Safety Analysis of Parameterised Networks with Non-Blocking Rendez-Vous and Broadcasts

Lucie Guillou, Arnaud Sangnier, Nathalie Sznajder

IRIF, Université Paris Cité  DIBRIS, Università di Genova  Lip6, Sorbonne Université

22nd February, 2024
Seminaire Move
Context

- Modelling Distributed / Concurrent Systems
  - Means of communication: shared memory/message passing
  - Number of entities: bounded / unbounded / dynamic
  - Scheduling: Synchronous / Asynchronous
  - Executed Protocol: same for everyone / presence of a leader

- Developing verification techniques
  - Which specification: safety / liveness / robustness
Parameterised Distributed Networks

- Unknown number of agents
- Each agent follows a protocol given as a finite-state machine
- Synchronous Communication
- Interleaving Semantics

...
Verification of Parameterised Distributed Networks

Is there a number of agents such that there exists a run leading to a bad configuration?
All agents execute the same protocol.

A Protocol
Communication

A process

the sender

,q, !!a, q'

A Broadcast
Communication

A Rendez-Vous
Communication

not able/ready to receive \( a \)

no process able to receive the message.

A Non-Blocking Sending

\( q, \text{!}a, q' \)
An Execution

Initial configuration with 4 processes:

\[ \begin{array}{llll}
P1 & P2 & P3 & P4 \\
0 & 0 & 0 & 0 \\
\end{array} \]

\[ \text{NB}(a) \rightarrow \begin{array}{llll}
P1 & P2 & P3 & P4 \\
1 & 0 & 0 & 0 \\
\end{array} \]

\[ \text{BR}(b) \rightarrow \begin{array}{llll}
P1 & P2 & P3 & P4 \\
1 & 1 & 4 & 4 \\
\end{array} \]

\[ \text{RDV}(d) \rightarrow \begin{array}{llll}
P1 & P2 & P3 & P4 \\
3 & 2 & 6 & 4 \\
\end{array} \]

\[ \text{BR}(a) \]

\[ \text{RDV}(a) \]
Motivation

Java  Threads  Programming

- Notify / NotifyAll messages
- Delzanno et. al. TACAS 2008
- Non-blocking sending
Verification Problems.

**STATE - COVER**

Given a protocol and a special state, is there a number of agents $N$ such that?

**CONF - COVER**

Given a protocol and a configuration, is there a number of agents $N$ such that?
State of the Art

<table>
<thead>
<tr>
<th>Broadcasts + Rendez-vous + Non-Blocking sending</th>
<th>State-COVER</th>
<th>Conf-COVER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decidable but Ackermann-hard</td>
<td>[Schmitz et al., CONCUR’15]</td>
<td>[Emerson et al., LICS’03]</td>
</tr>
<tr>
<td></td>
<td>[Esparza et al., LICS’99]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[Aminof et al., LPAR’15]</td>
<td></td>
</tr>
</tbody>
</table>
## State of the Art

<table>
<thead>
<tr>
<th>State-COVER</th>
<th>Conf-COVER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decidable but</td>
<td>Ackermann - hard</td>
</tr>
</tbody>
</table>

- Broadcasts + Rendezvous + Non-Broadcasting sending
- Broadcasts

References:
- [Schmitz et al. CONCUR'13]
- [Esparza et al. LICS'95]
- [Amini et al. LPAR'15]
- [Emerson et al. LICS'03]
## State of the Art

<table>
<thead>
<tr>
<th>Broadcasts + Rendez-vous + Non-Blocking sending</th>
<th>Broadcasts</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Decidable but Ackermann-hard</strong></td>
<td></td>
</tr>
<tr>
<td>[Schmitz et. al. CONCUR'13]</td>
<td></td>
</tr>
<tr>
<td>[Esparza et. al. LICS'95]</td>
<td></td>
</tr>
<tr>
<td>[Amiral et. al. LPAR'15]</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rendez-Vous without NB sending</th>
<th><strong>in Ptime</strong></th>
<th>[German et. al. 92]</th>
</tr>
</thead>
</table>

[1139]
## State of the Art

<table>
<thead>
<tr>
<th>Broadcasts + Rendez-Vous + Non-Blocking sending</th>
<th>Broadcasts</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>State-Cover</strong></td>
<td><strong>Conf-Cover</strong></td>
</tr>
<tr>
<td>Decidable but Ackermann-hard</td>
<td>[Schmitz et al. CONCUR'13]</td>
</tr>
<tr>
<td></td>
<td>[Esparza et al. LICS '95]</td>
</tr>
<tr>
<td></td>
<td>[Amir et al. LPAR'15]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rendez-Vous without NB sending</th>
<th>\text{in Ptime}</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[German et al. 92]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rendez-Vous and Non-Blocking Sending</th>
<th>?</th>
</tr>
</thead>
</table>
Part I: Without broadcasts

[Ex., Sangnier, Strajder, Comuna 2023]
RDV Protocol

Diagram:

0 -> 3
  | ?b
  | !a
  v
1

1 -> 2
  !d
  v
2

2 -> 0
  !b
  v

3 -> 4
  !c
  v
4

4 -> 5
  ?c
  v
5
Results.

Conf - COVER and State - COVER are Expspace-complete

To contrast with "Blocking Rendez-Vous":

- No Non-Blocking Sending
- Sending / Reception symmetric
- Both problems are in Ptime [German et. al. 1992]
Results.

[CONCUR'23]

Conf - COVER and State - COVER are \textit{ExpSPACE - complete}

- \textit{ExpSPACE - membership}:
  - proof inspired from Rackoff for VASS

- \textit{ExpSPACE - hardness}:
  - proof inspired from Lipton for VASS

⚠️ No direct translation to/from VASS.
Link with VASS

A VASS

Finite-State Machine

+ Counters (in \( \mathbb{N} \))

A VASS with two counters
Link with VASS

A VASS with two counters

An execution:

\begin{align*}
    x_1 &\quad l_0 \\ x_2 &\quad l_0 \\
    &\quad \rightarrow \\
    0 &\quad 0 \\
    &\quad \rightarrow \\
    3 &\quad 3 \\
    &\quad \rightarrow \\
    &\quad 4 \\
    &\quad \rightarrow \\
    &\quad 2 \\
    &\quad \rightarrow \\
    &\quad 3 \\
    &\quad \rightarrow \\
    3 &\quad 3 \\
    &\quad \rightarrow \\
    0 &\quad 0 
\end{align*}
**Link with VASS**

A VASS with two counters

**An execution:**

$$
\begin{align*}
\text{l}_0 & \rightarrow \text{l}_1 & \text{l}_1 & \rightarrow \text{l}_2 & \text{l}_2 & \rightarrow \text{l}_f \\
\text{x}_1 & = 0 & \rightarrow & \text{x}_1 & = 3 & \rightarrow & \text{x}_1 & = 3 \\
\text{x}_2 & = 0 & \rightarrow & \text{x}_2 & = 0 & \rightarrow & \text{x}_2 & = 3 \\
\end{align*}
$$

**lf is coverable**
Link with VASS

A Rendez-Vous

Translated

$q \leftarrow q - 1$
$q' \leftarrow q' + 1$
$p \leftarrow p - 1$
$p' \leftarrow p' + 1$
Link with VASS

A Non-Blocking Sending

\[ q \quad !a \quad q' \]

... Translated

\[ q \leftarrow q - 1 \]
\[ q' \leftarrow q' + 1 \]

but how to verify that others states are empty in \( l \)?
Why such a complexity gap?

In blaming Rendez-vous semantics, we have a nice property:

Copycat Lemma:
If a state $\bullet$ is coverable, then any configuration $\circ$ is coverable.

$\implies$ Conf: COVER and State: COVER in P-time
Why such a complexity gap?

Main ingredient: with non-blocking rendez-vous, we can isolate some processes.

At most one process on state 1.

NB(a) → 0 0 → ADV(a) → [1] → ...

0 0 → 1 0 → ...
Why such a complexity gap?

Main ingredient: with non-blocking rendez-vous, we can isolate some processes

![Diagram](image)

$\exists b, !c$

at most one process on \{1, 2, 3\}

0 0 \ldots \rightarrow 0 0 \ldots \rightarrow 2 0 \ldots \rightarrow \ldots \rightarrow 1 \ldots \rightarrow \ldots$

$\Rightarrow$ Conf - COVER and ExpSPACE - hard
A Restriction: Wait-Only Protocol

- Protocols where each state is either:
  - an action state
  - a waiting state

Natural Restriction

We suppose that the initial state is always an action state.
Results.

Conf - COVER and State - COVER are EXPSPACE - complete

**Why?**
Isolation mechanism is less powerful

**How?**
Abstraction on Configurations.
Inductive computation until saturation
Wait-Only Property

If an action state 9 is coverable...

then it is coverable by any number of processes...
Wait-Only Property

If an action state $q$ is coverable...

then it is coverable by any number of processes...

We can combine action states: if two action states $q$ and $p$, they are coverable at the same time.
Wait-Only Property

If an action state $q$ is coverable...

then it is coverable by any number of processes...

We can combine action states with one waiting state: if an action state $q$ and a waiting state $p$ are coverable, we can cover $q$ and $p$ at the same time.
Wait-Only Protocols

Why not more?

9 action state, p waiting state, s waiting state.
Wait-Only Protocols

Why not more?

$q$ action state, $p$ waiting state, $s$ waiting state.

 receptions can happen
Abstract Configurations.

Coverable states with any number of agents.

Reachable all together.

Toks

\(<\text{state, message}>\) (a token). Each state coverable by 1 agent.

Not necessarily reachable all together (message).

Gives us all the coverable configurations.
Greedy Algorithm to compute the right abstract configuration.
Computation

<table>
<thead>
<tr>
<th>S</th>
<th>Tokes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Node 0 can send "a" to node 1.
- Node 0 can receive "b" from node 2.
- Node 2 can receive "a" from node 3.
- Node 2 can send "b" to node 4.

- Node 4 can receive "b".
- Node 3 can receive "a".

- Node 1 has any number of agents.
- Node 4 has at most one agent.
Computation

<table>
<thead>
<tr>
<th></th>
<th>S</th>
<th>Tokas</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

any number of agents

at most one agent
Computation

At most one agent on state 2, reachable through a non-blocking sending of b.
Computation

At most one agent on state 2, reachable through a non-blocking sending of b.
Computation

<table>
<thead>
<tr>
<th>$S$</th>
<th>Tokens</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2, b</td>
</tr>
<tr>
<td>0, 1, 3, 4</td>
<td>2, b</td>
</tr>
</tbody>
</table>

Any number of agents $J$

At most one agent
Computation

\[ NB(a)^k \quad RDV(b)^k \]

\[
\begin{array}{c|c}
S & Toks \\
\hline
0 & \text{2, b} \\
0, 1, 3, 4 & \text{2, b} \\
\end{array}
\]

Any number of agents

\( \leq \) at most one agent
## Summary of Part I

<table>
<thead>
<tr>
<th>State-Cover</th>
<th>Conf-Cover</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decidable but Ackermann-hard</td>
<td>[\text{ExpSpace-Complete}]</td>
</tr>
<tr>
<td>[Schmitz et al. Concur 15]</td>
<td>[Concur 2023]</td>
</tr>
<tr>
<td>[Esparza et al. LICS 95]</td>
<td>[Concur 2023]</td>
</tr>
<tr>
<td>[Amir et al. LPAR 18]</td>
<td>[\text{Ptime - Complete}]</td>
</tr>
</tbody>
</table>

- **Broadcasts + Rendez-Vous + Non-Blocking Sending**
- **Rendez-Vous without NB sending**
- **Rendez-Vous and Non-Blocking Sending**
- **Wait-Only Protocols with Rendez-Vous and Non-Blocking Sending**
Part II: Adding broadcasts

[ e.g., Sangnier, Stnajder, submitted ]

New Results
Wait-Only Broadcast Protocols

- action states and waiting states
- What happens with broadcasts?
Wait-Only Broadcast Protocols

- action states and waiting states
- What happens with broadcasts?

State-COVER is P-time-Complete
Conf-COVER is PSPACE-Complete
State-Cover is Prime-Complete

Wait-Only Property

If an action state $q$ is coverable...

Then it is coverable by any number of processes...

We can combine action states with one waiting state: if an action state $q$ and a waiting state $p$ are coverable, we can cover $q$ and $p$ at the same time.
State-Cover is Prime-Complete

**Wait-Only Property**

If an action state $q$ is coverable...

```
0 ... 0  -->  *  ....  q
```

then it is coverable by any number of processes...

```
0 ... 0  -->  *  ....  q
0 ... 0  -->  ....  q
```

We can combine action states with one waiting state: if an action state $q$ and a waiting state $p$ are coverable, we can cover $q$ and $p$ at the same time.

Still holds when adding broadcasts.

Gives a simple inductive algorithm computing the set of all coverable states.
Conf-

COVER is PSPACE-Complete

- More complex algorithm
- Algorithm in polynomial space in the size of the protocol and of the configuration to cover
Conf-COVER is PSPACE-Complete

* More complex algorithm
* Algorithm in polynomial space in the size of the protocol and of the configuration to cover

\[ C_f: \text{the configuration to cover} \]
\[ |C_f|: \text{number of agents in } C_f = K \]
**Conf. COVER** is **PSPACE-Complete**

- More complex algorithm
- Algorithm in polynomial space in the size of the protocol and of the configuration to cover

\[ C_f : \text{the configuration to cover} \]
\[ ||C_f|| : \text{number of agents in } C_f. \ = \ K \]

We focus on \( K \) agents, knowing the set \( S \) of coverable states.
An Example

Diagram:

- Nodes: 0, 1, 2, 3, 4, 5, 6, 7
- Edges: ![Diagram](image)
- Labels:
  - Node 0: `!!a`, `!!d`
  - Node 1: `!!a`, `?c`
  - Node 2: `!b`, `?b`
  - Node 3: `?c`, `!?b`
  - Node 4: `!!d`, `?a`
  - Node 5: `!!c`, `!?b`
  - Node 6: `!!c`, `!?b`
  - Node 7: `!!d`, `?a`

Cf:

- Set: `{3, 3, 6}`
An Example

\[ S = \text{All states} \quad (\text{Coverable states}) \]

\[ \rightarrow \text{Focus on 3 processes.} \]
An Example

S = \{0, 1, 2, 3, 4, 5, 6, 7\}

P1 P2 P3

0 0 0

Cf.

3 3 6
An Example

\[ S = \{ 0, 1, 2, 3, 4, 5, 6, 7 \} \]
An Example

S = \{0, 1, 2, 3, 4, 5, 6, 7\}

0000 → 140 → 151

BR(a)

P1 P2 P3
An Example

\[ S = \{0, 1, 2, 3, 4, 5, 6, 7\} \]

Diagram:

- Nodes: 0, 1, 2, 3
- Edges:
  - \(!a \rightarrow 1\)
  - \(?c \rightarrow 2\)
  - \(!b \rightarrow 3\)
  - \(?a \rightarrow 4\)
  - \(!!c \rightarrow 5\)
  - \(?b \rightarrow 6\)
  - \(6 \rightarrow 7\)

Cf.

Transition System:

- States: 0, 1, 2, 3
- Actions: a, b, c
- Initial State: 0
- Initial Action: a
- Final States: 6, 7

BR(a):

- States: 1, 4, 0
- Initial State: 1
- Final State: 5

BR(c):

- States: 1, 5, 1
- Initial State: 1
- Final State: 6, 2

P1, P2, P3
An Example

\[ S = \{0, 1, 2, 3, 4, 5, 6, 7\} \]

\[ \text{ADV}(b) \rightarrow \begin{array}{c} 0000 \\ \end{array} \rightarrow ^{+} \begin{array}{c} 140 \\ \end{array} \rightarrow \begin{array}{c} 151 \\ \end{array} \rightarrow \begin{array}{c} 868 \\ \end{array} \]

\[ \text{Cf.} \]
An Example

S = \{0, 1, 2, 3, 4, 5, 6, 7\}

\[000 \xrightarrow{+} 140 \xrightarrow{\text{BR}(a)} 151\]
An Example

\[ S = \{ 0, 1, 8, 3, 4, 5, 6, 7 \} \]
An Example

\[ S = \{ 0, 1, 2, 3, 4, 5, 6, 7 \} \]

\( \text{NB}(b) \Rightarrow \text{NB}(b) \)

\( \text{BR}(a) \Rightarrow \text{BR}(a) \)

\( \text{Ext}(c) \Rightarrow \text{Ext}(c) \)
An Example

\[ S = \{ 0, 1, 2, 3, 4, 5, 6, 7 \} \]
An Example

\[ S = \{0, 1, 2, 3, 4, 5, 6, 7\} \]
An Example

$S = \{0, 1, 2, 3, 4, 5, 6, 7\}$

BR(a)

Ext(c)

NB(b)

BR(c)

ADV(b)

ADV(b)
An Example

\[ S = \{ 0, 1, 2, 3, 4, 5, 6, 7 \} \]

\[ \begin{align*}
000 & \rightarrow + & 140 & \rightarrow & 151 & \rightarrow & 252 \\
853 & \rightarrow \text{NB}(b) & 863 & \rightarrow \text{BR}(c) & 363 & \rightarrow \text{EXT}(b) & ?
\end{align*} \]
An Example

State 6 is a waiting state

No property for waiting state
An Example

Solution: we follow the process receiving "b" instead of the sender, and then switch processes.
An Example

$S = \{0, 1, 2, 3, 4, 5, 6, 7\}$

[Diagram of a graph with nodes and edges labeled with operations and numbers.]
An Example

\[ S = \{0, 1, 2, 3, 4, 5, 6, 7\} \]
PSPACE-completeness

in PSPACE:

Explore all configurations of cyclic agents non-deterministically in polynomial space and conclude with Savitch Theorem.

PSPACE-hard:

Reduction of the problem of the emptiness of the intersection of languages of a list of automata.
Conclusion and Future Work

- Good model for Java Threads Programming
  - non-blocking rendez-vous + broadcast
  - wait-only restrictions

- Liveness properties
- Dynamic creation of processes / objects
### Summary

<table>
<thead>
<tr>
<th>State- COVER</th>
<th>Conf- COVER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Broadcasts + Rendez-Vous + Non-Blocking sending</td>
<td>Decidable but Ackermann-hard</td>
</tr>
<tr>
<td>Broadcasts</td>
<td>[Schmitz et. al. Concur’15] [Emerson et. al. LICS’03]</td>
</tr>
<tr>
<td>Rendez-Vous without NB sending</td>
<td>in Ptime</td>
</tr>
<tr>
<td>Rendez-Vous and Non-Blocking Sending</td>
<td>EXPSPACE- Complete</td>
</tr>
<tr>
<td>Wait- Only Protocols with Rendez-Vous and Non-Blocking Sending</td>
<td>Ptime- Complete</td>
</tr>
<tr>
<td>Wait- Only Protocols with Rendez-Vous + Non-Blocking Sending + Broadcasts</td>
<td>Ptime- Complete</td>
</tr>
</tbody>
</table>