Parameterized verification of Broadcast networks of Register automata

Nicolas Waldburger
joint work with Lucie Guillou and Corto Mascle

ANR PaVeDyS

January 16th, 2024

To be published at FoSSaCS'24

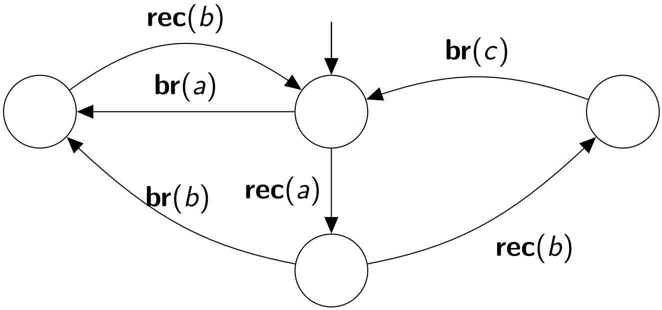
1. Broadcast networks
   - Basic model
   - With registers

2. Signature BNRA
   - Well quasi-orders
   - Decidability proof

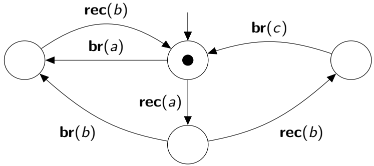
1. Broadcast networks
   - Basic model
   - With registers

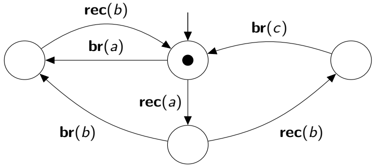
2. Signature BNRA
   - Well quasi-orders
   - Decidability proof

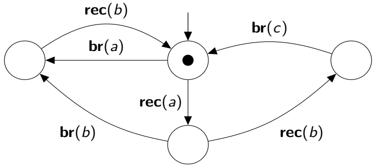
Broadcast networks



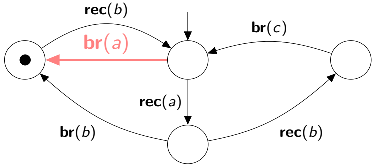
Broadcast networks

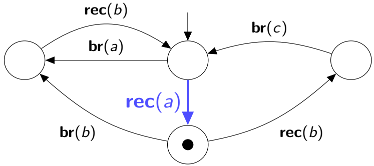


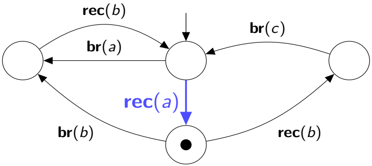




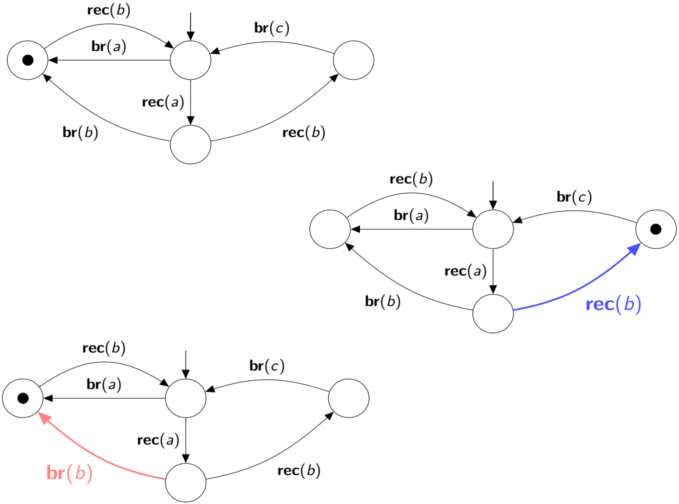
Broadcast networks



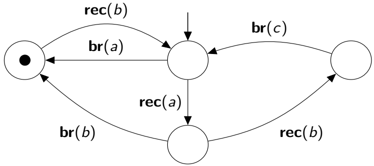


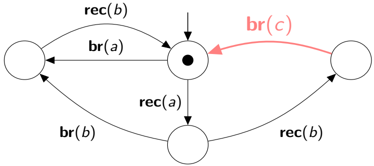


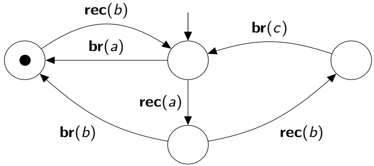
Broadcast networks



Broadcast networks







Broadcast Networks

Definition[1]

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

Broadcast Networks

Definition[1]

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

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

Broadcast Networks

Definition[1]

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

- 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 PTIME[1,2].

[1] Delzanno, Sangnier, Zavattaro, CONCUR'10
[2] Fournier, PhD thesis, 2015

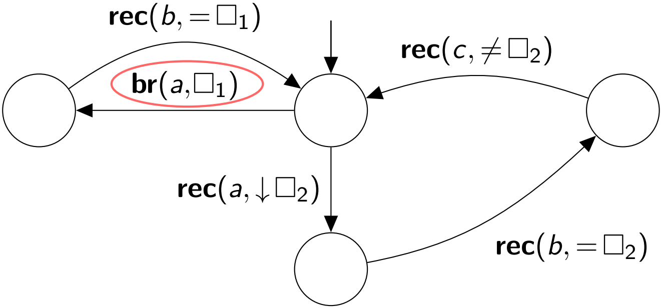
1. Broadcast networks
   - Basic model
   - With registers
2. Signature BNRA
   - Well quasi-orders
   - Decidability proof

Registers

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

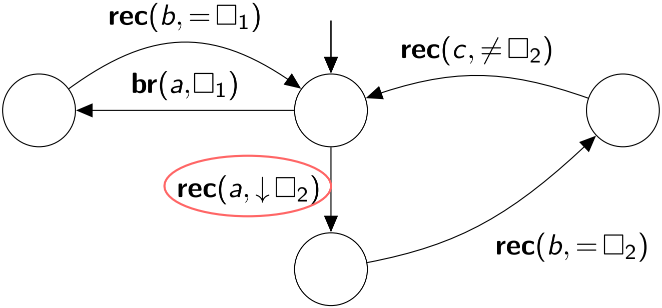
Registers

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



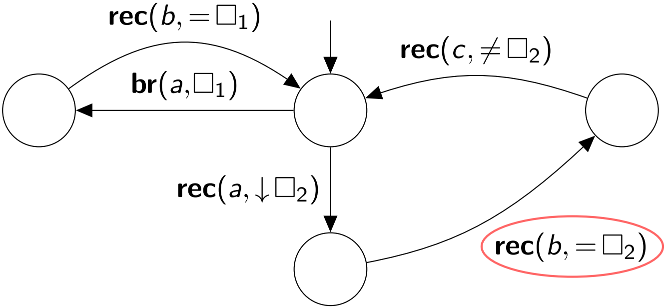
Registers

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



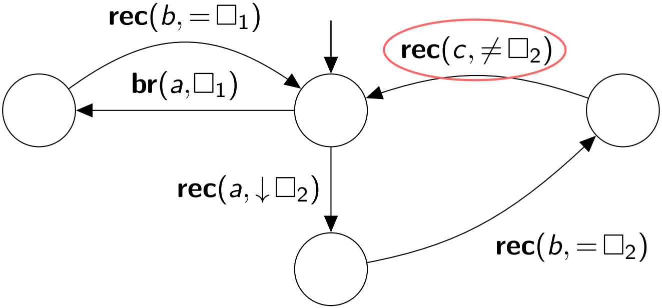
Registers

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



Registers

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



Broadcast Networks of Register Automata (BNRA)[3]

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

[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 $\mathbb{N}$.
Initially, all registers of all agents contain distinct values.

[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 $\mathbb{N}$.
Initially, all registers of all agents contain distinct values.

Messages also contain values: $(m, v) \in M \times \mathbb{N}$. An agent can:

- Broadcast a message with a register value $\mathbf{br}(m, r_i)$

[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 $\mathbb{N}$.
Initially, all registers of all agents contain distinct values.

Messages also contain values: $(m, v) \in M \times \mathbb{N}$. An agent can:

- Broadcast a message with a register value $\mathbf{br}(m, r_i)$
- Receive messages $\mathbf{rec}(m, r_i, op)$, with *op* either
  - store the value $\downarrow$,
  - test it for equality $=, \neq$
  - or do nothing $*$.

[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 $\mathbb{N}$.
Initially, all registers of all agents contain distinct values.

Messages also contain values: $(m, v) \in M \times \mathbb{N}$. An agent can:

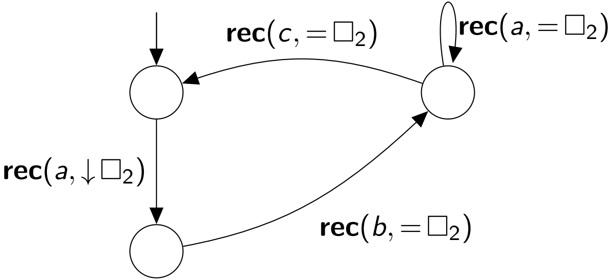
- Broadcast a message with a register value $\mathbf{br}(m, r_i)$
- Receive messages $\mathbf{rec}(m, r_i, op)$, with *op* either
  - store the value $\downarrow$,
  - test it for equality $=, \neq$
  - or do nothing $*$.

Remark: the model where one allows to send two messages per broadcast is undecidable[3].

[3] Delzanno, Sangnier, Traverso, RP'13

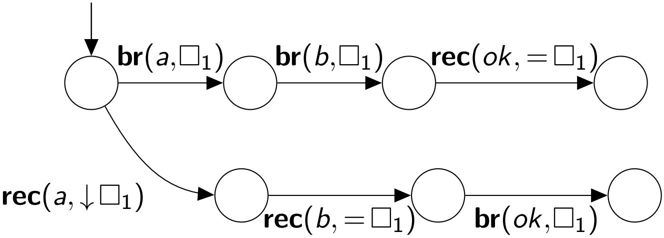
Things we can do

We can check that a sequence of messages all come from the same agent.



Things we can do

We can check that a sequence of messages we sent was received.



Parameterized verification principles

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

Parameterized verification principles

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

Copycat principle

Given a run ρ, we can construct a run made of many copies of ρ running in parallel.

Main theorem

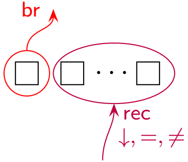
COVER is decidable for BNRA.

Signature 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.

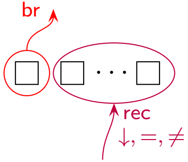


Signature BNRA

Signature BNRA

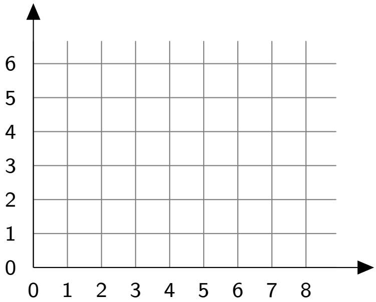
An agent never modifies its first register, and only broadcasts with the value of its first signature.
Other registers are used to store and compare values received.

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

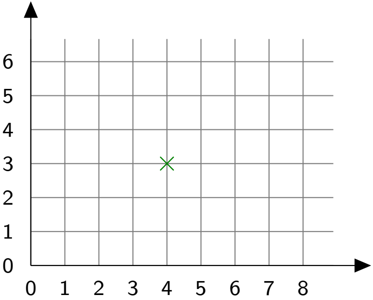


Messages received with the same value come from the same agent.

Well quasi-orders

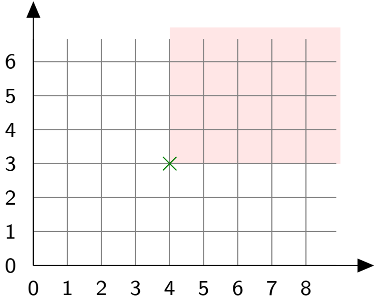


Well quasi-orders



$(4, 3)$

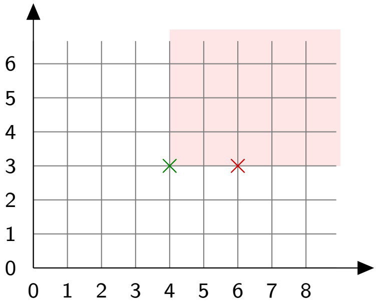
Well quasi-orders



$(4, 3)$

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

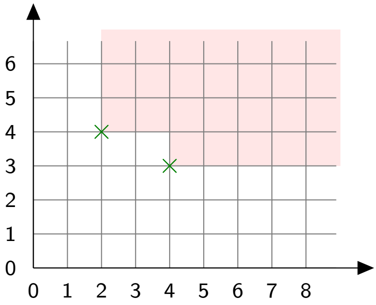
Well quasi-orders



$(4,3) \rightarrow (6,3)$

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

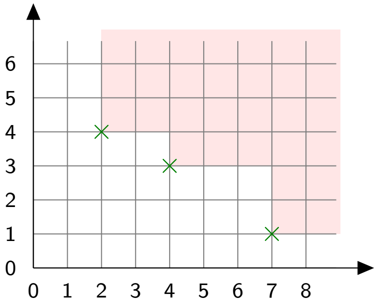
Well quasi-orders



$(4,3) \to (2,4)$

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

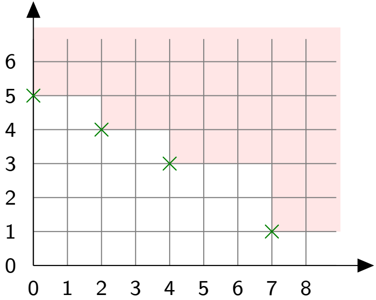
Well quasi-orders



$(4, 3) \rightarrow (2, 4) \rightarrow (7, 1)$

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

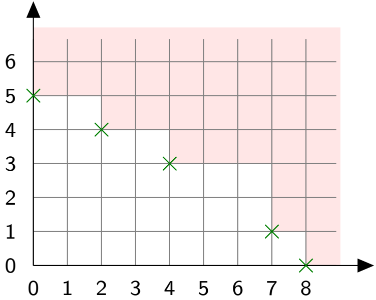
Well quasi-orders



$(4,3) \rightarrow (2,4) \rightarrow (7,1) \rightarrow (0,5)$

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

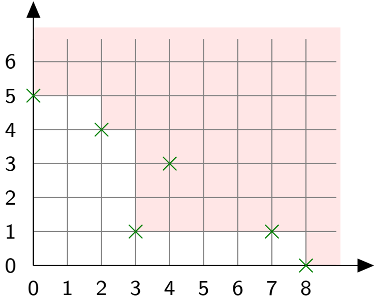
Well quasi-orders



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

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

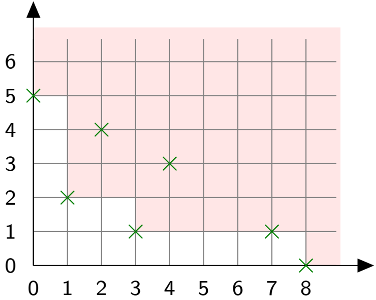
Well quasi-orders



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

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

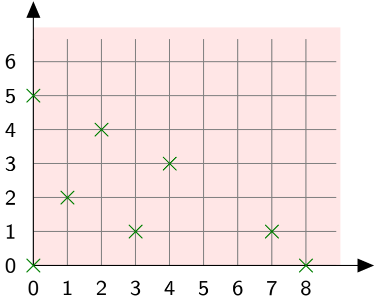
Well quasi-orders



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

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

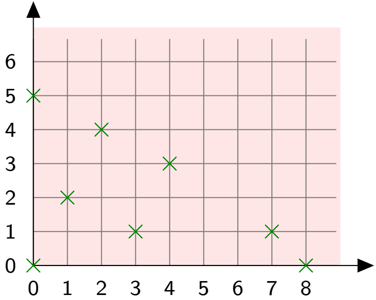
Well quasi-orders



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

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

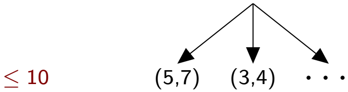
Well quasi-orders



- You cannot pick a point higher on both coordinates than one of the previous ones.
- Your ith point (x_i, y_i) has to be such that $|x_i|, |y_i| \leq 10^i$.

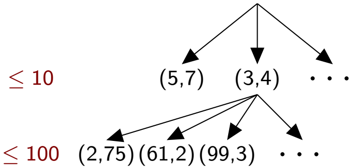
$$(4,3) \to (2,4) \to (7,1) \to (0,5) \to (8,0) \to (3,1) \to (1,2) \to (0,0)$$

Well quasi-orders



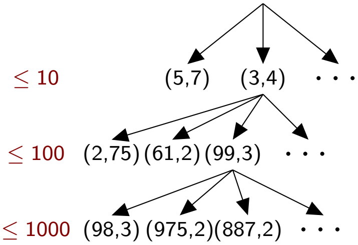
- You cannot pick a point higher on both coordinates than one of the previous ones.
- Your ith point (x_i, y_i) has to be such that $|x_i|, |y_i| \leq 10^i$.

Well quasi-orders



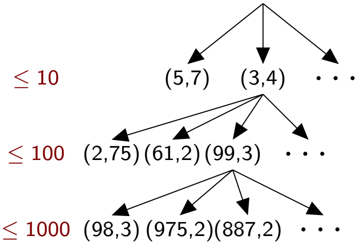
- You cannot pick a point higher on both coordinates than one of the previous ones.
- Your ith point (x_i, y_i) has to be such that $|x_i|, |y_i| \leq 10^i$.

Well quasi-orders



- You cannot pick a point higher on both coordinates than one of the previous ones.
- Your ith point (x_i, y_i) has to be such that $|x_i|, |y_i| \leq 10^i$.

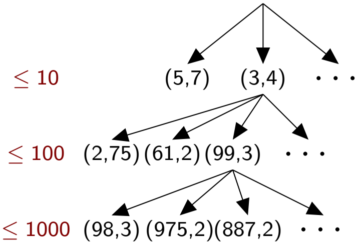
Well quasi-orders



- You cannot pick a point higher on both coordinates than one of the previous ones.
- Your ith point (x_i, y_i) has to be such that $|x_i|, |y_i| \leq 10^i$.

König's lemma $\rightarrow$ this tree is finite.
In fact, there is a computable bound on the length of the longest branch.

Well quasi-orders



- You cannot pick a point higher on both coordinates than one of the previous ones.
- Your ith point (x_i, y_i) has to be such that $|x_i|, |y_i| \leq 10^i$.

König's lemma $\rightarrow$ this tree is finite.
In fact, there is a computable bound on the length of the longest branch.
We can enumerate all possible such trees!

Well quasi-orders: Subwords

Higman's lemma

For all finite alphabet Σ, the subword order $\preceq$ is a well quasi-order over Σ^*.

Well quasi-orders: Subwords

Higman's lemma

For all finite alphabet Σ, the subword order $\preceq$ is a well quasi-order over Σ^*.

$\Leftrightarrow$ Any sequence $w_0, w_1, w_2, \ldots$ of words over Σ such that $w_i \not\preceq w_j$ for all $i < j$ is **finite**.

Well quasi-orders: Subwords

Higman's lemma

For all finite alphabet Σ, the subword order $\preceq$ is a well quasi-order over Σ^*.

$\Leftrightarrow$ Any sequence $w_0, w_1, w_2, \ldots$ of words over Σ such that $w_i \not\preceq w_j$ for all $i < j$ is **finite**.

Given a finite alphabet Σ and a computable function $B : \mathbb{N} \to \mathbb{N}$, the set of sequences $(w_i)_{i\in\mathbb{N}}$ over Σ such that

- $w_i \not\preceq w_j$ for all $i < j$
- $|w_i| \leq B(i)$ for all i

is finite and computable.

1. Broadcast networks
   - Basic model
   - With registers
2. Signature BNRA
   - Well quasi-orders
   - Decidability proof

Towards a tree abstraction

Assume that there is a valid run ρ for COVER.

Observation 1

If agent a broadcasts to agents b and c, we can make copy agent a so that b and c receive messages from distinct agents.

We can modify ρ so that each agent sends messages to one single agent.

Towards a tree abstraction

Assume that there is a valid run ρ for COVER.

Observation 1

If agent a broadcasts to agents b and c, we can make copy agent a so that b and c receive messages from distinct agents.

We can modify ρ so that each agent sends messages to one single agent.

Observation 2

If a broadcasts m_1 to b then b broadcasts m_2 to a, we can make a copy a' of a that broadcasts to b and then stops; $a' \to b \to a$.

More generally, we can guarantee that the graph of "who sends messages to whom" has no cycle: it's a tree (or a forest) !

Tree unfoldings

$$\xrightarrow[\mathbf{rec}(m_1, v_1)\mathbf{rec}(m_2, v_2)\mathbf{rec}(m_3, v_1)]{\mathbf{br}(m_0, v_0) \qquad\qquad\qquad\qquad\qquad\qquad \mathbf{br}(m, v_0)}$$

Tree unfoldings

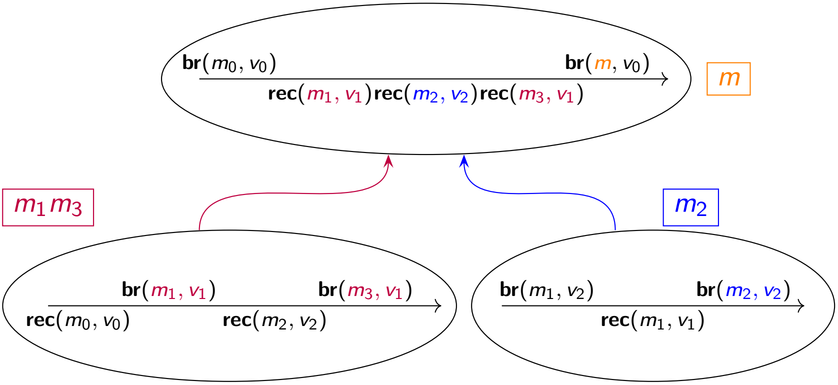
$$\xrightarrow[\mathbf{rec}(m_1, v_1)\,\mathbf{rec}(m_2, v_2)\,\mathbf{rec}(m_3, v_1)]{\mathbf{br}(m_0, v_0) \qquad\qquad\qquad\qquad\qquad\qquad \mathbf{br}(m, v_0)}$$

Tree unfoldings

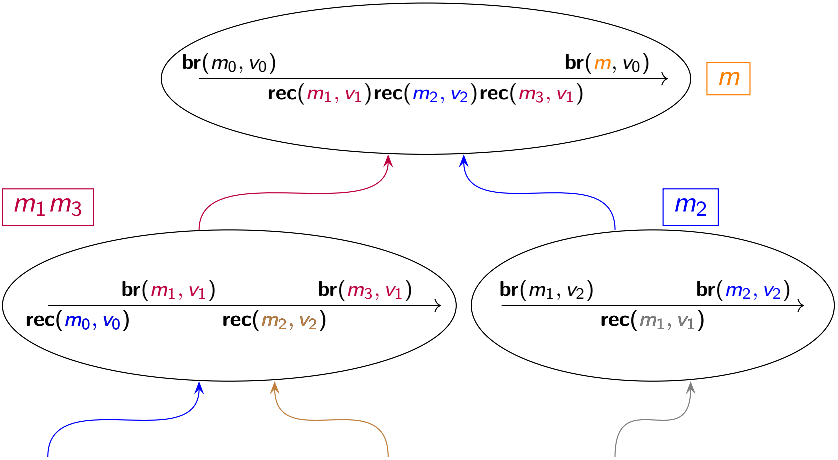
$$\xrightarrow[\mathbf{rec}(m_1, v_1)\mathbf{rec}(m_2, v_2)\mathbf{rec}(m_3, v_1)]{\mathbf{br}(m_0, v_0) \qquad\qquad\qquad\qquad \mathbf{br}(m, v_0)}$$

$$\xrightarrow[\mathbf{rec}(m_0, v_0) \qquad\qquad \mathbf{rec}(m_2, v_2)]{\mathbf{br}(m_1, v_1) \qquad\qquad \mathbf{br}(m_3, v_1)} \qquad \xrightarrow[\mathbf{rec}(m_1, v_1)]{\mathbf{br}(m_1, v_2) \qquad\qquad \mathbf{br}(m_2, v_2)}$$

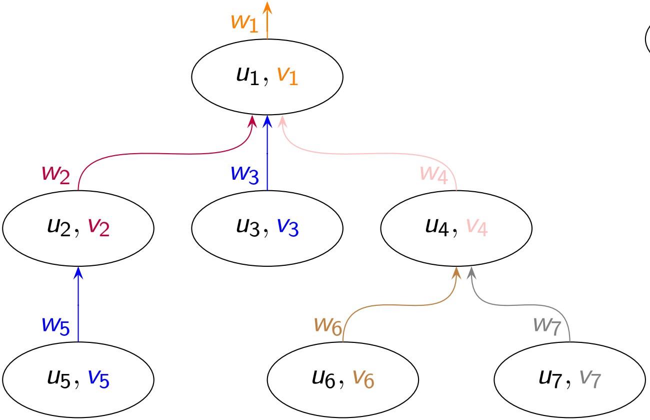
Tree unfoldings

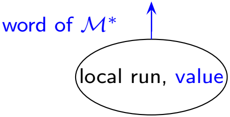


Tree unfoldings



Tree unfoldings

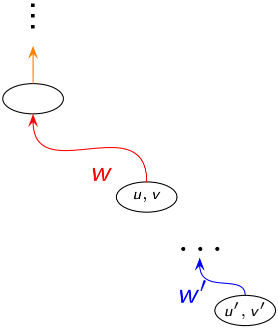




Branch reduction

Lemma

If a node labelled w has a descendant labelled w' with w a subword of w' then the tree can be reduced.

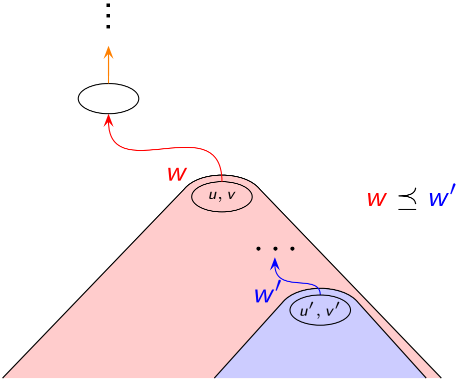


$w \preceq w'$

Branch reduction

Lemma

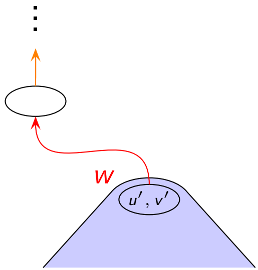
If a node labelled w has a descendant labelled w' with w a subword of w' then the tree can be reduced.



Branch reduction

Lemma

If a node labelled w has a descendant labelled w' with w a subword of w' then the tree can be reduced.



$w \preceq w'$

Branch reduction

- We can assume that a node labelled w has no descendant labelled $w' \succeq w$.

Branch reduction

- We can assume that a node labelled w has no descendant labelled $w' \succeq w$.
- In order to bound the size of the tree, we need a bound on the growth of the size of the nodes.

Branch reduction

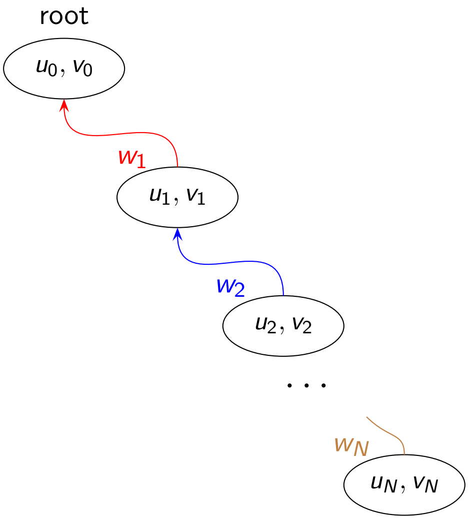
- We can assume that a node labelled w has no descendant labelled $w' \succeq w$.
- In order to bound the size of the tree, we need a bound on the growth of the size of the nodes.

Shortening long local runs

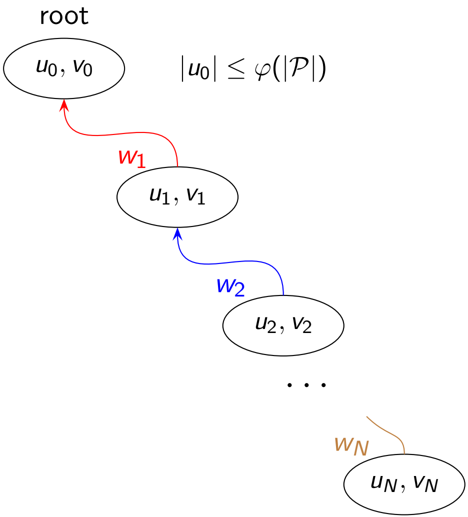
There exists a primitive recursive function $\varphi : \mathbb{N} \to \mathbb{N}$ such that, if an agent must broadcast k messages, its local run does not need to have more than $k\,\varphi(|\mathcal{P}|)$ steps.

$|\mathcal{P}|$: size of the protocol
Proof by shortening arguments (a bit involved)

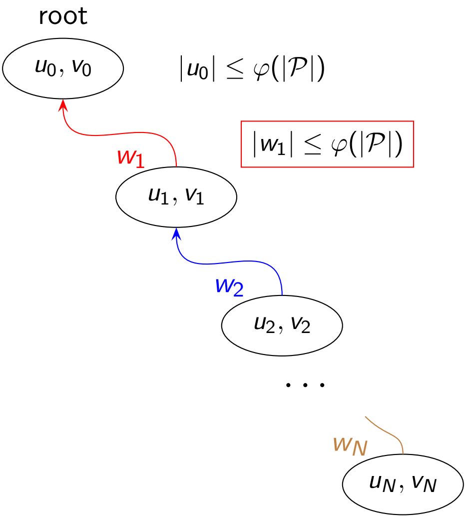
Bounding the branches



Bounding the branches



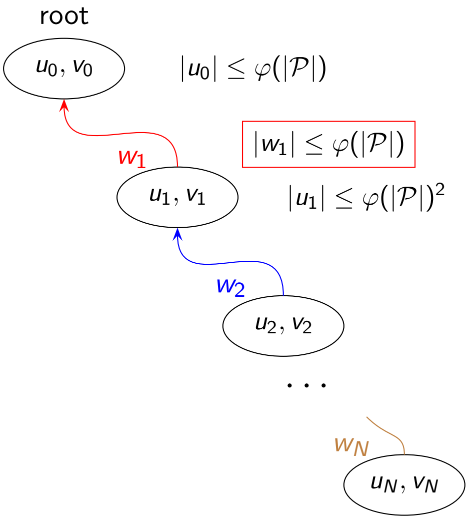
Bounding the branches



$|u_0| \leq \varphi(|\mathcal{P}|)$

$\boxed{|w_1| \leq \varphi(|\mathcal{P}|)}$

Bounding the branches



$|u_0| \leq \varphi(|\mathcal{P}|)$

$\boxed{|w_1| \leq \varphi(|\mathcal{P}|)}$

$|u_1| \leq \varphi(|\mathcal{P}|)^2$

Bounding the branches

root

u_0, v_0 $\qquad |u_0| \leq \varphi(|\mathcal{P}|)$

w_1 $\qquad \boxed{|w_1| \leq \varphi(|\mathcal{P}|)}$

u_1, v_1 $\qquad |u_1| \leq \varphi(|\mathcal{P}|)^2$

w_2 $\qquad \boxed{|w_2| \leq \varphi(|\mathcal{P}|)^2}$

u_2, v_2 $\qquad |u_2| \leq \varphi(|\mathcal{P}|)^3$

$\cdots$

w_N $\qquad \boxed{|w_N| \leq \varphi(|\mathcal{P}|)^N}$

u_N, v_N $\qquad |u_N| \leq \varphi(|\mathcal{P}|)^{N+1}$

Decidability and complexity

Bounds

We use the previous argument to bound (in an irreducible tree):

- 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 an irreducible tree):

- 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 irreducible 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 an irreducible tree):

- 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 irreducible trees in finite time, therefore

Theorem

The $\textsc{Cover}$ problem for signature BNRA is decidable and in $\mathbf{F}_{\omega^\omega}$.

Decidability and complexity

Bounds

We use the previous argument to bound (in an irreducible tree):

- 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 irreducible trees in finite time, therefore

Theorem

The COVER problem for signature BNRA is decidable and in $\mathbf{F}_{\omega^\omega}$.

Can be extended to the non-signature case.

Complexity lower bounds

Lossy Channel Systems

A *Lossy Channel System* (LCS) is a transition system with a FIFO queue and unreliable writes.

Theorem

LCS reachability is $\mathbf{F}_{\omega^\omega}$-hard[a].

[a] Schnoebelen, Information Processing Letters '08

Complexity lower bounds

Lossy Channel Systems

A *Lossy Channel System* (LCS) is a transition system with a FIFO queue and unreliable writes.

Theorem

LCS reachability is $\mathbf{F}_{\omega^\omega}$-hard[a].

[a] Schnoebelen, Information Processing Letters '08

Theorem

$\textsc{Cover}$ in BNRA is $\mathbf{F}_{\omega^\omega}$-complete, even for signature protocols with two registers.

Complexity lower bounds

Lossy Channel Systems

A *Lossy Channel System* (LCS) is a transition system with a FIFO queue and unreliable writes.

Theorem

LCS reachability is $\mathbf{F}_{\omega^\omega}$-hard[a].

[a] Schnoebelen, Information Processing Letters '08

Theorem

Cover in BNRA is $\mathbf{F}_{\omega^\omega}$-complete, even for signature protocols with two registers.

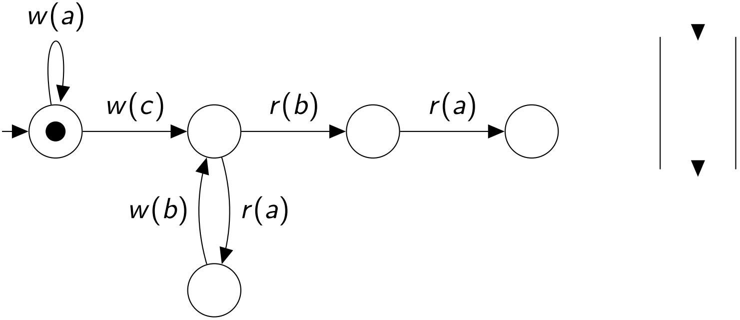
It is however NP-complete when each agent has only one register.

Conclusion

Thank you for your attention!

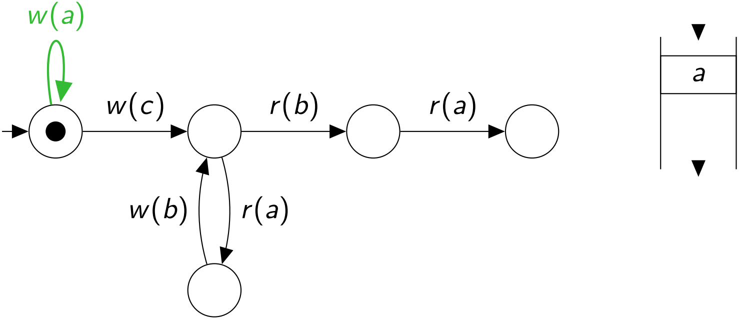
Complexity: encoding Lossy Channel Systems

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



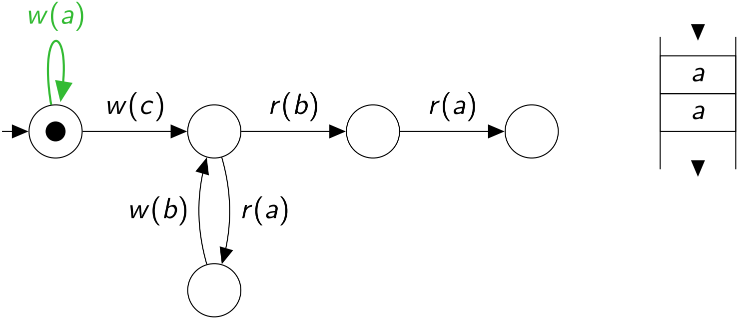
Complexity: encoding Lossy Channel Systems

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



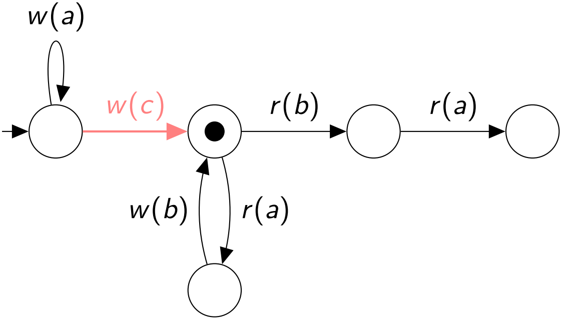
Complexity: encoding Lossy Channel Systems

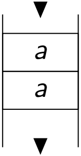
Lossy Channel System = Transition system with FIFO memory + unreliable writes.



Complexity: encoding Lossy Channel Systems

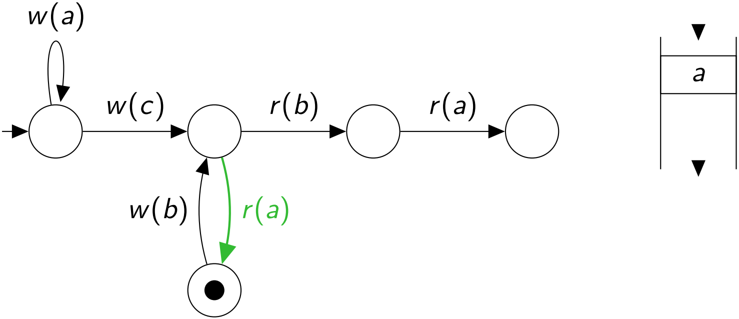
Lossy Channel System = Transition system with FIFO memory + unreliable writes.





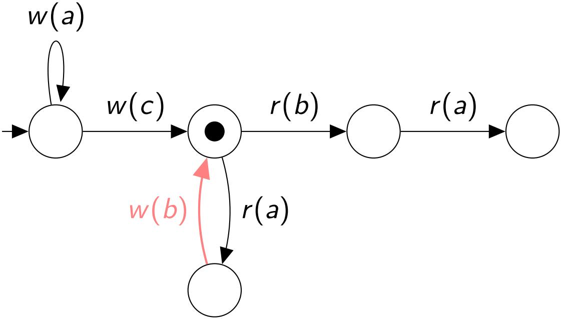
Complexity: encoding Lossy Channel Systems

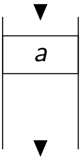
Lossy Channel System = Transition system with FIFO memory + unreliable writes.



Complexity: encoding Lossy Channel Systems

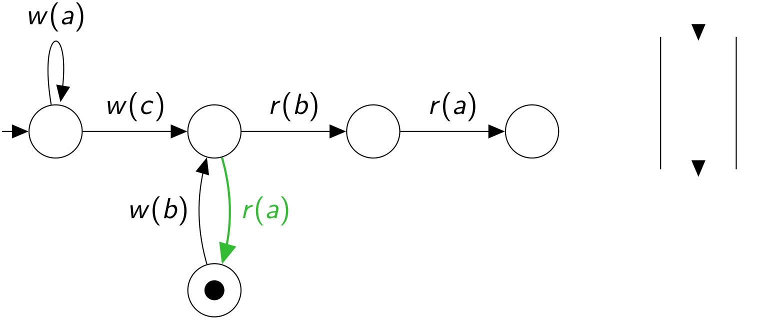
Lossy Channel System = Transition system with FIFO memory + unreliable writes.





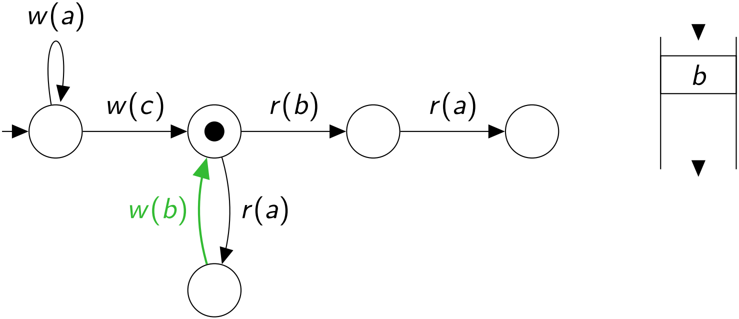
Complexity: encoding Lossy Channel Systems

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



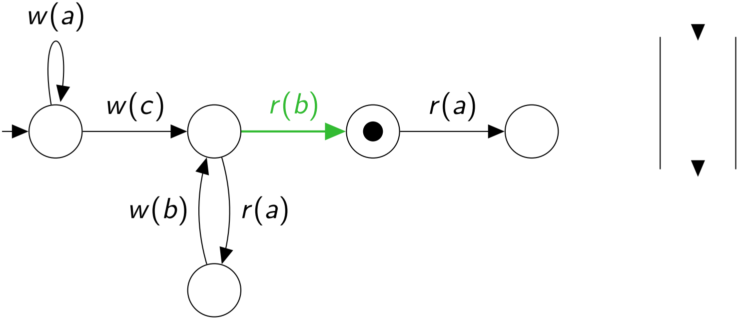
Complexity: encoding Lossy Channel Systems

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



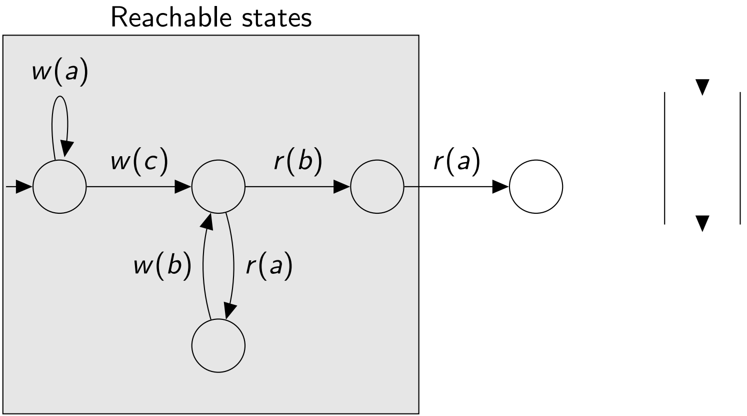
Complexity: encoding Lossy Channel Systems

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



Complexity: encoding Lossy Channel Systems

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

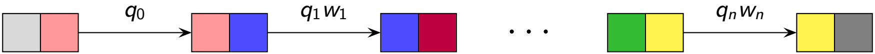


Complexity: encoding Lossy Channel Systems

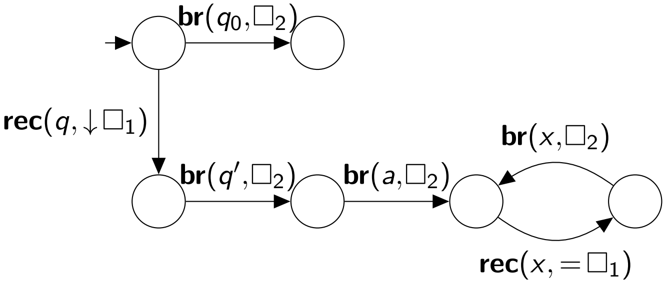
We simulate an LCS through a chain of agents that each apply a transition.

Each agent stores:

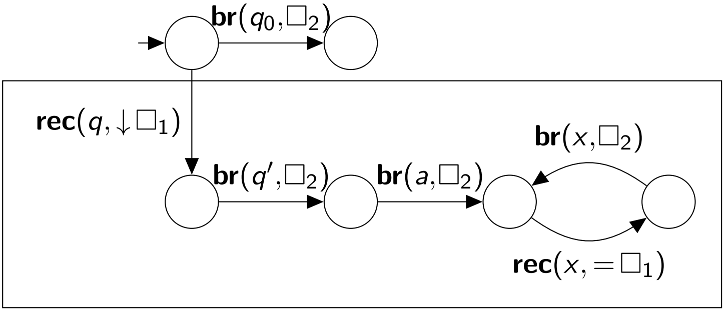
- An identifier for itself
- Its predecessor's identifier



Complexity: encoding Lossy Channel Systems



Complexity: encoding Lossy Channel Systems



For each transition $q \xrightarrow{w(a)} q'$ of the LCS

Complexity results

$\mathbf{F}_{\omega^\omega}$ = Hyper-Ackermannian complexity class.

Theorem

LCS reachability is $\mathbf{F}_{\omega^\omega}$-hard[a].

[a] Schnoebelen, Information Processing Letters '08

Complexity results

$\mathbf{F}_{\omega^\omega}$ = Hyper-Ackermannian complexity class.

Theorem

LCS reachability is $\mathbf{F}_{\omega^\omega}$-hard[a].

[a] Schnoebelen, Information Processing Letters '08

Theorem

$\textsc{Cover}$ in BNRA is $\mathbf{F}_{\omega^\omega}$-complete, even for signature protocols with two registers.

Complexity results

$\mathbf{F}_{\omega^\omega}$ = Hyper-Ackermannian complexity class.

Theorem

LCS reachability is $\mathbf{F}_{\omega^\omega}$-hard[a].

[a] Schnoebelen, Information Processing Letters '08

Theorem

Cover in BNRA is $\mathbf{F}_{\omega^\omega}$-complete, even for signature protocols with two registers.

Theorem

Cover in BNRA with one register is **NP**-complete.