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

¬F

specification
Landscape & Objectives

System model

set of possible traces

Behavior

set of admissible traces

\[ L(A) \]

model checking

\\[ \models \]

\[ L(A) \subseteq L(\varphi) ? \]

Specification
LTL specification

Behavior

Finite automata

L(A) \cap L(A') \equiv \emptyset

LTL specification

\neg \varphi \quad \varphi
Models of Distributed Systems
Landscape & Objectives

Topology
- tree, ring, star, …

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

System model

Specification

\[ \forall n \] for all \( n \)
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**

**Specification**
Landscape & Objectives

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

**Topology**
- tree, ring, star, ...

**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**

**Specification**

\[ \emptyset \]
Landscape & Objectives

System model

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

Several sources of infinity / unboundedness
Landscape & Objectives

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

Topology
- tree, ring, star, ...

System model

Several sources of infinity / unboundedness
Landscape & Objectives

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

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

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

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

Topology
- tree, ring, star, …

Several sources of infinity / unboundedness
Landscape & Objectives

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

**Topology**
- tree, ring, star, ...

**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**

**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
Distributed algorithms

Leader election [Franklin ’82]
Distributed algorithms

Leader election [Franklin ’82]

Distributed algorithm

Behavior

5
7
42
19
47
23

active

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

passive

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

round
Distributed algorithms

Leader election [Franklin ’82]

Distributed algorithm

active

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

fwd

passive

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

Behavior
Distributed algorithms

Leader election [Franklin ’82]

Distributed algorithm

Behavior

left ! id  right ! id
id > ?left
\land id > ?right

active

left ! id  right ! id
id < ?left
\lor id < ?right

fwd

passive
Distributed algorithms

Leader election [Franklin '82]
Distributed algorithms

**Distributed algorithm**

- **active**
  - left ! id, right ! id
  - id > ?left ∧ id > ?right

- **leader**
  - left ! id, right ! id
  - id = ?left

- **passive**
  - left ! id, right ! id
  - id < ?left ∨ id < ?right

**Behavior**

**Leader election [Franklin ’82]**
Distributed algorithms

Leader election [Franklin ’82]

Distributed algorithm

Behavior
Distributed algorithms

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

found

initial state: active

registers: id, r, r₁, r₂

$t_1 = \langle active: \text{left!id ; right!id ; left?r₁ ; right?r₂ ; } r₁ < \text{id} ; r₂ < \text{id ; goto active} \rangle$

$t_2 = \langle active: \text{________________________ ; } id < r₁ ; \text{goto passive} \rangle$

$t_3 = \langle active: \text{________________________ ; } id < r₂ ; \text{goto passive} \rangle$

$t_4 = \langle active: \text{________________________ ; } id = r₁ ; r := \text{id ; goto found} \rangle$

$t_5 = \langle passive: \text{fwd ; left?r ; goto passive} \rangle$
Behaviors

Distributed algorithm

Cylinders
Arbitrary length and width
Labelled with data from an infinite domain
Specification language
Distributed algorithms

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:

$$\langle \rightarrow \ast \rangle (\neg \langle \rightarrow \rangle \land \langle \text{go-to-} \rangle \land [\downarrow \ast] (id \leq \langle \text{go-to-} \rangle id))$$
Distributed algorithms

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:

\[
\langle \leftarrow \star \rangle (\quad \neg \langle \leftarrow \rangle \quad \land \quad \langle \text{go-to-} \rangle \\
\land \quad [\downarrow \star] (\text{id} \leq \langle \text{go-to-} \rangle \text{id}))
\]
Distributed algorithms

«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:

\[
\langle \rightarrow^* \rangle ( \neg \langle \rightarrow \rangle \land \langle \text{go-to-} \rangle \land \left[ \downarrow^* \right] (id \leq \langle \text{go-to-} \rangle id))
\]
Distributed algorithms

Behavior

Distributed algorithm

Leader election [Franklin ’82]

«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:

\[
\langle \rightarrow \ast \rangle ( 
\langle \rightarrow \ast \rangle \land \langle \text{go-to-} \rangle \\
\land [\downarrow ^*] (\text{id} \leq \langle \text{go-to-} \rangle \text{id})
\]

\[
\text{go-to-} = (\neg \downarrow ^*)^* 
\]
Distributed algorithms

Distributed algorithm

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

Behavior

Leader election [Franklin ’82]

«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:

$$\langle \leftarrow \rangle^* \left( \neg \langle \rightarrow \rangle \land \langle \text{go-to-} \rangle \right)$$

$$\land \left[ \leftarrow \right] (id \leq \langle \text{go-to-} \rangle id)$$

go-to- = (\neg \downarrow)^*
Distributed algorithms

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:

\[
\langle \rightarrow^* \rangle ( \neg \langle \rightarrow \rangle \land \langle \text{go-to-} \rangle \land [\downarrow^*] (id \leq \langle \text{go-to-} \rangle \text{id}))
\]

go-to- = (\neg \downarrow)^*
Specifications
Data PDL

\[ \varphi, \varphi' ::= m \mid s \mid \neg \varphi \mid \varphi \land \varphi' \mid \varphi \Rightarrow \varphi' \mid [\pi] \varphi \mid \langle \pi \rangle r \times \langle \pi' \rangle r' \]

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

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

Specifications

Data PDL

\[ \varphi, \varphi' ::= m \mid s \mid \neg \varphi \mid \varphi \land \varphi' \mid \varphi \Rightarrow \varphi' \mid [\pi]\varphi \mid \langle \pi \rangle r \Join \langle \pi' \rangle r' \]

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

\[ s \in S, r, r' \in Reg, \Join \in \{\,=, \neq, <, \leq\}, \text{ and } d \in \{\epsilon, \leftarrow, \rightarrow, \uparrow, \downarrow\}. \]
Specifications
Data PDL

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

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

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

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


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

UNDECIDABLE
Reduction to automata?

We do not know how to translate Data PDL to automata.
Data abstraction

A

\( A = \neg \wedge \wedge \neg \)

unsatisfiable over grids

\( \iff A \models \varphi \)

Distributed algorithm

PDL with loop (over finite alphabet)

Data PDL
Data abstraction: symbolic runs + tracking data

Distributed algorithm

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
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
Data abstraction: symbolic runs + tracking data

- Register updates

Distributed algorithm
Data abstraction: symbolic runs + tracking data

- Register updates
Data abstraction: symbolic runs + tracking data

- Register updates
Data abstraction: symbolic runs + tracking data

- Register updates
Data abstraction: symbolic runs + tracking data

- Register updates
Data abstraction: symbolic runs + tracking data

- Register updates

(r₁,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

π₁:(r₁,id)-path
π₂:(r₂,id)-path
Data abstraction: symbolic runs + tracking data

- Register updates
- Register equality check

\[ \pi_1 : (r_1, \text{id})\text{-path} \]
\[ \pi_2 : (r_2, \text{id})\text{-path} \]
\[ \text{loop}( \pi_1 \pi_2^{-1} ) \]

can be expressed in PDL with loop
Data abstraction: symbolic runs + tracking data

- Register updates
- Register equality check
- Register comparison
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
\[ r_{i0} < r_{i1}; (r_{i1},r_{i2})\text{-path}; r_{i2} < r_{i3}; (r_{i3},r_{i4})\text{-path}; \ldots ; r_{in} < r_{i0} \]

- 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
\[ r_{i_0} < r_{i_1}; (r_{i_1}, r_{i_2})\text{-path}; r_{i_2} < r_{i_3}; (r_{i_3}, r_{i_4})\text{-path}; \ldots ; r_{i_n} < r_{i_0} \]

- Register updates
- Register equality check
- Register comparison

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

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

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

$$\langle \rightarrow^* \rangle \left( \neg \langle \rightarrow \rangle \land \langle \text{go-to-} \rangle \right) \land \left[ \downarrow^* \right] \left( \text{id} \leq \langle \text{go-to-} \rangle \text{id} \right)$$

$$\Rightarrow \text{go-to-} = \left( \neg \downarrow \right)^* \text{id}$$
Distributed algorithms

Behavior

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

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

\[
\langle \rightarrow^* \rangle \left( \neg \langle \rightarrow \rangle \land \langle \text{go-to-}id \rangle \land [\downarrow^*] (\text{id} \leq \langle \text{go-to-}id \rangle) \right) \\
\land \left( \neg \langle \rightarrow \rangle \Downarrow^* \right) \varphi
\]

go-to- = (\neg \Downarrow)^*
Distributed algorithms

Distributed algorithm

Behavior

Data PDL

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

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

\[
\langle \rightarrow^* \rangle \ ( \neg \langle \rightarrow \rangle \land \langle \text{go-to-} \rangle \land [\downarrow^*] (\text{id} \leq \langle \text{go-to-} \rangle \text{id})
\]

\[
\text{go-to-} = (\neg \downarrow)*\]
Distributed algorithms

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

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

$$\langle \to^* \rangle \left( \neg \langle \to \rangle \land \langle \text{go-to-} \rangle \right)$$

$$\land \left[ \downarrow^* \right] \left( \text{id} \leq \langle \text{go-to-} \rangle \text{id} \right)$$

$$\varphi$$

go-to- = (\neg \downarrow)^*
Distributed algorithms

Distributed algorithm

Behavior

Data PDL

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

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

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

Loop ($\pi . (r,r')$)-<-path . ($\pi'$)-1)

$\oplus$

left ! id right ! id

$\rightarrow^*$

$\neg \langle \rightarrow \rangle \land \langle \text{go-to-} \rangle$

$\land \left[ \downarrow^* \right] \left( \text{id} \leq \langle \text{go-to-} \rangle \text{id} \right)$

$\varphi$

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

Distributed algorithm

Data PDL

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

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

\[
\langle \pi \rangle r \leq \langle \pi' \rangle r'
\]

Loop \( \pi \cdot (r,r')\langle \text{-path} \cdot (\pi')^{-1} \)
Distributed algorithms

Behavior

no loop

no evidence of \( \varphi \)

there are pids making \( \varphi \) false

\(-\text{path}\)

there is loop

\( \varphi \) holds here

\( \text{go-to-} \)

Go back

\( \neg \)

Loop

\( \pi . (r,r') \leftarrow \text{-path} . (\pi')^{-1} \)

For all \( n \), pid distributions, accepting runs, and processes:

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

\( \langle \text{go-to-} \rangle = (\neg \text{id} \downarrow)^* \text{id} \)
Data abstraction

Distributed algorithm

PDL with loop (over finite alphabet)

Data PDL
Data abstraction

Distributed algorithm

PDL with loop (over finite alphabet)

Data PDL


UNDECIDABLE

unsatisfiable
over grids

\[ A \models \varphi \]
Under approximate verification

Distributed algorithm

Data PDL

A

PDL with loop (over finite alphabet)

unsatisfiable over grids

undecidable

restrict to bounded number of rounds

\[ \langle A \rangle^{\forall} \wedge \neg \langle A \rangle^{\exists} \]

\[ \equiv \ A \models \varphi \]
Distributed algorithm

Data PDL

A

PDL with loop (over finite alphabet)

unsatisfiable over grids

exponentially smaller than # of processes

rewritable

undecidable

restrict to bounded number of rounds

\[ \mathcal{A} \models \varphi \]
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

\[\Rightarrow\]

PDL with loop over words

\[\Rightarrow\]

Alternating 2-way Automata

\[\Rightarrow\]

PSPACE

[Göller-Lohrey-Lutz '08]  [Serre '08]
Distributed algorithms

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
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!