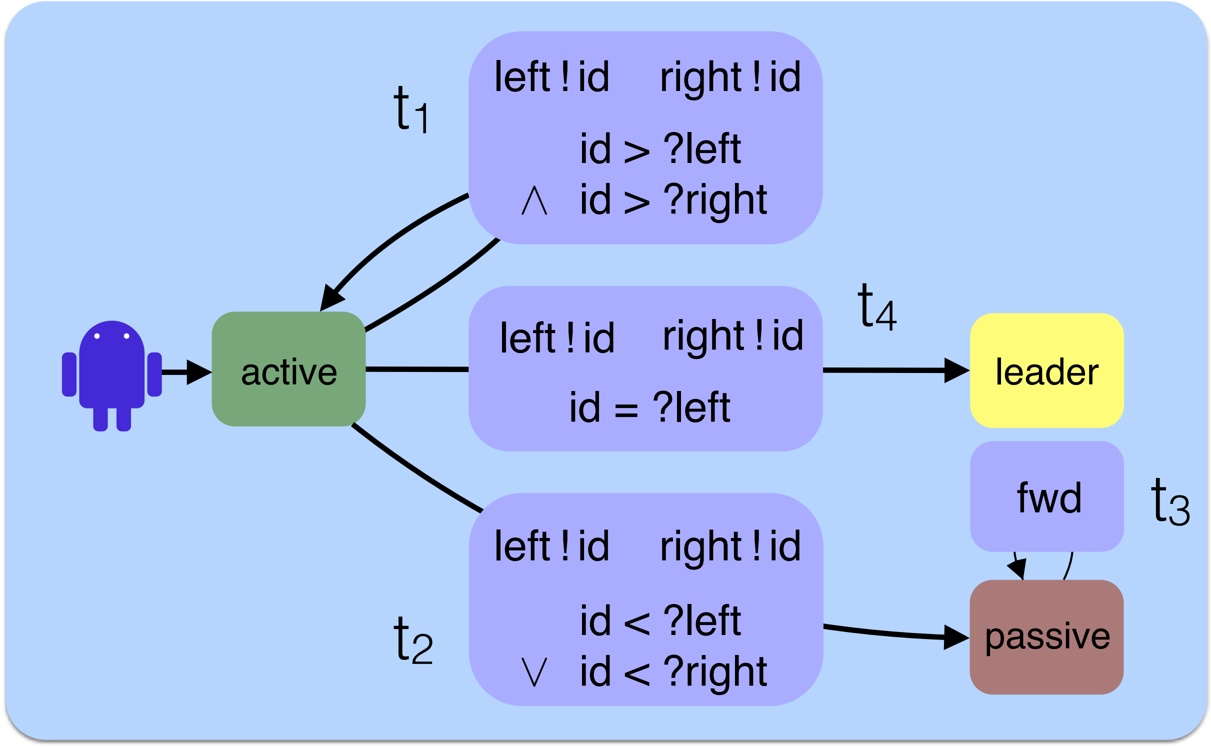
$$\text{go-to-}\square \;=\; (\neg \square \downarrow)^{*}\, \square$$

Data PDL

Distributed algorithms



Behavior



Distributed algorithm

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

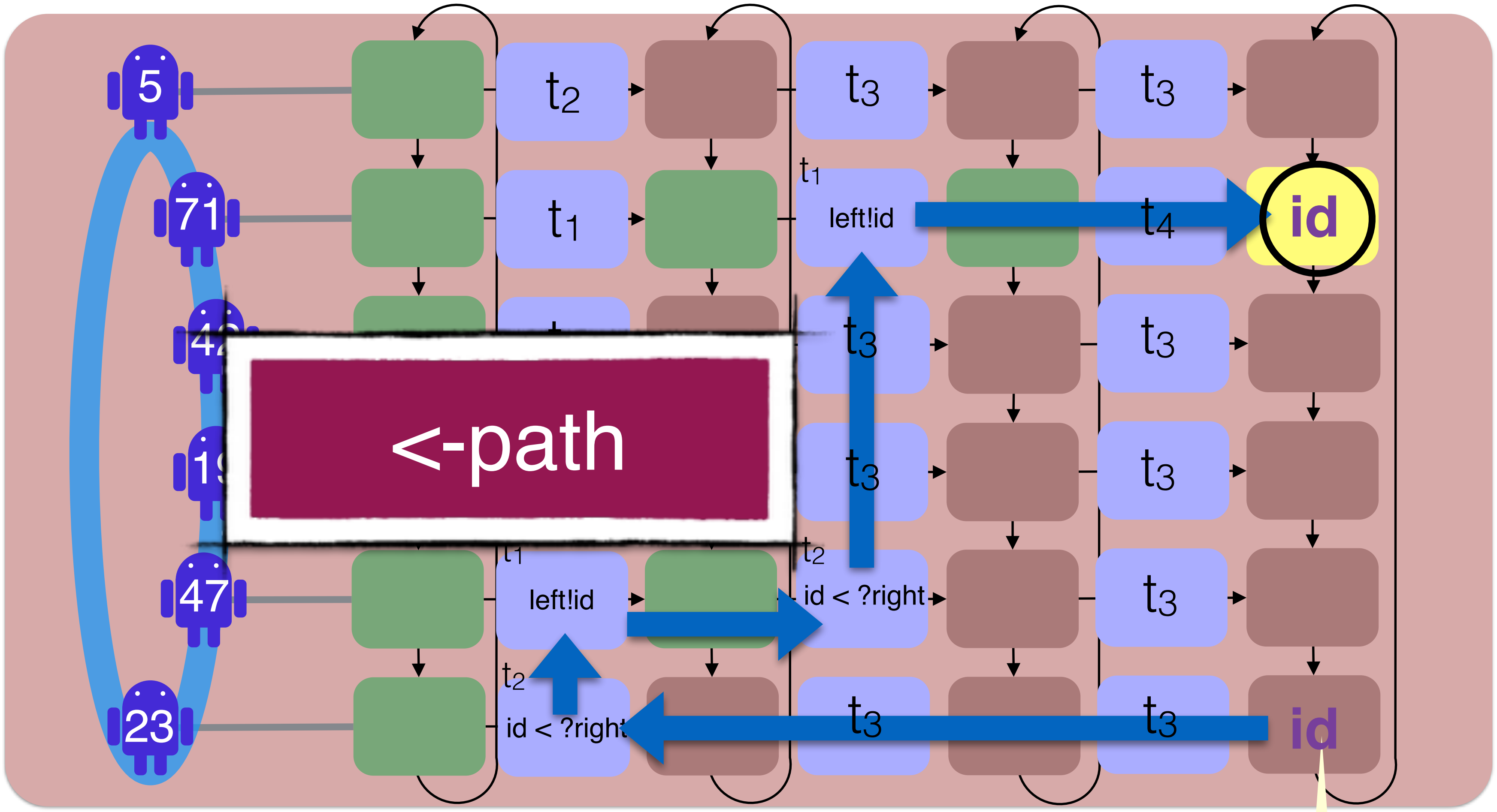
For all *n*, pid distributions, **accepting runs**, and processes:

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

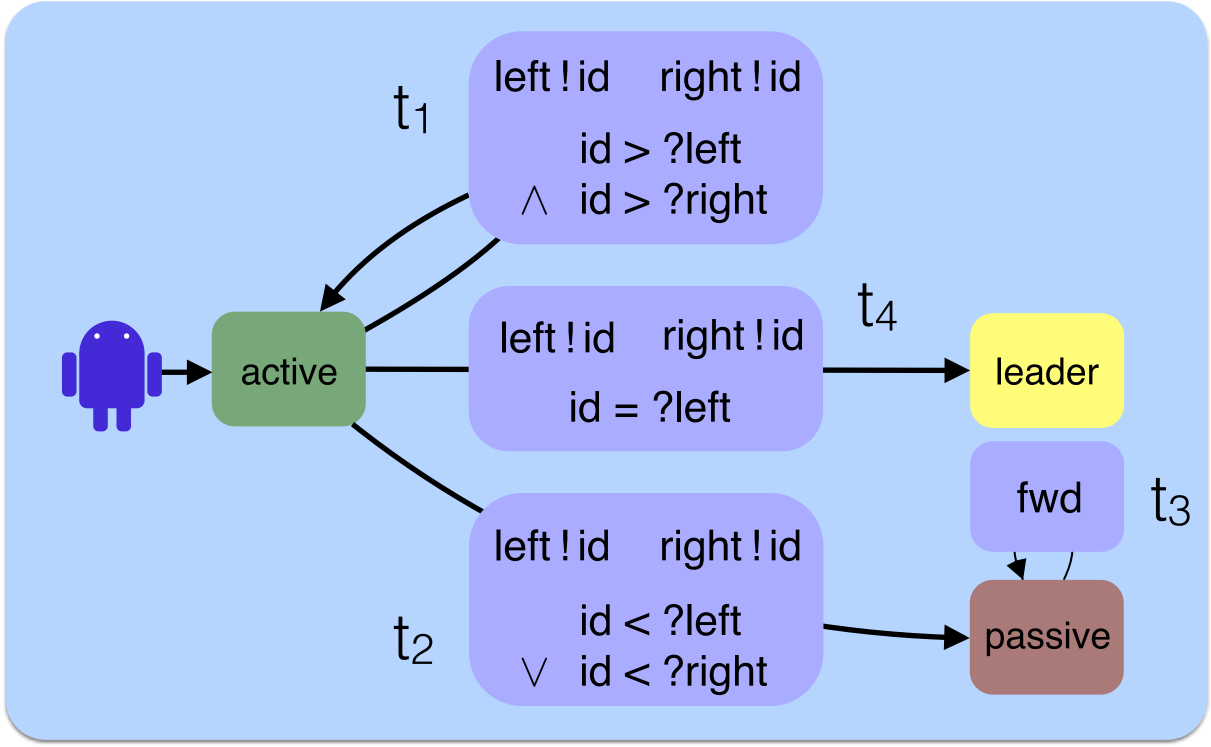
$$\text{go-to-}\square \;=\; (\neg \square \downarrow)^{*}\, \square$$

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:

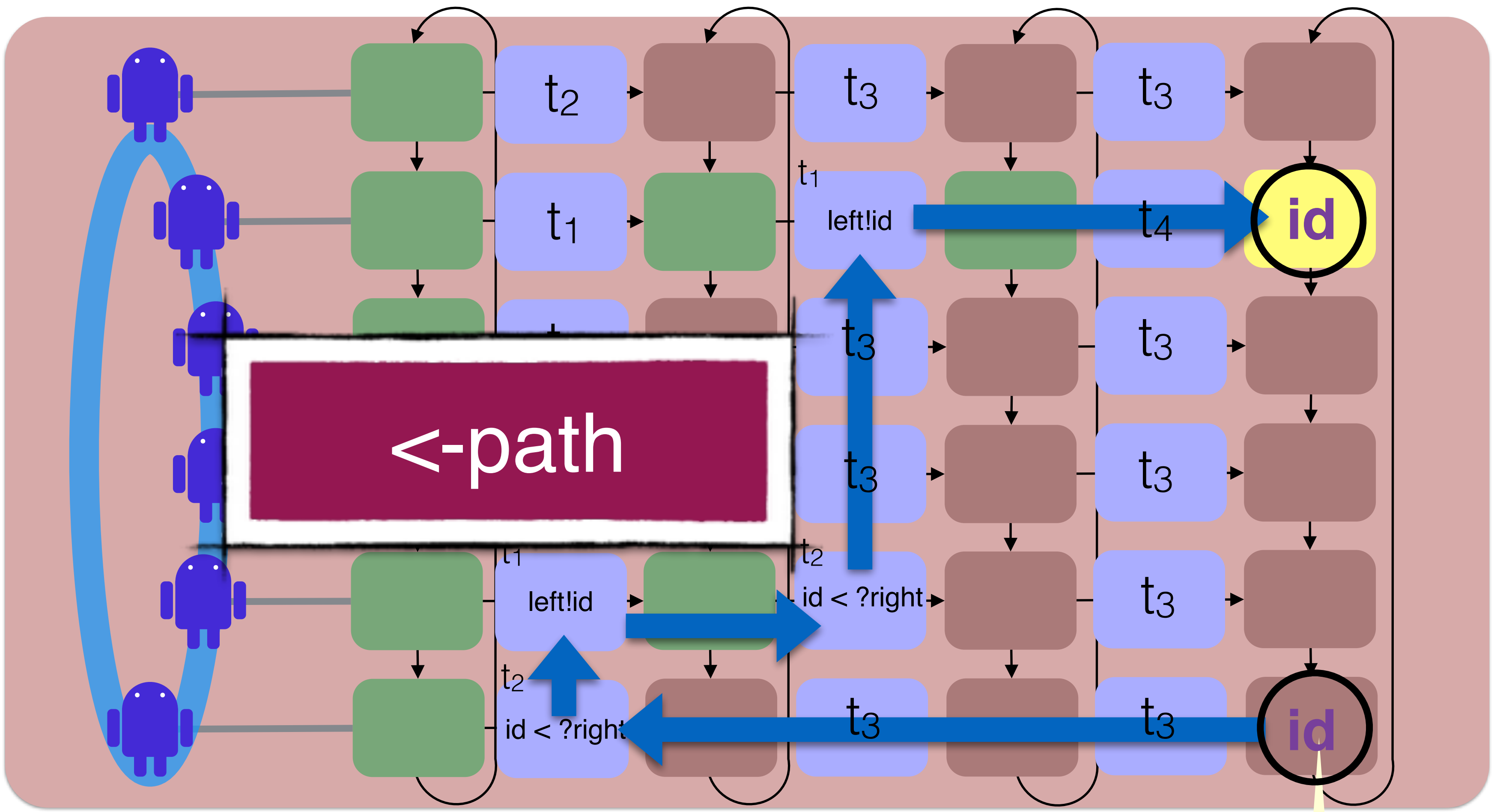
$$\langle \longrightarrow^* \rangle \; (\quad \neg \langle \longrightarrow \rangle \;\wedge\; \langle \text{go-to-}\square \rangle$$

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

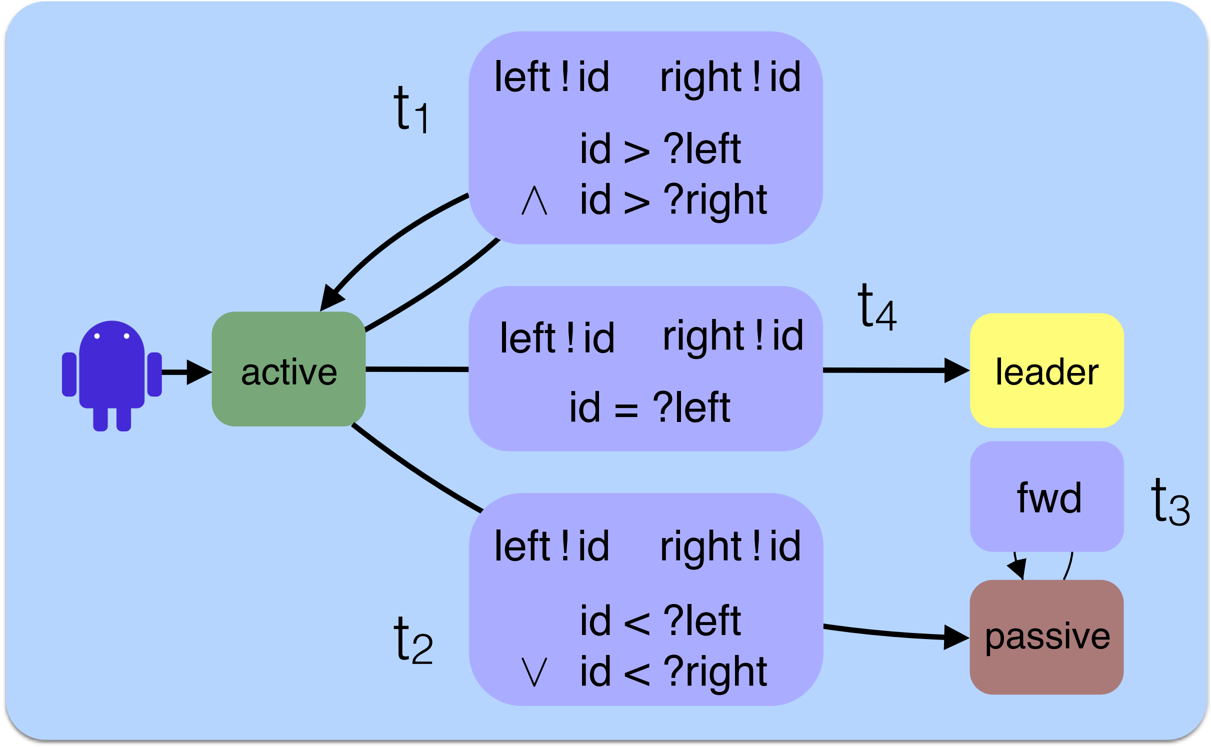
$$\text{go-to-}\square \;=\; (\neg \square \downarrow)^* \; \square$$

Data PDL

Distributed algorithms



Behavior



Distributed algorithm

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

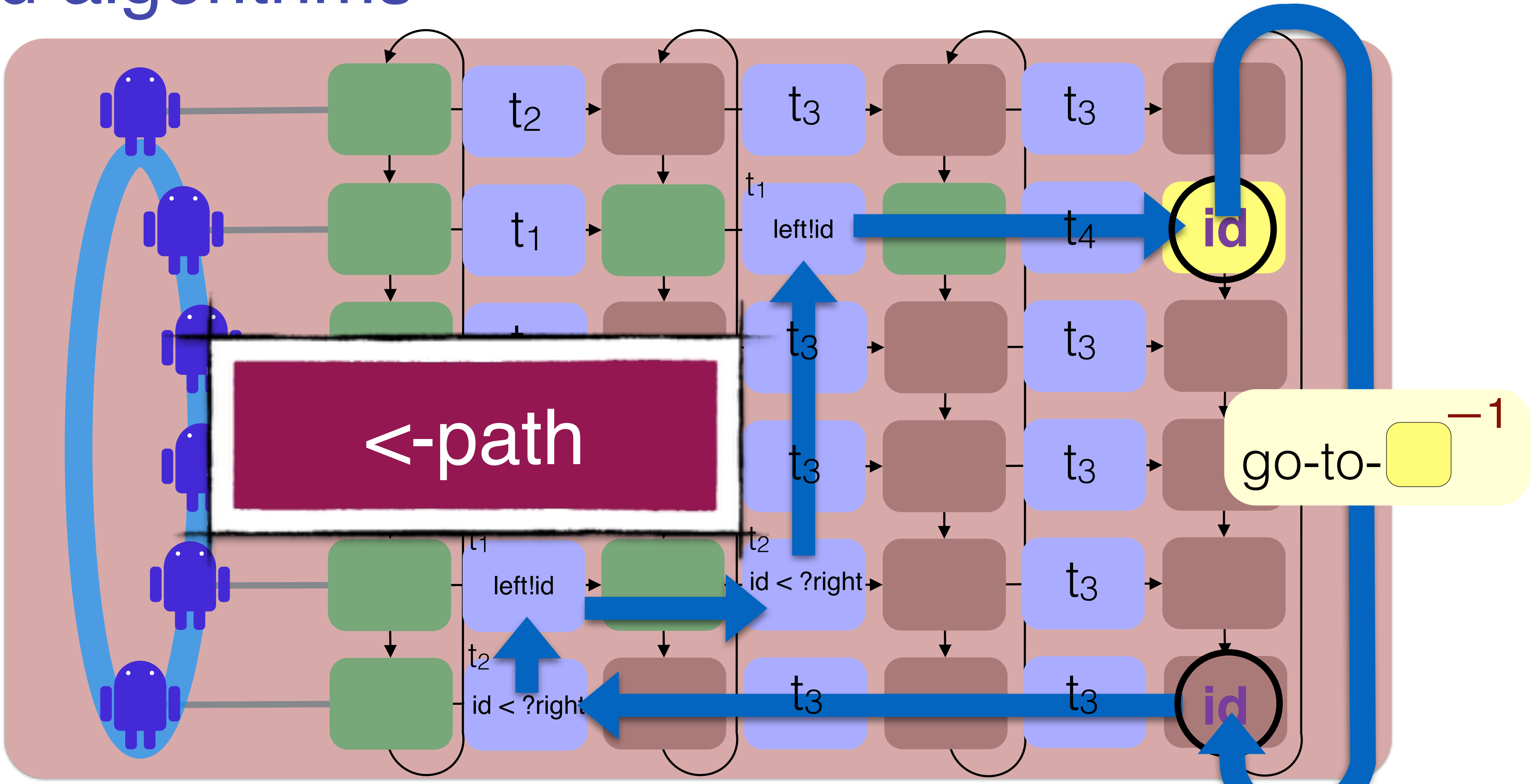
For all *n*, pid distributions, **accepting runs**, and processes:

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

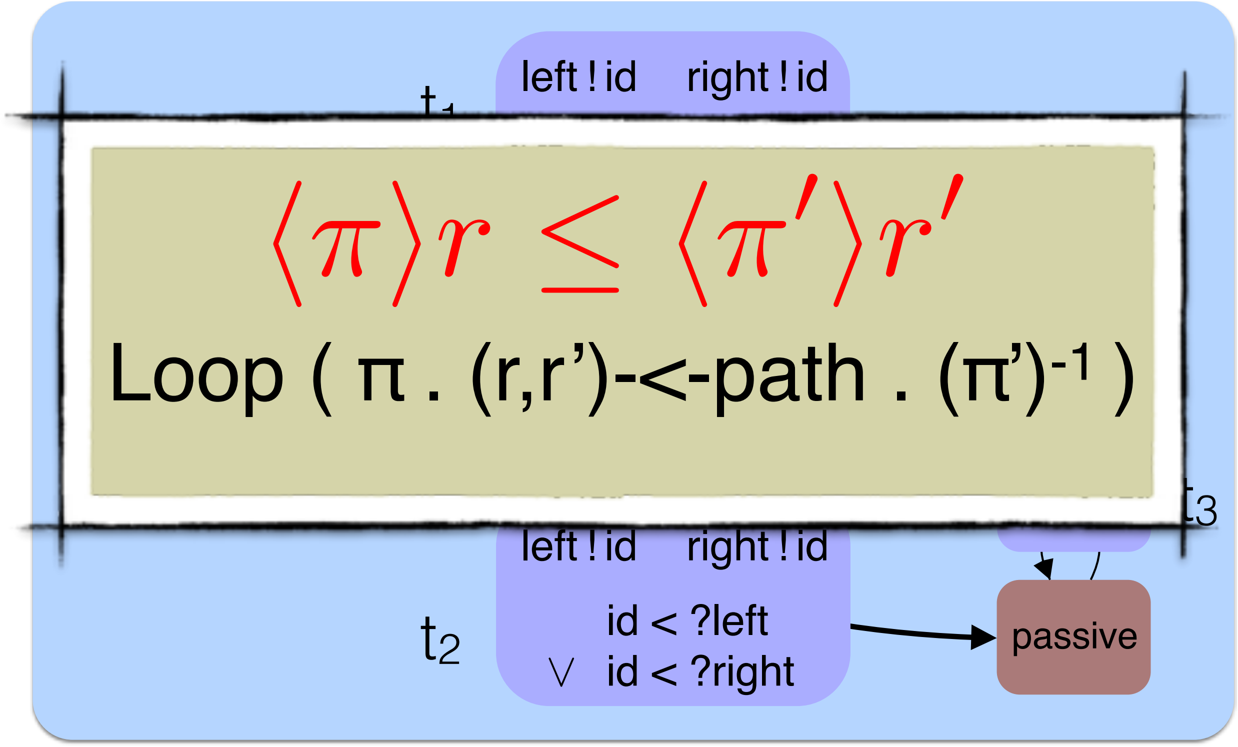
$$\text{go-to-}\square \;=\; (\neg \square \downarrow)^* \,\square$$

Data PDL

Distributed algorithms



Behavior



$$\langle\pi\rangle r \leq \langle\pi'\rangle r'$$

Loop (π . (r,r’)-<-path . $(\pi')^{-1}$)

Distributed algorithm

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

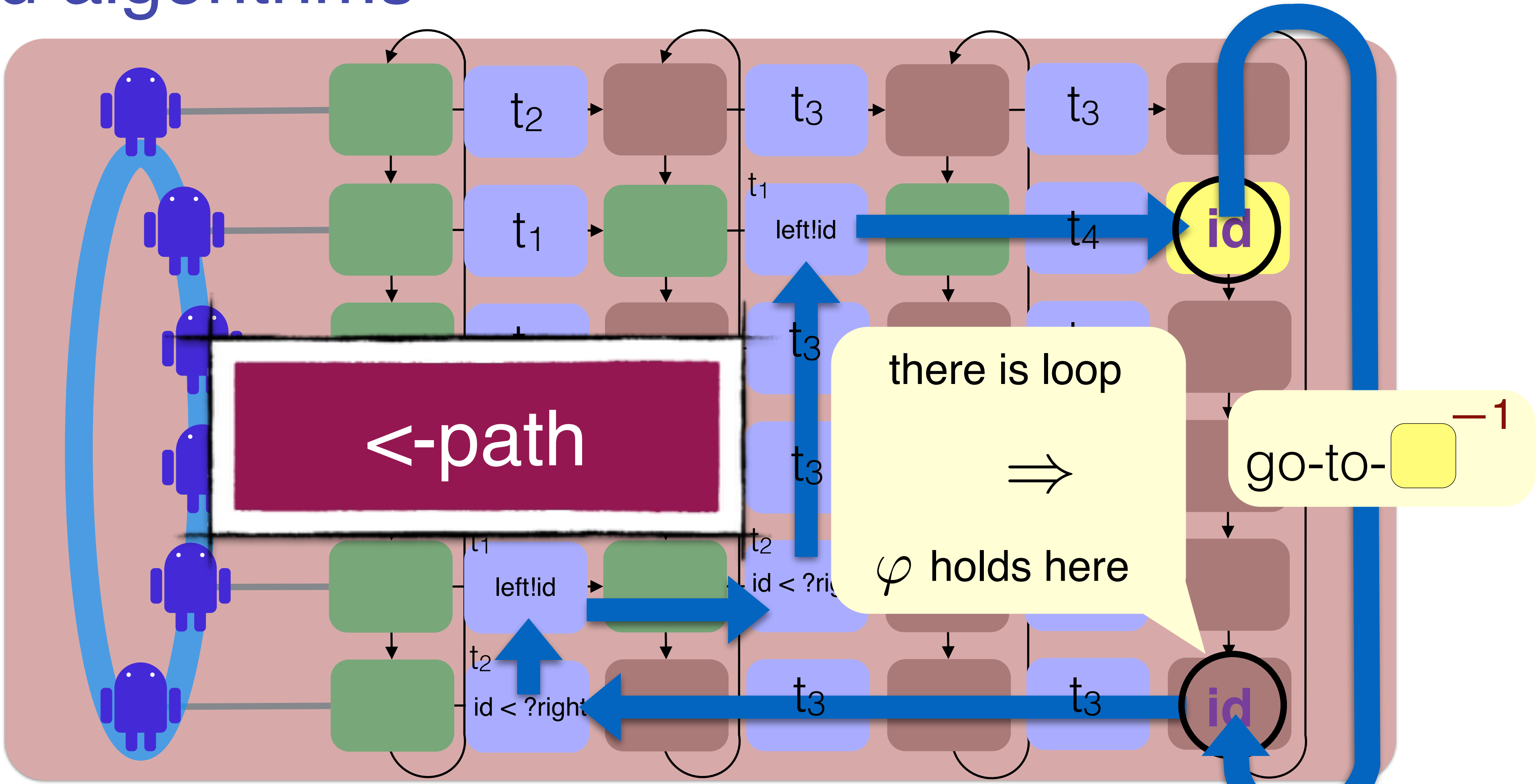
For all *n*, pid distributions, **accepting runs**, and processes:

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

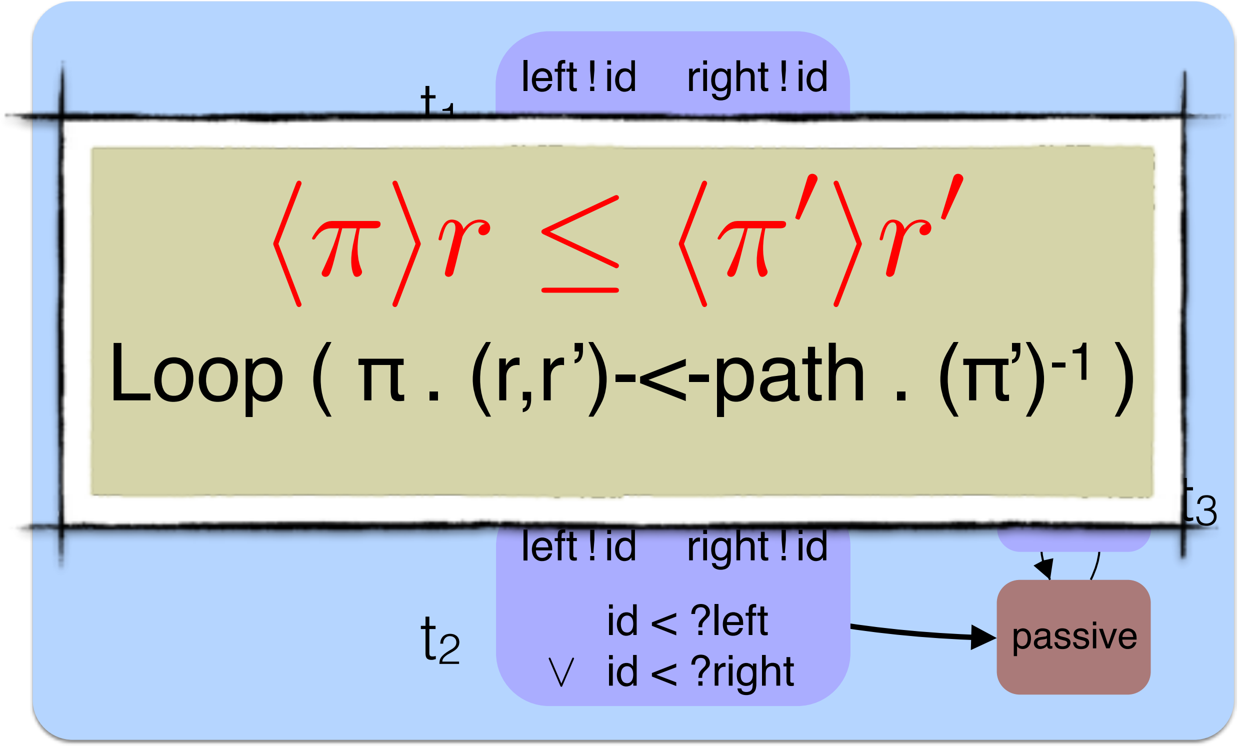
$$\text{go-to-}\square \;=\; (\neg\,\square \downarrow)^* \,\square$$

Data PDL

Distributed algorithms



Behavior



$$\langle\pi\rangle r \leq \langle\pi'\rangle r'$$

Loop (π . (r,r’)-<-path . (π’)$^{-1}$)

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:

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

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

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

Data PDL

Distributed algorithms

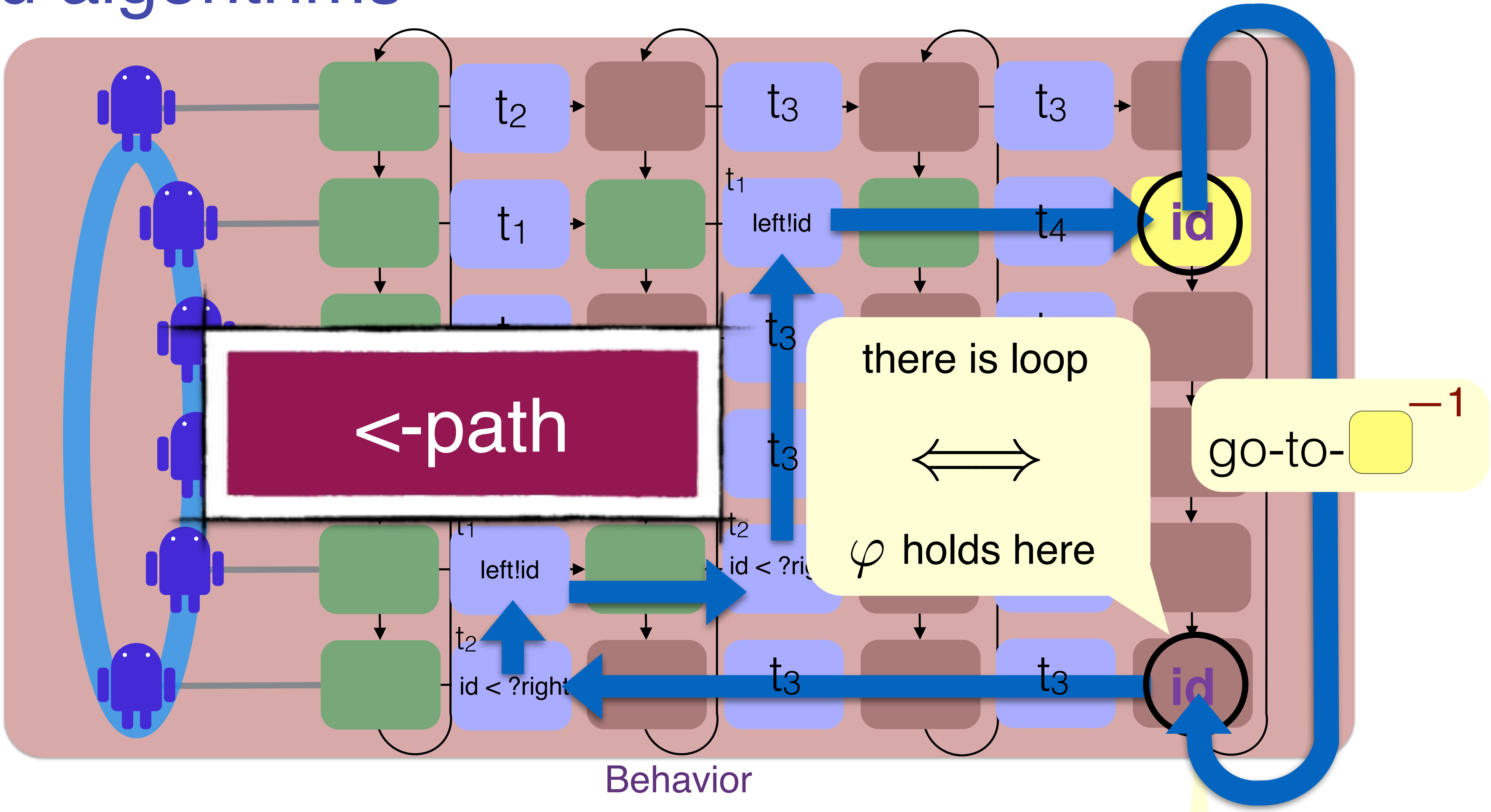
no loop

$\Rightarrow$

no evidence of φ

$\Rightarrow$

there are pids making φ false



<-path

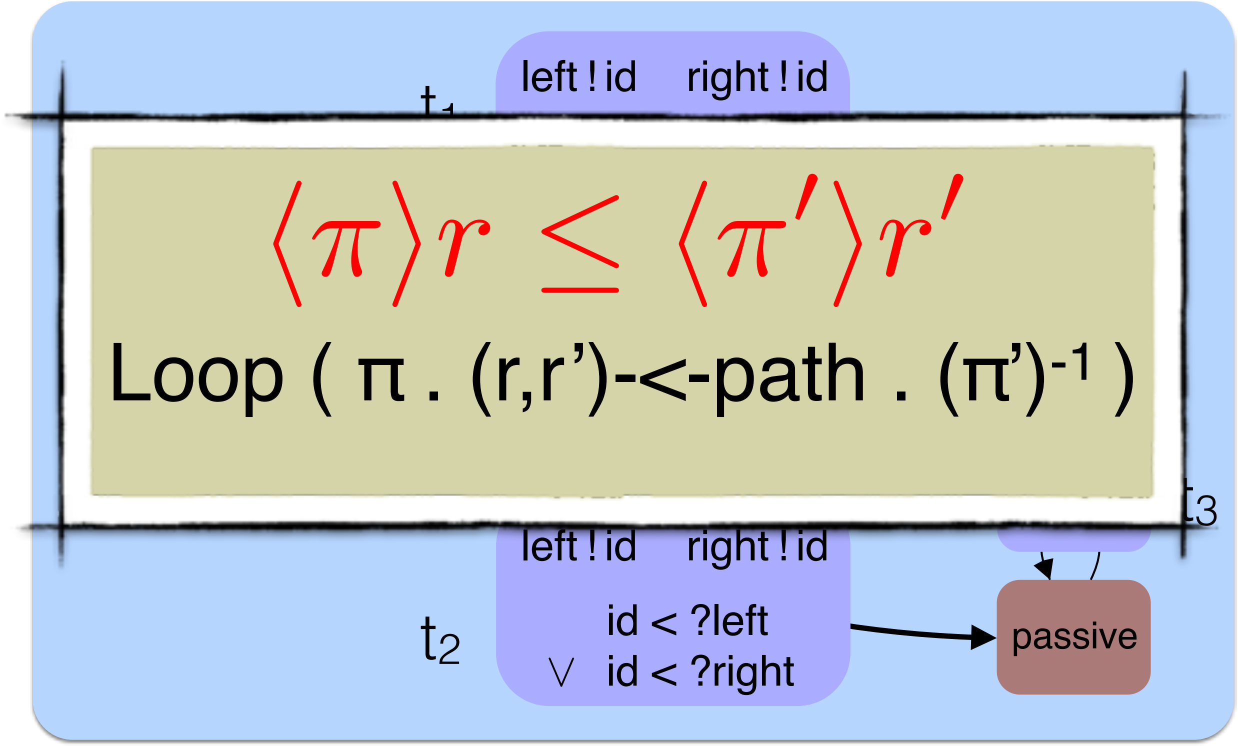
there is loop

$\Longleftrightarrow$

φ holds here

go-to-□ $^{-1}$

Behavior



$$\langle \pi \rangle r \leq \langle \pi' \rangle r'$$

Loop (π . (r,r')-<-path . (π')$^{-1}$)

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:

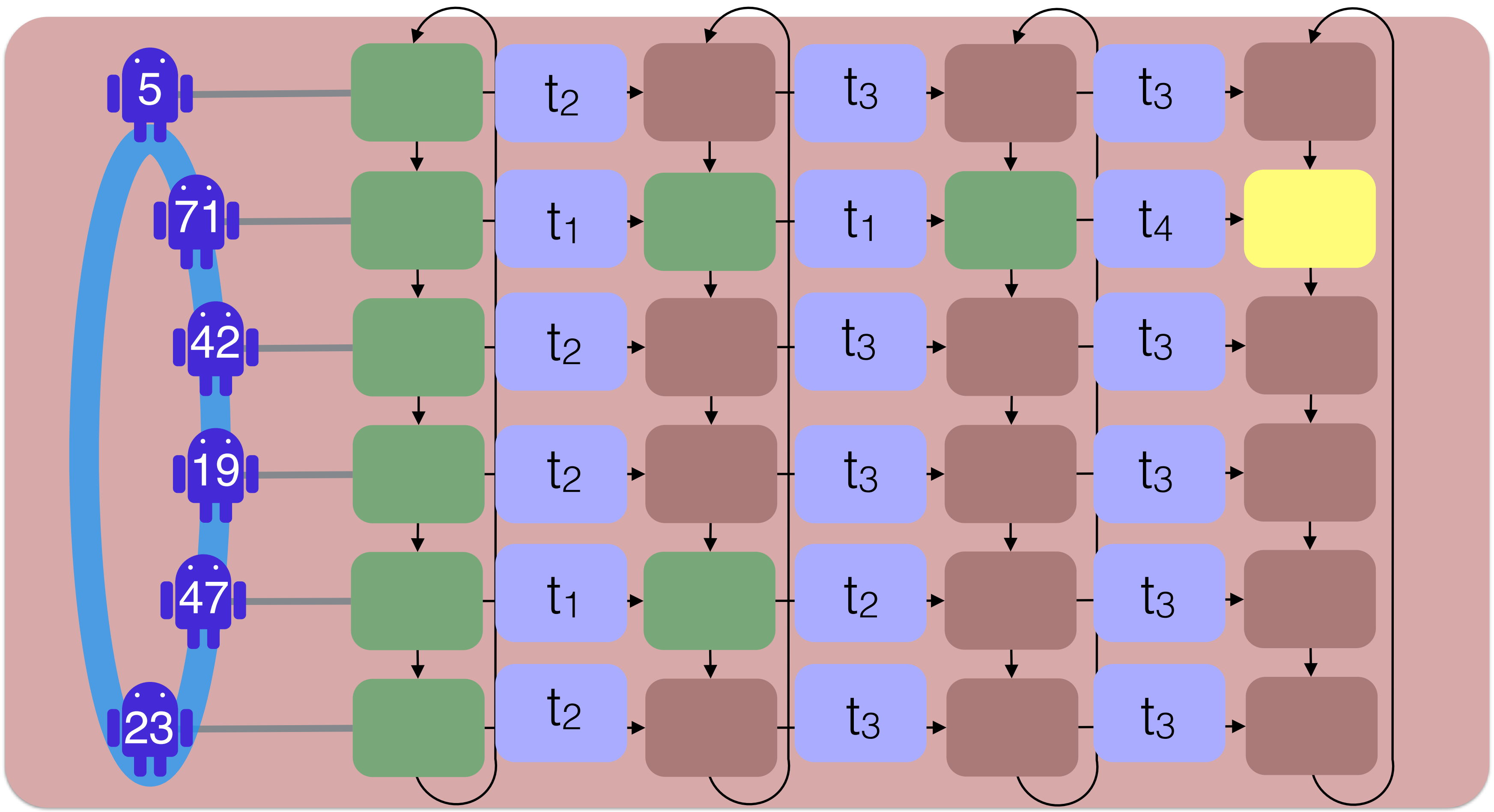
$$\langle \rightarrow^* \rangle \; (\quad \neg \langle \rightarrow \rangle \;\; \wedge \;\; \langle \text{go-to-}\square \rangle$$

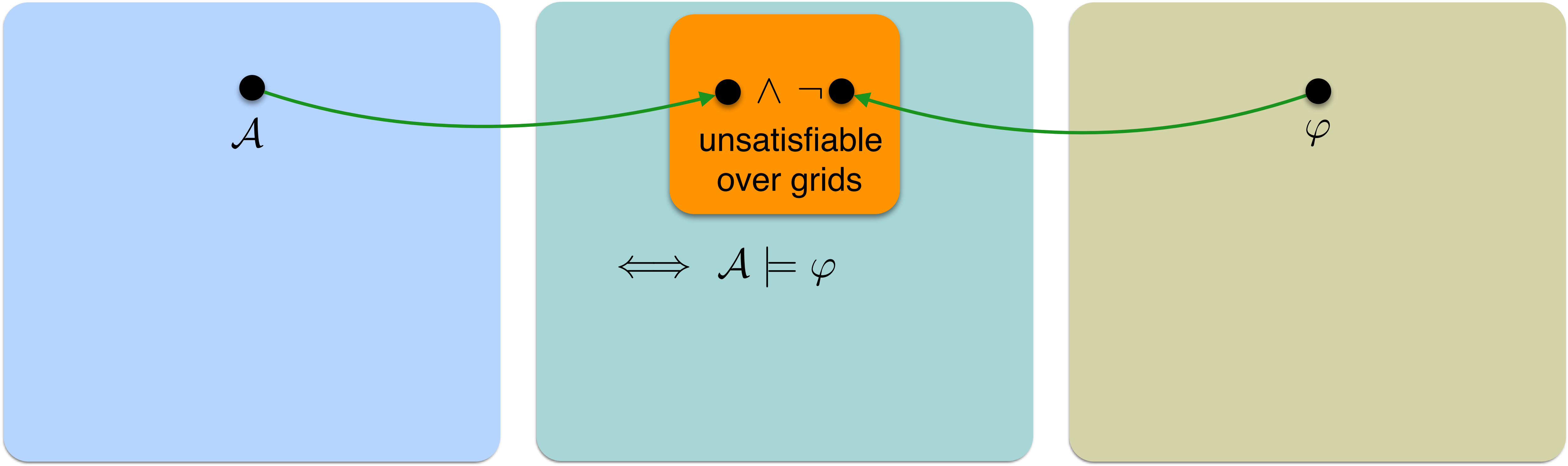
$$\wedge \; [\downarrow^*] \; \underbrace{(\mathbf{id} \leq \langle \text{go-to-}\square \rangle \, \mathbf{id})}_{\varphi})$$

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

Data PDL

Data abstraction





$\bullet \wedge \neg \bullet$

unsatisfiable over grids

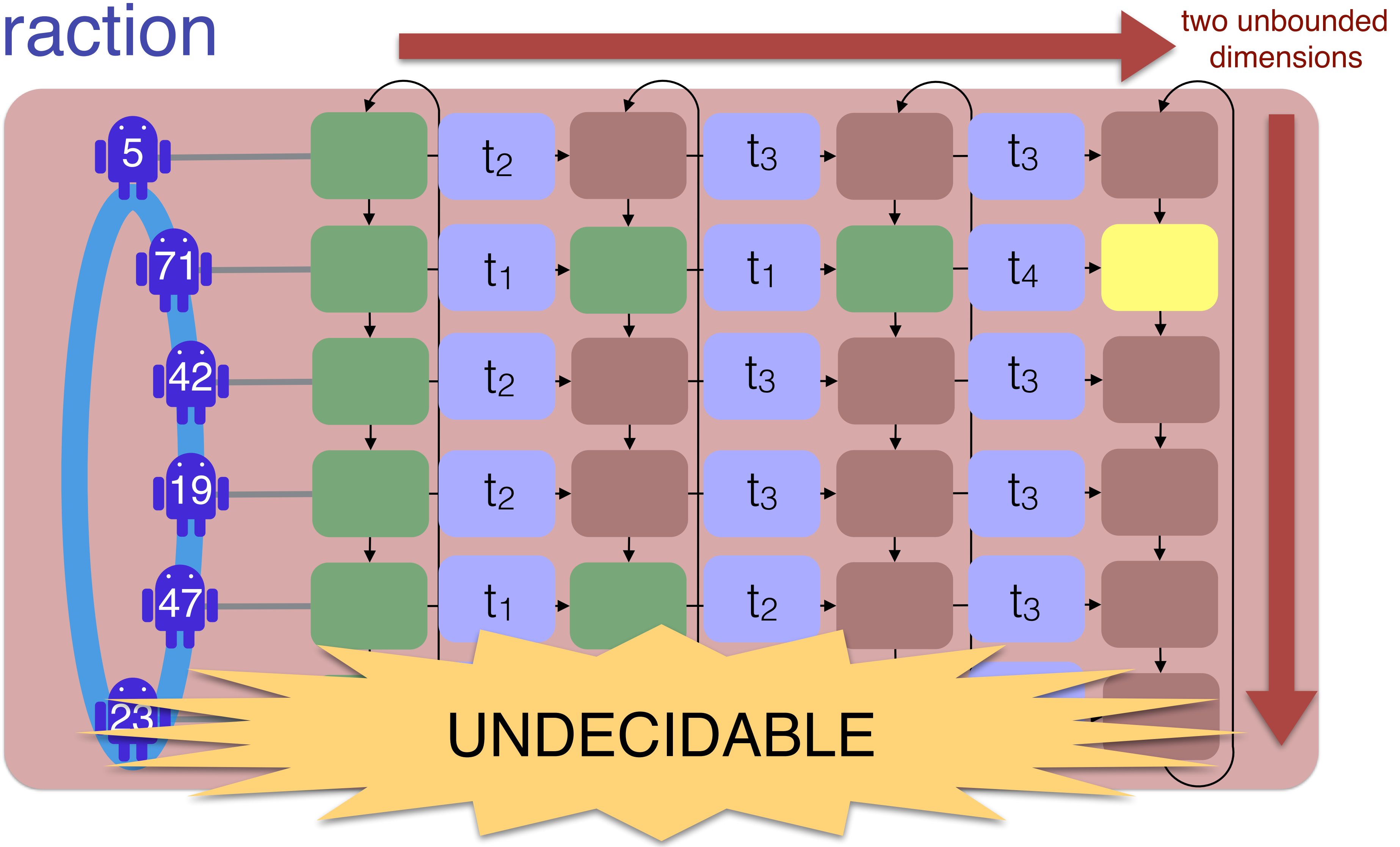
$\iff \mathcal{A} \models \varphi$

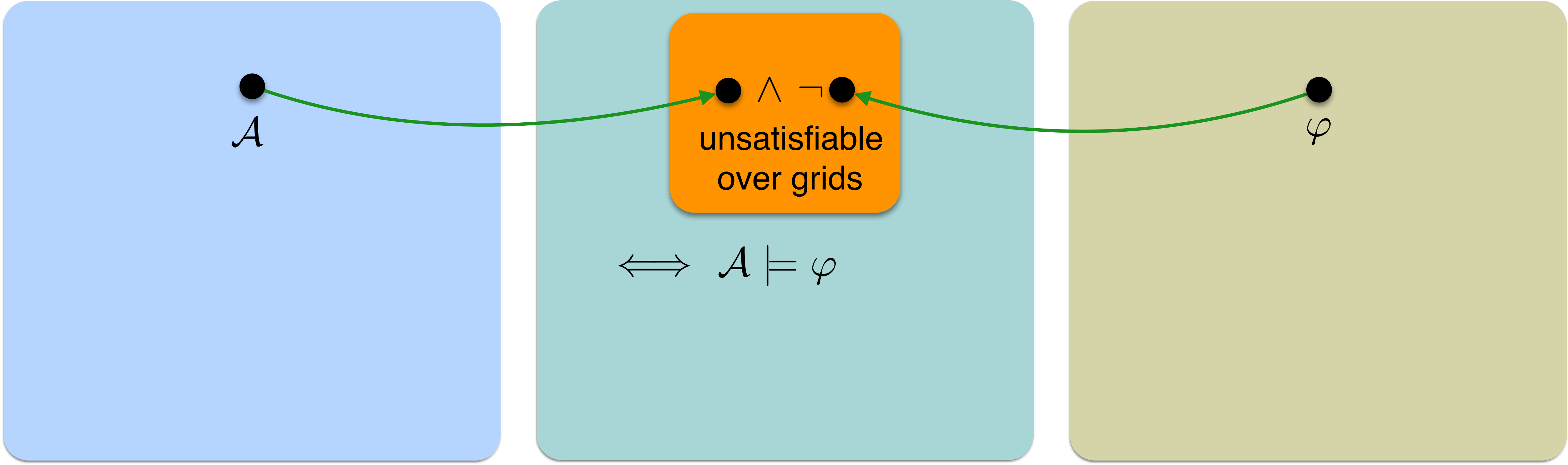
Distributed algorithm

PDL with loop (over finite alphabet)

Data PDL

Data abstraction





unsatisfiable over grids

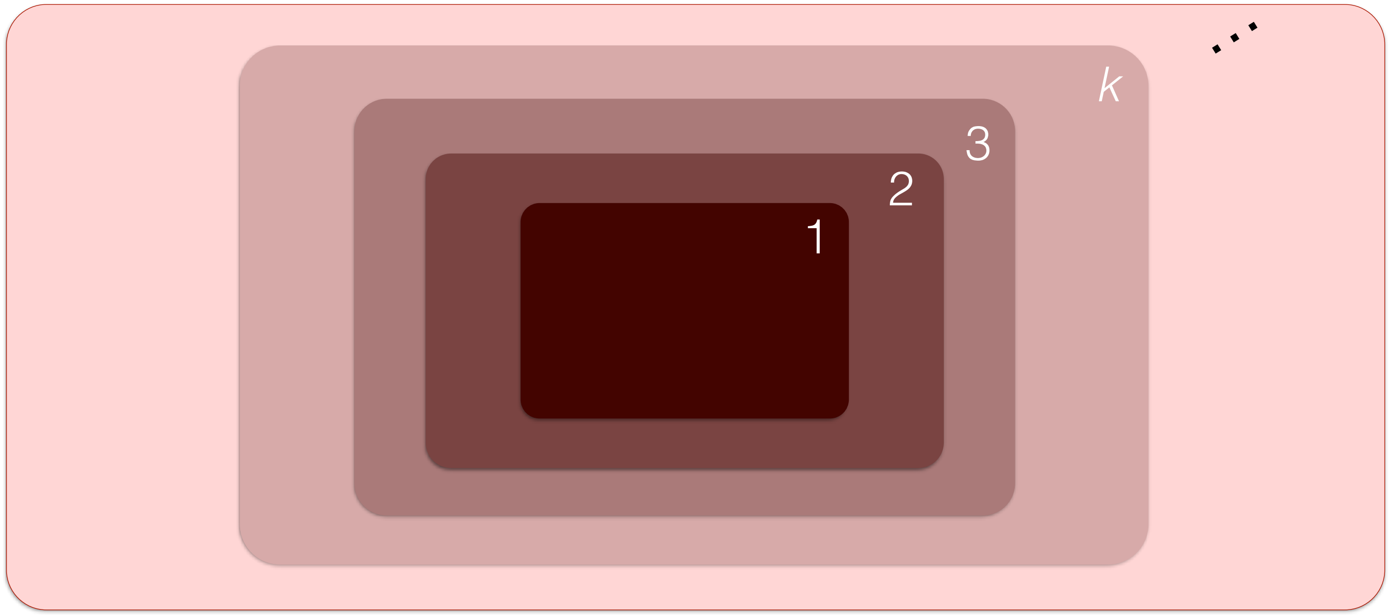
$\iff \mathcal{A} \models \varphi$

Distributed algorithm

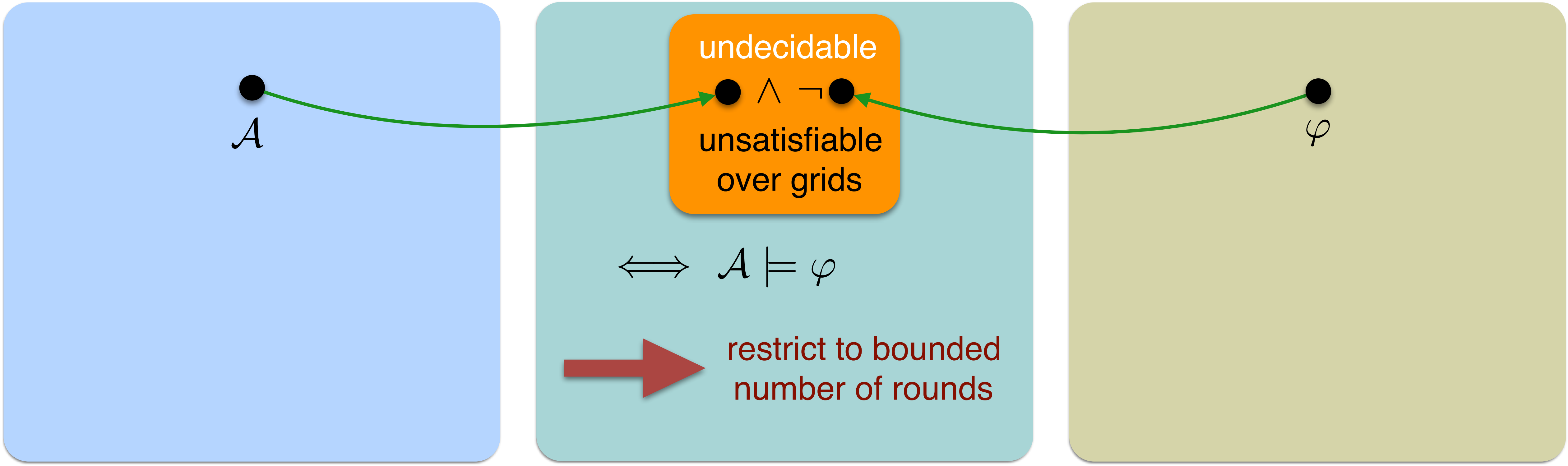
PDL with loop (over finite alphabet)

Data PDL

Under approximate verification



Behavior



undecidable

● $\wedge$ $\neg$ ●

unsatisfiable over grids

$\iff \mathcal{A} \models \varphi$

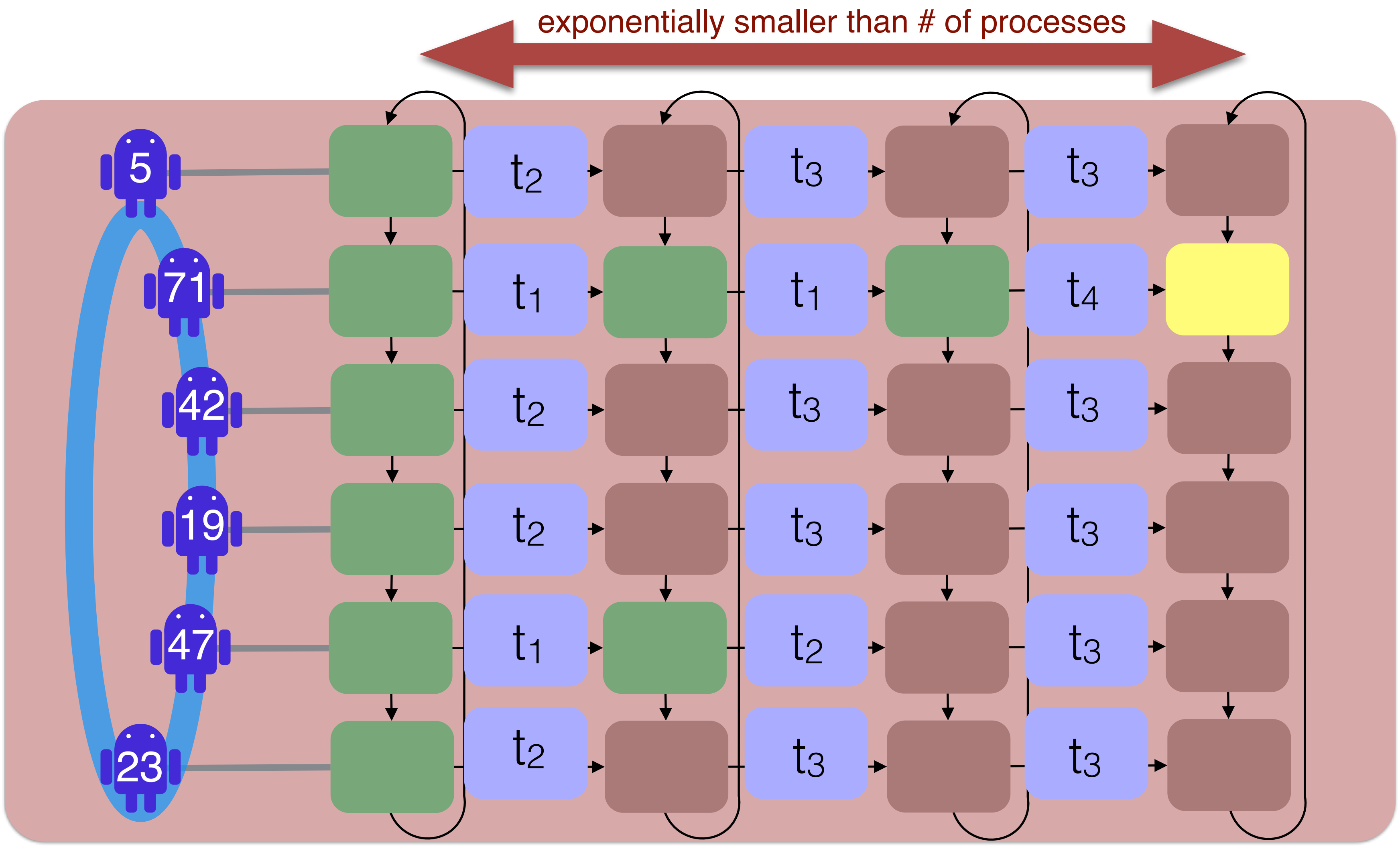
restrict to bounded number of rounds

Distributed algorithm

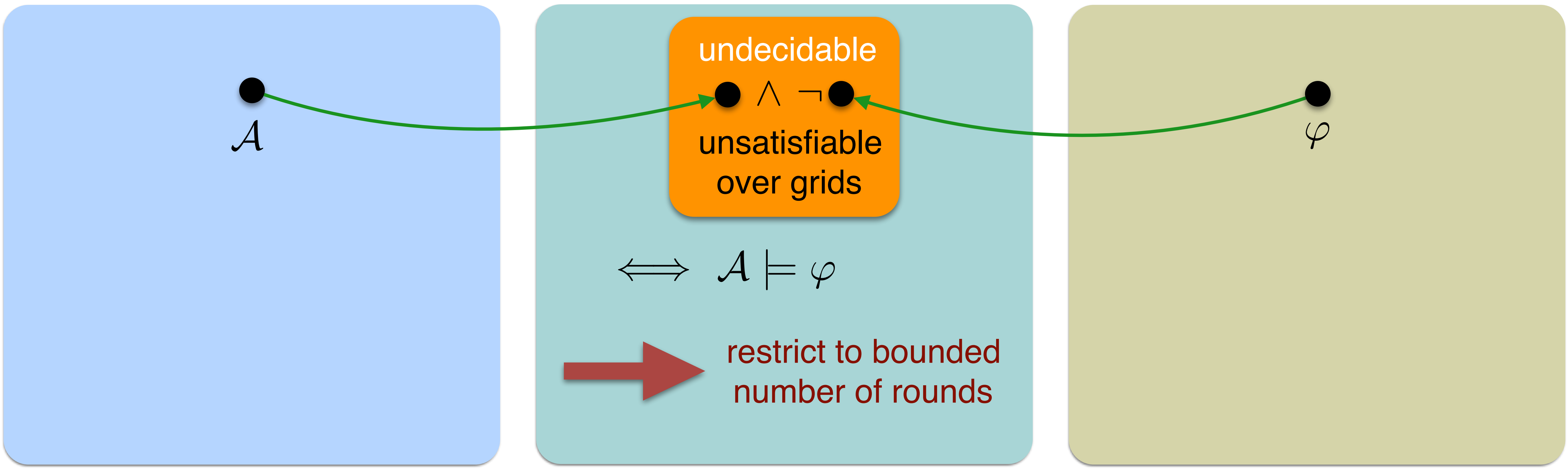
PDL with loop (over finite alphabet)

Data PDL

exponentially smaller than # of processes



5	🟩	t_2	🟫	t_3	🟫	t_3	🟫
71	🟩	t_1	🟩	t_1	🟩	t_4	🟨
42	🟩	t_2	🟫	t_3	🟫	t_3	🟫
19	🟩	t_2	🟫	t_3	🟫	t_3	🟫
47	🟩	t_1	🟩	t_2	🟫	t_3	🟫
23	🟩	t_2	🟫	t_3	🟫	t_3	🟫



undecidable

$\bullet \land \neg \bullet$

unsatisfiable over grids

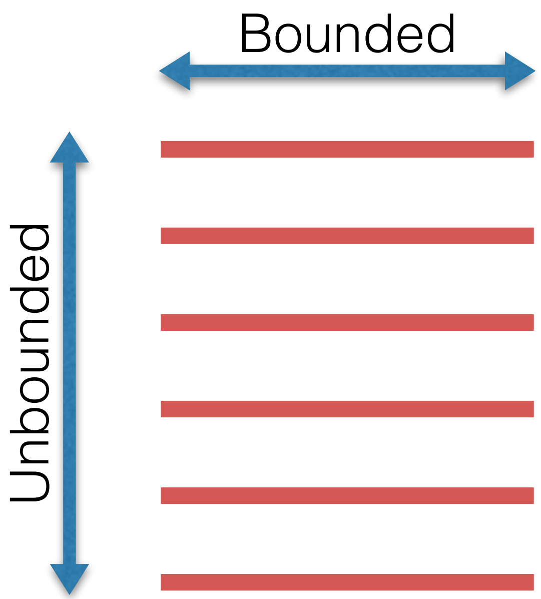
$\iff \mathcal{A} \models \varphi$

restrict to bounded number of rounds

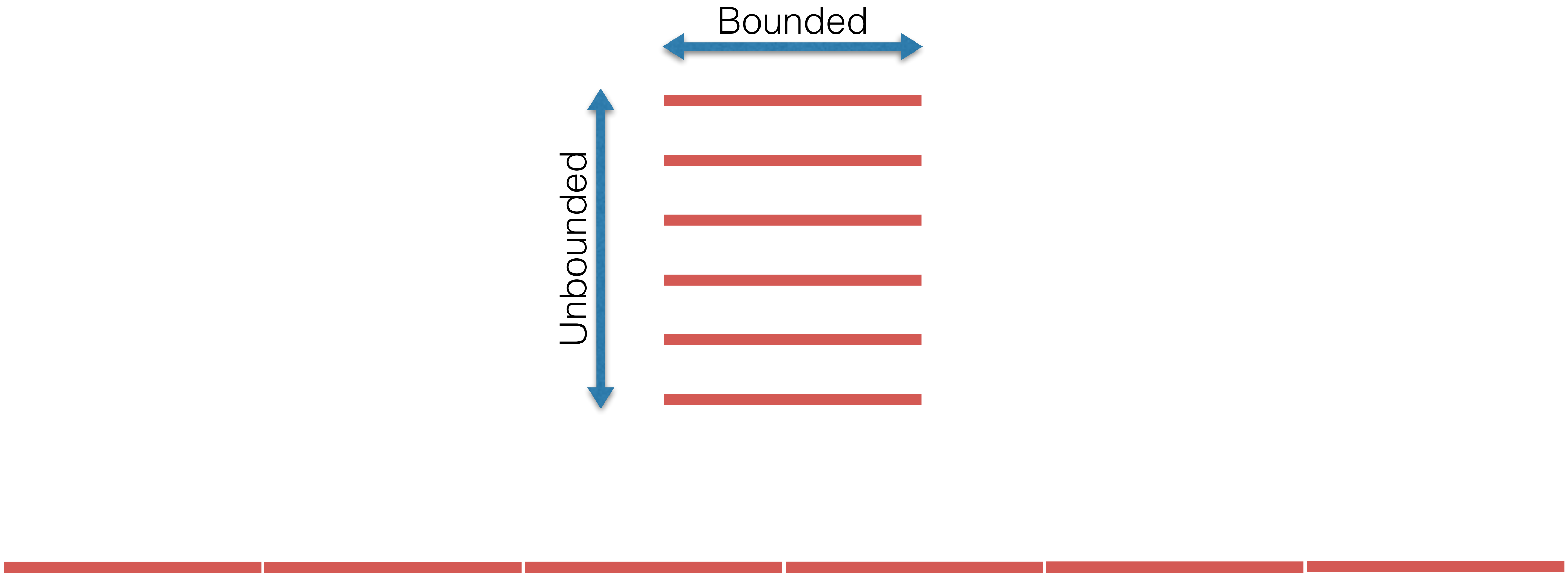
Distributed algorithm

PDL with loop (over finite alphabet)

Data PDL



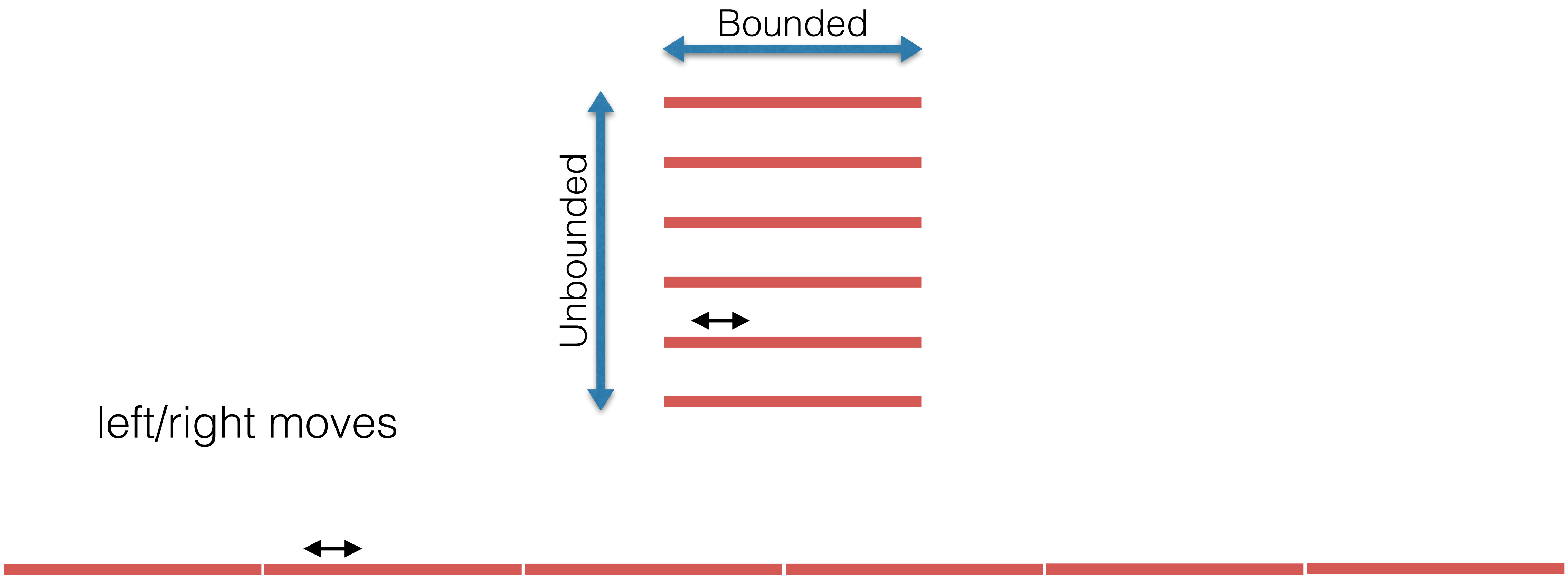
PDL with loop over bounded grids



PDL with loop over bounded grids

⇨

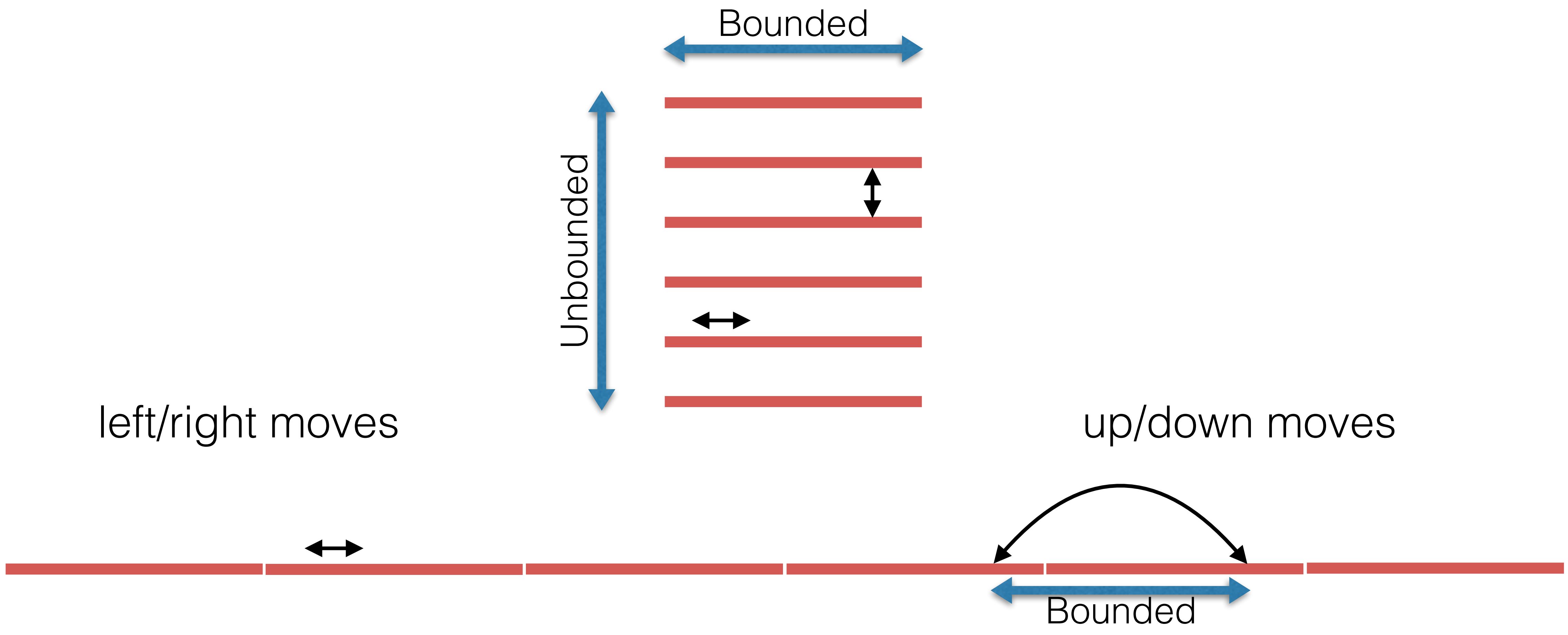
PDL with loop over words



PDL with loop over bounded grids

⇨

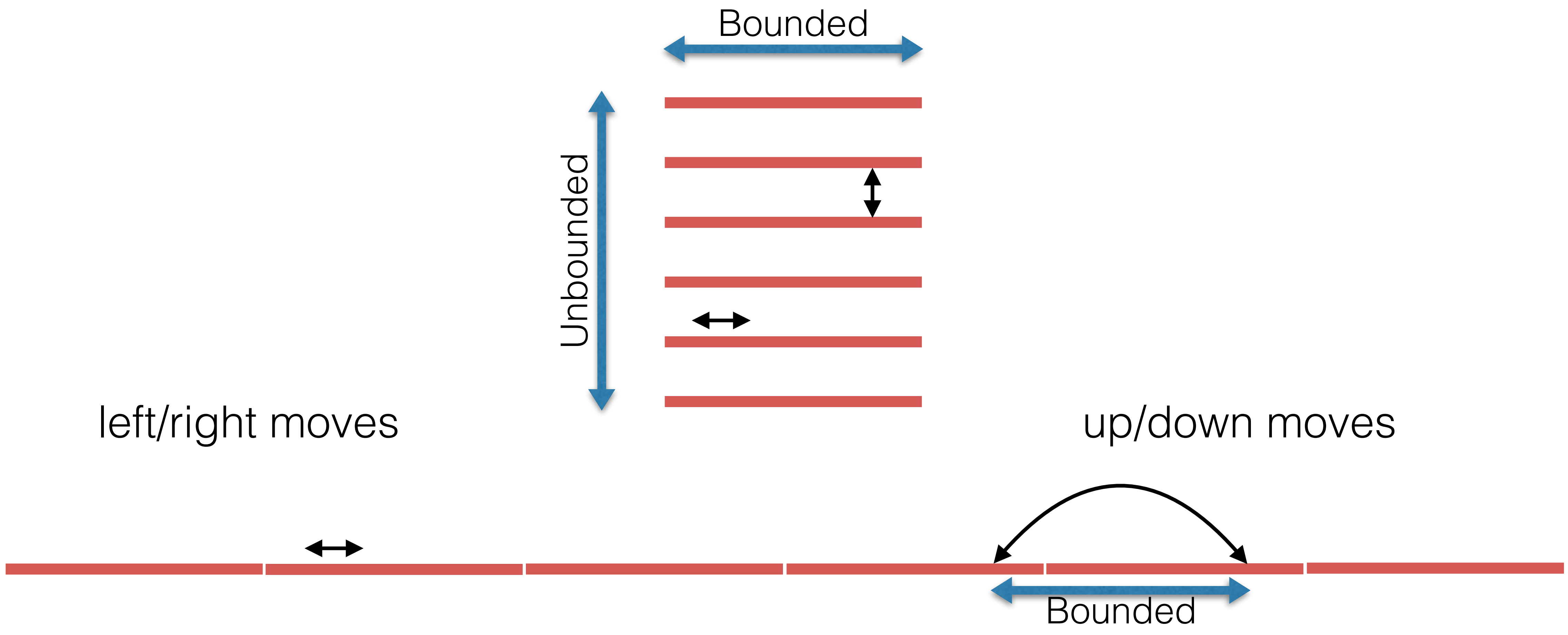
PDL with loop over words



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

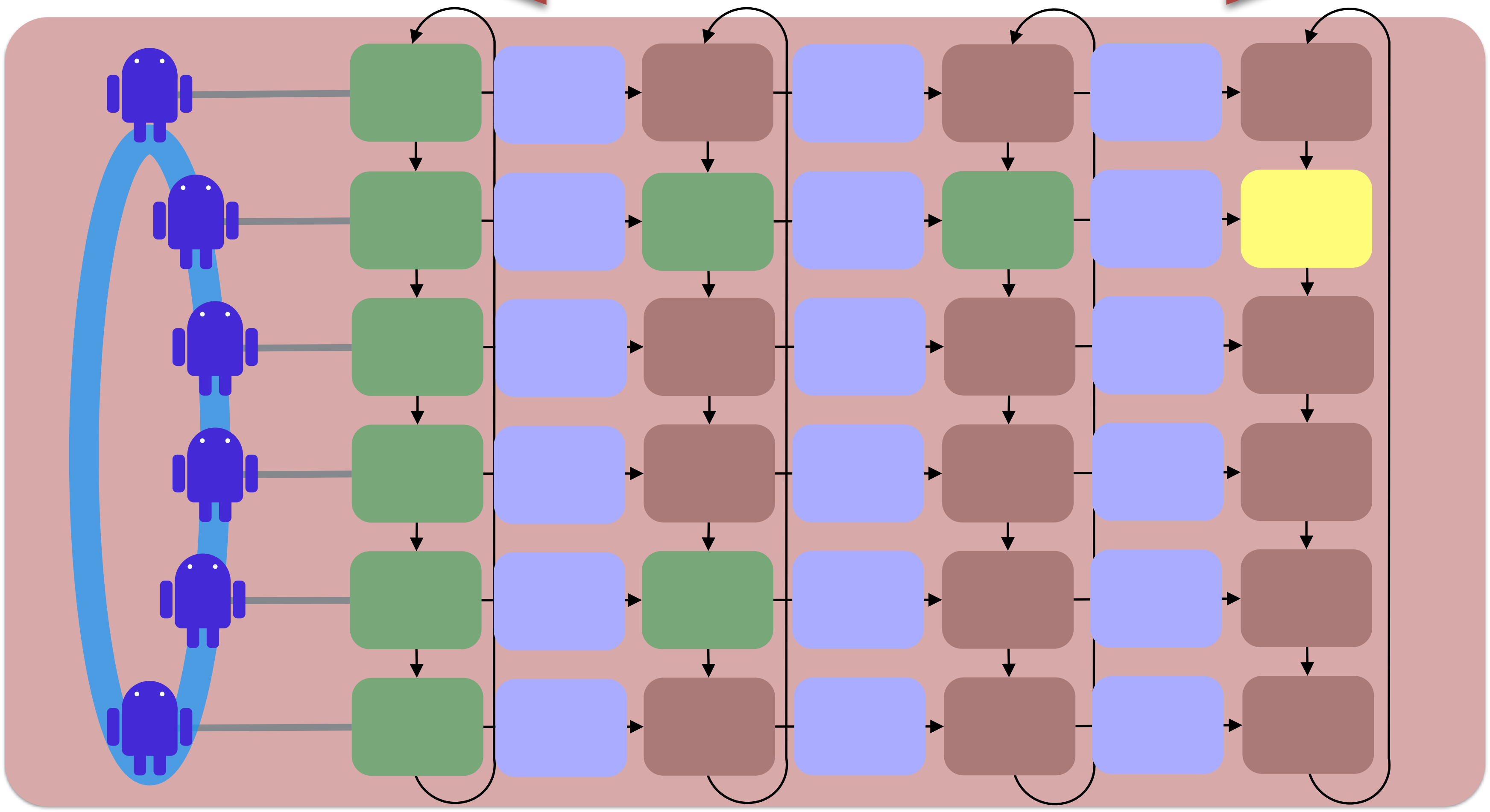
⇨

PSPACE

[Göller-Lohrey-Lutz ’08] [Serre ’08]

Distributed algorithms

exponentially smaller than # of processes



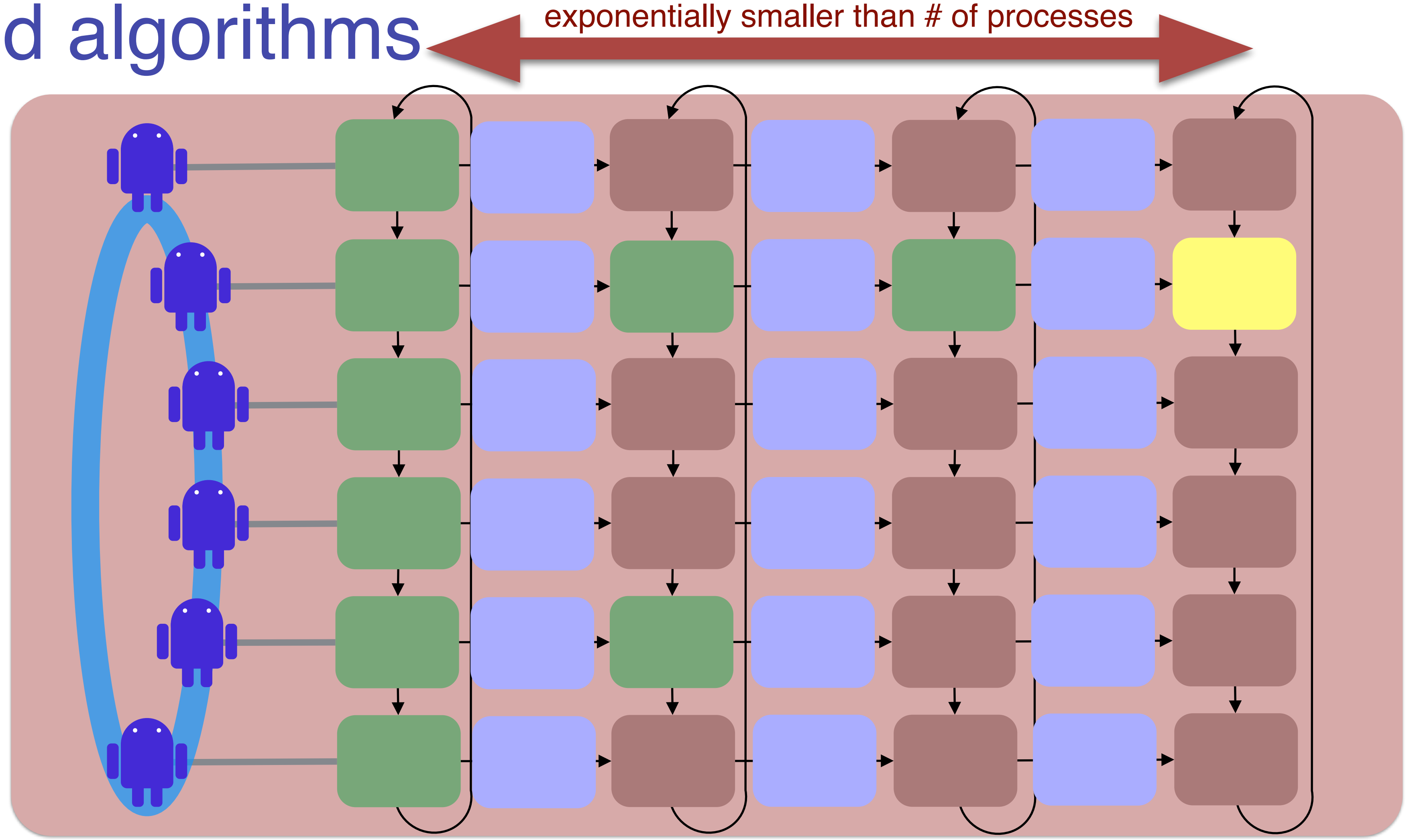
Theorem (Aiswarya-Bollig-Gastin; CONCUR ’15).

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

* with registers, register guards, and register updates

** unary encoding of # of rounds

Distributed algorithms



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

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

Summary

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.

Conclusions

Getting rid of Data

Translation of Distributed Algorithms and DataPDL to PDL with loops over finitely labelled cylinders

Independent of the restriction to rings

Independent of the number of rounds

Future work..

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

Thank you!