Parameterized Verification of Broadcast Networks of Register Automata

Nicolas Waldburger
Lucie Guillou
Corto Mascle

Funded by ANR PaVeDyS
March 11th, 2024
To be published at FoSSaCS'24
Introduction of the model

Decidability of $\text{Cover}$ for signature BNRA

Decidability of $\text{Cover}$ in the general case

Complexity lower bound
Broadcast networks
Broadcast networks

\[ \text{rec}(b) \]
\[ \text{br}(a) \]
\[ \text{br}(b) \]
\[ \text{rec}(a) \]
\[ \text{br}(c) \]

Nicolas Waldburger
Introduction of the model

Broadcast networks

![Diagram of broadcast networks with labeled transitions and nodes](image-url)
Broadcast networks

![Diagram of broadcast networks]

Introduction of the model
Introduction of the model

Broadcast networks
(Reconfigurable) Broadcast Network $= (Q, \mathcal{M}, \Delta, q_0)$ with
$\Delta \subseteq Q \times \{\text{br}(m), \text{rec}(m) \mid m \in \mathcal{M}\} \times Q.$

---

$^1$Delzanno, Sangnier, Zavattaro, CONCUR’10
Broadcast Networks

Definition

(Reconfigurable) Broadcast Network \( = (Q, M, \Delta, q_0) \) with \( \Delta \subseteq Q \times \{\text{br}(m), \text{rec}(m) \mid m \in M\} \times Q \)

- Arbitrarily many agents at the start
- One step = an agent broadcasts a message \( m \), some (arbitrary subset of) other agents receive it.

\(^1\)Delzanno, Sangnier, Zavattaro, CONCUR’10
Broadcast Networks

Definition

(Reconfigurable) Broadcast Network $= (Q, M, \Delta, q_0)$ with $\Delta \subseteq Q \times \{br(m), rec(m) \mid m \in M\} \times Q$.

- Arbitrarily many agents at the start
- One step = an agent broadcasts a message $m$, some (arbitrary subset of) other agents receive it.

Problems

**Cover**: Is there a run in which an agent reaches $q_f$?

**Target**: Is there a run in which all agents reach $q_f$ simultaneously?

Both problems are decidable in $\text{PTIME}^{12}$.

---

$^1$Delzanno, Sangnier, Zavattaro, CONCUR’10

Adding registers

Each agent now has local registers $\square_1, \ldots, \square_r$, containing values in $\mathbb{N}$. 
Adding registers

Each agent now has local registers $\square_1, \ldots, \square_r$, containing values in $\mathbb{N}$.
Adding registers

Each agent now has local registers $\Box_1, \ldots, \Box_r$, containing values in $\mathbb{N}$. 

\begin{align*}
\text{rec}(b, = \Box_1) \\
\text{br}(a, \Box_1) \\
\text{rec}(a, \downarrow \Box_2) \\
\text{rec}(c, \neq \Box_2) \\
\text{rec}(b, = \Box_2)
\end{align*}
Adding registers

Each agent now has local **registers** $\square_1, \ldots, \square_r$, containing values in $\mathbb{N}$.

Nicolas Waldburger
Parameterized Verification of BNRA

6 / 42
Adding registers

Each agent now has local registers $\square_1, \ldots, \square_r$, containing values in $\mathbb{N}$.
Each agent now has local registers $\square_1, \ldots, \square_r$, containing values in $\mathbb{N}$.  

---

3Delzanno, Sangnier, Traverso, RP’13
Broadcast Networks of Register Automata (BNRA)\textsuperscript{3}

Each agent now has local registers $\square_1, \ldots, \square_r$, containing values in $\mathbb{N}$. Initially, all registers of all agents contain distinct values.

\textsuperscript{3}Delzanno, Sangnier, Traverso, RP’13
Broadcast Networks of Register Automata (BNRA)\(^3\)

Each agent now has local \textit{registers} \(\square_1, \ldots, \square_r\), containing values in \(\mathbb{N}\).
\textbf{Initially, all registers of all agents contain distinct values.}

A message is a pair \((m, v) \in \mathcal{M} \times \mathbb{N}\). An agent can:
\begin{itemize}
  \item Broadcast a message symbol along with a register value: \(\textbf{br}(m, r_i)\)
\end{itemize}

\[^3\text{Delzanno, Sangnier, Traverso, RP'13}\]
Broadcast Networks of Register Automata (BNRA)\(^3\)

Each agent now has local registers \(\square_1, \ldots, \square_r\), containing values in \(\mathbb{N}\). Initially, all registers of all agents contain distinct values.

A message is a pair \((m, v) \in M \times \mathbb{N}\). An agent can:

- Broadcast a message symbol along with a register value: \(\text{br}(m, r_i)\)

- Receive a message of a given symbol \(m\): \(\text{rec}(m, \text{op})\), with \(\text{op}\) one of the following:
  - store the value in register \(\square_i\): \(\downarrow \square_i\),
  - test it for equality with register \(\square_i\): \(= \square_i, \neq \square_i\)
  - or discard the received value: \(*\).

---

\(^3\)Delzanno, Sangnier, Traverso, RP’13
Broadcast Networks of Register Automata (BNRA)\(^3\)

Each agent now has local *registers* \(\square_1, \ldots, \square_r\), containing values in \(\mathbb{N}\). \textbf{Initially, all registers of all agents contain distinct values.}\n
A message is a pair \((m, v) \in M \times \mathbb{N}\). An agent can:

- Broadcast a message symbol along with a register value: \texttt{br}(m, r_i)

- Receive a message of a given symbol \(m\): \texttt{rec}(m, \text{op}), with \text{op} one of the following:
  - store the value in register \(\square_i\): \(\downarrow \square_i\),
  - test it for equality with register \(\square_i\): \(= \square_i, \neq \square_i\)
  - or discard the received value: \(*\).

This model was first defined in \(^3\), where the authors prove that this model is undecidable if several values can be appended to the same message. They also wrongly claimed that, with one value per message (our model), coverability is decidable in \(\text{PSPACE}\).

\(^3\)Delzanno, Sangnier, Traverso, RP’13
We can check that messages received come from the same agent. Here a word in $a b a^* c$ must be received with all messages having the same value:
Things we can do

We can check that a sequence of messages we sent was received. Here the top branch sends $a \ b$, the bottom branch receives $a \ b$ and sends an acknowledgement.
Parameterized verification principles

Our parameterized problems

**Cover**: Is there a number of agents \( n \), a run over \( n \) agents in which an agent reaches \( q_f \)?

**Target**: Is there a number of agents \( n \), a run over \( n \) agents in which all agents reach \( q_f \) simultaneously?
Parameterized verification principles

Our parameterized problems

**Cover**: Is there a *number of agents* $n$, a run over $n$ agents in which an *agent* reaches $q_f$?

**Target**: Is there a *number of agents* $n$, a run over $n$ agents in which all *agents* reach $q_f$ *simultaneously*?

- Unlimited supply of agents.
- For **Cover**, we can add as many agents as we need at no cost.
Parameterized verification principles

Our parameterized problems

**Cover**: Is there a *number of agents* $n$, a run over $n$ agents in which an agent reaches $q_f$?

**Target**: Is there a *number of agents* $n$, a run over $n$ agents in which all agents reach $q_f$ *simultaneously*?

- Unlimited supply of agents.
- For Cover, we can add as many agents as we need at no cost.

Copycat principle

Given a run $\rho$, we can construct a run made of many copies of $\rho$ running in parallel.
Parameterized verification principles

Our parameterized problems

**Cover**: Is there a number of agents $n$, a run over $n$ agents in which an agent reaches $q_f$?

**Target**: Is there a number of agents $n$, a run over $n$ agents in which all agents reach $q_f$ simultaneously?

- Unlimited supply of agents.
- For Cover, we can add as many agents as we need at no cost.

Copycat principle

Given a run $\rho$, we can construct a run made of many copies of $\rho$ running in parallel.

Main theorem

Cover is decidable for BNRA.
Signature BNRA

An agent never modifies its first register, and only broadcasts with the value of its first signature. Other registers are used to store and compare values received.

The first register acts as an identity with which agents sign their messages.
Signature BNRA

An agent never modifies its first register, and only broadcasts with the value of its first signature. Other registers are used to store and compare values received.

The first register acts as an identity with which agents sign their messages.

Messages received with the same value come from the same agent.
Tree witnesses for \textsc{Cover}

\[
\begin{array}{c}
\text{br}(m_0, v_0) \\
\text{rec}(m_1, v_1) \text{rec}(m_2, v_2) \text{rec}(m_3, v_1)
\end{array}
\frac{\text{br}(m, v_0)}{\text{rec}(m_1, v_1) \text{rec}(m_2, v_2) \text{rec}(m_3, v_1)} \rightarrow q_f
\]
Tree witnesses for $\text{COVER}$

\[
\frac{\text{br}(m_0, v_0)}{\text{rec}(m_1, v_1) \text{rec}(m_2, v_2) \text{rec}(m_3, v_1)} \frac{\text{br}(m, v_0)}{q_f}
\]
Tree witnesses for \textsc{Cover}

$$\begin{align*}
\text{br}(m_0, v_0) & \quad \text{rec}(m_1, v_1) \quad \text{rec}(m_2, v_2) \quad \text{rec}(m_3, v_1) \\
\text{br}(m, v_0) & \quad \text{br}(m_1, v_1) \quad \text{br}(m_3, v_1) \\
\text{rec}(m_0, v_0) & \quad \text{rec}(m_2, v_2) \\
\text{br}(m_1, v_2) & \quad \text{br}(m_2, v_2) \\
\text{rec}(m_1, v_1) & \quad \text{rec}(m_1, v_1)
\end{align*}$$
Tree witnesses for \textsc{Cover}

\[
\begin{align*}
\text{br}(m_0, v_0) & \quad \text{rec}(m_1, v_1) \quad \text{rec}(m_2, v_2) \quad \text{rec}(m_3, v_1) \\
\text{br}(m, v_0) & \quad q_f
\end{align*}
\]
Decidability of \textsc{Cover} for signature BNRA

Tree witnesses for \textsc{Cover}

\[ \text{br}(m_0,v_0) \] \[ \text{rec}(m_1,v_1) \text{rec}(m_2,v_2) \text{rec}(m_3,v_1) \] \[ \text{br}(m,v_0) \] \[ q_f \]

\[ m_1m_3 \]

\[ \text{br}(m_1,v_1) \] \[ \text{rec}(m_0,v_0) \] \[ \text{rec}(m_2,v_2) \]

\[ m_2 \]

\[ \text{br}(m_1,v_2) \] \[ \text{br}(m_2,v_2) \] \[ \text{rec}(m_1,v_1) \]
Tree witnesses for \textsc{Cover}

Lemma

There is a tree witness if and only if the instance of \textsc{Cover} is positive.

For decidability, we need to bound the size of well-chosen tree witnesses.
Tree witnesses for \textsc{Cover}

**Lemma**

There is a tree witness if and only if the instance of \textsc{Cover} is positive.

For decidability, we need to bound the size of well-chosen tree witnesses.
Branch reduction

**Lemma**

If a node labelled $w$ has a descendant labelled $w'$ with $w$ a subword of $w'$ (written $w \preceq w'$) then the tree can be shortened.

![Diagram](image)
Branch reduction

**Lemma**

If a node labelled \( w \) has a descendant labelled \( w' \) with \( w \) a subword of \( w' \) (written \( w \preceq w' \)) then the tree can be shortened.
Branch reduction

**Lemma**

If a node labelled $w$ has a descendant labelled $w'$ with $w$ a subword of $w'$ (written $w \preceq w'$) then the tree can be shortened.

\[
\begin{array}{c}
\text{If a node labelled } w \text{ has a descendant labelled } w' \text{ with } w \text{ a subword of } w' \\
\text{(written } w \preceq w') \text{ then the tree can be shortened.}
\end{array}
\]
Branch reduction

**Lemma**

If a node labelled $w$ has a descendant labelled $w'$ with $w$ a subword of $w'$ (written $w \preceq w'$) then the tree can be shortened.

After iterating this shortening procedure, we end up with a tree in which a node labelled $w$ has no descendant labelled $w' \succeq w$. 
Decidability of \textsc{Cover} for signature \textsc{BNRA}

Well quasi-orders

This order on $\mathbb{N}^2$ is a well quasi-order: every bad sequence is finite.

Higman’s lemma

For a finite alphabet $\Sigma$, the subword order $\preceq$ is a well quasi-order over $\Sigma^*$. In other words, there is no infinite bad sequence $w_0, w_1, w_2, \ldots$ in $\Sigma^*$, i.e., such that $w_i \preceq \not\preceq w_j$ for all $i < j$. 

Nicolas Waldburger
Parameterized Verification of \textsc{BNRA}
Well quasi-orders

You cannot pick a point higher on both coordinates than one of the previous points.

\[(4, 3)\]

This order on \(\mathbb{N}^2\) is a well quasi-order: every bad sequence is finite.

Higman's lemma

For a finite alphabet \(\Sigma\), the subword order \(\preceq\) is a well quasi-order over \(\Sigma^*\).

In other words, there is no infinite bad sequence \(w_0, w_1, w_2, \ldots\) in \(\Sigma^*\), i.e., such that \(w_i \preceq \neg w_j\) for all \(i < j\).
Well quasi-orders

You cannot pick a point higher on both coordinates than one of the previous points.

(4, 3)
Well quasi-orders

You cannot pick a point higher on both coordinates than one of the previous points.

\((4, 3) \rightarrow (6, 3)\)
Well quasi-orders

You cannot pick a point higher on both coordinates than one of the previous points.

\[(4, 3) \rightarrow (2, 4)\]
Well quasi-orders

You cannot pick a point higher on both coordinates than one of the previous points.

\[(4, 3) \rightarrow (2, 4) \rightarrow (7, 1)\]
Well quasi-orders

You cannot pick a point higher on both coordinates than one of the previous points.

\[(4, 3) \rightarrow (2, 4) \rightarrow (7, 1) \rightarrow (0, 5)\]
Well quasi-orders

You cannot pick a point higher on both coordinates than one of the previous points.

\((4, 3) \rightarrow (2, 4) \rightarrow (7, 1) \rightarrow (0, 5) \rightarrow (8, 0)\)
Well quasi-orders

You cannot pick a point higher on both coordinates than one of the previous points.

$$(4, 3) \rightarrow (2, 4) \rightarrow (7, 1) \rightarrow (0, 5) \rightarrow (8, 0) \rightarrow (3, 1)$$
Well quasi-orders

You cannot pick a point higher on both coordinates than one of the previous points.

(4, 3) \rightarrow (2, 4) \rightarrow (7, 1) \rightarrow (0, 5) \rightarrow (8, 0) \rightarrow (3, 1) \rightarrow (1, 2)
Well quasi-orders

You cannot pick a point higher on both coordinates than one of the previous points.

\[(4, 3) \rightarrow (2, 4) \rightarrow (7, 1) \rightarrow (0, 5) \rightarrow (8, 0) \rightarrow (3, 1) \rightarrow (1, 2) \rightarrow (0, 0)\]
Well quasi-orders

You cannot pick a point higher on both coordinates than one of the previous points.

\[(4, 3) \rightarrow (2, 4) \rightarrow (7, 1) \rightarrow (0, 5) \rightarrow (8, 0) \rightarrow (3, 1) \rightarrow (1, 2) \rightarrow (0, 0)\]

This order on \( \mathbb{N}^2 \) is a well quasi-order: every bad sequence is finite.
Well quasi-orders

(4, 3) → (2, 4) → (7, 1) → (0, 5) → (8, 0) → (3, 1) → (1, 2) → (0, 0)

This order on \( \mathbb{N}^2 \) is a well quasi-order: every bad sequence is finite.

Higman’s lemma

For a finite alphabet \( \Sigma \), the subword order \( \preceq \) is a well quasi-order over \( \Sigma^* \). In other words, there is no infinite bad sequence \( w_0, w_1, w_2, \ldots \) in \( \Sigma^* \), i.e., such that \( w_i \not\preceq w_j \) for all \( i < j \).
If \( q_f \) can be covered, then there is a witness of the execution of the form:

Every branch forms a bad sequence. Because \( \preceq \) is a well quasi-order, we know that every branch of the tree is finite...
Decidability of \texttt{Cover} for signature BNRA

Back to the trees

If $q_f$ can be covered, then there is a witness of the execution of the form:

$$u_0 \xrightarrow{} q_f \xrightarrow{} \text{word of } M^*$$

Every branch forms a bad sequence. Because $\preceq$ is a well quasi-order, we know that every branch of the tree is finite... Not useful!
We need a bound on the size of the tree, so that we can iterate over every possible such tree in finite time.
Bounds on the length of sequences

Obviously, there is no general bound on the length of a bad sequence: the sequence $m^k, m^{k-1}, \ldots, m$ with $m \in \mathcal{M}$ is a bad sequence of length $k$. 

---

4 Schmitz, Schnoebelen, ICALP’11

Nicolas Waldburger
Parameterized Verification of BNRA
Bounds on the length of sequences

Obviously, there is no general bound on the length of a bad sequence: the sequence \( m^k, m^{k-1}, \ldots, m \) with \( m \in \mathcal{M} \) is a bad sequence of length \( k \). However, there is a bound if we have some control on the size of the elements of the sequence:

**Length function theorem**

Given a finite alphabet \( \Sigma \) and a computable function \( F : \mathbb{N} \to \mathbb{N} \), there is a computable bound \( B \) such that every sequence \( (w_i)_{i \in \mathbb{N}} \) over \( \Sigma \) such that

1. \( w_i \not\leq w_j \) for all \( i < j \) (bad sequence) and
2. \( |w_i| \leq F(i) \) for all \( i \)

has length at most \( B \).

---

\(^4\)Schmitz, Schnoebelen, ICALP’11
Applying the length function theorem

Consider a branch of a tree of minimal size:

\[ u_0 \rightarrow u_1 \rightarrow u_2 \rightarrow \ldots \rightarrow u_n \]

We need to bound the number of steps that an agent has to perform to perform a task: we need a function \( f \) such that

\[ |u_i| \leq f(|w_i|) \]
Applying the length function theorem

Consider a branch of a tree of minimal size:

\[ u_0 \xrightarrow{} q_f \]
\[ w_1 \quad |w_1| \leq |u_0| \]
\[ u_1 \]
\[ w_2 \quad |w_2| \leq |u_1| \]
\[ u_2 \]
\[ w_3 \quad |w_3| \leq |u_2| \]
\[ \quad \ldots \]
\[ w_n \quad |w_n| \leq |u_{n-1}| \]
\[ u_n \]

We need to bound the number of steps that an agent has to perform to perform a task: we need a function \( f \) such that \( |u_i| \leq f(|w_i|) \).
Bounding local runs

By induction on the number of *active* registers. Register $\square_i$ is *active* when some storing action $\downarrow \square_i$ is performed.

<table>
<thead>
<tr>
<th>$q$</th>
<th>$q'$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_1$</td>
<td>$v'_1$</td>
</tr>
<tr>
<td>$v_2$</td>
<td>$v'_2$</td>
</tr>
<tr>
<td>$v_3$</td>
<td>$v'_3$</td>
</tr>
<tr>
<td>$v_4$</td>
<td>$v'_4$</td>
</tr>
</tbody>
</table>
Bounding local runs

By induction on the number of *active* registers. Register $\square_i$ is *active* when some storing action $\downarrow \square_i$ is performed.

0 active registers

\[
\begin{array}{c|c}
q & q' \\
\hline
v_1 & v_1' \\
v_2 & v_2' \\
v_3 & v_3' \\
v_4 & v_4'
\end{array}
\]
Bounding local runs

By induction on the number of active registers. Register $\square_i$ is active when some storing action $\downarrow \square_i$ is performed.

0 active registers

<table>
<thead>
<tr>
<th>$q$</th>
<th>$q''$</th>
<th>$q''$</th>
<th>$q'$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_1$</td>
<td></td>
<td></td>
<td>$v'_1$</td>
</tr>
<tr>
<td>$v_2$</td>
<td></td>
<td></td>
<td>$v'_2$</td>
</tr>
<tr>
<td>$v_3$</td>
<td></td>
<td></td>
<td>$v'_3$</td>
</tr>
<tr>
<td>$v_4$</td>
<td></td>
<td></td>
<td>$v'_4$</td>
</tr>
</tbody>
</table>
Bounding local runs

By induction on the number of *active* registers.
Register \( \Box_i \) is *active* when some storing action \( \downarrow \Box_i \) is performed.

0 active registers
Bounding local runs

By induction on the number of active registers. Register $\square_i$ is active when some storing action $\downarrow \square_i$ is performed.

1 active registers

![Diagram showing a local run with 1 active register]

$|Q| > |Q'|$
Bounding local runs

By induction on the number of *active* registers. Register $\Box_i$ is *active* when some storing action $\downarrow \Box_i$ is performed.

1 active registers

<table>
<thead>
<tr>
<th>$q$</th>
<th>$q''$</th>
<th>$q''$</th>
<th>$q'$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_1$</td>
<td></td>
<td></td>
<td>$v'_1$</td>
</tr>
<tr>
<td>$v_2$</td>
<td></td>
<td></td>
<td>$v'_2$</td>
</tr>
<tr>
<td>$v_3$</td>
<td></td>
<td></td>
<td>$v'_3$</td>
</tr>
<tr>
<td>$v_4$</td>
<td></td>
<td></td>
<td>$v'_4$</td>
</tr>
</tbody>
</table>

$|Q| > |Q| + 1$
Bounding local runs

By induction on the number of *active* registers. Register $\Box_i$ is *active* when some storing action $\downarrow \Box_i$ is performed.

1 active registers

<table>
<thead>
<tr>
<th>$q$</th>
<th>$q''$</th>
<th>$q'$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_1$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$v_2$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$v_3$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$v_4$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$\Delta = |Q| - 1$
Bounding local runs

By induction on the number of *active* registers. Register $\square_i$ is *active* when some storing action $\downarrow \square_i$ is performed.

1 active registers

\[
\begin{array}{ccccccc}
q & v_1 & v_2 & v_3 & v_4 & q' \\
q' & v'_1 & v'_2 & v'_3 & v'_4 & \\
\end{array}
\]

\[
|Q|(|Q| + 1)
\]
Bounding local runs

By induction on the number of active registers. Register \( \square_i \) is active when some storing action \( \downarrow \square_i \) is performed.

1 active registers

\[
\begin{array}{c}
q \quad q'' \\
q' \quad q'' \\
q'' \\
q'
\end{array}
\]

\( v_1 \)
\( v_2 \)
\( v_3 \)
\( v_4 \)

\( v_1' \)
\( v_2' \)
\( v_3' \)
\( v_4' \)

\( Q \quad (|Q| + 1) \)
Bounding local runs

By induction on the number of active registers. Register $\square_i$ is active when some storing action $\downarrow \square_i$ is performed.

1 active registers

<table>
<thead>
<tr>
<th>$q$</th>
<th>$q''$</th>
<th>$q'$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_1$</td>
<td>$\square$</td>
<td>$\square$</td>
</tr>
<tr>
<td>$v_2$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$v_3$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$v_4$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Bounding local runs

By induction on the number of active registers. Register $\square_i$ is active when some storing action $\downarrow \square_i$ is performed.

$m$ active registers

Say we can reduce any local run with $< m$ active registers of length $\geq K$. 

$q$

$v_1$
$v_2$
$v_3$
$v_4$

$K$

$q'$

$v'_1$
$v'_2$
$v'_3$
$v'_4$
Bounding local runs

By induction on the number of active registers.
Register $\square_i$ is active when some storing action $\downarrow \square_i$ is performed.

$m$ active registers

Say we can reduce any local run with $< m$ active registers of length $\geq K$.

$q$

\[ q' \]

$\Delta$

\[ K \]

\[ K \]

\[ \geq |\Delta|^K \]
Bounding local runs

By induction on the number of *active* registers. Register $\square_i$ is *active* when some storing action $\downarrow \square_i$ is performed.

$m$ active registers

Say we can reduce any local run with $< m$ active registers of length $\geq K$.

Parameterized Verification of BNRA
Bounding the tree

**Lemma**

There is a function $\varphi$ such that if an agent has a local run between two local configurations, then it has one such local run of length $\leq \varphi(|\Delta|, r)$.

$\Delta$: set of transitions  
$r$: number of registers.

**Corollary**

If an agent has a local run that broadcasts $w$, then it has one such local run of length $\leq |w| \varphi(|\Delta|, r)$. 
Decidability of \textit{Cover} for signature BNRA

Bounding the branches

\begin{align*}
  u_0, v_0 & \rightarrow q_f \\
  w_1 & \quad |w_1| \leq |u_0| \\
  u_1, v_1 & \rightarrow w_2 \\
  |w_2| & \leq |u_1| \\
  u_2, v_2 & \rightarrow w_3 \\
  |w_3| & \leq |u_2| \\
  \vdots & \\
  w_n & \quad |w_n| \leq |u_{n-1}| \\
  u_n, v_n &
\end{align*}
Bounding the branches

- $u_0, v_0$ connected to $q_f$
- $|w_1| \leq |u_0|$, $|w_1| \leq \phi(|\Delta|, r)$
- $u_1, v_1$
- $|w_2| \leq |u_1|$, $|w_2| \leq \phi(|\Delta|, r)^2$
- $u_2, v_2$
- $|w_3| \leq |u_2|$, $|w_3| \leq \phi(|\Delta|, r)^3$
- $\ldots$
- $w_n \leq |u_{n-1}|$, $|w_n| \leq \phi(|\Delta|, r)^n$
Bounding the branches

\[ \begin{align*}
u_0, v_0 &\rightarrow q_f \quad |u_0| \leq \phi(|\Delta|, r) \\
w_1 &\quad |w_1| \leq |u_0| \quad |w_1| \leq \phi(|\Delta|, r) \\
u_1, v_1 &\quad |u_1| \leq |w_1| \phi(|\Delta|, r) \leq \phi(|\Delta|, r)^2 \\
w_2 &\quad |w_2| \leq |u_1| \quad |w_2| \leq \phi(|\Delta|, r)^2 \\
u_2, v_2 &\quad |u_2| \leq |w_2| \phi(|\Delta|, r) \leq \phi(|\Delta|, r)^3 \\
w_3 &\quad |w_3| \leq |u_2| \quad |w_3| \leq \phi(|\Delta|, r)^3 \\
\vdots &\vdots \\
w_n &\quad |w_n| \leq |u_{n-1}| \quad |w_n| \leq \phi(|\Delta|, r)^n \\
u_n, v_n &
\end{align*} \]

Length function theorem: we obtain a computable bound \( B(|\Delta|, r) \) such that \( n \leq B(|\Delta|, r) \): \( B \) bounds the height of a witness tree for \textsc{Cover}!
Decidability and complexity

**Bounds**

We use the previous argument to bound (in well-chosen witness trees):

- the length of all branches,
- the size of every node,
- the maximal degree of the tree.

This bounds the total space needed to store such a tree.
Decidability and complexity

Bounds

We use the previous argument to bound (in well-chosen witness trees):

- the length of all branches,
- the size of every node,
- the maximal degree of the tree.

This bounds the total space needed to store such a tree.

We can enumerate all such trees in finite time, therefore

Theorem

The Cover problem for signature BNRA is decidable
Decidability and complexity

Bounds

We use the previous argument to bound (in well-chosen witness trees):

- the length of all branches,
- the size of every node,
- the maximal degree of the tree.

This bounds the total space needed to store such a tree.

We can enumerate all such trees in finite time, therefore

Theorem

The Cover problem for signature BNRA is decidable and in $F_{\omega^\omega}$.

The length function theorem in fact gives us a bound for the height of our trees that is a function in hyper-Ackermannian class $F_{\omega^\omega}$ of $|\Delta|$ and $r$. 
Agents can broadcast messages with values that they received before.

An agent $a$ now receives two types of messages:

- Messages with values that belonged to other agents initially.
- Messages with values that $a$ had initially, that it had broadcast and that someone else stored and broadcasts.
Observation

An agent may do this:

\[
\text{br}(a, \square_1) \text{ br}(b, \square_1) \text{ rec}(c, = \square_1) \text{ rec}(d, = \square_1) \text{ rec}(c, = \square_1)
\]
Observation

An agent may do this:

\[ \text{br}(a, \square_1) \text{br}(b, \square_1) \text{rec}(c, = \square_1) \text{rec}(d, = \square_1) \text{rec}(c, = \square_1) \]

To witness that this is feasible, we must exhibit:

- A run that, after receiving \((a, v)(b, v)\), broadcasts \((c, v)\), and
- A run that, after receiving \((a, v)(b, v)(c, v)^*\), broadcasts \((d, v)\).
Decidability of Cover in the general case

Observation

An agent may do this:

\[ \text{br}(a, \square_1) \text{ br}(b, \square_1) \text{ rec}(c, = \square_1) \text{ rec}(d, = \square_1) \text{ rec}(c, = \square_1) \]

To witness that this is feasible, we must exhibit:

1. A run that, after receiving \((a, v)(b, v)\), broadcasts \((c, v)\), and
2. A run that, after receiving \((a, v)(b, v)(c, v)^*\), broadcasts \((d, v)\).

We add contract nodes labelled \(w \rightarrow m\) that witness a local run that, if it receives word \(w\) with value \(v\), can broadcast \((m, v)\).
Decidability of Cover in the general case

Our new tree witnesses

\[
\frac{\text{br}(m_0, v_0) \ \text{br}(m_1, v_0)}{\text{rec}(m_2, v_2) \ \text{rec}(m_3, v_0) \ \text{rec}(m_3, v_0)}
\]
Our new tree witnesses

\[
\begin{align*}
\text{br}(m_0, v_0) &\quad \text{br}(m_1, v_0) \\
\text{rec}(m_2, v_2) \text{rec}(m_3, v_0) \text{rec}(m_3, v_0)
\end{align*}
\]
Our new tree witnesses

\[
\begin{align*}
\text{br}(m_0, v_0) \quad \text{br}(m_1, v_0) \\
\text{rec}(m_2, v_2) \text{rec}(m_3, v_0) \text{rec}(m_3, v_0)
\end{align*}
\]
Our new tree witnesses

\[ \text{br}(m_0, v_0) \quad \text{br}(m_1, v_0) \]
\[ \text{rec}(m_2, v_2) \text{rec}(m_3, v_0) \text{rec}(m_3, v_0) \]

\[ m_0 m_1 \rightarrow m_3 \]

\[ \text{br}(m_3, v_0) \]
\[ \text{rec}(m_0, v_0) \text{rec}(m_1, v_0) \text{rec}(m_2, v_2) \]

\[ \text{br}(m_1, v_2) \quad \text{br}(m_2, v_2) \]
\[ \text{rec}(m_1, v_2) \]
Our new tree witnesses

\[ \text{br}(m_0, v_0) \quad \text{br}(m_1, v_0) \]

\[ \overset{\text{rec}(m_2, v_2) \text{rec}(m_3, v_0) \text{rec}(m_3, v_0)}{\rightarrow} \]

\[ m_0 m_1 \rightarrow m_3 \]

\[ \text{br}(m_3, v_0) \quad \overset{\text{rec}(m_0, v_0) \text{rec}(m_1, v_0) \text{rec}(m_2, v_2)}{\rightarrow} \quad \text{br}(m_1, v_2) \quad \overset{\text{br}(m_2, v_2)}{\rightarrow} \quad \text{br}(m_2, v_2) \]
Decidability of \textsc{Cover} in the general case

Branch reductions
Branch reductions

Decidability of \textsc{Cover} in the general case
Decidability of \textsc{Cover} in the general case

Branch reductions

\[ u', w_1 \leq w_2, \quad w_1 \rightarrow m, \quad u', v' \leq w_2 \]
Things are more complicated than before

Problem: The number of messages that a node must broadcast now depends on its \(w \rightarrow m\) children, and not just on its father.
Decidability of Cover in the general case

Rearranging our trees

\[ u_1 \rightarrow w_2 \rightarrow u_2 \rightarrow w_5 \rightarrow u_5 \]

\[ w_3 \rightarrow m_3 \rightarrow u_3 \]

\[ w_6 \rightarrow m_4 \rightarrow u_4 \rightarrow w_7 \rightarrow u_7 \]
Rearranging our trees

$u_3 \rightarrow m_3$

$w_3 \rightarrow m_3$

$w_2$

$u_2$

$w_5$

$u_5$

$w_4 \rightarrow m_4$

$w_4 \rightarrow m_4$

$w_6$

$u_1$

$w_6$

$u_6$

$w_7$

$u_7$
Rearranging the tree

**Definition**

The *altitude* of a node is

- 0 if it is the root
- its father’s altitude +1 if it is labelled $w \rightarrow m$
- its father’s altitude −1 if it is labelled $w$
Bounding the altitude

Let \( A \) be the maximal altitude in the tree, we follow a branch reaching it.

\[
\begin{align*}
|w_0| & \leq \psi(|\Delta|, r) \\
|w_1| & \leq \psi(|\Delta|, r) \\
|w_2| & \leq \psi(|\Delta|, r) \\
|w_{A-1}| & \leq \psi(|\Delta|, r) \\
|w_A| & \leq \psi(|\Delta|, r)
\end{align*}
\]
Bounding the altitude

Let $A$ be the maximal altitude in the tree, we follow a branch reaching it.
Bounding the altitude

Let $A$ be the maximal altitude in the tree, we follow a branch reaching it.

altitude

$|w_0| \leq \psi(|\Delta|, r)$

$|w_1| \leq \psi(|\Delta|, r)$

$|w_2| \leq \psi(|\Delta|, r)$

$|w_3| \leq \psi(|\Delta|, r)$

$|w_A| \leq \psi(|\Delta|, r)$

$\text{root}$
Bounding the altitude

Let $A$ be the maximal altitude in the tree, we follow a branch reaching it.

Let $A$ be the maximal altitude in the tree, we follow a branch reaching it.

$$|w_0| \leq \psi(|\Delta|, r)$$

$$|w_1| \leq \psi(|\Delta|, r)$$

$$|w_2| \leq \psi(|\Delta|, r)$$

$$|w_3| \leq \psi(|\Delta|, r)$$

$$|w_{A-1}| \leq \psi(|\Delta|, r)$$

$$|w_{A-2}| \leq \psi(|\Delta|, r)$$

$$|w_{A-3}| \leq \psi(|\Delta|, r)$$

$$|w_A| \leq \psi(|\Delta|, r)$$
Bounding the altitude

Let $A$ be the maximal altitude in the tree, we follow a branch reaching it.

- $|w_0| \leq \psi(|\Delta|, r)$
- $|w_1| \leq \psi(|\Delta|, r)^2$
- $|w_2| \leq \psi(|\Delta|, r)^3$
- $|w_3| \leq \psi(|\Delta|, r)^4$
- $|w_A| \leq \psi(|\Delta|, r)^{A+1}$
Bounding the altitude

We have bounds on the maximal altitude and the size of the root. Let $R$ be the size of the root, $-B$ the minimal altitude.

\begin{align*}
0 & \leq R \\
-1 & \leq R \\
-B + 3 & \leq R \\
-B + 2 & \leq R \\
-B + 1 & \leq R \\
-B & \leq R
\end{align*}
Bounding the altitude

We have bounds on the maximal altitude and the size of the root. Let $R$ be the size of the root, $-B$ the minimal altitude.

\[
\begin{align*}
-0 & \quad \text{root} \\
-1 & \\
-B+3 & \\
-B+2 & \\
-B+1 & \\
-B & \\
\end{align*}
\]
Bounding the altitude

We have bounds on the maximal altitude and the size of the root. Let $R$ be the size of the root, $-B$ the minimal altitude.

\[
\begin{align*}
|w_0| & \leq R \\
|w_{-1}| & \leq R \psi(|\Delta|, r) \\
|w_{-2}| & \leq R \psi(|\Delta|, r) \\
|w_{-3}| & \leq R \psi(|\Delta|, r) \\
|w_{1}| & \leq R \psi(|\Delta|, r) \\
|w_{B-1}| & \leq R \psi(|\Delta|, r) \\
|w_{B-2}| & \leq R \psi(|\Delta|, r) \\
|w_{B-3}| & \leq R \psi(|\Delta|, r) \\
|w_B| & \leq R \psi(|\Delta|, r)
\end{align*}
\]
Bounding the altitude

We have bounds on the maximal altitude and the size of the root. Let $R$ be the size of the root, $-B$ the minimal altitude.

Let $R$ be the size of the root, $-B$ the minimal altitude.

$\begin{align*}
|w_0| & \leq R \\
|w_{B-1}| & \leq R \\
|w_{B-2}| & \leq R \\
|w_{B-3}| & \leq R \\
\end{align*}$
Bounding the altitude

We have bounds on the maximal altitude and the size of the root. Let $R$ be the size of the root, $-B$ the minimal altitude.

- $|w_0| \leq R$
- $|w_1| \leq R\psi(|\Delta|, r)$
- $|w_{B-3}| \leq R\psi(|\Delta|, r)^{B-3}$
- $|w_{B-2}| \leq R\psi(|\Delta|, r)^{B-2}$
- $|w_{B-1}| \leq R\psi(|\Delta|, r)^{B-1}$
- $|w_B| \leq R\psi(|\Delta|, r)^B$
Decidability

We have obtained bounds of the height of our witness trees; from there, we can easily bound the space needed to store such trees.
Decidability

We have obtained bounds of the height of our witness trees; from there, we can easily bound the space needed to store such trees. We can simply enumerate witness trees, thus

**Theorem**

$\text{Cover}$ for BNRA is decidable (and in class $F_{\omega^\omega}$).
Decidability

We have obtained bounds of the height of our witness trees; from there, we can easily bound the space needed to store such trees. We can simply enumerate witness trees, thus

**Theorem**

COVER for BNRA is decidable (and in class $F_{\omega\omega}$).

By contrast,

**Theorem**

TARGET is undecidable for BNRA.
Complexity lower bound

A matching lower bound

**Theorem**

**Cover** in BNRA is \( F_{\omega^\omega} \)-hard, even in signature protocols with two registers per agent.
A matching lower bound

Theorem

COVER in BNRA is $\mathbf{F}_{\omega^\omega}$-hard, even in signature protocols with two registers per agent.

We proceed by reduction from lossy channel systems:

Theorem\textsuperscript{5}

Lossy channel system reachability is $\mathbf{F}_{\omega^\omega}$-hard.

\textsuperscript{5}Schnoebelen, Information Processing Letters '08
Lossy Channel Systems

Lossy Channel System = Transition system with FIFO memory + unreliable writes.

Reachable states:
- $w(a)$
- $w(c)$
- $r(b)$
- $r(a)$

Lossy channel system reachability asks if one can reach a given state. This problem is decidable but has very high complexity: it is $F\omega$-complete.
Lossy Channel Systems

Lossy Channel System = Transition system with FIFO memory + unreliable writes.

\[ w(a) \]
\[ w(c) \]
\[ w(b) \]
\[ r(b) \]
\[ r(a) \]

Reachable states \( w(a) \) \( w(c) \) \( r(b) \) \( r(a) \)
Lossy Channel Systems

Lossy Channel System = Transition system with FIFO memory + unreliable writes.
Lossy Channel Systems

Lossy Channel System = Transition system with FIFO memory + unreliable writes.

Lossy channel system reachability asks if one can reach a given state. This problem is decidable but has very high complexity: it is $\mathcal{F}\omega$-complete.
Lossy Channel Systems

Lossy Channel System = Transition system with FIFO memory + unreliable writes.

Parameterized Verification of BNRA
Lossy Channel Systems

Lossy Channel System = Transition system with FIFO memory + unreliable writes.
Lossy Channel Systems

Lossy Channel System = Transition system with FIFO memory + unreliable writes.

Lossy channel system reachability asks if one can reach a given state. This problem is decidable but has very high complexity: it is $\text{F}^{\omega}_\omega$-complete.
Lossy Channel Systems

Lossy Channel System = Transition system with FIFO memory + unreliable writes.

Reachable states $w(a) \rightarrow w(c) \rightarrow r(b) \rightarrow r(a) \rightarrow w(b) \rightarrow r(a)$.
Lossy Channel Systems

Lossy Channel System = Transition system with FIFO memory + unreliable writes.

Reachable states with transitions:

$w(a)$

$w(c)$

$r(b)$

$r(a)$

$w(b)$

$r(a)$

Complexity lower bound
Lossy Channel Systems

Lossy Channel System = Transition system with FIFO memory + unreliable writes.

Reachable states

\[ w(a) \quad w(c) \quad r(b) \quad r(a) \]

\[ w(b) \quad r(a) \]

Lossy channel system reachability asks if one can reach a given state. This problem is decidable but has very high complexity: it is \( \text{F}\omega\) -complete.
Lossy Channel Systems

Lossy Channel System = Transition system with FIFO memory + unreliable writes.

Lossy channel system reachability asks if one can reach a given state. This problem is decidable but has very high complexity: it is $F_{\omega\omega}$-complete.
Encoding Lossy Channel Systems in BNRA

We simulate a lossy channel system through a chain of agents that each apply a transition. Each agent stores:

- An identifier for itself
- Its predecessor’s identifier

$q_0, \epsilon \quad q_1, w_1 \quad \cdots \quad q_n, w_n$
Encoding write transitions of Lossy Channel Systems

Gadget for a transition $q \xrightarrow{\text{write}(a)} q'$ of the lossy channel system
Encoding read transitions of Lossy Channel Systems

Gadget for a transition $q \xrightarrow{\text{read}(a)} q'$ of the lossy channel system
Theorem

\textsc{Cover} in BNRA is \( F_{\omega^\omega} \)-complete.
Summary of complexity results

Theorem

$\text{COVER}$ in BNRA is $F_{\omega}^{\omega}$-complete.

Theorem

$\text{COVER}$ for BNRA with one register per agent is NP-complete.
Thank you for your attention!
Turning the communication graph into a tree

Tasks:
- $\epsilon_7 \rightarrow a, b, c, d_7 \rightarrow a, d$
- $a_7 \rightarrow c, a d_7 \rightarrow q, f$
- $a$ with same value

Nicolas Waldburger
Parameterized Verification of BNRA
Turning the communication graph into a tree

Diagram:

- Node X
- Node Y
- Node Z
- Edge a from X to Y
- Edge b from X to Z
- Edge a' from Y to X

Tasks:
- ϵ

Parameterized Verification of BNRA
Turning the communication graph into a tree
Turning the communication graph into a tree

Tasks:
- $\epsilon \rightarrow a$, $b$, $c$
- $a \rightarrow c$, $d$
- $a \rightarrow q$, $f$

Nicolas Waldburger
Parameterized Verification of BNRA
41 / 42
Turning the communication graph into a tree

Tasks: $\epsilon \xrightarrow{} a$, $b$, $c \xrightarrow{} a$, $d$.

Tasks: $a \xrightarrow{} c$, $d$, $q_f \xrightarrow{} f$, $a$.

Nicolas Waldburger
Parameterized Verification of BNRA
Turning the communication graph into a tree

Tasks:
- $\epsilon \mapsto a$
- $b, c \mapsto ad$

Tasks:
- $a \mapsto c$
- $ad \mapsto q_f$

$ad = \text{receive } a \text{ then } d \text{ with same value}$

Tasks:
- $a \mapsto b$

Nicolas Waldburger
Parameterized Verification of BNRA
Turning the communication graph into a tree

Tasks:

1. $\epsilon \mapsto a$
2. $b, c \mapsto ad$
3. $a \mapsto c$
4. $a \mapsto d$
5. $ad \mapsto q_f$

Tasks: $a \mapsto b$

Tasks: $a \mapsto c$

Tasks: $ad \mapsto q_f$