Parameterized Verification of Broadcast Networks of Register Automata

Nicolas Waldburger

Lucie Guillou　　Corto Mascle

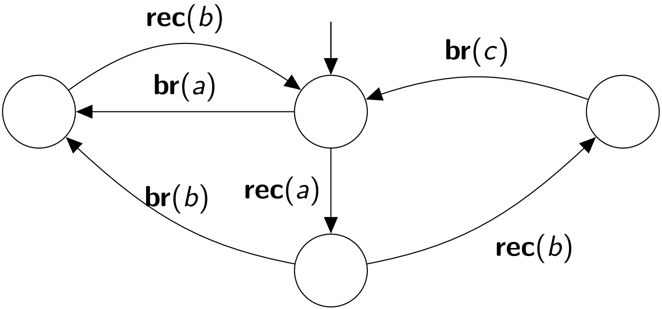
Funded by ANR PaVeDyS

March 11th, 2024

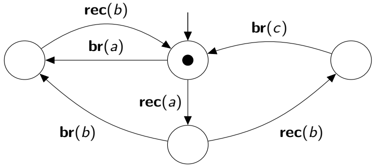
To be published at FoSSaCS'24

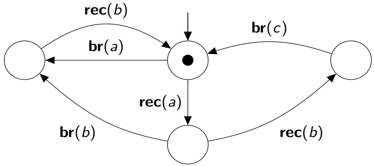
1. Introduction of the model
2. Decidability of COVER for signature BNRA
3. Decidability of COVER in the general case
4. Complexity lower bound

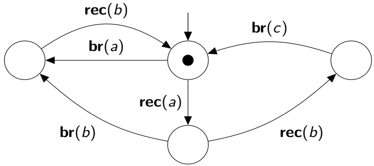
Broadcast networks



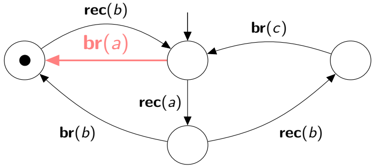
Broadcast networks

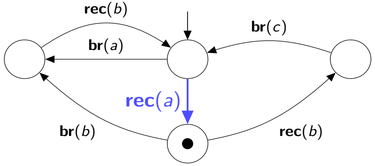


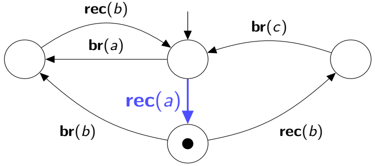




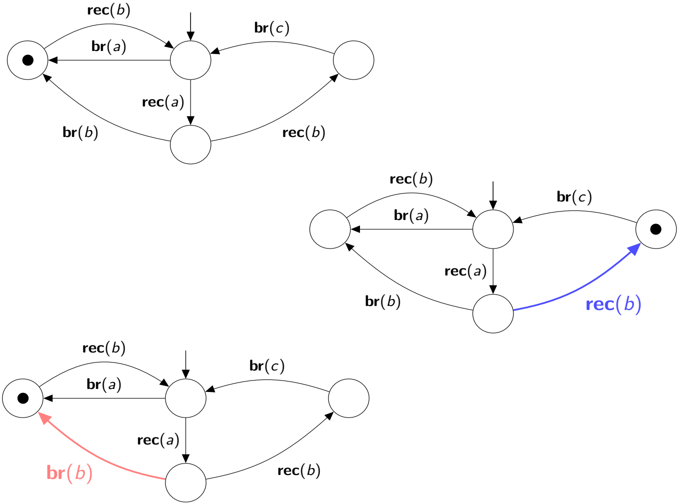
Broadcast networks



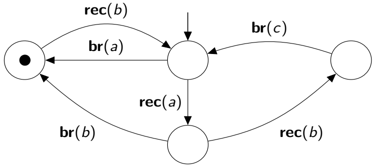


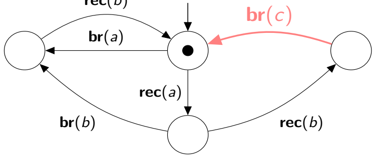


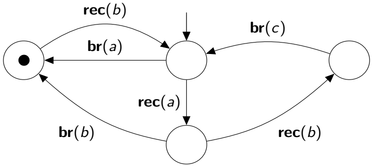
Broadcast networks



Broadcast networks







Broadcast Networks

Definition[1]

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

[1] Delzanno, Sangnier, Zavattaro, CONCUR'10

Broadcast Networks

Definition[1]

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

- 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[1]

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

- 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].

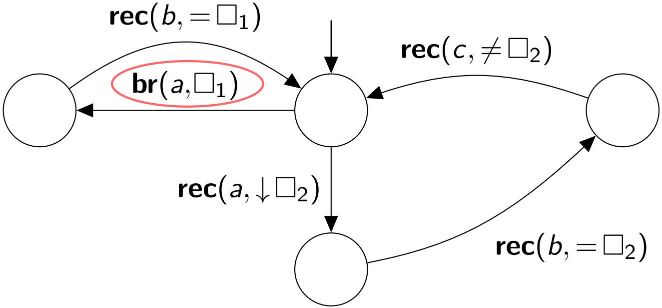
[1] Delzanno, Sangnier, Zavattaro, CONCUR'10
[2] Fournier, PhD thesis, 2015

Adding registers

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

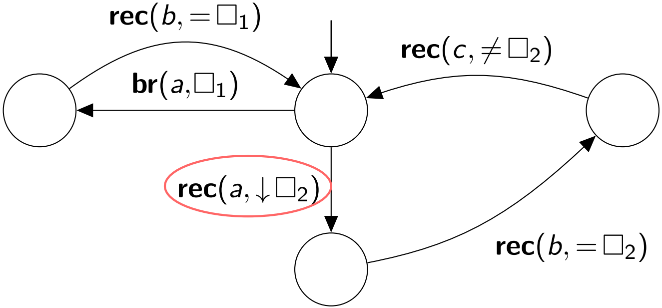
Adding registers

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



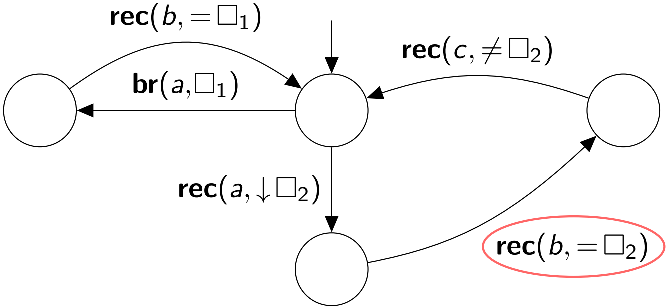
Adding registers

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



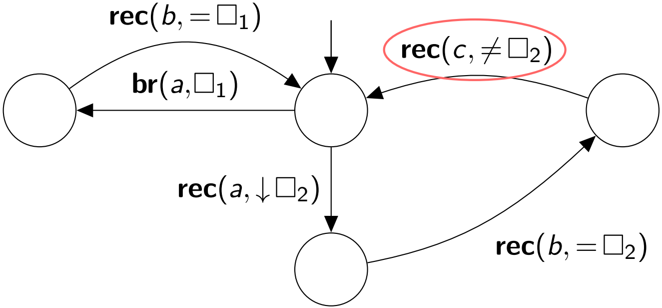
Adding registers

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



Adding registers

Each agent now has local *registers* $\square_1, \ldots, \square_r$, 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.

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

- Broadcast a message symbol along 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.

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

- Broadcast a message symbol along with a register value: $\mathbf{br}(m, r_i)$
- Receive a message of a given symbol m: $\mathbf{rec}(m, \mathrm{op})$, with 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 $\mathbb{N}$.
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:

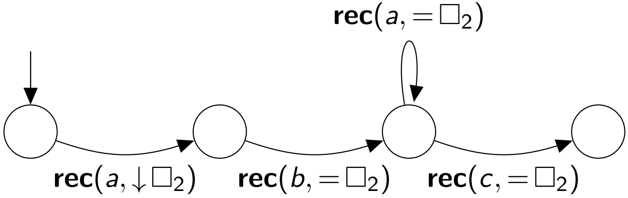
- Broadcast a message symbol along with a register value: $\mathbf{br}(m, r_i)$
- Receive a message of a given symbol m: $\mathbf{rec}(m, \mathrm{op})$, with op one of the following:
  - store the value in register $\square_i$: $\downarrow\square_i$,
  - test it for equality with register $\square_i$: $=\square_i, \neq\square_i$
  - 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 PSPACE.

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

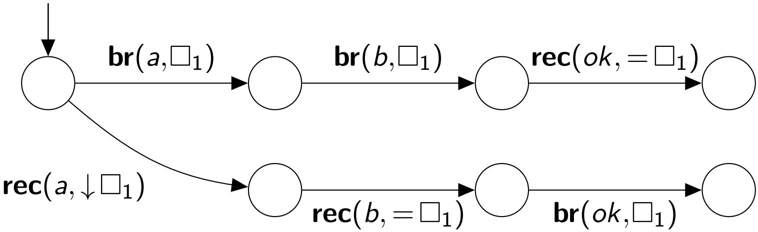
Things we can do

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

Main theorem

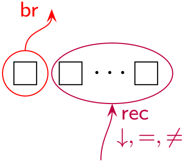
COVER is decidable for BNRA.

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.

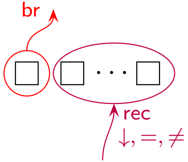


Signature BNRA

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.

Tree witnesses for COVER

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

Tree witnesses for COVER

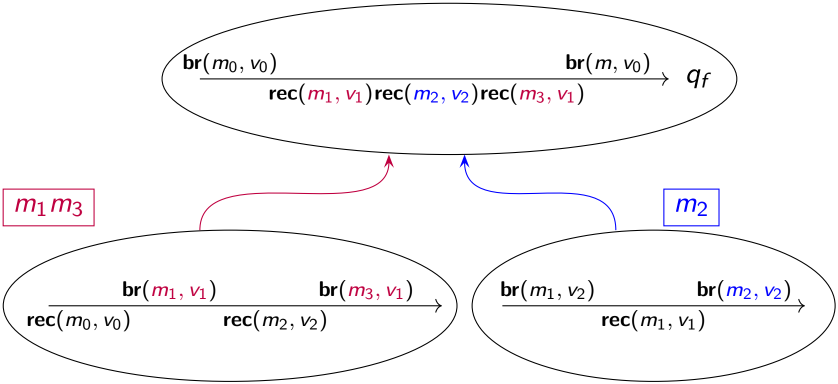
$$\xrightarrow[\mathbf{rec}(m_1, v_1)\mathbf{rec}(m_2, v_2)\mathbf{rec}(m_3, v_1)]{\mathbf{br}(m_0, v_0) \qquad\qquad\qquad\qquad\qquad\qquad \mathbf{br}(m, v_0)} q_f$$

Tree witnesses for COVER

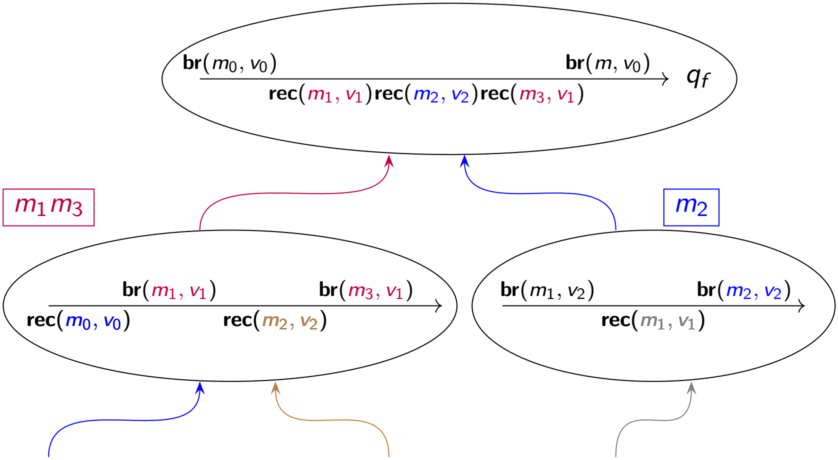
$$\xrightarrow[\mathbf{rec}(m_1, v_1)\mathbf{rec}(m_2, v_2)\mathbf{rec}(m_3, v_1)]{\mathbf{br}(m_0, v_0) \qquad \qquad \mathbf{br}(m, v_0)} q_f$$

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

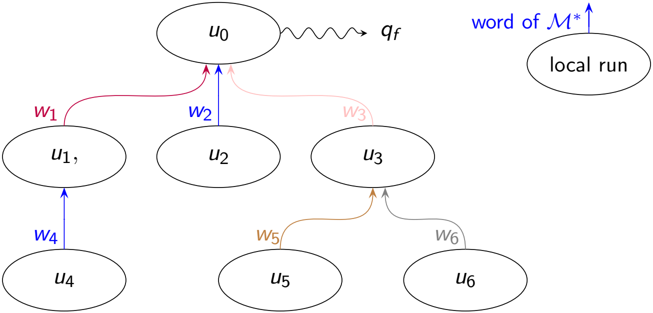
Tree witnesses for COVER



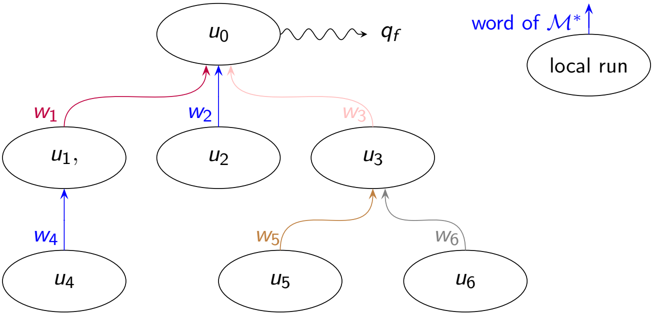
Tree witnesses for COVER



Tree witnesses for COVER



Tree witnesses for COVER



Lemma

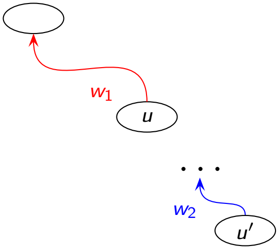
There is a tree witness if and only if the instance of COVER is positive.

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.

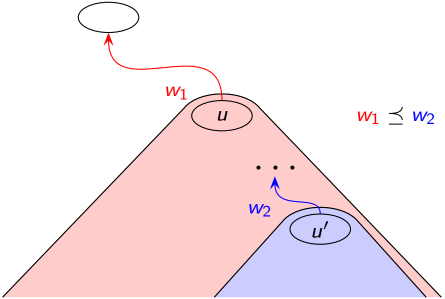


$w_1 \preceq w_2$

Branch reduction

Lemma

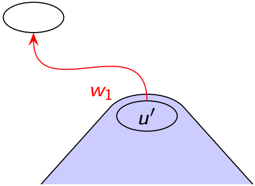
If a node labelled w has a descendant labelled w' with w a subword of w' (written $w \preceq w'$) then the tree can be shortened.



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.

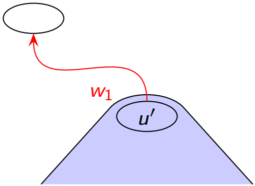


$w_1 \preceq w_2$

Branch reduction

Lemma

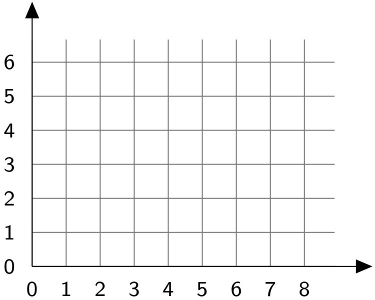
If a node labelled w has a descendant labelled w' with w a subword of w' (written $w \preceq w'$) then the tree can be shortened.



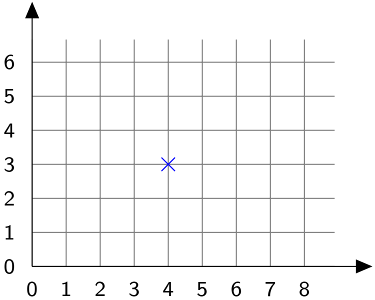
$w_1 \preceq w_2$

After iterating this shortening procedure, we end up with a tree in which a node labelled w has no descendant labelled $w' \succeq w$.

Well quasi-orders

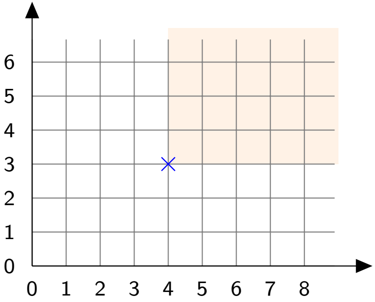


Well quasi-orders



$(4, 3)$

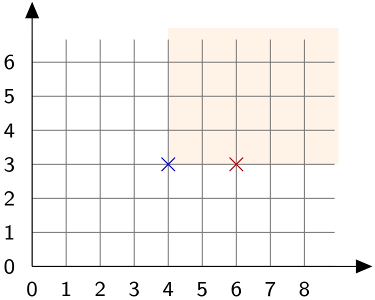
Well quasi-orders



$(4, 3)$

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

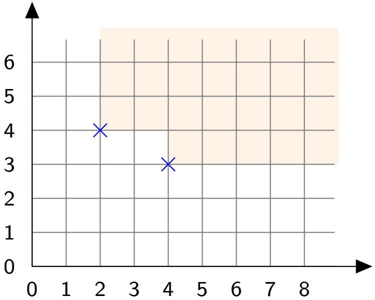
Well quasi-orders



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

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

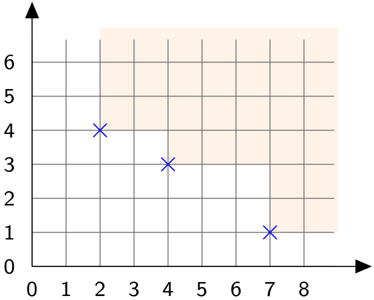
Well quasi-orders



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

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

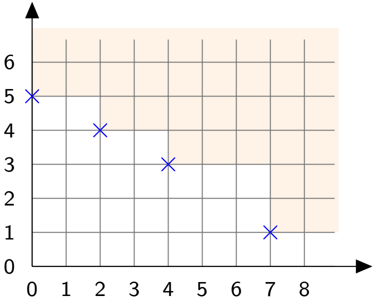
Well quasi-orders



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

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

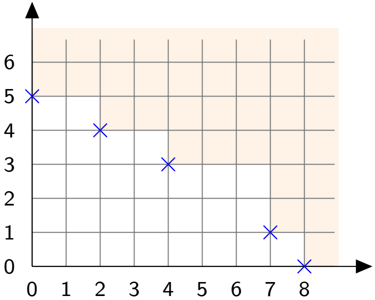
Well quasi-orders



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

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

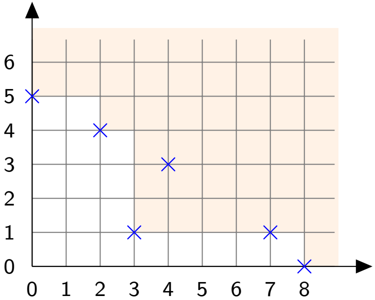
Well quasi-orders



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

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

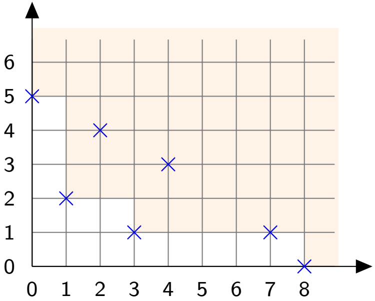
Well quasi-orders



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

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

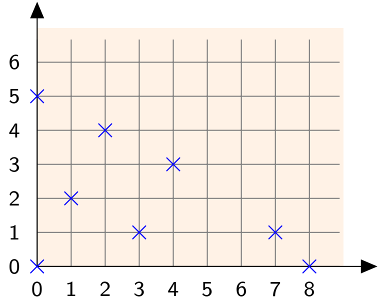
Well quasi-orders



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

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

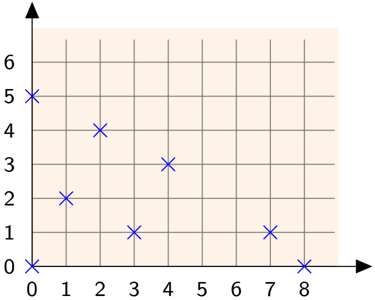
Well quasi-orders



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

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

Well quasi-orders

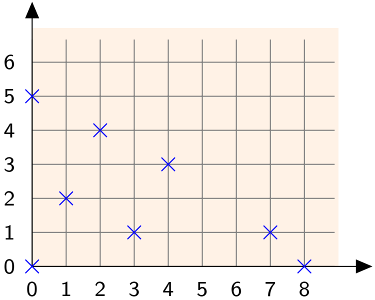


You cannot pick a point higher on both coordinates than one of the previous points.

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

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

Well quasi-orders



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

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

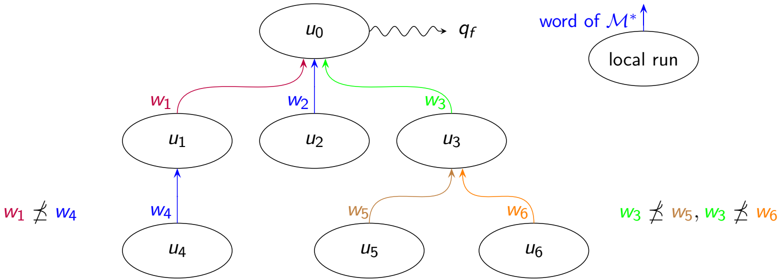
This order on $\mathbb{N}^2$ is a *well quasi-order* : every bad sequence is finite.

Higman's lemma

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

Back to the trees

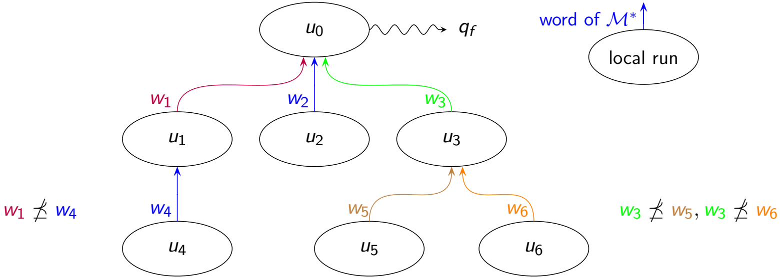
If q_f can be covered, then there is a witness of the execution of the form:



Every branch forms a bad sequence. Because $\preceq$ is a well quasi-order, we know that every branch of the tree is finite...

Back to the trees

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... 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 $m^k, m^{k-1}, \dots, m$ with $m \in \mathcal{M}$ is a bad sequence of length k.

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[4]

Given a finite alphabet Σ 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 Σ such that

- $w_i \not\preceq w_j$ for all $i < j$ (bad sequence) and
- $|w_i| \leq F(i)$ for all i

has length at most B.

[4] Schmitz, Schnoebelen, ICALP'11

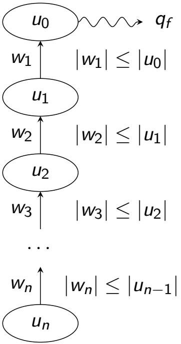
Applying the length function theorem

Consider a branch of a tree of minimal size:

$$\begin{array}{c} u_0 \rightsquigarrow q_f \\ \uparrow w_1 \\ u_1 \\ \uparrow w_2 \\ u_2 \\ \uparrow w_3 \\ \cdots \\ \uparrow w_n \\ u_n \end{array}$$

Applying the length function theorem

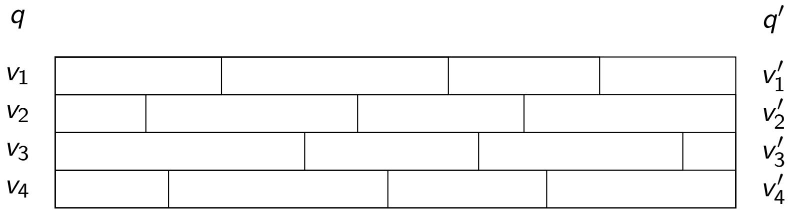
Consider a branch of a tree of minimal size:



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

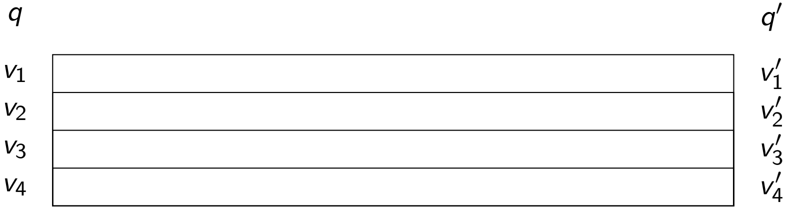
By induction on the number of *active* registers.
Register $\square_i$ is *active* when some storing action $\downarrow\square_i$ is performed.



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.

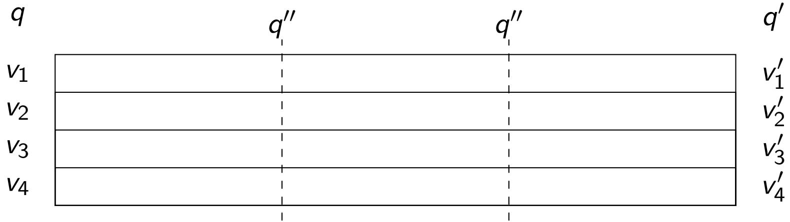
0 active registers



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.

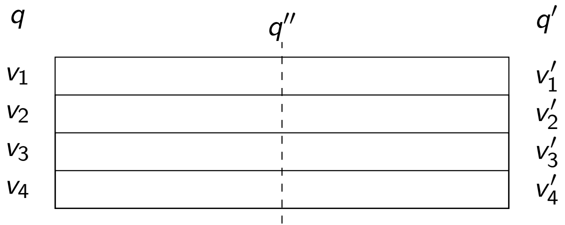
0 active registers



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.

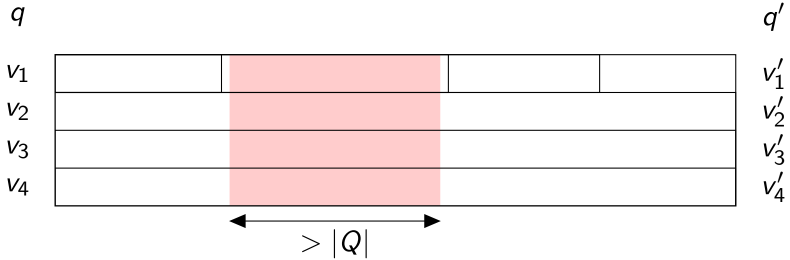
0 active registers



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.

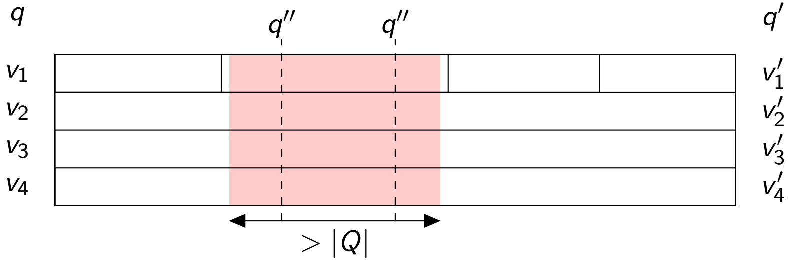
1 active registers



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.

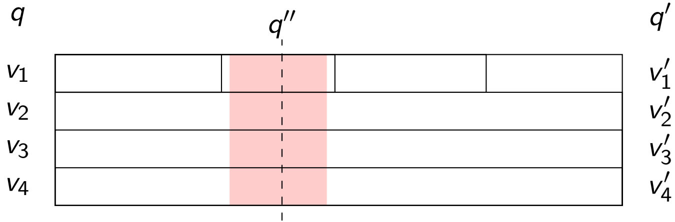
1 active registers



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.

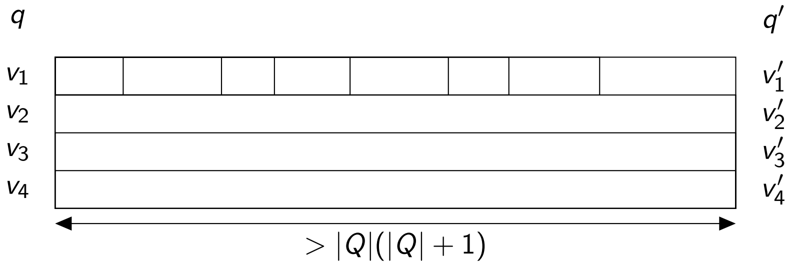
1 active registers



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.

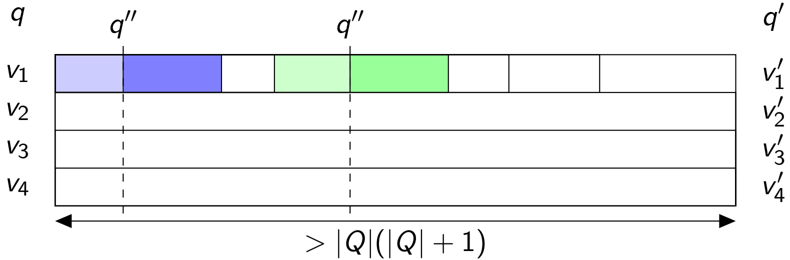
1 active registers



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.

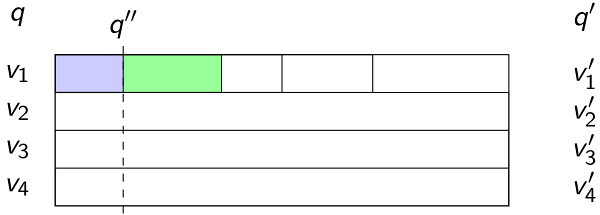
1 active registers



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

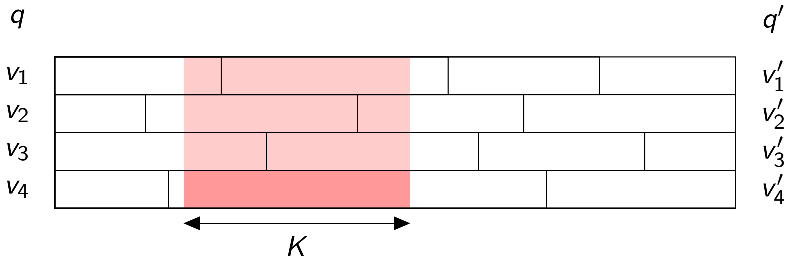


Bounding local runs

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

m active registers

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

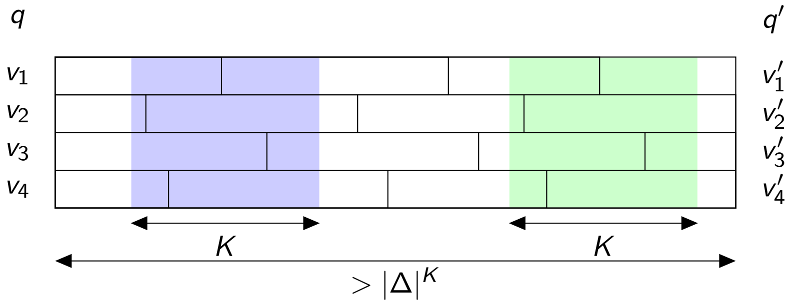


Bounding local runs

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 $\geq K$.

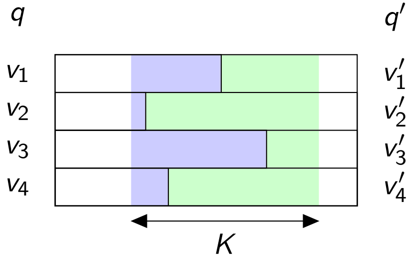


Bounding local runs

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 $\geq K$.



Bounding the tree

Lemma

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

Δ: set of transitions $\quad$ r: number of registers.

Corollary

If an agent has a local run that broadcasts w, then it has one such local run of length $\leq |w|\,\varphi(|\Delta|, r)$.

Bounding the branches

$$\begin{tikzcd}
u_0, v_0 \arrow[r, squiggly] & q_f \\
u_1, v_1 \arrow[u, "w_1", "|w_1| \leq |u_0|"'] & \\
u_2, v_2 \arrow[u, "w_2", "|w_2| \leq |u_1|"'] & \\
\cdots \arrow[u, "w_3", "|w_3| \leq |u_2|"'] & \\
u_n, v_n \arrow[u, "w_n", "|w_n| \leq |u_{n-1}|"'] &
\end{tikzcd}$$

Bounding the branches

$$
\begin{array}{llll}
(u_0, v_0) \rightsquigarrow q_f & & & |u_0| \leq \phi(|\Delta|, r) \\
w_1 \uparrow & & |w_1| \leq |u_0| & |w_1| \leq \phi(|\Delta|, r) \\
(u_1, v_1) & & & |u_1| \leq |w_1|\,\phi(|\Delta|, r) \leq \phi(|\Delta|, r)^2 \\
w_2 \uparrow & & |w_2| \leq |u_1| & |w_2| \leq \phi(|\Delta|, r)^2 \\
(u_2, v_2) & & & |u_2| \leq |w_2|\,\phi(|\Delta|, r) \leq \phi(|\Delta|, r)^3 \\
w_3 \uparrow & & |w_3| \leq |u_2| & |w_3| \leq \phi(|\Delta|, r)^3 \\
\ldots & & & \ldots \\
w_n \uparrow & & |w_n| \leq |u_{n-1}| & |w_n| \leq \phi(|\Delta|, r)^n \\
(u_n, v_n) & & &
\end{array}
$$

Bounding the branches

$$
\begin{array}{llll}
(u_0, v_0) \rightsquigarrow q_f & & & |u_0| \leq \phi(|\Delta|, r) \\
w_1 \uparrow & & |w_1| \leq |u_0| & |w_1| \leq \phi(|\Delta|, r) \\
(u_1, v_1) & & & |u_1| \leq |w_1|\,\phi(|\Delta|, r) \leq \phi(|\Delta|, r)^2 \\
w_2 \uparrow & & |w_2| \leq |u_1| & |w_2| \leq \phi(|\Delta|, r)^2 \\
(u_2, v_2) & & & |u_2| \leq |w_2|\,\phi(|\Delta|, r) \leq \phi(|\Delta|, r)^3 \\
w_3 \uparrow & & |w_3| \leq |u_2| & |w_3| \leq \phi(|\Delta|, r)^3 \\
\ldots & & & \ldots \\
w_n \uparrow & & |w_n| \leq |u_{n-1}| & |w_n| \leq \phi(|\Delta|, r)^n \\
(u_n, v_n) & & &
\end{array}
$$

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 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 $\mathbf{F}_{\omega^\omega}$.

The length function theorem in fact gives us a bound for the height of our trees that is a function in hyper-Ackermannian class $\mathcal{F}_{\omega^\omega}$ of $|\Delta|$ and r.

General case

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:

$$\mathbf{br}(a, \square_1)\, \mathbf{br}(b, \square_1)\, \mathbf{rec}(c, = \square_1)\, \mathbf{rec}(d, = \square_1)\, \mathbf{rec}(c, = \square_1)$$

Observation

An agent may do this:

$$\mathbf{br}(a,\square_1)\,\mathbf{br}(b,\square_1)\,\mathbf{rec}(c,=\square_1)\,\mathbf{rec}(d,=\square_1)\,\mathbf{rec}(c,=\square_1)$$

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).

Observation

An agent may do this:

$$\mathbf{br}(a,\square_1)\,\mathbf{br}(b,\square_1)\,\mathbf{rec}(c,=\square_1)\,\mathbf{rec}(d,=\square_1)\,\mathbf{rec}(c,=\square_1)$$

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).

We add contract nodes labelled $\boxed{w \to m}$ that witness a local run that, if it receives word w with value v, can broadcast (m,v).

Our new tree witnesses

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

Our new tree witnesses

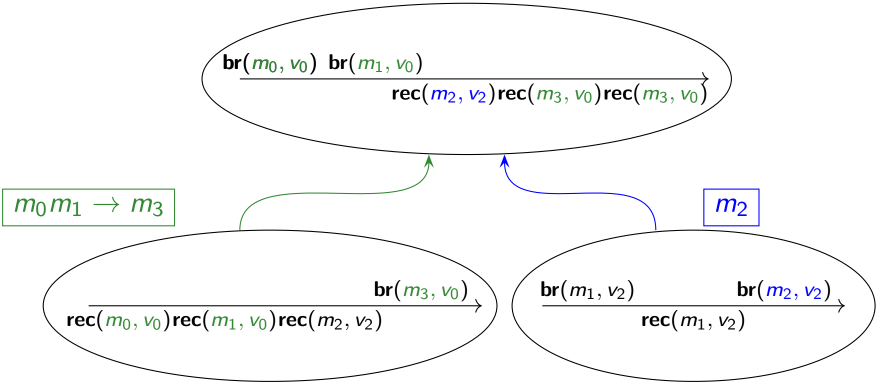
$$\xrightarrow[\mathbf{rec}(m_2, v_2)\mathbf{rec}(m_3, v_0)\mathbf{rec}(m_3, v_0)]{\mathbf{br}(m_0, v_0)\ \mathbf{br}(m_1, v_0)}$$

Our new tree witnesses

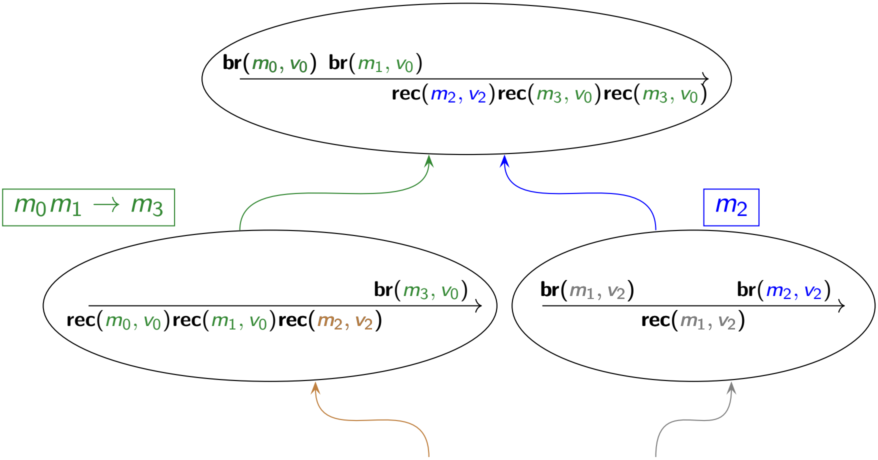
$$\xrightarrow[\mathbf{rec}(m_2, v_2)\mathbf{rec}(m_3, v_0)\mathbf{rec}(m_3, v_0)]{\mathbf{br}(m_0, v_0)\ \ \mathbf{br}(m_1, v_0)}$$

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

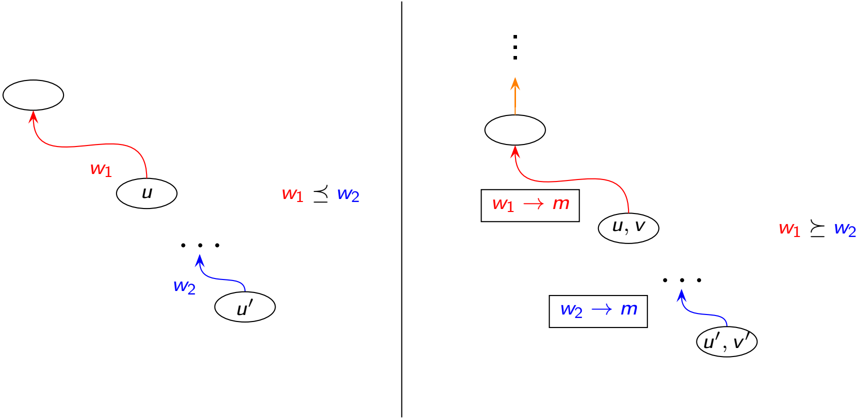
Our new tree witnesses



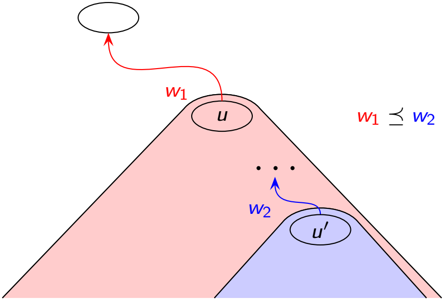
Our new tree witnesses

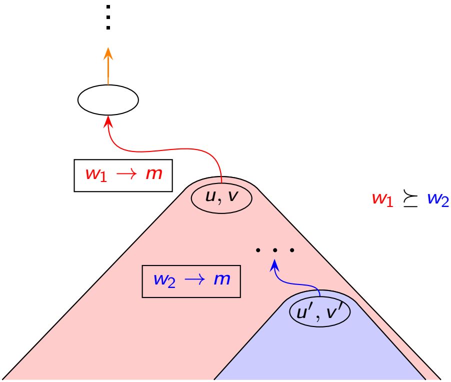


Branch reductions

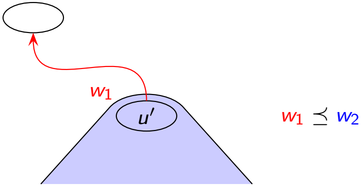


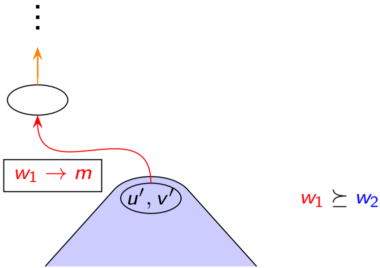
Branch reductions



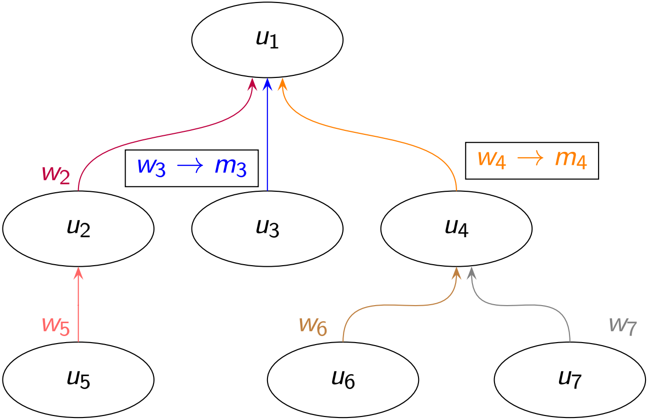


Branch reductions



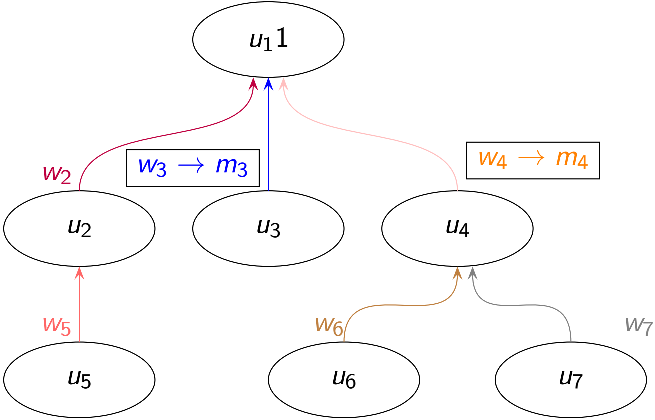


Things are more complicated than before

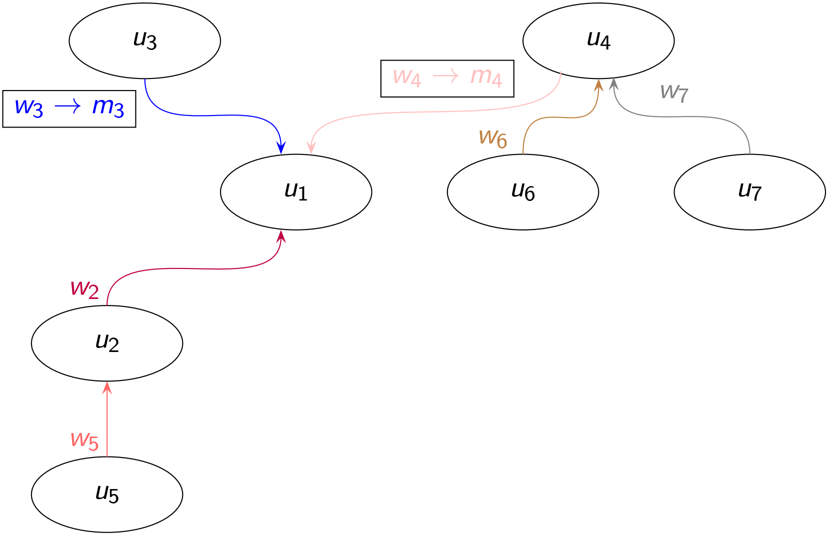


Problem: The number of messages that a node must broadcast now depends on its $\boxed{w \rightarrow m}$ children, and not just on its father.

Rearranging our trees



Rearranging our trees



Rearranging the tree

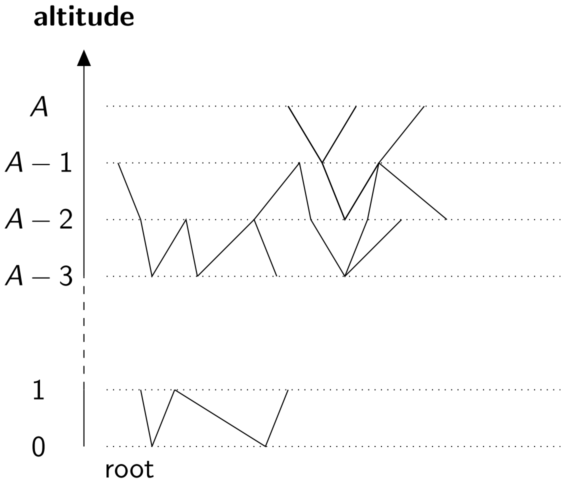
Definition

The *altitude* of a node is

- 0 if it is the root
- its father's altitude $+1$ if it is labelled $\boxed{w \to m}$
- its father's altitude -1 if it is labelled w

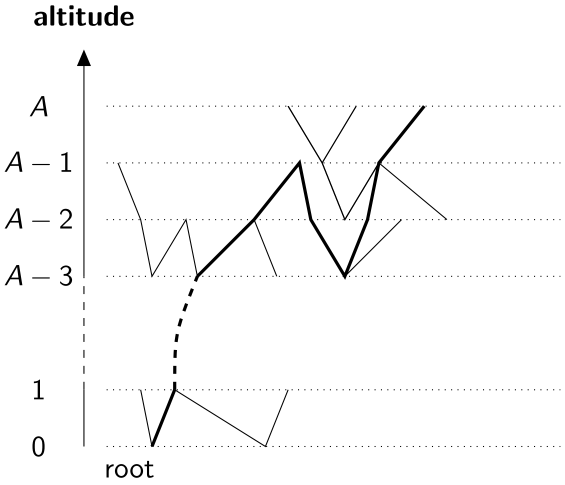
Bounding the altitude

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



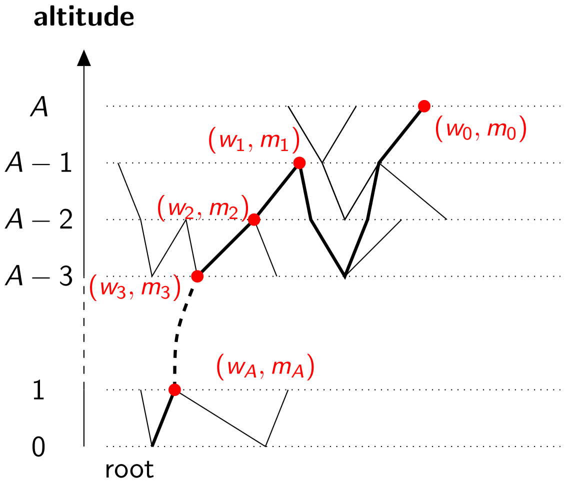
Bounding the altitude

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



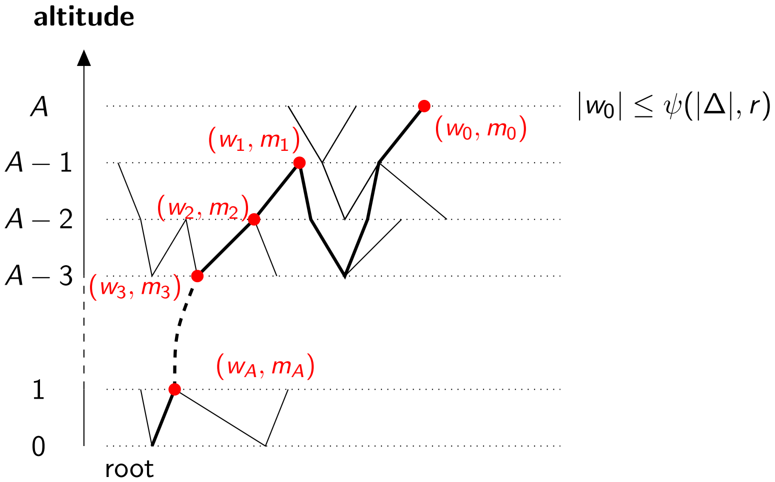
Bounding the altitude

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



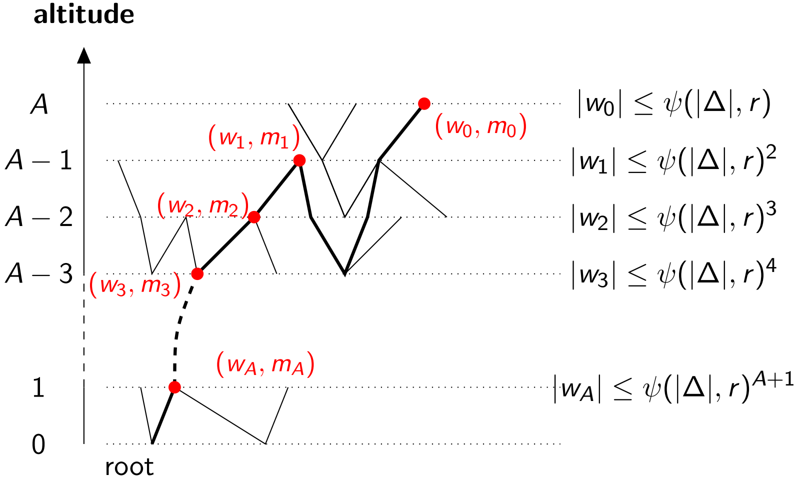
Bounding the altitude

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



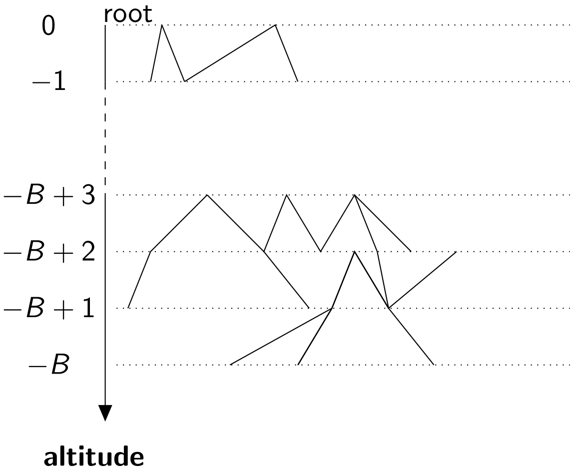
Bounding the altitude

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



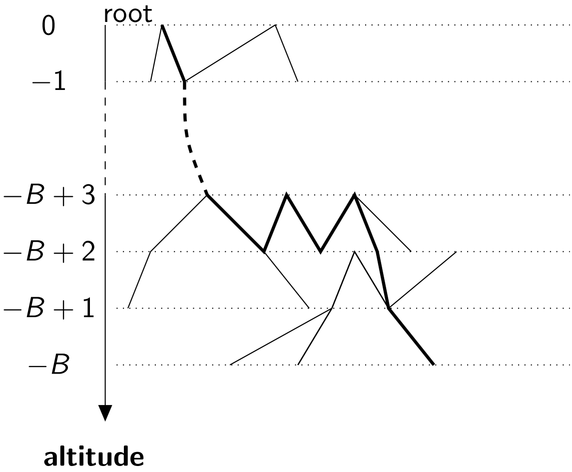
Bounding the altitude

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.



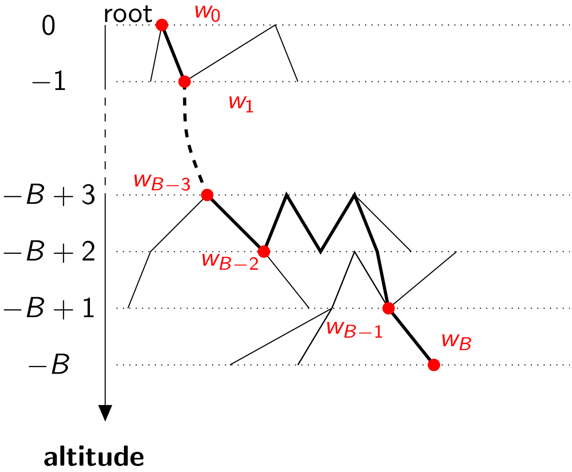
Bounding the altitude

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.



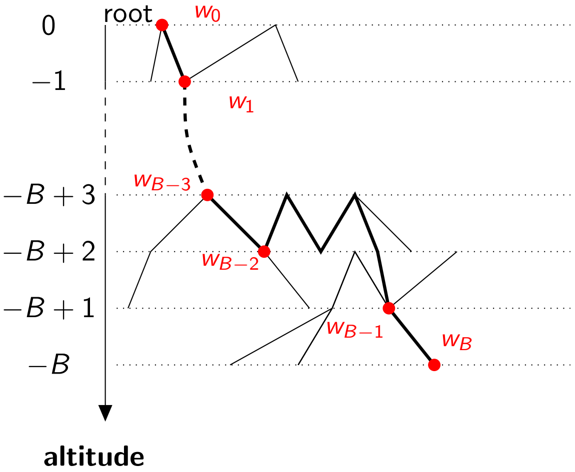
Bounding the altitude

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.



Bounding the altitude

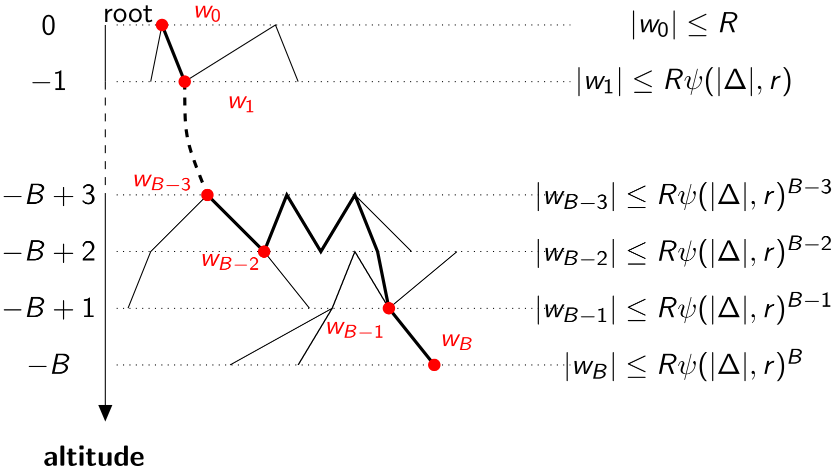
We have bounds on the maximal altitude and the size of the root.
Let R be the size of the root, $-B$ the minimal altitude.



$|w_0| \leq R$

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.



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

COVER for BNRA is decidable (and in class $\mathbf{F}_{\omega^\omega}$).

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 $\mathbf{F}_{\omega^\omega}$).

By contrast,

Theorem

TARGET is undecidable for BNRA.

A matching lower bound

Theorem

$\textsc{Cover}$ in BNRA is $\mathbf{F}_{\omega^\omega}$-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*:

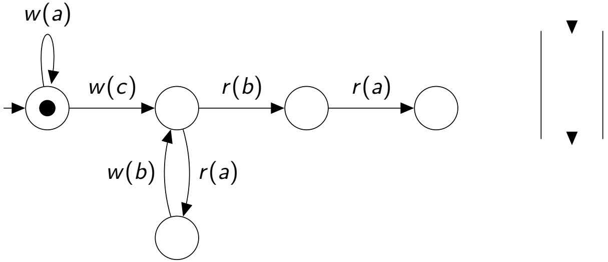
Theorem[5]

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

[5] Schnoebelen, Information Processing Letters '08

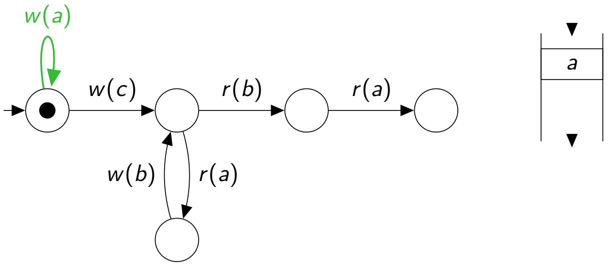
Lossy Channel Systems

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



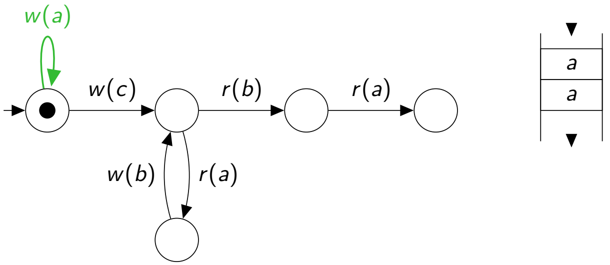
Lossy Channel Systems

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



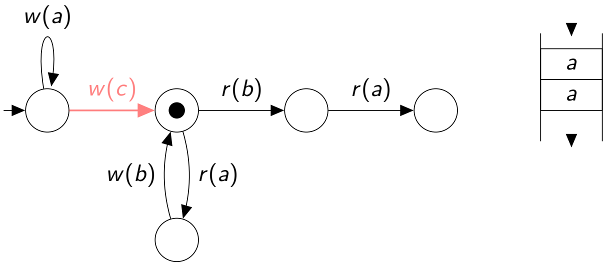
Lossy Channel Systems

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



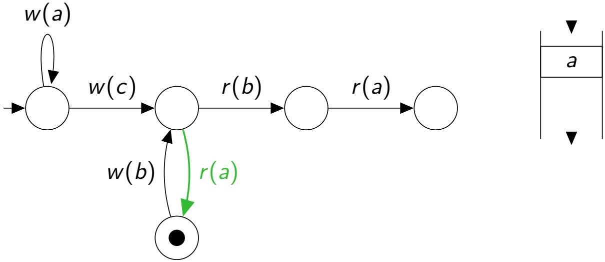
Lossy Channel Systems

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



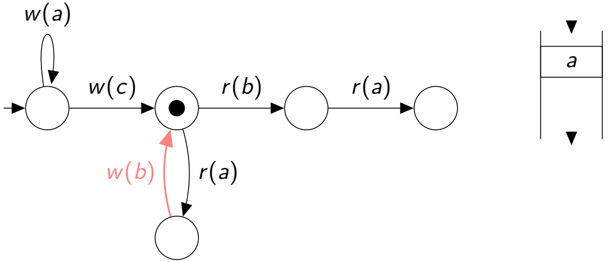
Lossy Channel Systems

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



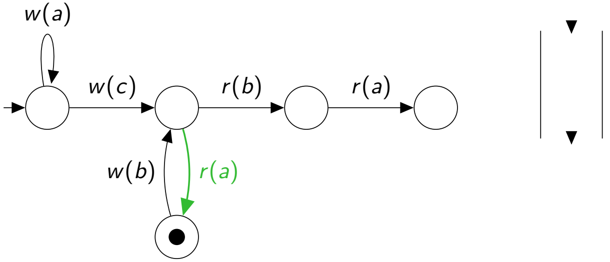
Lossy Channel Systems

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



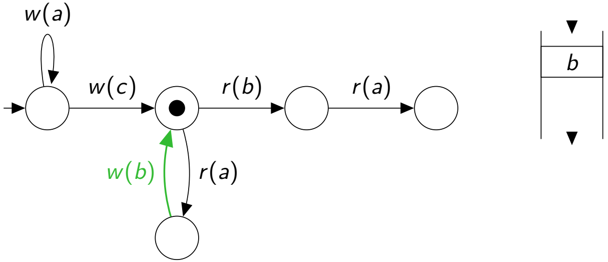
Lossy Channel Systems

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



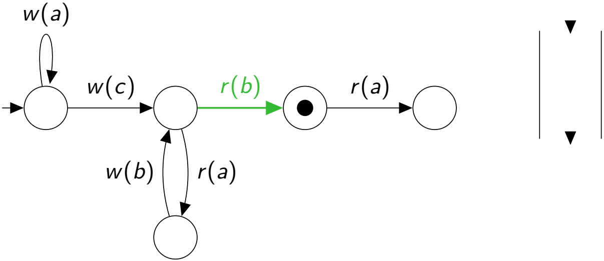
Lossy Channel Systems

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



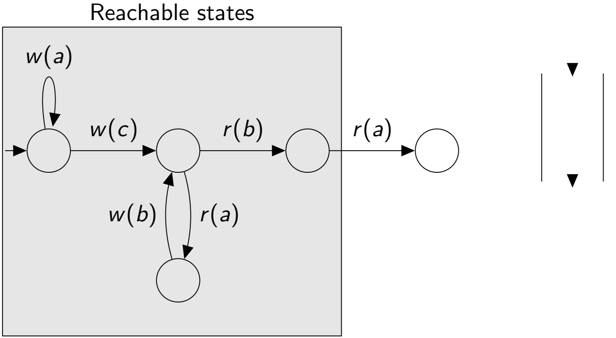
Lossy Channel Systems

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



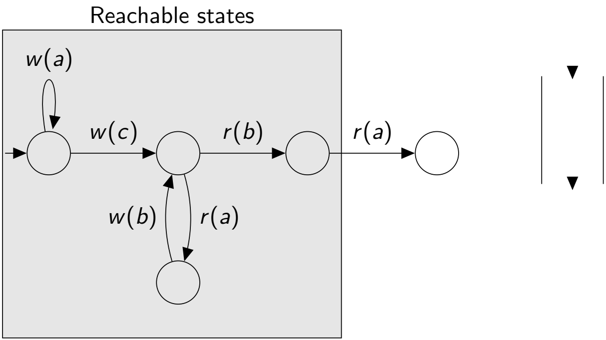
Lossy Channel Systems

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



Lossy Channel Systems

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

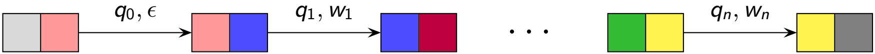


Lossy channel system reachability asks if one can reach a given state. This problem is decidable but has very high complexity: it is $\mathbf{F}_{\omega^\omega}$-complete.

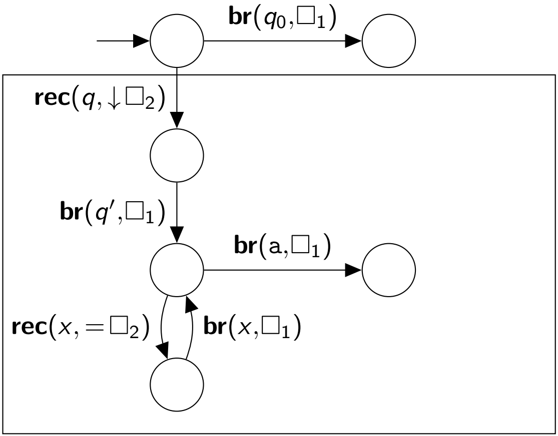
Encoding Lossy Channel Systems in BNRA

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

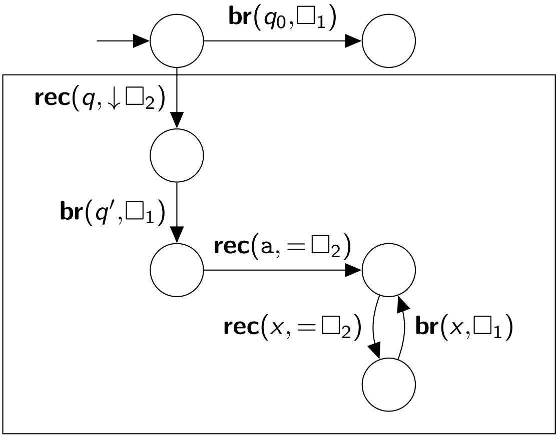


Encoding write transitions of Lossy Channel Systems



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 $q \xrightarrow{\mathtt{read(a)}} q'$ of the lossy channel system

Summary of complexity results

Theorem

COVER in BNRA is $\mathbf{F}_{\omega^\omega}$-complete.

Summary of complexity results

Theorem

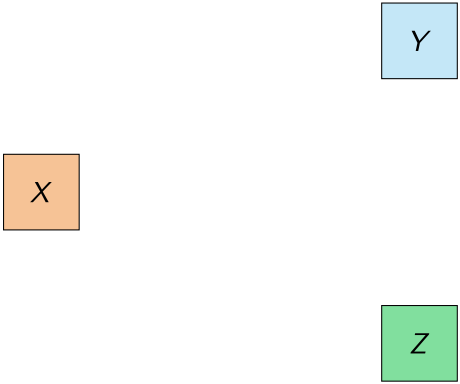
COVER in BNRA is $\mathbf{F}_{\omega^\omega}$-complete.

Theorem

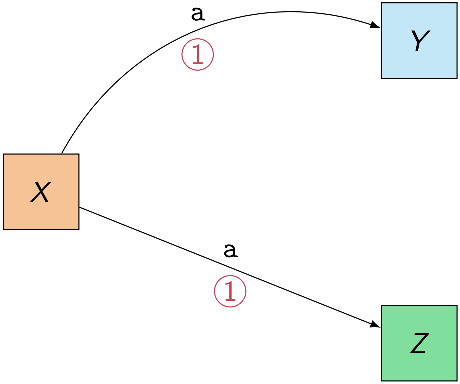
COVER for BNRA with one register per agent is NP-complete.

Thank you for your attention!

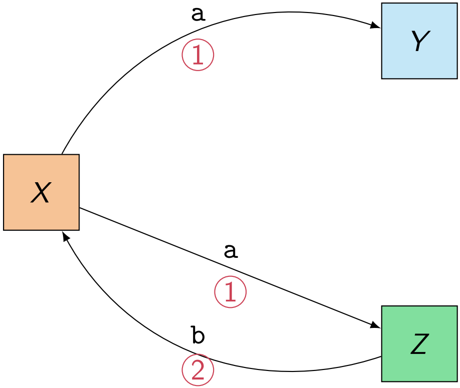
Turning the communication graph into a tree



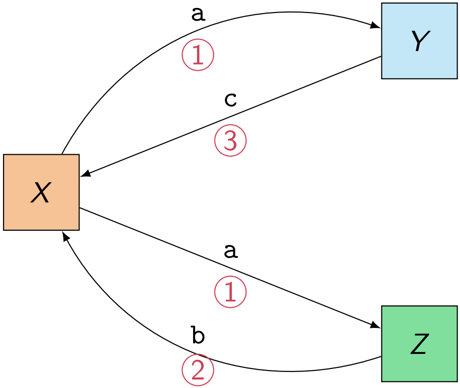
Turning the communication graph into a tree



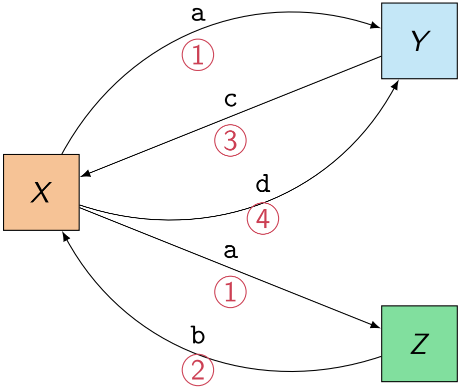
Turning the communication graph into a tree



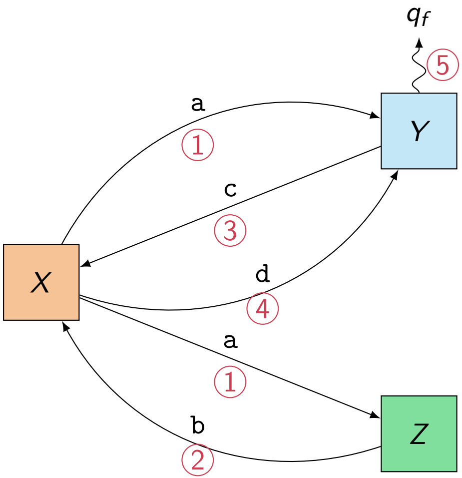
Turning the communication graph into a tree



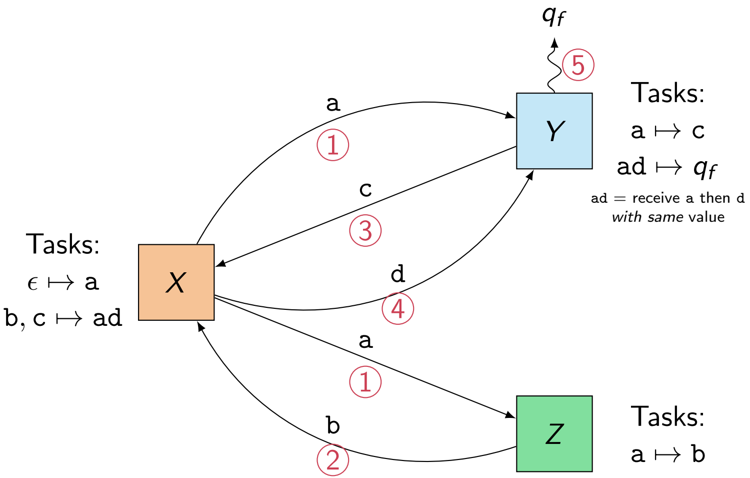
Turning the communication graph into a tree



Turning the communication graph into a tree



Turning the communication graph into a tree



Turning the communication graph into a tree