Automata-theoretic Verification of Distributed Algorithms

C. Aiswarya, Benedikt Bollig, Paul Gastin

FRIDAy 5th June, 2015
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?

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

- On ring topologies
- Left neighbour and a right neighbour
- Number of process is unknown and unbounded
- Processes have unique pids (integers — unbounded data)
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
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
The max-pid process is elected as the leader.

- candidate supported by its right neighbour
- checks whether the left neighbour is a local maxima
- becomes passive if not an active supporter
- passive processes enter a “fwd” state to enable communication between the active processes

Leader Election Algorithms

Dolev-Klawe-Rhodeh82
The max-pid process is elected as the leader by its right neighbour.

Candidate supported by its right neighbour.

Checks whether the left neighbour is a local maxima.

Becomes passive if not an active supporter.

Passive processes enter a "fwd" state to enable communication between the active processes.
The max-pid process is elected as the leader

- candidate supported by its right neighbour
- checks whether the left neighbour is a local maxima
- becomes passive if not an active supporter
- passive processes enter a “fwd” state to enable communication between the active processes
Can we model check distributed algorithms?

Model checking

- 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..

System $S$

Specification $\varphi$

$S \models \varphi?$

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?

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

two sources of infinity
Finite state systems

Unknown number of processes

Unknown number of processes handling data

Parametrised verification

Data (automata/logics)

Systems handling unbounded data


A formal model for distributed algorithms

An automata-like way of writing DA

- 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, r_1, r_2

\[ t_1 = \langle \text{active: left!id ; right!id ; left?r}_1 ; \text{right?r}_2 ; r_1 < id ; r_2 < id ; \text{goto active} \rangle \]
\[ t_2 = \langle \text{active: } \text{id} < r_1 ; \text{goto passive} \rangle \]
\[ t_3 = \langle \text{active: } \text{id} < r_2 ; \text{goto passive} \rangle \]
\[ t_4 = \langle \text{active: } \text{id} = r_1 ; r := id ; \text{goto found} \rangle \]
\[ t_5 = \langle \text{passive: fwd ; left?r ; goto passive} \rangle \]
Run

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

Franklin82

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

\[ \Phi ::= \forall_{rings} \forall_{runs} \forall_m \varphi \]
\[ \varphi, \varphi' ::= m \mid s \mid \neg \varphi \mid \varphi \land \varphi' \mid \varphi \Rightarrow \varphi' \mid [\pi]\varphi \mid \langle \pi \rangle r \Join \langle \pi' \rangle r' \]
\[ \pi, \pi' ::= \{ \varphi \}? \mid d \mid \pi + \pi' \mid \pi \cdot \pi' \mid \pi^* \]

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

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

\[ \langle \pi \rangle r < \langle \pi' \rangle r' \text{ allowed only if } \pi \text{ and } \pi' \text{ are unambiguous} \]

\[ \langle (\downarrow + \rightarrow)^* \rangle (P \Rightarrow (\langle \cdot \text{id} \rangle \neq \langle \cdot \rangle r) \]

\[ \begin{array}{cccccccc}
P & A & P & P & P & P & P & P \\
P & F & P & P & P & P & P & P \\
\end{array} \]
Specifications

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

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

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

$\langle \pi \rangle r < \langle \pi' \rangle r'$ allowed only if $\pi$ and $\pi'$ are unambiguous
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...

$S \models \phi$?

\[ \checkmark \]

Refine $S$

(Fix bugs)

Undecidable

Distributed Algorithms

Data PDL
Undecidable

Infinite grid labelled by infinite data

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

specification, DA

$\xrightarrow{}$

PDL with loops over finitely labelled grids
Undecidable

finitely labelled
Infinite grid vs. PDL
labelled by infinite data
Round bounded model checking

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

Decidable
Conclusions

Round bounded model checking of distributed algorithms over rings decidable

Translation of specification and DA to PDL over finitely labelled grids

Independent of the number of rounds, and even the restriction

Other restrictions?

Other topologies?
Namaste!

Merci!

Thank you!