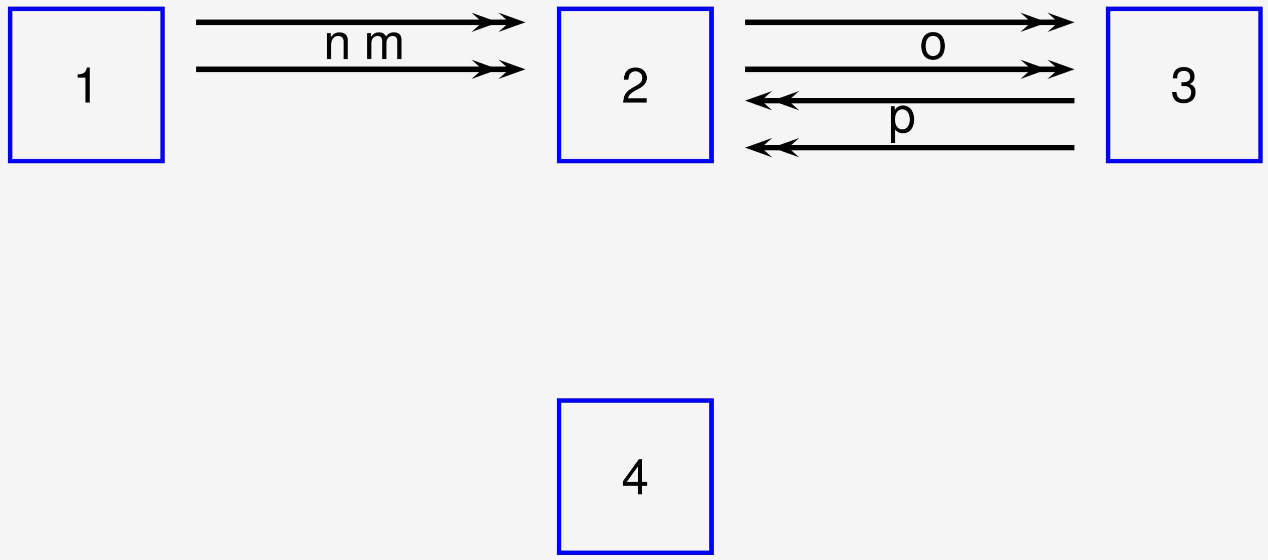
Model Checking Concurrent Systems with Unboundedly Many Processes Using Data Logics

Ahmet Kara

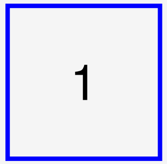
MOVEP 2012, Marseille

technische universität
dortmund

Interaction of Unboundedly Many Processes



Interaction of Unboundedly Many Processes



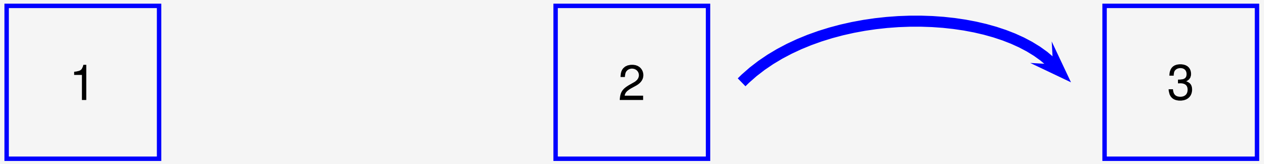
- A system run

Interaction of Unboundedly Many Processes

- A system run

spawn
1 **2**

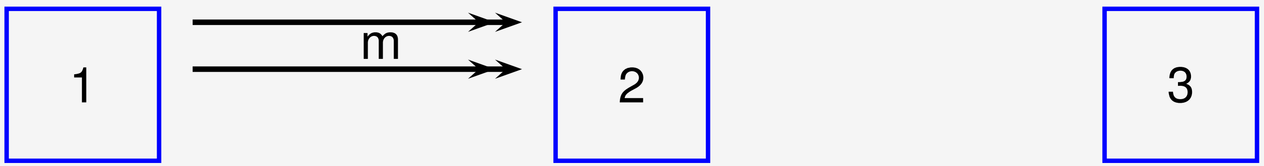
Interaction of Unboundedly Many Processes



- A system run

spawn	*spawn*
1	**2**
2	**3**

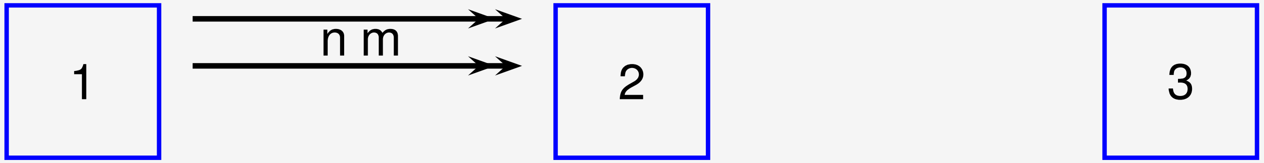
Interaction of Unboundedly Many Processes



- A system run

spawn	*spawn*	*se*(m)
$\mathbf{1}$	$\mathbf{2}$	$\mathbf{1}$
$\mathbf{2}$	$\mathbf{3}$	$\mathbf{2}$

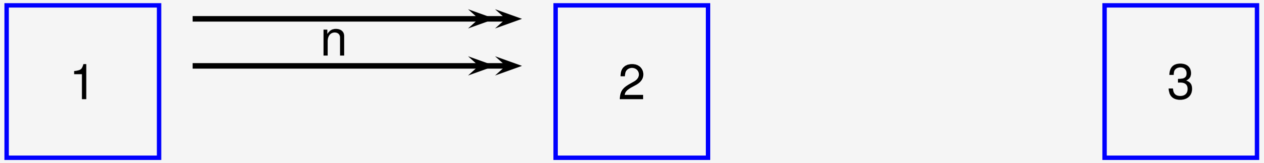
Interaction of Unboundedly Many Processes



- A system run

spawn	*spawn*	*se*(m)	*se*(n)
1	**2**	**1**	**1**
2	**3**	**2**	**2**

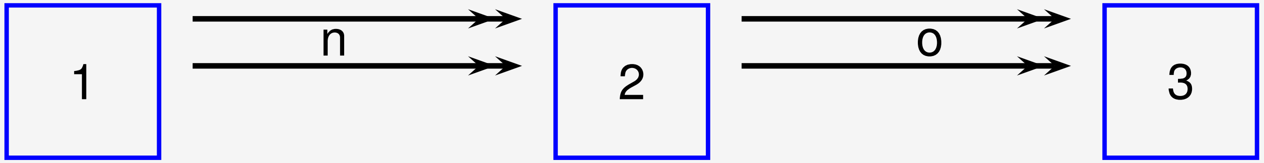
Interaction of Unboundedly Many Processes



- A system run

spawn	*spawn*	*se*(m)	*se*(n)	*rec*(m)
$\mathbf{1}$	$\mathbf{2}$	$\mathbf{1}$	$\mathbf{1}$	$\mathbf{2}$
$\mathbf{2}$	$\mathbf{3}$	$\mathbf{2}$	$\mathbf{2}$	$\mathbf{1}$

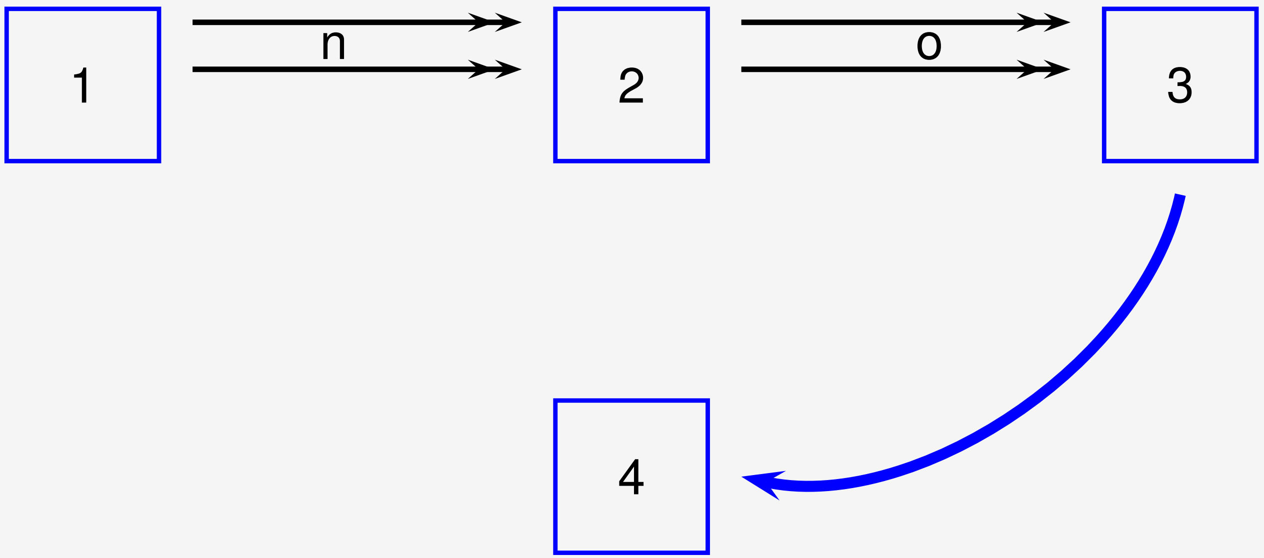
Interaction of Unboundedly Many Processes



- A system run

spawn	spawn	se(m)	se(n)	rec(m)	se(o)
1	2	1	1	2	2
2	3	2	2	1	3

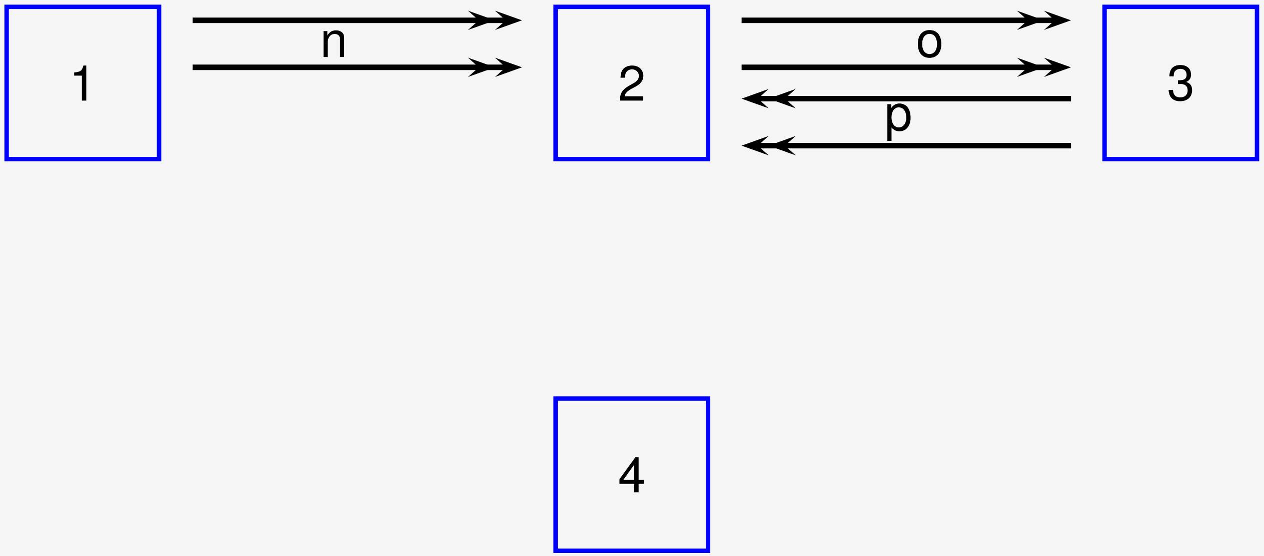
Interaction of Unboundedly Many Processes



- A system run

spawn	spawn	$se(m)$	$se(n)$	$rec(m)$	$se(o)$	spawn
$\mathbf{1}$ $\mathbf{2}$	$\mathbf{2}$ $\mathbf{3}$	$\mathbf{1}$ $\mathbf{2}$	$\mathbf{1}$ $\mathbf{2}$	$\mathbf{2}$ $\mathbf{1}$	$\mathbf{2}$ $\mathbf{3}$	$\mathbf{3}$ $\mathbf{4}$

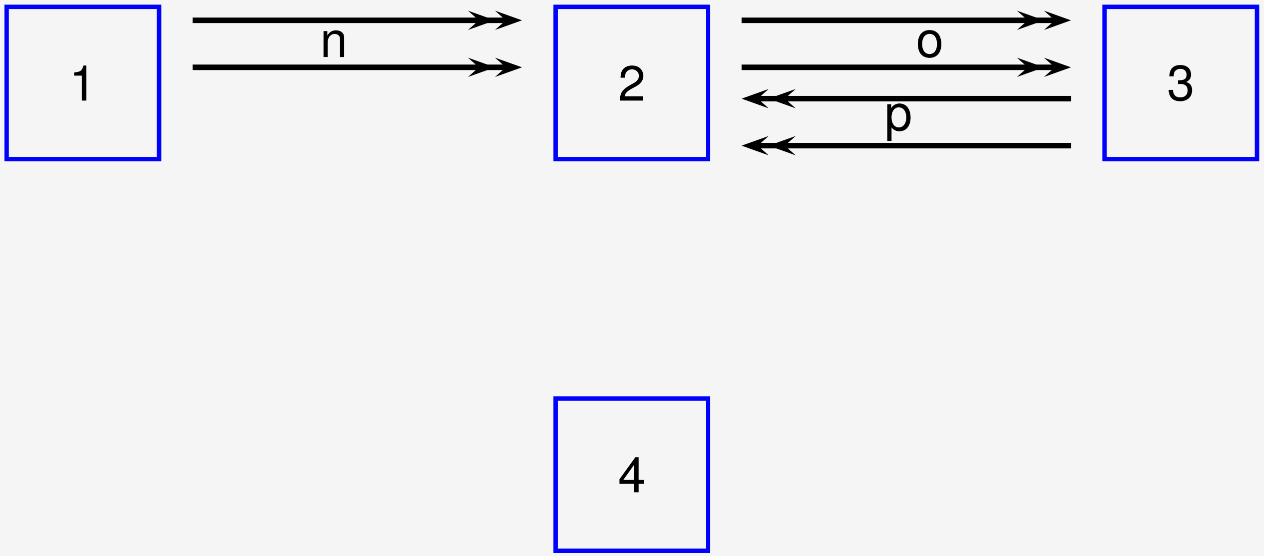
Interaction of Unboundedly Many Processes



- A system run

spawn	spawn	se(m)	se(n)	rec(m)	se(o)	spawn	se(p)
1 **2**	**2** **3**	**1** **2**	**1** **2**	**2** **1**	**2** **3**	**3** **4**	**3** **2**

Interaction of Unboundedly Many Processes



- A system run

spawn	*spawn*	*se*(m)	*se*(n)	*rec*(m)	*se*(o)	*spawn*	*se*(p)
$\mathbf{1}$	$\mathbf{2}$	$\mathbf{1}$	$\mathbf{1}$	$\mathbf{2}$	$\mathbf{2}$	$\mathbf{3}$	$\mathbf{3}$
$\mathbf{2}$	$\mathbf{3}$	$\mathbf{2}$	$\mathbf{2}$	$\mathbf{1}$	$\mathbf{3}$	$\mathbf{4}$	$\mathbf{2}$

- A system property

„Every sent message is received eventually."

$$\bigwedge_m \mathbf{G}(se(m) \rightarrow \downarrow x.\mathbf{F} rec(m) \wedge x_{@_1} \sim @_2 \wedge x_{@_2} \sim @_1)$$

Words and Data Words

A Word over $\Sigma = \{a, b, c\}$

c	c	a	c	a	b	c	b

Words and Data Words

A Word over $\Sigma = \{a, b, c\}$

c	c	a	c	a	b	c	b

A Data Word over $\Sigma = \{a, b, c\}$

c	c	a	c	a	b	c	b
1	4	3	2	2	3	7	2
7	4	2	5	9	1	3	2

Definition: Data Words

- Let
  - Σ be a finite alphabet
  - $\mathcal{D}$ be an infinite set of data values
- $w \in (\Sigma \times \mathcal{D}^m)^*$ is an m-dimensional data word over Σ

Words and Data Words

A Word over $\Sigma = \{a, b, c\}$

c	c	a	c	a	b	c	b

A Data Word over $\Sigma = \{a, b, c\}$

c	c	a	c	a	b	c	b
1 7	4 4	3 2	2 5	2 9	3 1	7 3	2 2

- $\mathcal{D} = \{1, 2, 3, \ldots\}$

Definition: Data Words

- Let
  - Σ be a finite alphabet
  - $\mathcal{D}$ be an infinite set of data values
- $w \in (\Sigma \times \mathcal{D}^m)^*$ is an m-dimensional data word over Σ

Logics on Data Words – Data Logics

- Even very weak logics on data words have an undecidable satisfiability problem.
  - First order logic with only three variables is not decidable [Bojańczyk et al. 06]
  - **LTL** is in general not decidable [Demri et al. 06]

→ Focus on restricted logics where the only predicate on data values is the equality relation

Logics on Data Words – Freeze LTL ($\mathbf{LTL}^{\Downarrow}$)

- Freeze LTL ($\mathbf{LTL}^{\Downarrow}$) [Demri et al. 06]:
  - contains the usual temporal operators like $\mathbf{X}, \mathbf{F}, \mathbf{U}, \ldots$
  - allows to put a variable x on a position
  - allows to compare the data values of the x-position with the data values of a current position

Logics on Data Words – Freeze LTL ($\mathrm{LTL}^{\Downarrow}$)

- Freeze LTL ($\mathbf{LTL}^{\Downarrow}$) [Demri et al. 06]:
  - contains the usual temporal operators like $\mathbf{X}, \mathbf{F}, \mathbf{U}, \ldots$
  - allows to put a variable x on a position
  - allows to compare the data values of the x-position with the data values of a current position

Example

"There is a b-position such that an a-position with the same data value follows somewhere in the future."

a	c	b	a	b	a	b	c
1	2	3	1	4	3	7	2

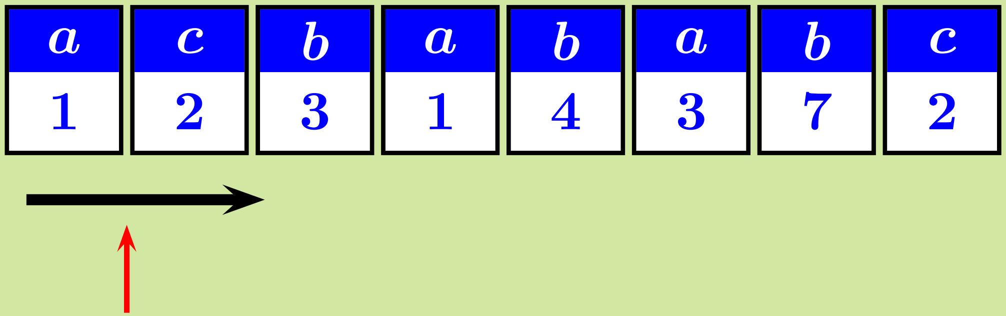
$$\mathbf{F}(b \wedge \downarrow x.\mathbf{F}(a \wedge x_{@_1} \sim @_1)))$$

Logics on Data Words – Freeze LTL ($\mathrm{LTL}^{\Downarrow}$)

- Freeze LTL ($\mathbf{LTL}^{\Downarrow}$) [Demri et al. 06]:
  - contains the usual temporal operators like $\mathbf{X}, \mathbf{F}, \mathbf{U}, \ldots$
  - allows to put a variable x on a position
  - allows to compare the data values of the x-position with the data values of a current position

Example

"There is a b-position such that an a-position with the same data value follows somewhere in the future."



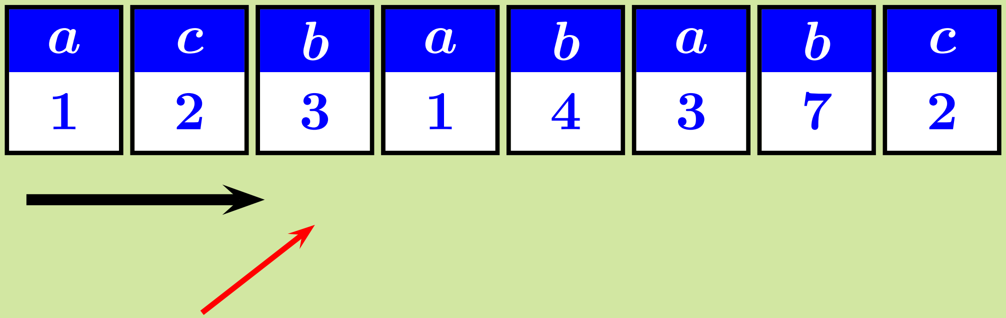
$$\mathbf{F}(b \wedge {\downarrow}x.\mathbf{F}(a \wedge x_{@_1} \sim @_1)))$$

Logics on Data Words – Freeze LTL ($\mathbf{LTL}^{\Downarrow}$)

- Freeze LTL ($\mathbf{LTL}^{\Downarrow}$) [Demri et al. 06]:
  - contains the usual temporal operators like $\mathbf{X}, \mathbf{F}, \mathbf{U}, \ldots$
  - allows to put a variable x on a position
  - allows to compare the data values of the x-position with the data values of a current position

Example

"There is a b-position such that an a-position with the same data value follows somewhere in the future."

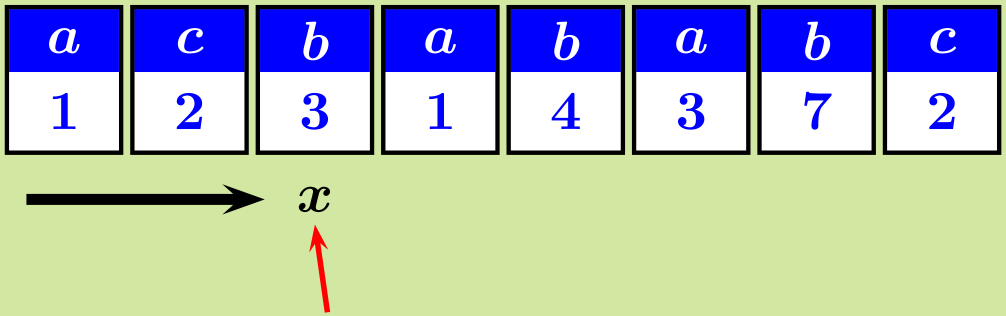


$$\mathbf{F}(b \wedge \downarrow x.\mathbf{F}(a \wedge x_{@_1} \sim @_1)))$$

- Freeze LTL ($\mathbf{LTL}^{\Downarrow}$) [Demri et al. 06]:
  - contains the usual temporal operators like $\mathbf{X}, \mathbf{F}, \mathbf{U}, \ldots$
  - allows to put a variable x on a position
  - allows to compare the data values of the x-position with the data values of a current position

Example

"There is a b-position such that an a-position with the same data value follows somewhere in the future."



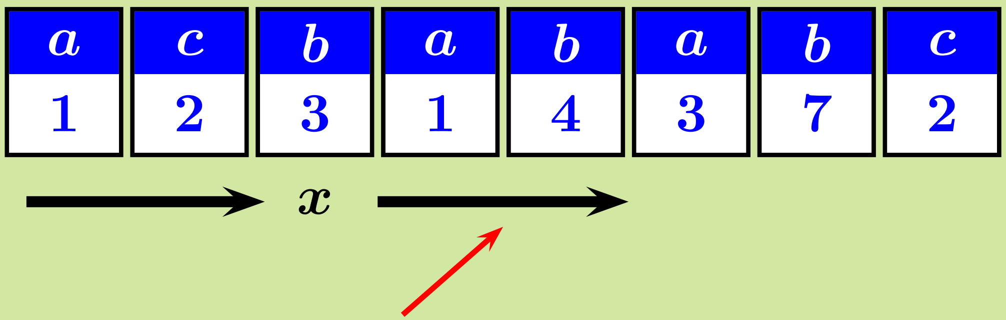
$$\mathbf{F}(b \wedge \downarrow x.\mathbf{F}(a \wedge x_{@_1} \sim @_1)))$$

Logics on Data Words – Freeze LTL ($\mathbf{LTL}^{\Downarrow}$)

- Freeze LTL ($\mathbf{LTL}^{\Downarrow}$) [Demri et al. 06]:
  - contains the usual temporal operators like $\mathbf{X}, \mathbf{F}, \mathbf{U}, \ldots$
  - allows to put a variable x on a position
  - allows to compare the data values of the x-position with the data values of a current position

Example

"There is a b-position such that an a-position with the same data value follows somewhere in the future."

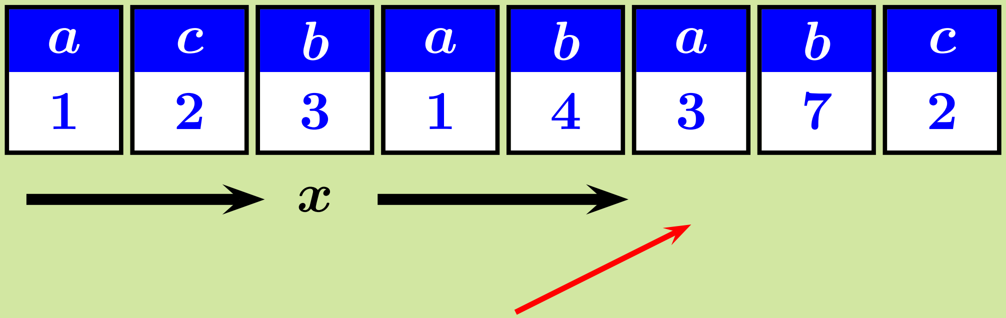


$$\mathbf{F}(b \wedge \downarrow x.\mathbf{F}(a \wedge x_{@_1} \sim @_1)))$$

- Freeze LTL ($\mathbf{LTL}^{\Downarrow}$) [Demri et al. 06]:
  - contains the usual temporal operators like $\mathbf{X}, \mathbf{F}, \mathbf{U}, \ldots$
  - allows to put a variable x on a position
  - allows to compare the data values of the x-position with the data values of a current position

Example

"There is a b-position such that an a-position with the same data value follows somewhere in the future."



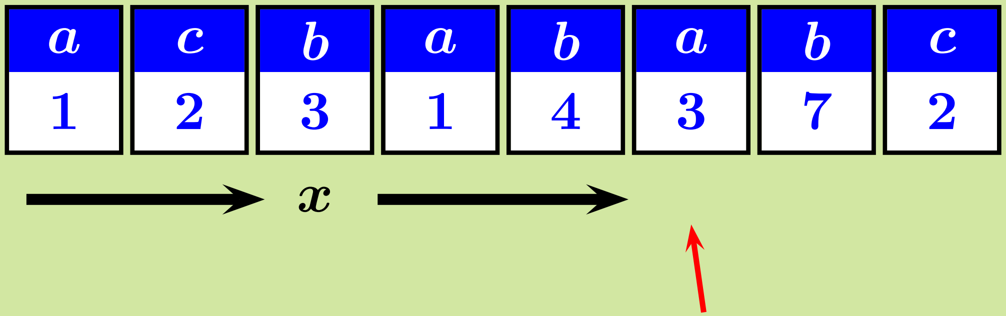
$$\mathbf{F}(b \wedge \downarrow x.\mathbf{F}(a \wedge x_{@_1} \sim @_1)))$$

Logics on Data Words – Freeze LTL ($\mathbf{LTL}^{\Downarrow}$)

- Freeze LTL ($\mathbf{LTL}^{\Downarrow}$) [Demri et al. 06]:
  - contains the usual temporal operators like $\mathbf{X}, \mathbf{F}, \mathbf{U}, \ldots$
  - allows to put a variable x on a position
  - allows to compare the data values of the x-position with the data values of a current position

Example

"There is a b-position such that an a-position with the same data value follows somewhere in the future."



$$\mathbf{F}(b \wedge \downarrow x.\mathbf{F}(a \wedge x_{@_1} \sim @_1)))$$

Logics on Data Words – Freeze LTL ($\mathbf{LTL}^{\Downarrow}$)

- Freeze LTL ($\mathbf{LTL}^{\Downarrow}$) [Demri et al. 06]:
  - contains the usual temporal operators like $\mathbf{X}, \mathbf{F}, \mathbf{U}, \ldots$
  - allows to put a variable x on a position
  - allows to compare the data values of the x-position with the data values of a current position

Example

"The first data value of every b-position is different from the second data value of its next position."

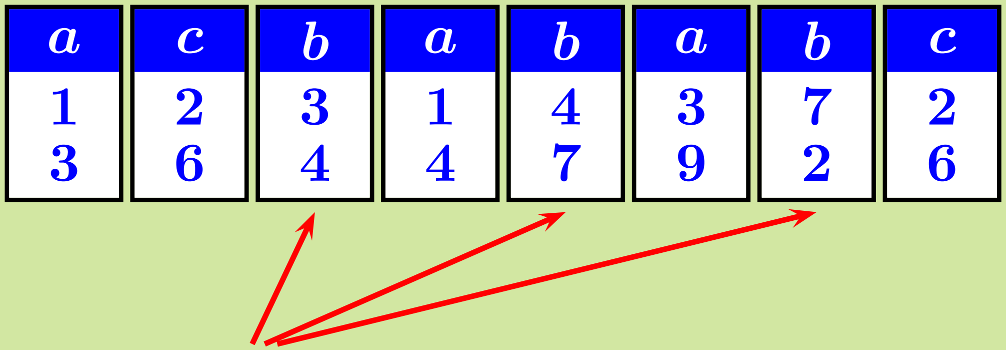
a	c	b	a	b	a	b	c
1	2	3	1	4	3	7	2
3	6	4	4	7	9	2	6

$$\mathbf{G}(b \rightarrow \downarrow x.\mathbf{X}(x_{@_1} \not\sim @_2))$$

- Freeze LTL ($\mathbf{LTL}^{\Downarrow}$) [Demri et al. 06]:
  - contains the usual temporal operators like $\mathbf{X}, \mathbf{F}, \mathbf{U}, \ldots$
  - allows to put a variable x on a position
  - allows to compare the data values of the x-position with the data values of a current position

Example

"The first data value of every b-position is different from the second data value of its next position."



$$\mathbf{G}(b \rightarrow \downarrow x.\mathbf{X}(x_{@_1} \not\sim @_2))$$

- Freeze LTL ($\mathbf{LTL}^{\Downarrow}$) [Demri et al. 06]:
  - contains the usual temporal operators like $\mathbf{X}, \mathbf{F}, \mathbf{U}, \ldots$
  - allows to put a variable x on a position
  - allows to compare the data values of the x-position with the data values of a current position

Example

"The first data value of every b-position is different from the second data value of its next position."

a	c	b	a	b	a	b	c
1	2	3	1	4	3	7	2
3	6	4	4	7	9	2	6
		x		x		x	

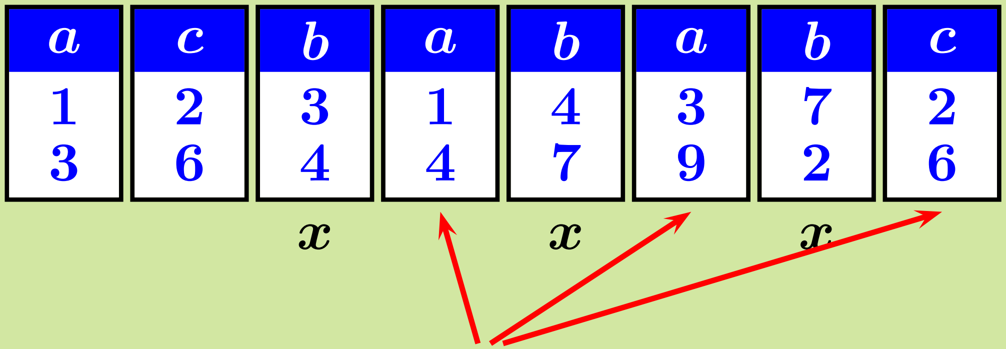
$$\mathbf{G}(b \rightarrow \downarrow x.\mathbf{X}(x_{@_1} \not\sim @_2))$$

Logics on Data Words – Freeze LTL ($\mathrm{LTL}^{\Downarrow}$)

- Freeze LTL ($\mathbf{LTL}^{\Downarrow}$) [Demri et al. 06]:
  - contains the usual temporal operators like $\mathbf{X}, \mathbf{F}, \mathbf{U}, \ldots$
  - allows to put a variable x on a position
  - allows to compare the data values of the x-position with the data values of a current position

Example

"The first data value of every b-position is different from the second data value of its next position."



$$\mathbf{G}(b \rightarrow \downarrow x.\mathbf{X}(x_{@_1} \not\sim @_2))$$

Logics on Data Words – Freeze LTL ($\mathbf{LTL}^{\Downarrow}$)

- Freeze LTL ($\mathbf{LTL}^{\Downarrow}$) [Demri et al. 06]:
  - contains the usual temporal operators like $\mathbf{X}, \mathbf{F}, \mathbf{U}, \ldots$
  - allows to put a variable x on a position
  - allows to compare the data values of the x-position with the data values of a current position

Example

"The first data value of every b-position is different from the second data value of its next position."

a	c	b	a	b	a	b	c
1	2	3	1	4	3	7	2
3	6	4	4	7	9	2	6
		x		x		x	

$$\mathbf{G}(b \rightarrow {\downarrow}x.\mathbf{X}(x_{@_1} \not\sim @_2))$$

Logics on Data Words – Freeze LTL ($\mathrm{LTL}^{\Downarrow}$)

- Freeze LTL ($\mathbf{LTL}^{\Downarrow}$) [Demri et al. 06]:
  - contains the usual temporal operators like $\mathbf{X}, \mathbf{F}, \mathbf{U}, \ldots$
  - allows to put a variable x on a position
  - allows to compare the data values of the x-position with the data values of a current position

Example

"The first data value of every b-position is different from the second data value of its next position."

a	c	b	a	b	a	b	c
1	2	3	1	4	3	7	2
3	6	4	4	7	9	2	6
		x		x		x	

$$\mathbf{G}(b \rightarrow \downarrow x.\mathbf{X}(x_{@_1} \not\sim @_2))$$

Theorem [Demri et al. 06]

- Satisfiability is decidable on
  - 1-dimensional data words if
  - only one variable and
  - only future operators are used.
- Complexity: not primitive recursive

Logics on Data Words – Freeze LTL ($\mathrm{LTL}^{\Downarrow}$)

- Freeze LTL ($\mathbf{LTL}^{\Downarrow}$) [Demri et al. 06]:
  - contains the usual temporal operators like $\mathbf{X}, \mathbf{F}, \mathbf{U}, \ldots$
  - allows to put a variable x on a position
  - allows to compare the data values of the x-position with the data values of a current position

Example

"The first data value of every b-position is different from the second data value of its next position."

a	c	b	a	b	a	b	c
1	2	3	1	4	3	7	2
3	6	4	4	7	9	2	6
		x		x		x	

$$\mathbf{G}(b \rightarrow \downarrow x.\mathbf{X}(x_{@_1} \not\sim @_2))$$

Theorem [Demri et al. 06]

- Satisfiability is decidable on
  - 1-dimensional data words if
  - only one variable and
  - only future operators are used.
- Complexity: not primitive recursive
- Satisfiability is undecidable if
  - more than one variable or
  - past operators are added.

Logics on Data Words – Data LTL

- Data LTL [K. et al. 06]:
  - allows navigation on consecutive position via $\mathbf{X}, \mathbf{F}, \mathbf{U}, \ldots$
  - allows navigation on positions carrying the same data value via $\mathbf{X}^{=}, \mathbf{F}^{=}, \mathbf{U}^{=}, \ldots$

Logics on Data Words – Data LTL

- Data LTL [K. et al. 06]:
  - allows navigation on consecutive position via $\mathbf{X}, \mathbf{F}, \mathbf{U}, \ldots$
  - allows navigation on positions carrying the same data value via $\mathbf{X}^{=}, \mathbf{F}^{=}, \mathbf{U}^{=}, \ldots$

Example

"There is some position such that on the subword induced by its first data value it holds a until b."

d	a	c	a	a	b	b	c
3	3	5	4	6	4	3	5
4	6	7	3	3	2	4	2

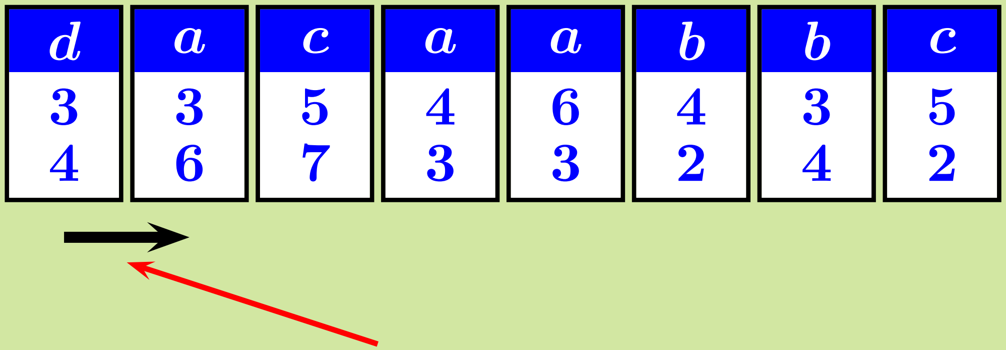
$$\mathbf{F}\mathcal{C}_{@_1}(a\mathbf{U}^{=}b)$$

Logics on Data Words – Data LTL

- Data LTL [K. et al. 06]:
  - allows navigation on consecutive position via $\mathbf{X}, \mathbf{F}, \mathbf{U}, \ldots$
  - allows navigation on positions carrying the same data value via $\mathbf{X}^{=}, \mathbf{F}^{=}, \mathbf{U}^{=}, \ldots$

Example

"There is some position such that on the subword induced by its first data value it holds a until b."



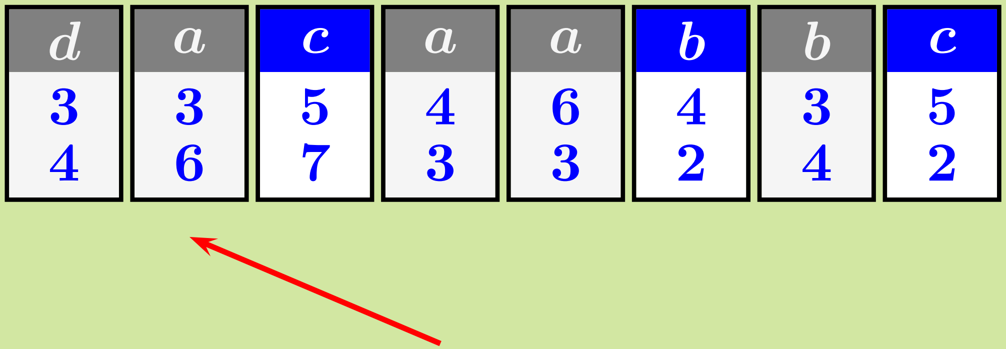
$\mathbf{F}\mathcal{C}_{@_1}(a\mathbf{U}^{=}b)$

Logics on Data Words – Data LTL

- Data LTL [K. et al. 06]:
  - allows navigation on consecutive position via $\mathbf{X}, \mathbf{F}, \mathbf{U}, \ldots$
  - allows navigation on positions carrying the same data value via $\mathbf{X}^{=}, \mathbf{F}^{=}, \mathbf{U}^{=}, \ldots$

Example

"There is some position such that on the subword induced by its first data value it holds a until b."



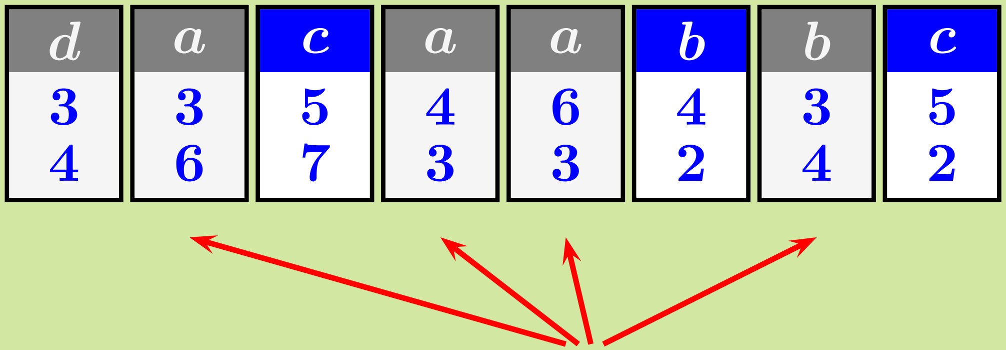
$$\mathbf{F}\mathcal{C}_{@_1}(a\mathbf{U}^{=}b)$$

Logics on Data Words – Data LTL

- Data LTL [K. et al. 06]:
  - allows navigation on consecutive position via $\mathbf{X}, \mathbf{F}, \mathbf{U}, \ldots$
  - allows navigation on positions carrying the same data value via $\mathbf{X}^{=}, \mathbf{F}^{=}, \mathbf{U}^{=}, \ldots$

Example

"There is some position such that on the subword induced by its first data value it holds a until b."



$$\mathbf{F}\mathcal{C}_{@_1}(a\mathbf{U}^{=}b)$$

Logics on Data Words – Data LTL

- Data LTL [K. et al. 06]:
  - allows navigation on consecutive position via $\mathbf{X}, \mathbf{F}, \mathbf{U}, \ldots$
  - allows navigation on positions carrying the same data value via $\mathbf{X}^{=}, \mathbf{F}^{=}, \mathbf{U}^{=}, \ldots$

Example

"There is some position such that on the subword induced by its first data value it holds $\boldsymbol{a}$ until $\boldsymbol{b}$."

d	a	c	a	a	b	b	c
3	3	5	4	6	4	3	5
4	6	7	3	3	2	4	2

$$\mathbf{F}\mathcal{C}_{@_1}(a\mathbf{U}^{=}b)$$

Theorem ([K. et al. 06])

- Satisfiability is decidable on
  - multi-dimensional data words with
  - future and past operators.
- Precise complexity not known but presumably very bad.
- Satisfiability is undecidable if
  - navigation along tuples is allowed

Investigations on Data Logics

- In many papers it is mentioned that system verification is one of the main motivations for the investigation of data logics:
  - Data values can be used to represent process IDs and data words to represent system runs.
  - Data logics can be used to specify system properties.
- Nevertheless, the most investigated question is rather satisfiability than model checking.

Our Main Motivation

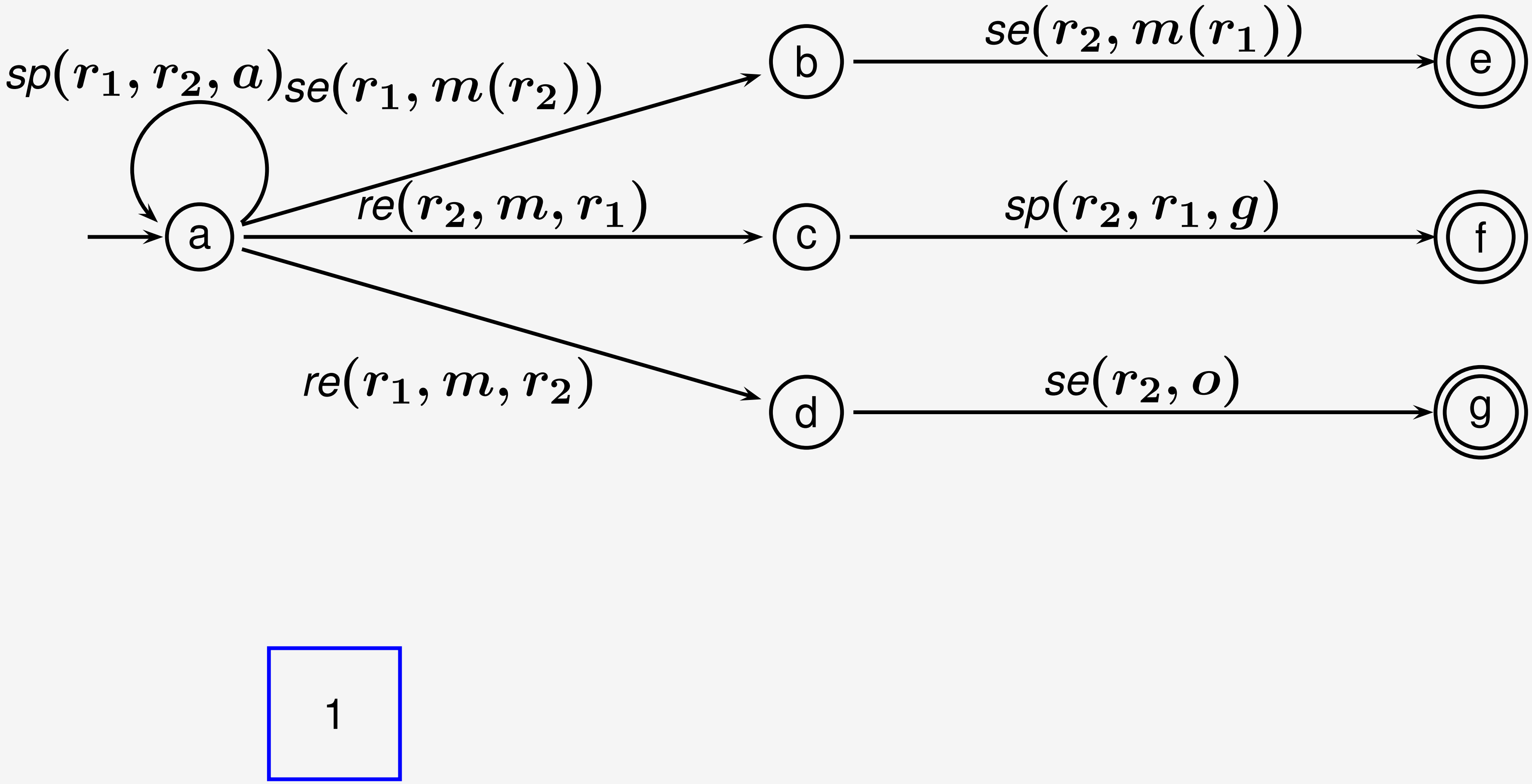
- We want to consider the model checking problem with data logics on models which
  - describe the behavior of concurrent systems with unboundedly many processes, and
  - produce system runs which can be represented by data words if process IDs are identified by data values.

→ Model Checking on models producing restricted data words can deliver good decidability and complexity results.

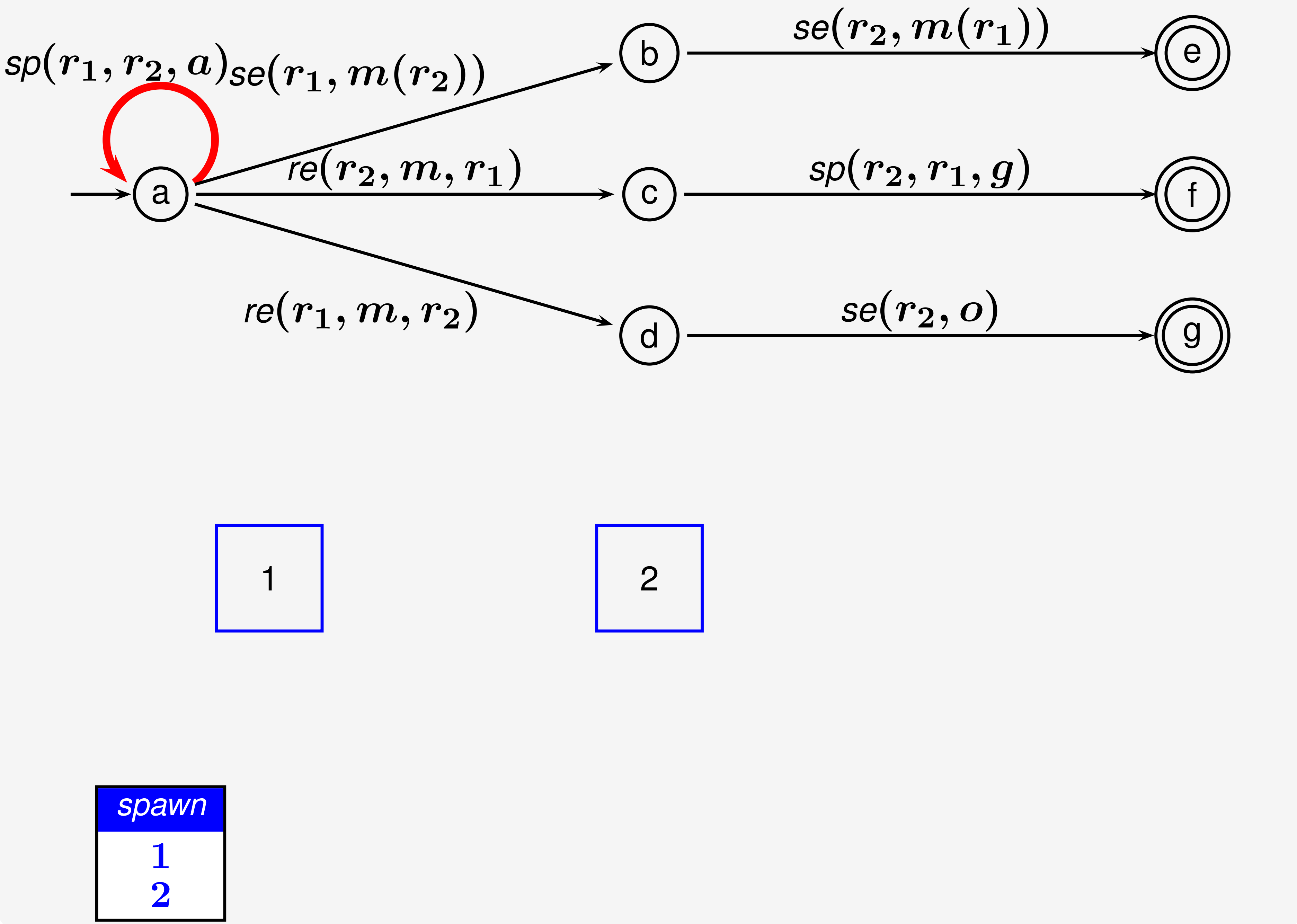
Dynamic Communicating Automata (DCA)

- Introduced by [Bollig and Hélouët 10]
- Extension of communicating finite state machines [Brand and Zafiropulo 83]
- Allows the creation of fresh processes
- Communication between processes through communication channels
- Maintenance of communication by storing process ID in registers

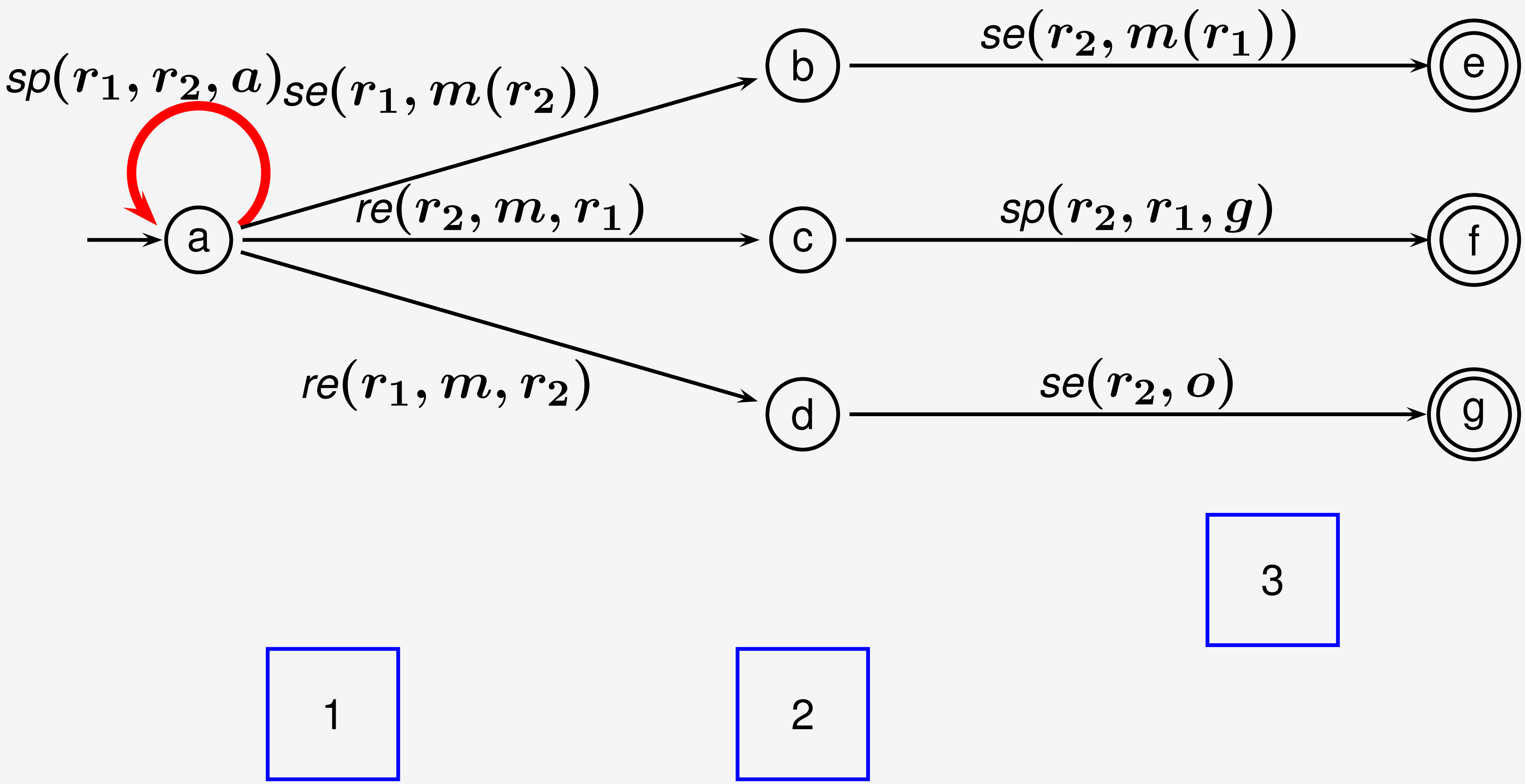
A 2-variable DCA



A 2-variable DCA

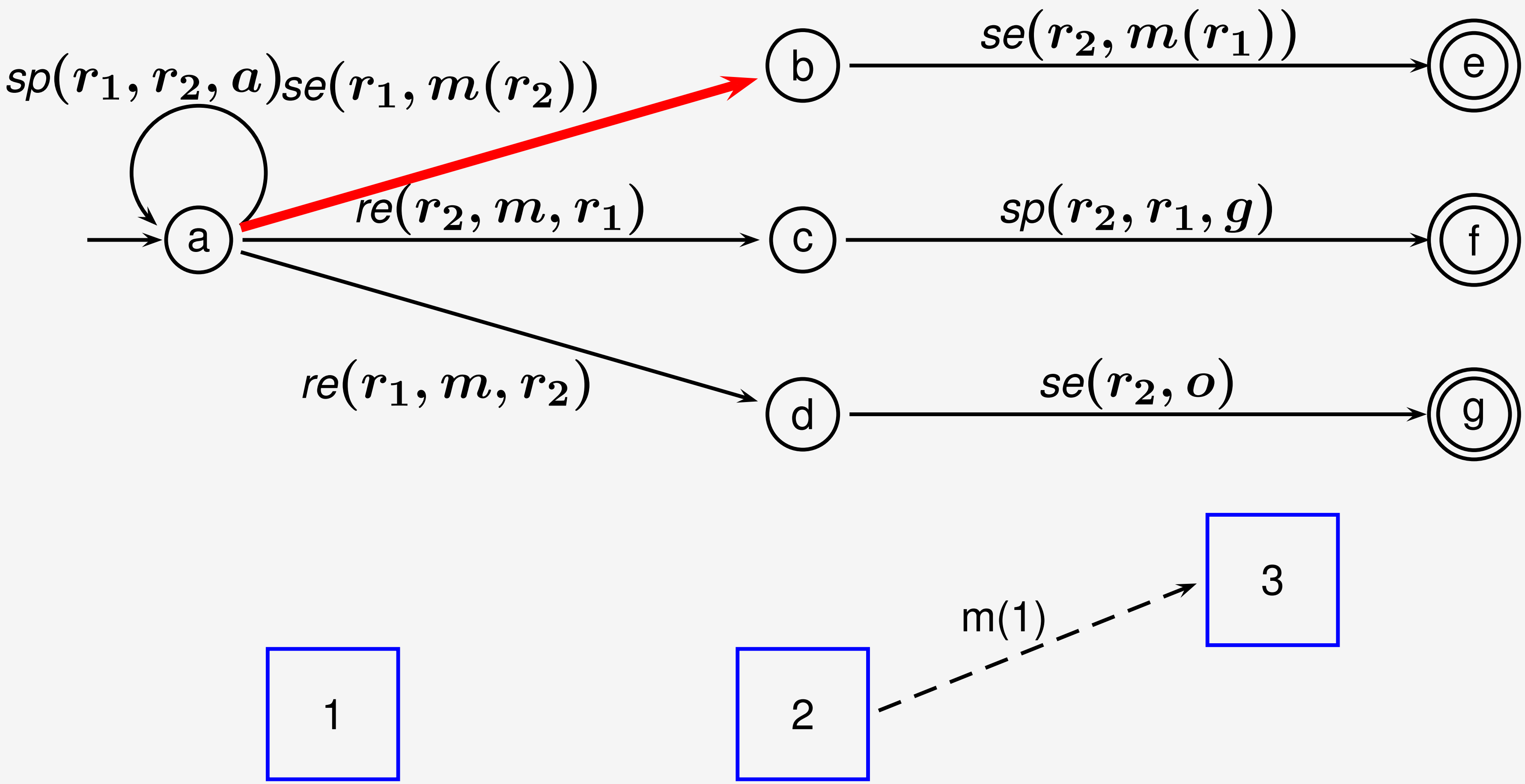


A 2-variable DCA



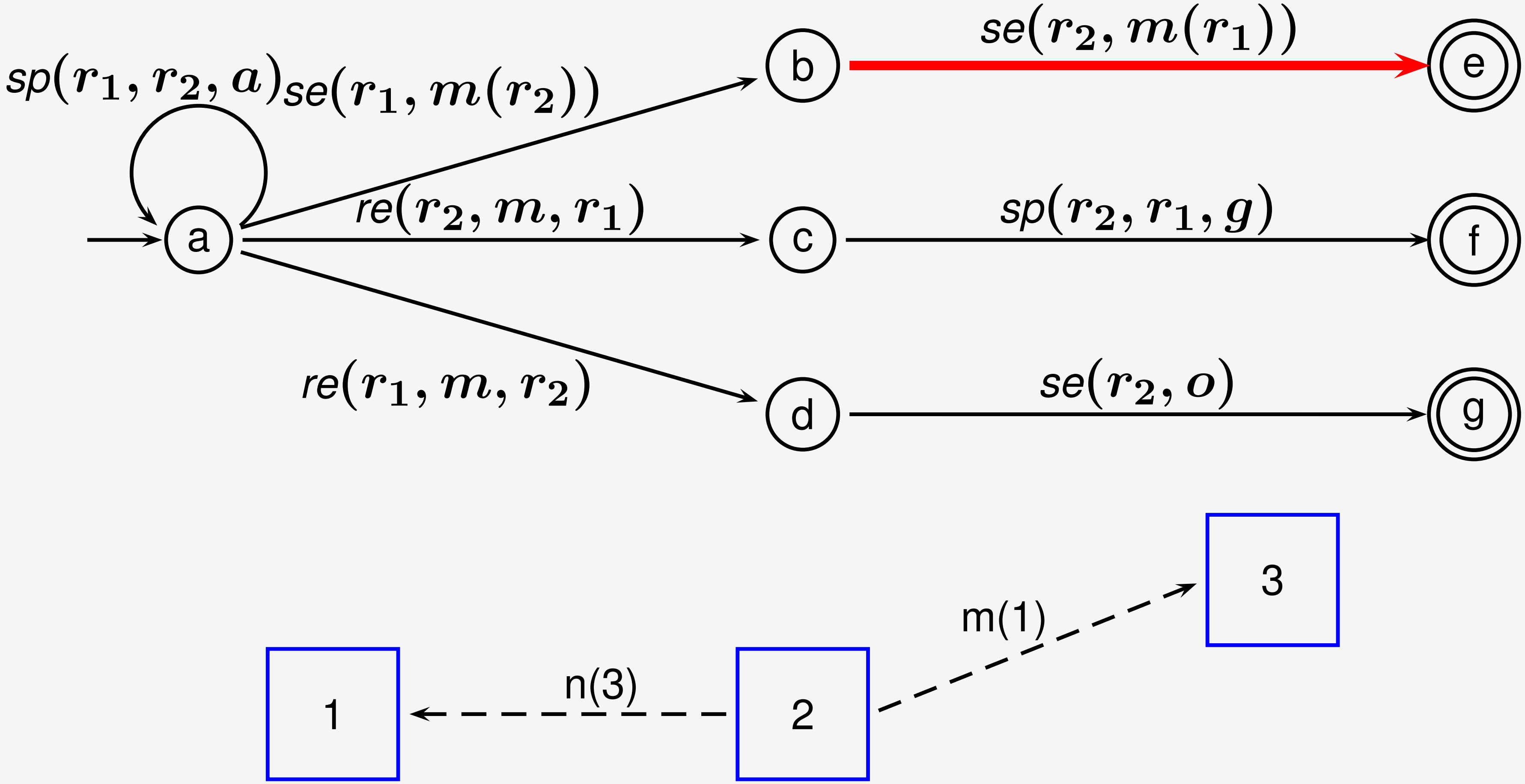
spawn	spawn
1 2	2 3

A 2-variable DCA



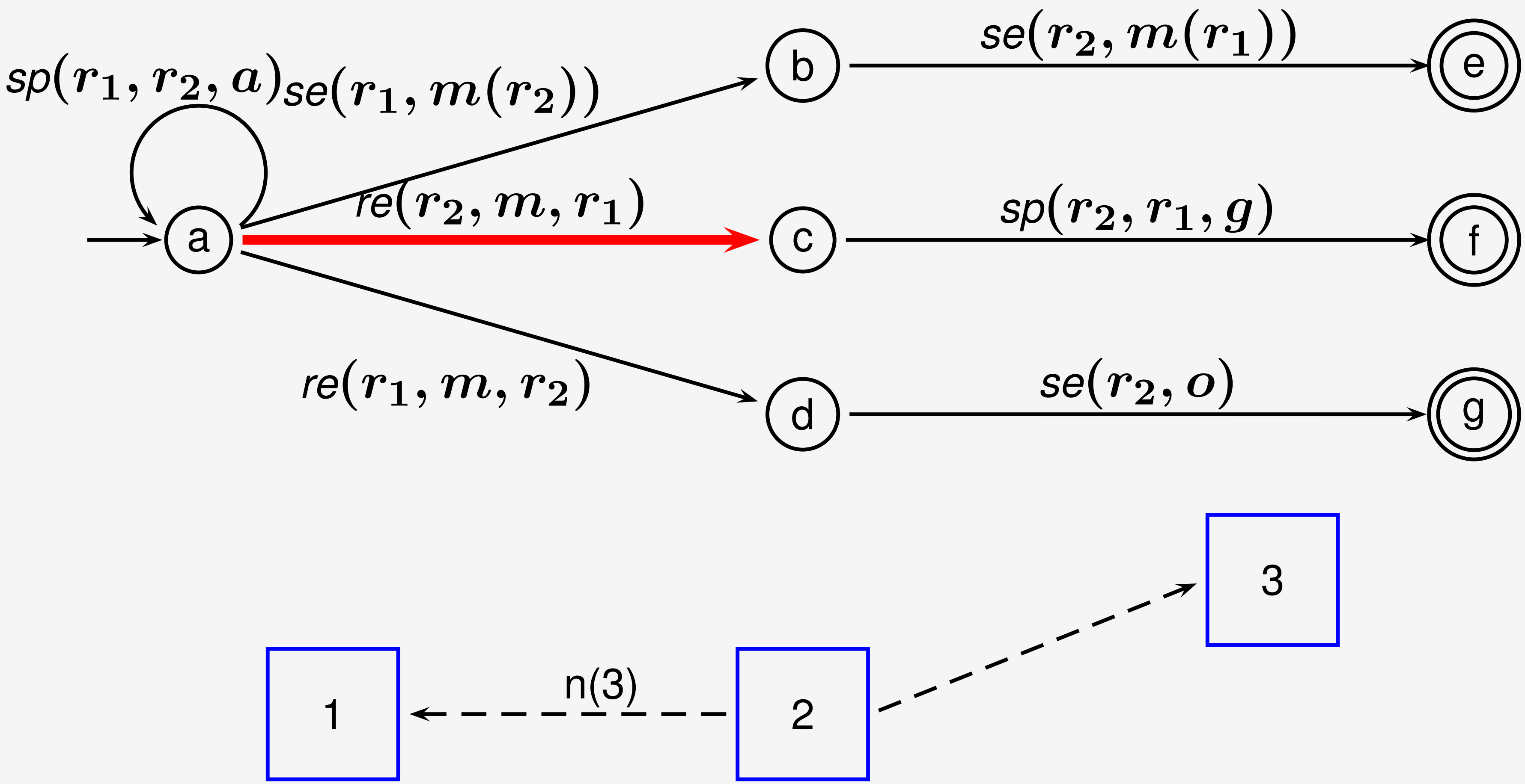
spawn	*spawn*	*se*(m)
1	2	2
2	3	3

A 2-variable DCA



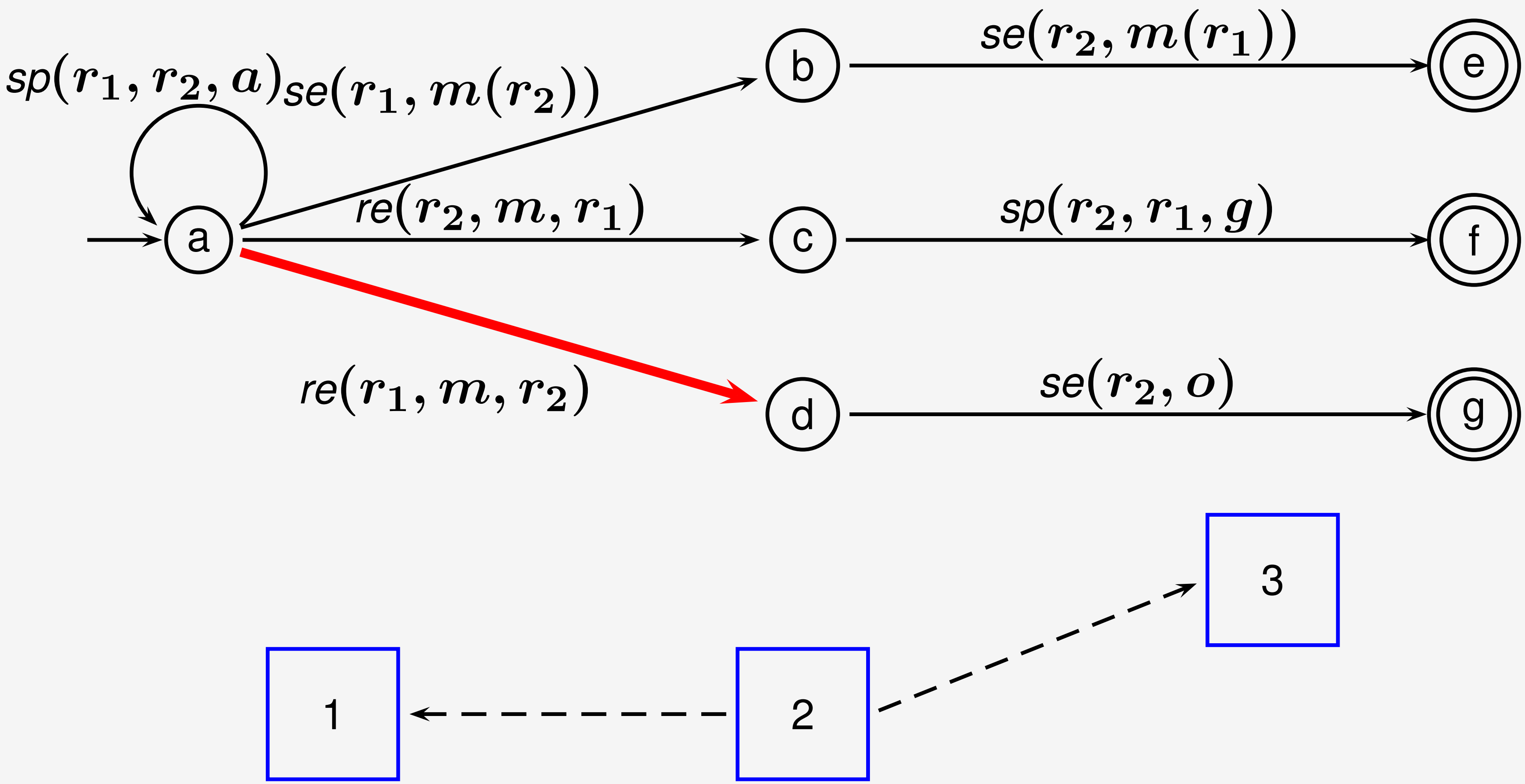
spawn	spawn	$se(m)$	$se(n)$
1	2	2	2
2	3	3	1

A 2-variable DCA



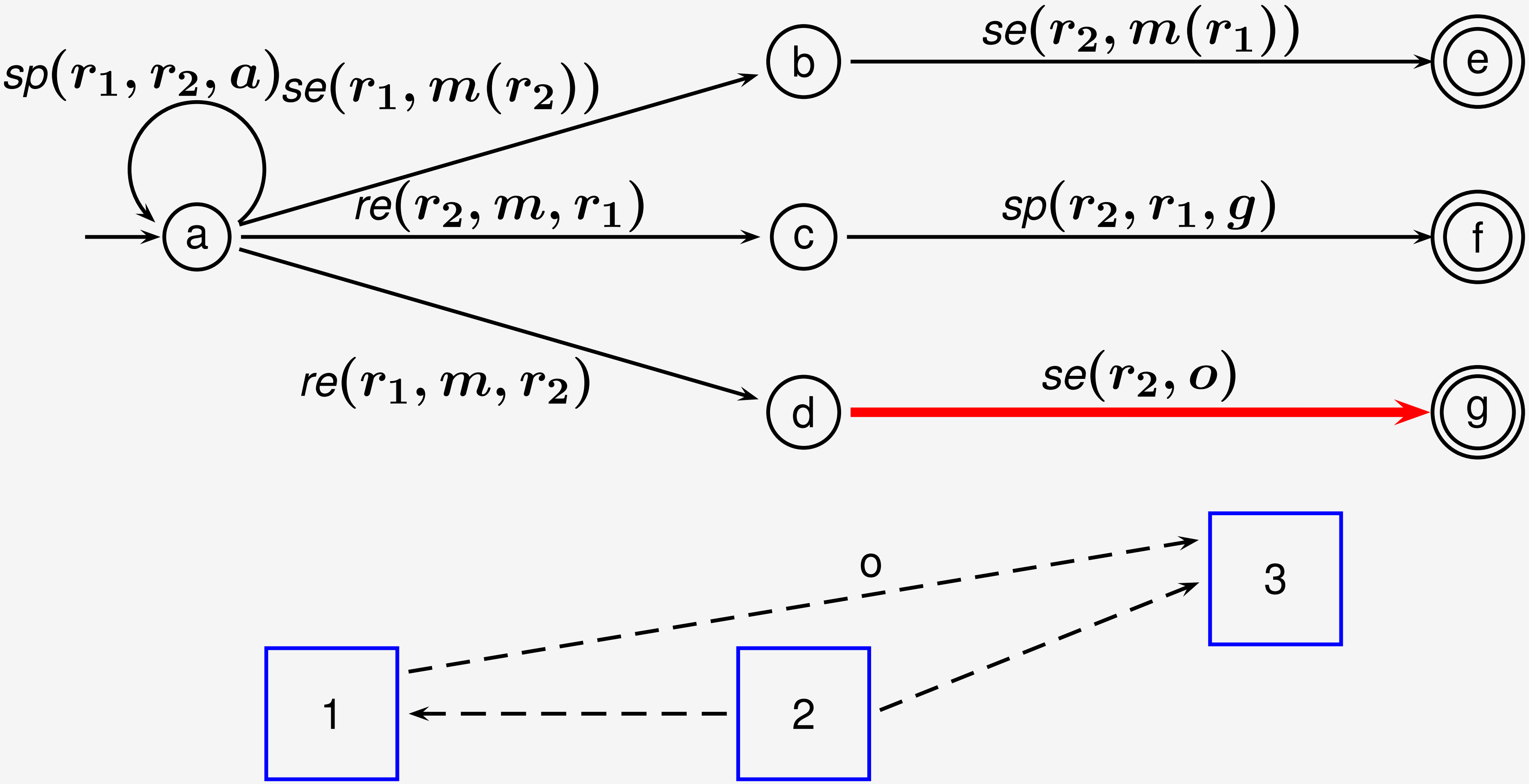
spawn	*spawn*	$se(m)$	$se(n)$	$rec(m)$
1	**2**	**2**	**2**	**3**
2	**3**	**3**	**1**	**2**

A 2-variable DCA



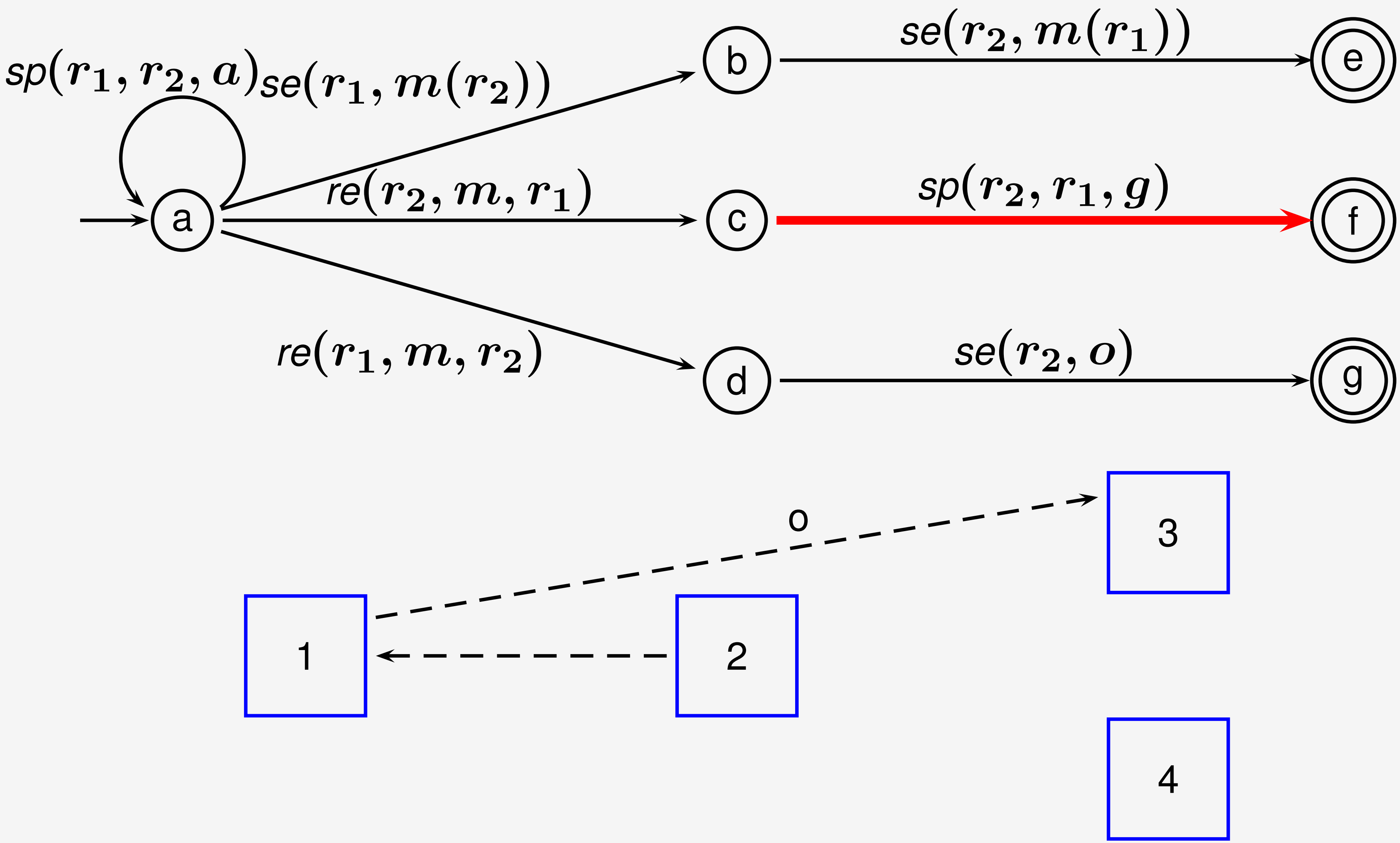
spawn	*spawn*	*se*(m)	*se*(n)	*rec*(m)	*rec*(n)
1	**2**	**2**	**2**	**3**	**1**
2	**3**	**3**	**1**	**2**	**2**

A 2-variable DCA



spawn	*spawn*	*se*(m)	*se*(n)	*rec*(m)	*rec*(n)	*se*(o)
1	2	2	2	3	1	1
2	3	3	1	2	2	2

A 2-variable DCA



spawn	*spawn*	*se*(m)	*se*(n)	*rec*(m)	*rec*(n)	*se*(o)	*spawn*
1	2	2	2	3	1	1	3
2	3	3	1	2	2	2	4

The Model Checking Problem on DCA

The Model Checking Problem on DCA

Given: A DCA $\mathcal{A}$ and a formula φ of a data logic
Question: Does φ hold on all accepting runs of $\mathcal{A}$?

First Insights – Undecidability Results

Theorem

The nonemptiness problem for 2-variable-DCA with bounded channels is undecidable.

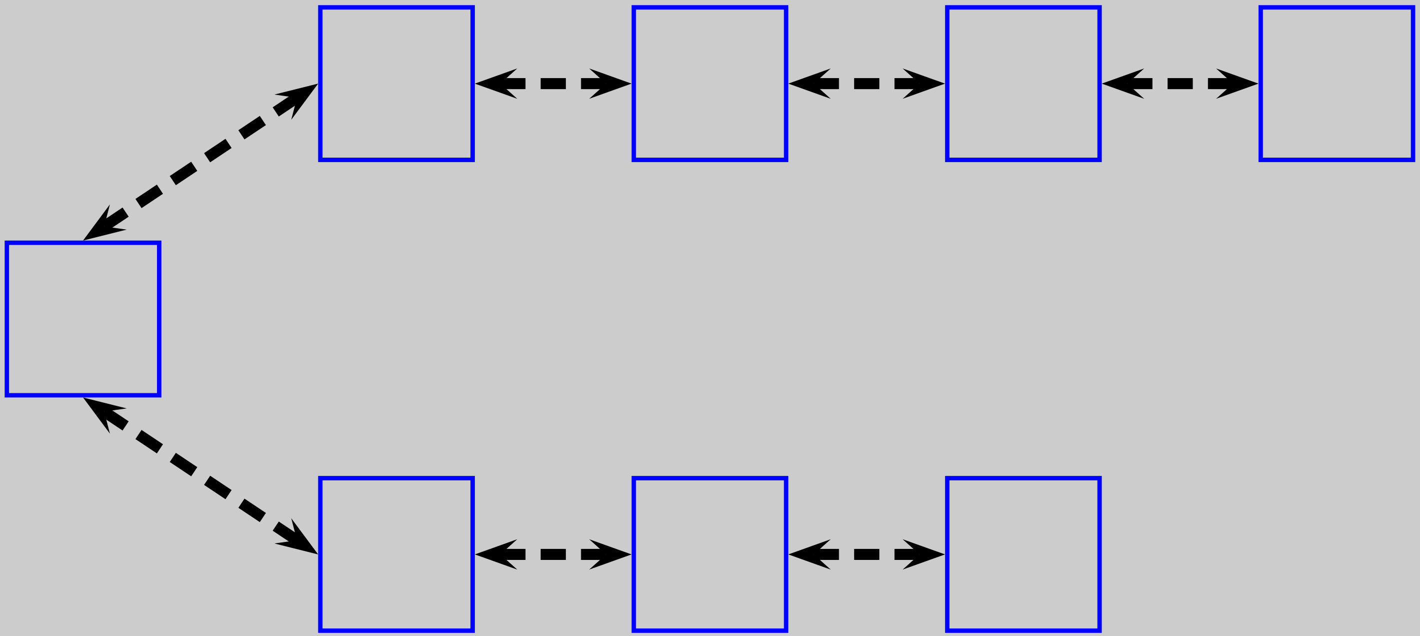
First Insights – Undecidability Results

Theorem

The nonemptiness problem for 2-variable-DCA with bounded channels is undecidable.

Proof idea

- By reduction from the nonemptiness problem for 2-counter automata.
  - A chain of processes can represent a counter value.



First Insights – A Decidability Result

Theorem

The model checking problem for 1-variable-DCA with bounded channel capacities and Data LTL is decidable.

First Insights – A Decidability Result

Theorem

The model checking problem for 1-variable-DCA with bounded channel capacities and Data LTL is decidable.

Proof idea

1. Given a DCA $\mathcal{A}$ and a formula φ we decide whether there is an accepting run satisfying $\neg\varphi$.

First Insights – A Decidability Result

Theorem

The model checking problem for 1-variable-DCA with bounded channel capacities and Data LTL is decidable.

Proof idea

1. Given a DCA $\mathcal{A}$ and a formula φ we decide whether there is an accepting run satisfying $\neg\varphi$.
2. Reduction to a reachability problem in an infinite state system.

$$\begin{array}{c} \mathcal{H} \\ (a_1, \mathcal{F}_1, i_1) \\ \vdots \\ (a_n, \mathcal{F}_n, i_n) \\ (b_1, \mathcal{G}_1, c_1, \mathcal{H}_n, i'_1) \\ \vdots \\ (b_m, \mathcal{G}_m, c_m, \mathcal{H}_m, i'_m) \end{array}$$

First Insights – A Decidability Result

Theorem

The model checking problem for 1-variable-DCA with bounded channel capacities and Data LTL is decidable.

Proof idea

1. Given a DCA $\mathcal{A}$ and a formula φ we decide whether there is an accepting run satisfying $\neg\varphi$.
2. Reduction to a reachability problem in an infinite state system.

$\mathcal{H}_0$

$(a_0, \mathcal{F}_0, 1)$

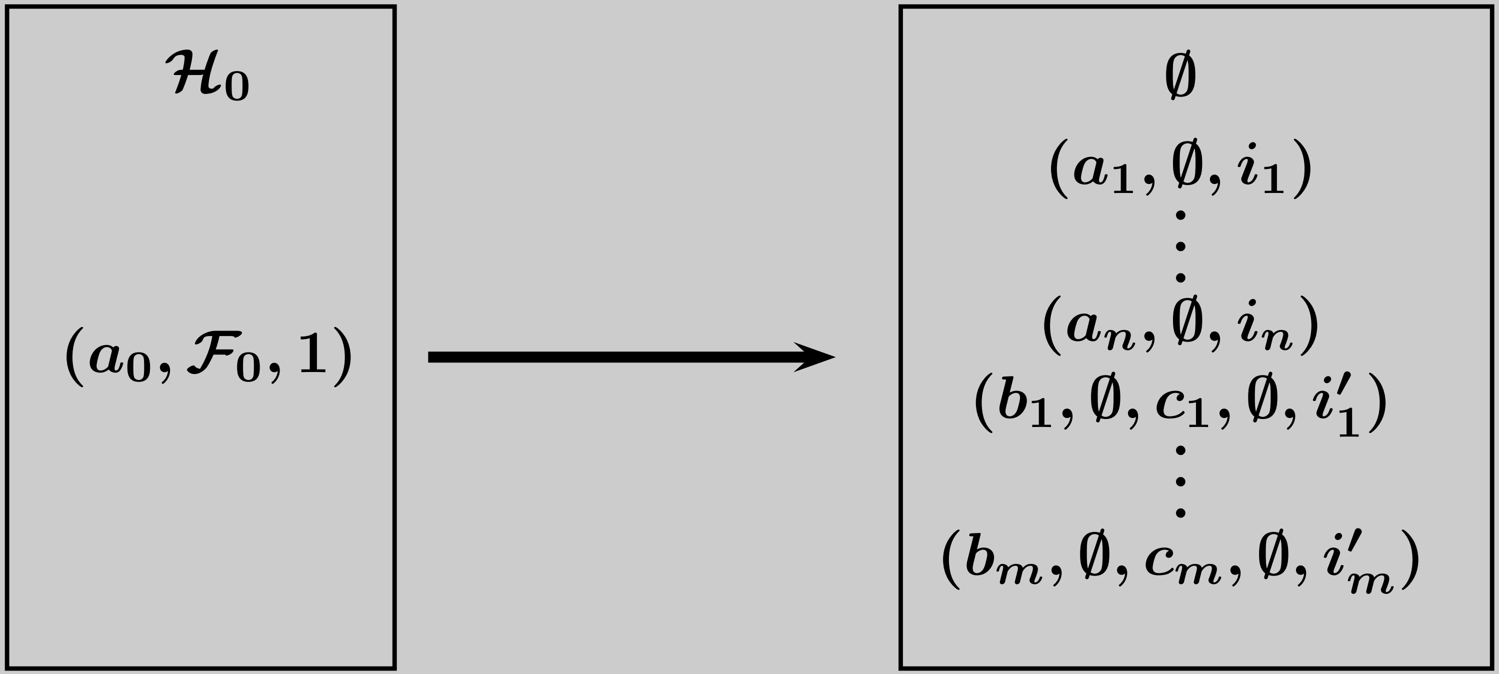
First Insights – A Decidability Result

Theorem

The model checking problem for 1-variable-DCA with bounded channel capacities and Data LTL is decidable.

Proof idea

1. Given a DCA $\mathcal{A}$ and a formula φ we decide whether there is an accepting run satisfying $\neg\varphi$.
2. Reduction to a reachability problem in an infinite state system.



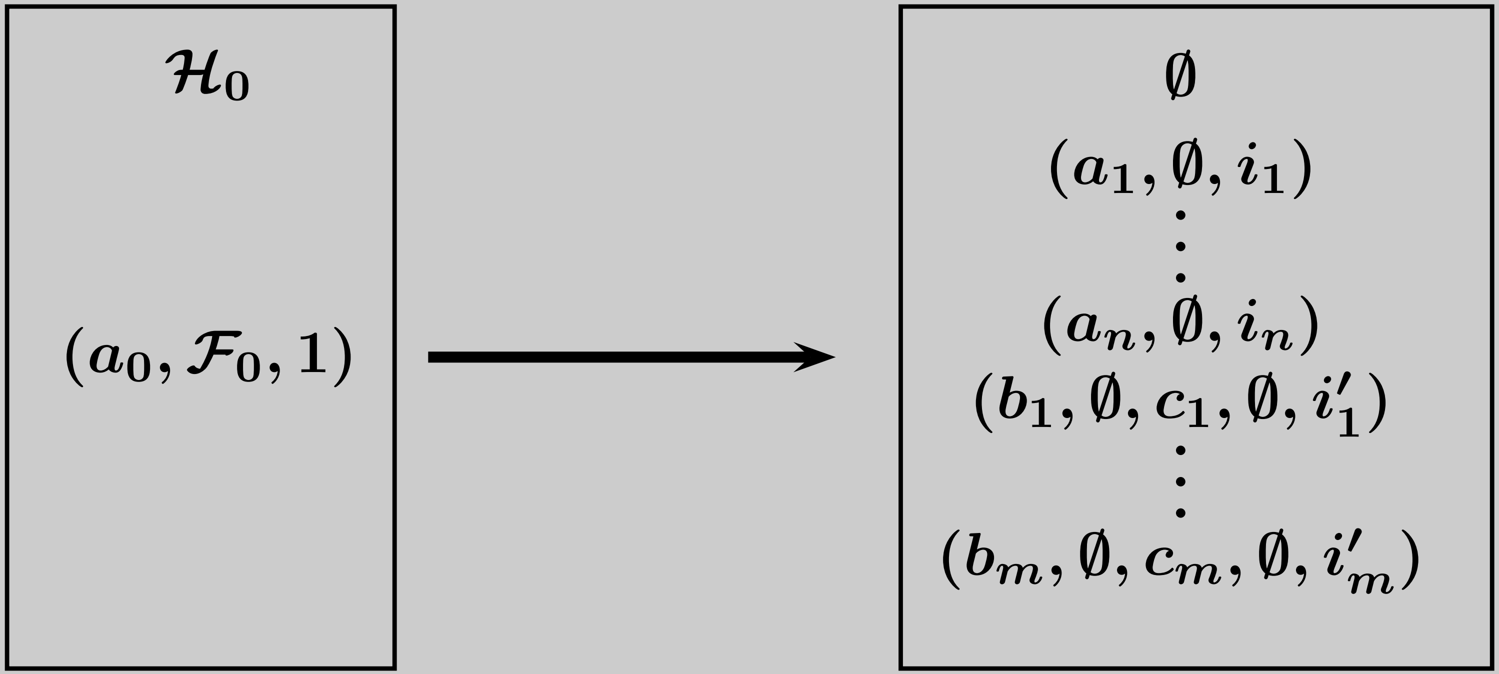
First Insights – A Decidability Result

Theorem

The model checking problem for 1-variable-DCA with bounded channel capacities and Data LTL is decidable.

Proof idea

1. Given a DCA $\mathcal{A}$ and a formula φ we decide whether there is an accepting run satisfying $\neg\varphi$.
2. Reduction to a reachability problem in an infinite state system.



3. Reduction to the nonemptiness problem for multi-counter automata without zero-tests.

Further Directions

- Model Checking of DCA where communication paths between processes are always bounded remains decidable.
  - How can DCA be restricted such that this property holds on all runs?
- Consider model checking on models which describe the *global* behavior of a system: register automata, register pushdown automata, MSC-based models.