C. Aiswarya, Benedikt Bollig, Paul Gastin

Automata-theoretic Verification of Distributed Algorithms

Credits for the slides go to C. Aiswarya.

ALFA
16th June, 2015
Model checking

System $S$

Specification $\varphi$

$S \models \varphi$?

Refine $S$
(Fix bugs)
Model checking

- System - finite state automata modelling hardware / software

$S \models \varphi$?

Refine $S$ (Fix bugs)
Model checking

• System - finite state automata modelling hardware / software

• Specification - another finite state automata, LTL formula, first order logic formula…

\[ S \models \varphi? \]

Refine S
(Fix bugs)
Model checking

- System - finite state automata modelling hardware / software
- Specification - another finite state automata, LTL formula, first order logic formula…

Can we model check distributed algorithms?
Distributed algorithms
Distributed algorithms

- Identical finite-state processes
Distributed algorithms

- Identical finite-state processes
- On ring topologies
Distributed algorithms

- Identical finite-state processes
- On ring topologies
- Left neighbour and a right neighbour
Distributed algorithms

- Identical finite-state processes
- On ring topologies
- Left neighbour and a right neighbour
- Number of processes is unknown and unbounded
Distributed algorithms

- Identical finite-state processes
- On ring topologies
- Left neighbour and a right neighbour
- Number of processes is unknown and unbounded
- Processes have unique pids (integers — unbounded data)
Leader Election Algorithms

Franklin82

- The max-pid process is elected as the leader
- Proceeds in round
- Computes the local maxima among the active neighbours
- Becomes passive if not a local maxima
- Passive processes enter a “fwd” state to enable communication between the active processes
Leader Election Algorithms

- The max-pid process is elected as the leader
- Proceeds in round
- Computes the local maxima among the active neighbours
- Becomes passive if not a local maxima
- Passive processes enter a "fwd" state to enable communication between the active processes
Leader Election Algorithms

**Franklin82**

- The max-pid process is elected as the leader
- Proceeds in round
- Computes the local maxima among the active neighbours
- Becomes passive if not a local maxima
- Passive processes enter a “fwd” state to enable communication between the active processes
Model checking

- System - finite state automata serving as abstractions of hardware / software
- Specification - another finite state automata, LTL formula, first order logic formula…

Can we model check distributed algorithms?
Model checking

Can we model check distributed algorithms?

- Systems must model unbounded number of processes
- where processes handle unbounded data
- and communicate
- Specifications need to reason about pids, processes, left and right neighbours, compare pids...

Refine $S$ (Fix bugs)
Can we model check distributed algorithms?

two sources of infinity

Unknown number of processes

Systems handling unbounded data

I. Konnov, H. Veith, and J. Widder. Who is afraid of model checking distributed algorithms?, 2012.
Can we model check distributed algorithms?

Two sources of infinity:

- Unknown number of processes
- Systems handling unbounded data

- Unknown number of processes handling data
- Finite state systems
Unknown number of processes handling data

Unknown number of processes

Systems handling unbounded data

Finite state systems

Classical model checking
Unknown number of processes handling data

Systems handling unbounded data

Finite state systems

Classical model checking

Parametrised verification

Unknown number of processes
Unknown number of processes

Parametrised verification

Unknown number of processes handling data

Systems handling unbounded data

Finite state

Classical model checking


Finite state systems

Unknown number of processes

Unknown number of processes handling data

Parametrised verification

Systems handling unbounded data

Data (automata/logics)

Classical model checking

---


Finite state systems
Unknown number of processes
Parametrised verification
Unknown number of processes handling data
Data (automata/logics)
Systems handling unbounded data

Finite state systems

Unknown number of processes

Systems handling unbounded data

Data (automata/logics)

Parametrised verification

Unknown number of processes

??


A formal model for distributed algorithms

An automata-like way of writing DA

Every process can be described by:
A formal model for distributed algorithms
An automata-like way of writing DA

Every process can be described by:

• Set of states
A formal model for distributed algorithms

An automata-like way of writing DA

Every process can be described by:

- Set of states
- Initial state
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
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
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
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
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
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: active, passive
found
initial state: active
registers: id, r, r₁, r₂

\[ t₁ = \langle \text{active: left!id ; right!id ; left?r₁ ; right?r₂ ; r₁ < id ; r₂ < id ; goto active} \rangle \]
\[ t₂ = \langle \text{active: ______________ ; id < r₁ ; goto passive} \rangle \]
\[ t₃ = \langle \text{active: ______________ ; id < r₂ ; goto passive} \rangle \]
\[ t₄ = \langle \text{active: ______________ ; id = r₁ ; r := id ; goto found} \rangle \]
\[ t₅ = \langle \text{passive: fwd ; left?r ; goto passive} \rangle \]
Run

Franklin82

state: A, id = 5, r = 5, r1 = 5, r2 = 5
Run
Franklin82

5 71 42 19 47 23

A A A A A A

P A P P A P
Run

Franklin82

A A A A A A
P A P P A P
P A P P P P
P F P P P P
Run

Franklin82

state: P, id = 23, r = 47, r1 = 47, r2 = 5
Specifications (DataPDL)

\[ \Phi ::= \forall_{\text{rings}} \forall_{\text{runs}} \varphi \]
Specifications (DataPDL)

\[ \Phi ::= \forall_{rings} \forall_{runs} \varphi \]

Next, we define the semantics. Consider a run

\[ (\forall_{rings} \forall_{runs} \varphi) \]

Typically, one requires that a distributed algorithm is correct no matter what the

\[ \forall_{rings} \forall_{runs} \varphi \]

underlying ring is. Since we will bound the number of rounds, we moreover study a form of

\[ \forall_{rings} \forall_{runs} \varphi \]

partial correctness. Accordingly, a property is of the form

\[ \forall_{rings} \forall_{runs} \varphi \]

can be explored using regular expressions

\[ \forall_{rings} \forall_{runs} \varphi \]

to a temporal logic with past operators). By means of

\[ \forall_{rings} \forall_{runs} \varphi \]

the ring that can be retrieved at any time. Actually,

\[ \forall_{rings} \forall_{runs} \varphi \]

read as "for all rings, all runs, and all processes

\[ \forall_{rings} \forall_{runs} \varphi \]

can be combined with a regular expression

\[ \forall_{rings} \forall_{runs} \varphi \]

a given position/coordinate of the cylinder, we can check

\[ \forall_{rings} \forall_{runs} \varphi \]

This is satisfied. The most interesting construct in our logic

\[ \forall_{rings} \forall_{runs} \varphi \]

is to a \(-\)path (a path matching

\[ \forall_{rings} \forall_{runs} \varphi \]

a \(-\)path to some position where

\[ \forall_{rings} \forall_{runs} \varphi \]

is satisfied. The run conforms to the correctness property formulated in Example 3. In particular, in

\[ \forall_{rings} \forall_{runs} \varphi \]

Moreover, a non-colored row forms, together with the states above and below, a transition

\[ \forall_{rings} \forall_{runs} \varphi \]

in Figure 3 (for the moment, we may ignore the blue and violet lines). A colored row forms

\[ \forall_{rings} \forall_{runs} \varphi \]

in a cell refer to registers

\[ \forall_{rings} \forall_{runs} \varphi \]

the final configuration, all processes store the maximum pid in register

\[ \forall_{rings} \forall_{runs} \varphi \]

that the run conforms to the correctness property formulated in Example 3. In particular, in

\[ \forall_{rings} \forall_{runs} \varphi \]

r

\[ \forall_{rings} \forall_{runs} \varphi \]

is the number of processes in the underlying ring. A local

\[ \forall_{rings} \forall_{runs} \varphi \]

algorithm (e.g., "at the end, all processes store the maximal pid in register

\[ \forall_{rings} \forall_{runs} \varphi \]

coefficienticienticients (is used to

\[ \forall_{rings} \forall_{runs} \varphi \]

the respective pid from the cylinder). Next, we introduce a

\[ \forall_{rings} \forall_{runs} \varphi \]

Definition 6. and

\[ \forall_{rings} \forall_{runs} \varphi \]

the pid stored in

\[ \forall_{rings} \forall_{runs} \varphi \]

that lead to positions
Specifications (DataPDL)

\[ \Phi ::= \forall_{\text{rings}} \forall_{\text{runs}} \varphi \]

state: P, id = 23, r = 47, r1 = 47, r2 = 5
Specifications (DataPDL)

\[ \Phi ::= \forall_{\text{rings}} \forall_{\text{runs}} \varphi \]

\[ \text{state:P, id = 23, } \]
\[ r = 47, r1 = 47, r2 = 5 \]
Specifications (DataPDL)

\[ \Phi ::= \forall_{rings} \forall_{runs} \forall_{m} \varphi \]

state: P, id = 23, r = 47, r1 = 47, r2 = 5
Specifications (DataPDL)

\[ \Phi ::= \forall_{rings} \forall_{runs} \forall_m \varphi \]

The logic DataPDL allows us to "navigate" back and forth. This is like placing a pebble in a ring and then navigating to other rings. This is like reasoning about XML documents. A colored row forms a vertical path (a path matching a given position/coordinate of the cylinder, we can check partial correctness. Accordingly, a property is of the form \( \forall_m \varphi \). Typically, one requires that a distributed algorithm is correct no matter what the underlying ring is. Since we will bound the number of rounds, we moreover study a form of formal language to specify correctness properties. It is defined wrt. a given configuration. The three pids in a cell refer to registers \( P, A, R \). In Examples 2 and 3, we informally stated the correctness criterion for the presented distributed algorithm (e.g., "at the end, all processes store the maximal pid in register \( Reg_0 \)". Now, we will introduce our logic in full generality. Later, we will restrict the use of \( m \).

...
The logic DataPDL is given by the following grammar:

\[ \Phi ::= \forall_{\text{rings}} \forall_{\text{runs}} \forall_{\text{m}} \varphi \]

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

\[ \Phi ::= \forall_{\text{rings}} \forall_{\text{runs}} \forall_m \varphi \]

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

\[ \Phi ::= \forall_{rings} \forall_{runs} \forall m \varphi \]

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

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


where false

Definition 6. that lead to positions can be combined with a regular expression

can be explored using regular expressions

the ring that can be retrieved at any time. Actually,

read as "for all rings, all runs, and all processes

partial correctness. Accordingly, a property is of the form

underlying ring is. Since we will bound the number of rounds, we moreover study a form of

introduce a

algorithms (e.g., "at the end, all processes store the maximal pid in register

In Examples 2 and 3, we informally stated the correctness criterion for the presented

that the run conforms to the correctness property formulated in Example 3. In particular, in

Moreover, a non-colored row forms, together with the states above and below, a transition

in Figure 3 (for the moment, we may ignore the blue and violet lines). A colored row forms

in Figure 3 (for the moment, we may ignore the blue and violet lines). A colored row forms

Moreover, a non-colored row forms, together with the states above and below, a transition

in Figure 3 (for the moment, we may ignore the blue and violet lines). A colored row forms

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition

Moreover, a non-colored row forms, together with the states above and below, a transition
Specifications

\[ \Phi ::= \forall_{\text{rings}} \forall_{\text{runs}} \forall_m \varphi \]

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

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

\[ s \in S, \ r, r' \in \text{Reg}, \Join \in \{=, \neq, <, \leq\}, \text{and} \ d \in \{\epsilon, \leftarrow, \rightarrow, \uparrow, \downarrow\}. \]
Definition 6. the pid stored in $s$ is the state taken by a process, or whether we are on the marked process to a temporal logic with past operators). By means of avoiding to "get lost" in a ring when writing the property partial correctness. Accordingly, a property is of the form underlying ring is. Since we will bound the number of rounds, we moreover study a form of distributed algorithm (e.g., "at the end, all processes store the maximal pid in register $\text{Reg}$). Moreover, a non-colored row forms, together with the states above and below, a transition tuple. When looking at the step from a configuration, the three pids in a cell refer to registers in Figure 3 (for the moment, we may ignore the blue and violet lines). A colored row forms to a temporal logic with past operators). By means of avoiding to "get lost" in a ring when writing the property partial correctness. Accordingly, a property is of the form underlying ring is. Since we will bound the number of rounds, we moreover study a form of distributed algorithm (e.g., "at the end, all processes store the maximal pid in register $\text{Reg}$). Moreover, a non-colored row forms, together with the states above and below, a transition tuple. When looking at the step from a configuration, the three pids in a cell refer to registers in Figure 3 (for the moment, we may ignore the blue and violet lines). A colored row forms to a temporal logic with past operators). By means of avoiding to "get lost" in a ring when writing the property partial correctness. Accordingly, a property is of the form underlying ring is. Since we will bound the number of rounds, we moreover study a form of distributed algorithm (e.g., "at the end, all processes store the maximal pid in register $\text{Reg}$). Moreover, a non-colored row forms, together with the states above and below, a transition tuple. When looking at the step from a configuration, the three pids in a cell refer to registers in Figure 3 (for the moment, we may ignore the blue and violet lines). A colored row forms to a temporal logic with past operators). By means of avoiding to "get lost" in a ring when writing the property partial correctness. Accordingly, a property is of the form underlying ring is. Since we will bound the number of rounds, we moreover study a form of distributed algorithm (e.g., "at the end, all processes store the maximal pid in register $\text{Reg}$). Moreover, a non-colored row forms, together with the states above and below, a transition tuple. When looking at the step from a configuration, the three pids in a cell refer to registers in Figure 3 (for the moment, we may ignore the blue and violet lines). A colored row forms.

The Specification Language

The logic DataPDL is given by the following grammar:

$$\Phi ::= \forall \text{rings} \forall \text{runs} \forall m \varphi$$

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

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

$$s \in S, r, r' \in \text{Reg}, \circ \in \{=, \neq, <, \leq \}, \text{and } d \in \{\epsilon, \leftarrow, \rightarrow, \uparrow, \downarrow\}.$$
Next, we define a restricted logic, \( \Delta \), and positions for all rings \( \pi, \pi' := \{ \varphi \} \) \( s \) \( \pi \oplus \pi' \) \( \pi \cdot \pi' \) \( \pi^* \) \( s \in S, r, r' \in \text{Reg} \), \( \in \in \{ =, \neq, <, \leq \} \), and \( d \in \{ \epsilon, \leftarrow, \rightarrow, \uparrow, \downarrow \} \).

\[
\forall \text{rings} \forall \text{runs} \forall m [\downarrow^*]((\varphi_{\text{last}} \land \varphi_{\text{acc}}) \Rightarrow)
\]
\[ \Phi := \forall \text{rings} \forall \text{runs} \forall m \varphi \]
\[ \varphi, \varphi' := m \mid s \mid \neg \varphi \mid \varphi \land \varphi' \mid \varphi \Rightarrow \varphi' \mid [\pi] \varphi \mid (\pi)r \bowtie (\pi')r' \]
\[ \pi, \pi' := \{ \varphi \}? \mid d \mid \pi + \pi' \mid \pi \cdot \pi' \mid \pi^* \]
\[ s \in S, r, r' \in \text{Reg}, \bowtie \in \{ =, \neq, <, \leq \}, \text{and } d \in \{ \epsilon, \leftarrow, \rightarrow, \uparrow, \downarrow \}. \]

\[ \forall \text{rings} \forall \text{runs} \forall m \downarrow^* ( (\varphi_{\text{last}} \land \varphi_{\text{acc}}) \Rightarrow \downarrow \text{false} ) \]
In Examples 2 and 3, we informally stated the correctness criterion for the presented algorithm. Specifically, it says that, at the end (i.e., in the last configuration) of each accepting run, expressed by $\Phi_{\text{acc}}$, we must have that $\forall m \exists \mu \left[ \left( \forall \mu \left( \forall \pi \& [\pi] \Phi_{\text{acc}} \right) \right) \Rightarrow \left( \forall \mu \right) \right]$. Such a property is of the form $\forall m \exists \mu \left[ \left( \forall \mu \left( \forall \pi \& [\pi] \Phi_{\text{acc}} \right) \right) \Rightarrow \left( \forall \mu \right) \right]$, and $\forall m \exists \mu \left[ \left( \forall \mu \left( \forall \pi \& [\pi] \Phi_{\text{acc}} \right) \right) \Rightarrow \left( \forall \mu \right) \right]$.
Thus, it says that, at the end (i.e., in the last configuration) of each accepting run, expressed by terms of a binary relation \[ \cdot \]

It remains to define \( i, j \) and \( a \) given position/coordinate of the cylinder, we can check forth (read as “for all rings, all runs, and all processes

Moreover, a non-colored row forms, together with the states above and below, a transition

\[ = \]

\[ ÛÙ \]

Formally, for all rings

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

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

\[ \forall_{rings} \forall_{runs} \forall_{m} \varphi \]

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

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

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

\[ \forall_{rings} \forall_{runs} \forall_{m} [\downarrow^*] \left( (\varphi_{\text{last}} \land \varphi_{\text{acc}}) \Rightarrow (\varphi_{\text{found}} \land \varphi_{\text{max}} \land \varphi_{r=r} \land \varphi_{r=id}) \right) \]

\[ [\downarrow] \text{false} \]

\[ [\rightarrow^*](\text{passive} \lor \text{found}) \]
where

\[\Phi := \forall_{\text{rings}} \forall_{\text{runs}} \forall m \varphi\]

\[\varphi, \varphi' := m | s | \neg \varphi | \varphi \land \varphi' | \varphi \Rightarrow \varphi' | [\pi] \varphi | (\pi) r \Join (\pi') r'\]

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

s ∈ S, r, r' ∈ Reg, ∞ ∈ \{=, ≠, <, ≤\}, and d ∈ \{ε, ←, →, ↑, ↓\}.

\[\forall_{\text{rings}} \forall_{\text{runs}} \forall m [\downarrow^*] ((\varphi_{\text{last}} \land \varphi_{\text{acc}}) \Rightarrow (\varphi_{\text{found}} \land \varphi_{\text{max}} \land \varphi_{r=r} \land \varphi_{r=id}))\]

\[\langle \pi_{\text{found}} \rightarrow (\{\neg \text{found}\} ? \rightarrow)^* \rangle m\]
\[ \Phi ::= \forall_{\text{rings}} \forall_{\text{runs}} \forall_m \varphi \]

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

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

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

\[ \forall_{\text{rings}} \forall_{\text{runs}} \forall_m \downarrow^*(\varphi_{\text{last}} \land \varphi_{\text{acc}}) \Rightarrow (\varphi_{\text{found}} \land \varphi_{\text{max}} \land \varphi_{r=r} \land \varphi_{r=id}) \]

\[ \downarrow \text{false} \]

\[ \rightarrow^*(\text{passive} \lor \text{found}) \]

\[ \langle \pi_{\text{found}} \longrightarrow (\{\neg \text{found}\}? \longrightarrow)^* \rangle m \]

\[ (\{\neg \text{found}\}? \longrightarrow)^* \{\text{found}\}? \]
\[
\Phi := \forall_{\text{rings}} \forall_{\text{runs}} \forall_{m} \varphi \\
\varphi, \varphi' ::= m | s | \neg \varphi | \varphi \land \varphi' | \varphi \Rightarrow \varphi' | [\pi] \varphi | \langle \pi \rangle r \mathrel{\bowtie} \langle \pi' \rangle r' \\
\pi, \pi' ::= \{ \varphi \} ? | d | \pi + \pi' | \pi \cdot \pi' | \pi^* \\
s \in S, r, r' \in \text{Reg}, \bowtie \in \{ =, \neq, <, \leq \}, \text{and } d \in \{ \epsilon, \leftarrow, \rightarrow, \uparrow, \downarrow \}.
\]

\[
\forall_{\text{rings}} \forall_{\text{runs}} \forall_{m} [\downarrow^*] ( (\varphi_{\text{last}} \land \varphi_{\text{acc}}) \Rightarrow (\varphi_{\text{found}} \land \varphi_{\text{max}} \land \varphi_{r=r} \land \varphi_{r=id} ) )
\]

\[
[\downarrow] \text{false} \\
[\rightarrow^*] (\text{passive} \lor \text{found})
\]

\[
\langle \pi_{\text{found}} \rightarrow (\{ \neg \text{found} \} ? \rightarrow )^* \rangle m
\]

\[
(\{ \neg \text{found} \} ? \rightarrow )^* \{ \text{found} \} ?
\]

\[
[\rightarrow^*] (\langle \epsilon \rangle \text{id} \leq \langle \pi_{\text{found}} \rangle r)
\]
The other local formulas use path formulas. The semantics of a path formula \( \mathcal{L} \) is given in the form

\[
\mathcal{L} ::= \forall_{\text{paths}} \forall_{\text{runs}} \forall_{m} \mathcal{L}
\]

where \( m \) is a set of positions. Formally, for all rings \( \mathcal{R} \), we have

\[
\mathcal{L}[\rightarrow^*](\text{passive} \lor \text{found})
\]

and

\[
\neg(\langle \epsilon \rangle r \neq \langle \rightarrow^* \rangle r)
\]

Moreover,

\[
\langle \pi_{\text{found} \rightarrow} \{ \neg \text{found} \} \rightarrow^* \rangle m
\]

is satisfied. The most interesting construct in our logic are the regular constructs, typically one requires that a distributed algorithm is correct no matter what the underlying ring is. Since we will bound the number of rounds, we moreover study a form of

\[
\forall_{\text{rings}} \forall_{\text{runs}} \forall_{m} \mathcal{L}[\downarrow^*]((\varphi_{\text{last}} \land \varphi_{\text{acc}}) \Rightarrow (\varphi_{\text{found}} \land \varphi_{\text{max}} \land \varphi_{r=r} \land \varphi_{r=id}))
\]

where

\[
\varphi_{\text{last}} := m | s | \neg \varphi | \varphi \land \varphi' | \varphi \Rightarrow \varphi' | [\pi] \varphi | (\pi) r \Join (\pi') r'
\]

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

\[
s \in S, r, r' \in \text{Reg}, \infty \in \{ =, \neq, <, \leq \}, \text{and } d \in \{ \epsilon, \leftarrow, \rightarrow, \uparrow, \downarrow \}.
\]
We say that a path formula for all rings terms of a binary relation.

\[ \forall_{\text{rings}} \forall_{\text{runs}} \forall_{m} \varphi \]

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

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

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

\[ \Phi := \forall_{\text{rings}} \forall_{\text{runs}} \forall_{m} \varphi \]

\[ \forall_{\text{rings}} \forall_{\text{runs}} \forall_{m} \downarrow^* [(\varphi_{\text{last}} \land \varphi_{\text{acc}}) \Rightarrow (\varphi_{\text{found}} \land \varphi_{\text{max}} \land \varphi_{r=r} \land \varphi_{r=id})] \]

\[ \langle \pi_{\text{found}} \rightarrow (\{\neg \text{found}\}?\rightarrow)^* \rangle m \]

\[ \langle \pi_{\text{found}} \rangle (\langle \epsilon \rangle r = \langle \epsilon \rangle id) \]

\[ ([\rightarrow^*](\langle \epsilon \rangle id \leq \langle \pi_{\text{found}} \rangle r) \]

\[ (\{\neg \text{found}\}?\rightarrow)^* \{\text{found}\}? \]

\[ ([\rightarrow^*](\langle \epsilon \rangle r \neq \langle \rightarrow^* \rangle r) \]

\[ \langle \pi_{\text{found}} \rightarrow \rangle (\{\neg \text{found}\}?\rightarrow)^* \]
\[
\Phi := \forall_{\text{rings}} \forall_{\text{runs}} \forall_m \varphi
\]
\[
\varphi, \varphi' ::= m \mid s \mid \neg \varphi \mid \varphi \land \varphi' \mid \varphi \Rightarrow \varphi' \mid [\pi] \varphi \mid (\langle \pi \rangle r \bowtie \langle \pi' \rangle r') \mid \{\varphi\}?
\]
\[
\pi, \pi' ::= \{\varphi\}? \mid d \mid \pi + \pi' \mid \pi \cdot \pi' \mid \pi^*
\]
\[
s \in S, r, r' \in \text{Reg}, \bowtie \in \{=, \neq, <, \leq\}, \text{and } d \in \{\epsilon, \leftarrow, \rightarrow, \uparrow, \downarrow\}.
\]
\[
\varphi = (\langle \pi \rangle r \bowtie (\langle \pi' \rangle r') \text{ with } \bowtie \in \{<, \leq\} \text{ is such that } \pi \text{ and } \pi' \text{ are unambiguous. Moreover, } \varphi \text{ must not occur (i) in the scope of a negation, (ii) on the left-hand side of an implication } \_ \Rightarrow \_, \text{ or (iii) within a test } \{\_\}? \text{. Note that guards using } = \text{ and } \neq \text{ are still unrestricted.} \]

\[
\forall_{\text{rings}} \forall_{\text{runs}} \forall_m [\downarrow^*]( (\varphi_{\text{last}} \land \varphi_{\text{acc}}) \Rightarrow (\varphi_{\text{found}} \land \varphi_{\text{max}} \land \varphi_{r=r} \land \varphi_{r=id}))
\]

\[
[\downarrow]false [\rightarrow^*](\text{passive } \lor \text{ found}) \neg(\langle \epsilon \rangle r \neq \langle \rightarrow^* \rangle r)
\]

\[
\langle \pi_{\text{found}} \rightarrow (\{\neg \text{found}\}? \rightarrow)^* \rangle m \langle \pi_{\text{found}} \rangle(\langle \epsilon \rangle r = \langle \epsilon \rangle id)
\]

\[
(\{\neg \text{found}\}? \rightarrow)^* \{\text{found}\}?
\]

\[
[\rightarrow^*](\langle \epsilon \rangle id \leq \langle \pi_{\text{found}} \rangle r)
\]
Can we model check distributed algorithms?

- System - finite state automata serving as abstractions of hardware/software
- Specification - another finite state automata, LTL formula, first order logic formula...

\[ S \models \varphi? \]

Refine S
(Fix bugs)

Distributed Algorithms

DataPDL
Can we model check distributed algorithms?

- System - finite state automata serving as abstractions of hardware/software
- Specification - another finite state automata, LTL formula, first order logic formula…

\[
S \models \varphi
\]

Undecidable

Distributed Algorithms

DataPDL

Refine S
(Fix bugs)
<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
</tr>
</tbody>
</table>

**Undecidable**

Unbounded cylinder labelled by infinite data

state: P, id = 23, r = 47, r1 = 47, r2 = 5
Get rid of infinite data:

\[
\text{DA , DataPDL over cylinder } \implies \text{PDL+Loop over finitely labelled grids}
\]

Get rid of infinite data: symbolic runs + track origin

state: P, id = 23, r = 47, r1 = 47, r2 = 5
Get rid of infinite data: symbolic runs + track origin
Get rid of infinite data: symbolic runs + track origin
Get rid of infinite data: symbolic runs + track origin
Get rid of infinite data: symbolic runs + track origin
Get rid of infinite data: symbolic runs + track origin

```
| 5 | 71 | 42 | 19 | 47 | 23 |
```

```
A  A  A  A  A  A
r1 r2 t1 t2 t4 t4
P  P  P  P  P  P
```

```
right?r1
```

```
r1
```

```
t4 t3 t4 t4 t4 t4
F  P  P  P  P  P
```
Get rid of infinite data: symbolic runs + track origin
Get rid of infinite data: symbolic runs + track origin

```
A A A A A
5 71 42 19 47
```

```
A A A A A
t1 t2 t2 t2 t2
```

```
P A P P P
```

```
t4 t1 t4 t4 t4
```

```
P A P P P
```

```
t4 t3 t4 t4 t4
```

```
P F P P P
```

```
t4 t4 t4 t4 t4
```

```
P P P P P
```

```
A A A A A
left!r r
```

```
A A A A A
right!?r1 r1
```

Get rid of infinite data: symbolic runs + track origin

r not updated
r
left!r
right?r1
r1
Get rid of infinite data: symbolic runs + track origin
Get rid of infinite data: symbolic runs + track origin
Get rid of infinite data: symbolic runs + track origin
No contradictions!!
Get rid of infinite data: symbolic runs + track origin
Get rid of infinite data: symbolic runs + track origin

No contradictions!!

r2 < r1
No contradictions!!
Get rid of infinite data: symbolic runs + track origin

r2 < r1
Get rid of infinite data: symbolic runs + track origin

No contradictions!!

r2 < r1

r < r2
No contradictions!!
Get rid of infinite data: symbolic runs + track origin

\[ r_1 < r_2 \]

\[ r_2 < r_1 \]
No contradictions!!
Get rid of infinite data: symbolic runs + track origin
No contradictions!!
Get rid of infinite data: symbolic runs + track origin
No contradictions!!

Get rid of infinite data: symbolic runs + track origin
No contradictions!!
Get rid of infinite data: symbolic runs + track origin
No contradictions!!
Get rid of infinite data: symbolic runs + track origin
No contradictions!!
Get rid of infinite data: symbolic runs + track origin
No contradictions!!
Get rid of infinite data: symbolic runs + track origin

NO loop of the form (<-path)*
Get rid of infinite data: symbolic runs + track origin

DA, DataPDL over cylinder

⇒

PDL+Loop over finitely labelled grids
Get rid of infinite data: symbolic runs + track origin

DataPDL: For all rings …
Get rid of infinite data: symbolic runs + track origin

DataPDL: For all rings ...
Get rid of infinite data: symbolic runs + track origin

DataPDL: For all rings …
Get rid of infinite data: symbolic runs + track origin

- Loop formula
- If there is no such loop, formula is false for at least one ring

DataPDL: For all rings …
Get rid of infinite data: symbolic runs + track origin

- Loop formula
- If there is no such loop, formula is false for at least one ring

DataPDL: For all rings ...
Get rid of infinite data: symbolic runs + track origin

DA, DataPDL over cylinder

⇒

PDL+Loop over finitely labelled grids
<table>
<thead>
<tr>
<th>A</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>A</td>
</tr>
<tr>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>P</td>
<td>P</td>
</tr>
</tbody>
</table>

**Undecidable**

Unbounded cylinder labelled by infinite data

finitely labelled unbounded grid vs. PDL
Round bounded model checking

Does the DA satisfy the specification for all rings, and for all runs of at most k rounds?

Decidable
PDL with loop over bounded grids

\[ \Rightarrow \]

PDL with loop over words

\[ \Rightarrow \]

Alternating 2-way Automata

\[ \Rightarrow \]

PSPACE
PDL with loop over bounded grids

⇒

PDL with loop over words

⇒

Alternating 2-way Automata

⇒

PSPACE
Conclusions
Conclusions

Round bounded model checking of distributed algorithms over rings decidable
Conclusions

Round bounded model checking of distributed algorithms over rings decidable

Translation of DA and specification to PDL with loops over finitely labelled grids
Conclusions

Round bounded model checking of distributed algorithms over rings decidable

Translation of DA and specification to PDL with loops over finitely labelled grids

Independent of the number of rounds and the restriction to rings
Conclusions

Round bounded model checking of distributed algorithms over rings decidable

Translation of DA and specification to PDL with loops over finitely labelled grids

Independent of the number of rounds and the restriction to rings

Other restrictions?
Conclusions

Round bounded model checking of distributed algorithms over rings decidable

Translation of DA and specification to PDL with loops over finitely labelled grids

Independent of the number of rounds and the restriction to rings

Other restrictions?

Other topologies?
Namaste!

Merci!

Thank you!

Danke!

Namaste!
Janamdin Mubarak Ho

Bon anniversaire

Happy Birthday

Alles Gute zum Geburtstag

Volker!