Finite Asynchronous Automata

Julian Jarecki

Department of Computer Science,
SWT, Freiburg, Germany
http://swt.informatik.uni-freiburg.de

18. Juni 2012

Content

Content

Overview

- introduced by Wiesław Zielonka [Zie87]

Overview

- introduced by Wiesław Zielonka [Zie87]
- motivated by Trace Theory [Maz77]

Overview

- introduced by Wiesław Zielonka [Zie87]
- motivated by Trace Theory [Maz77]
- concurrent systems

Overview

- introduced by Wiesław Zielonka [Zie87]
- motivated by Trace Theory [Maz77]
- concurrent systems
- similar to Petri Nets [DR95]

Overview

- introduced by Wiesław Zielonka [Zie87]
- motivated by Trace Theory [Maz77]
- concurrent systems
- similar to Petri Nets [DR95]
- restriction to a finite statespace

Recap: Petri Nets

$$L_n = \{wc \mid w \in \{a, b\}^* \ \wedge \ |w|_a = |w|_b = n\}$$

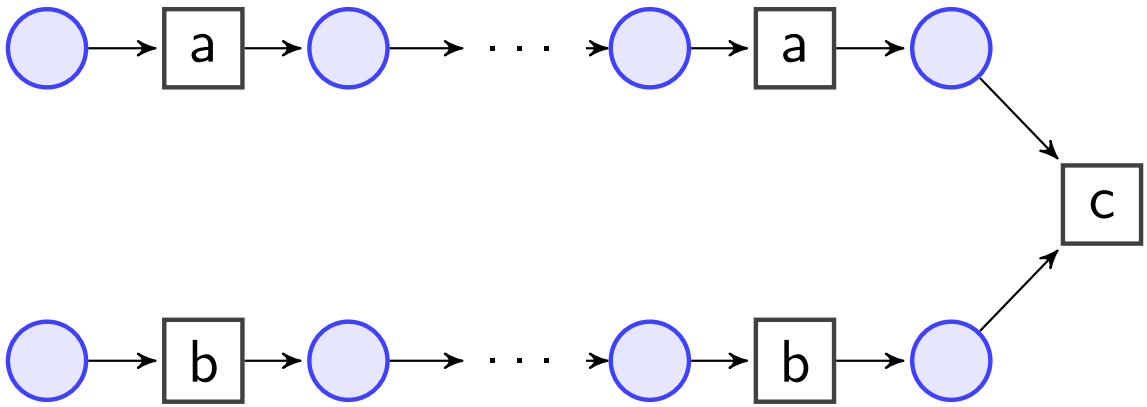
Recap: Petri Nets

$$L_n = \{wc \mid w \in \{a, b\}^* \ \wedge \ |w|_a = |w|_b = n\}$$

- $L_1 = \{abc, bac\}$
- $L_2 = \{aabbc, ababc, baabc, babac, bbaac\}$
- $L_3 = \{aaabbbc, aababbc, \ldots, bbbaaac\}$

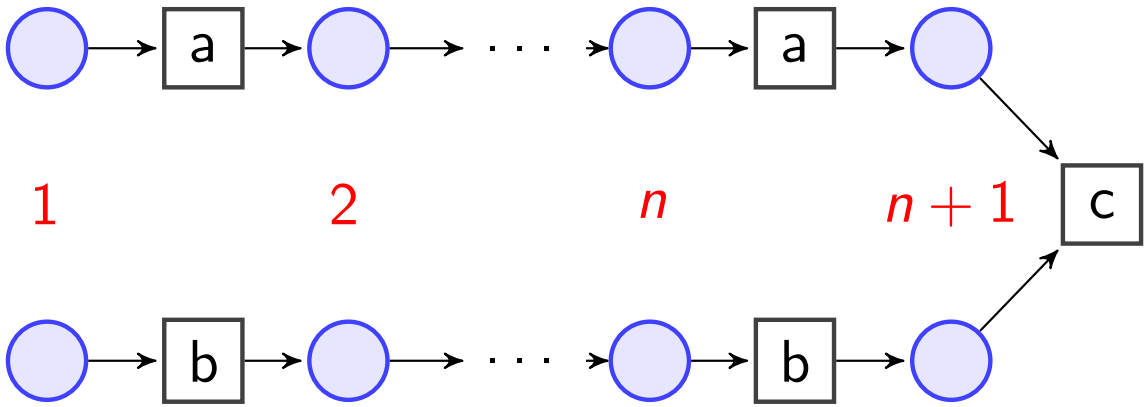
Recap: Petri Nets

$$L_n = \{wc \mid w \in \{a, b\}^* \ \wedge \ |w|_a = |w|_b = n\}$$



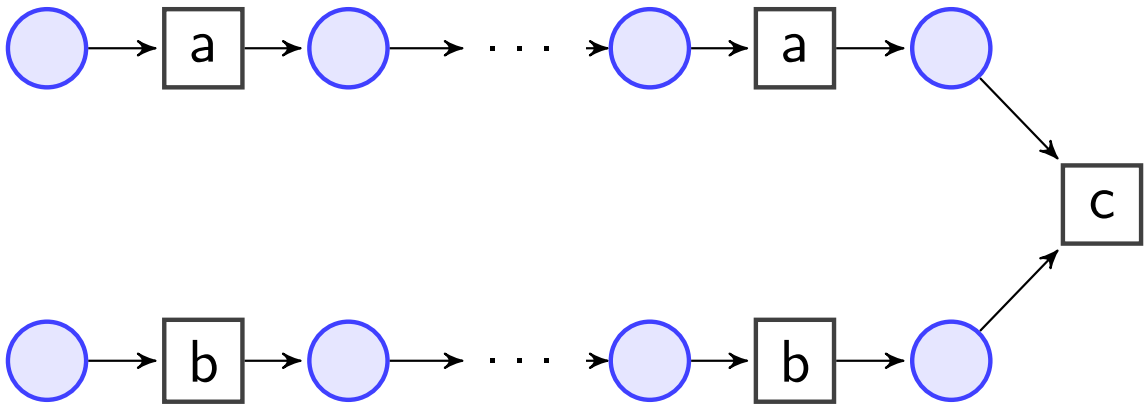
Recap: Petri Nets

$$L_n = \{wc \mid w \in \{a, b\}^* \ \wedge \ |w|_a = |w|_b = n\}$$



Recap: Petri Nets

$$L_n = \{wc \mid w \in \{a, b\}^* \ \wedge \ |w|_a = |w|_b = n\}$$

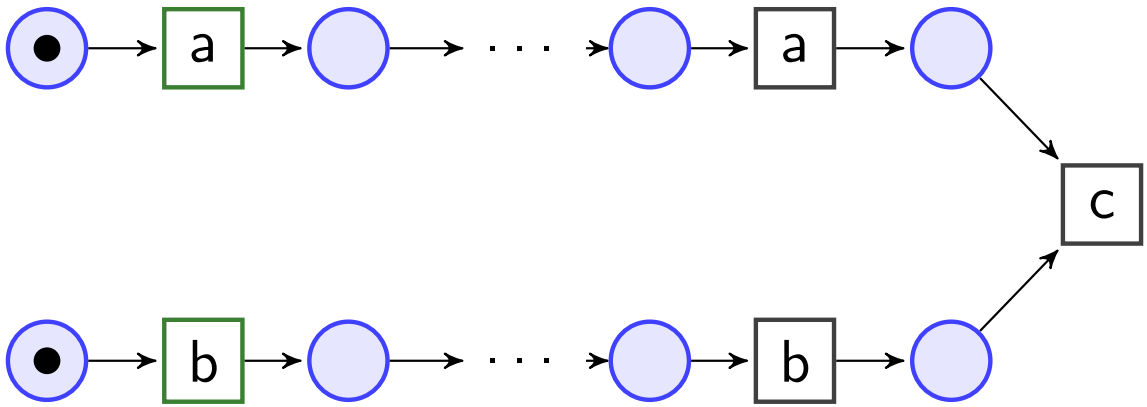


Exactly one final marking:

$$p \mapsto 0$$

Recap: Petri Nets

$$L_n = \{wc \mid w \in \{a, b\}^* \ \wedge \ |w|_a = |w|_b = n\}$$

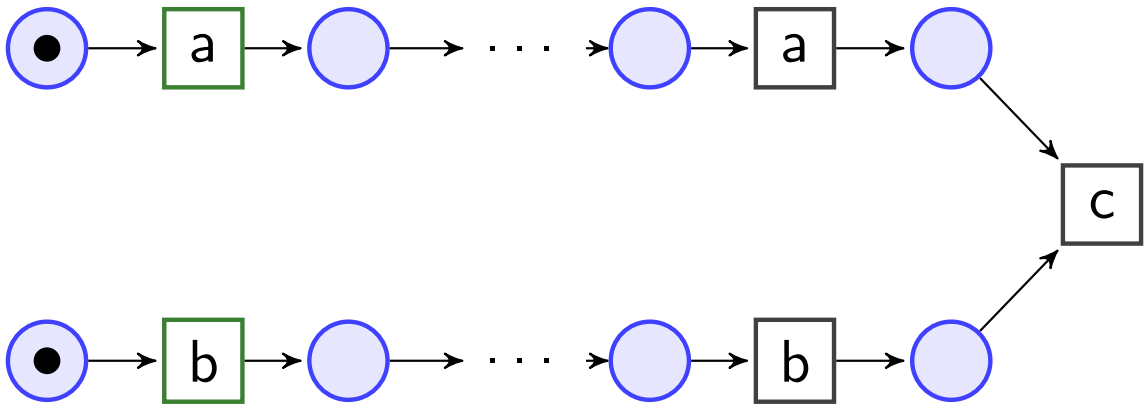


Exactly one final marking:

$$p \mapsto 0$$

Recap: Petri Nets

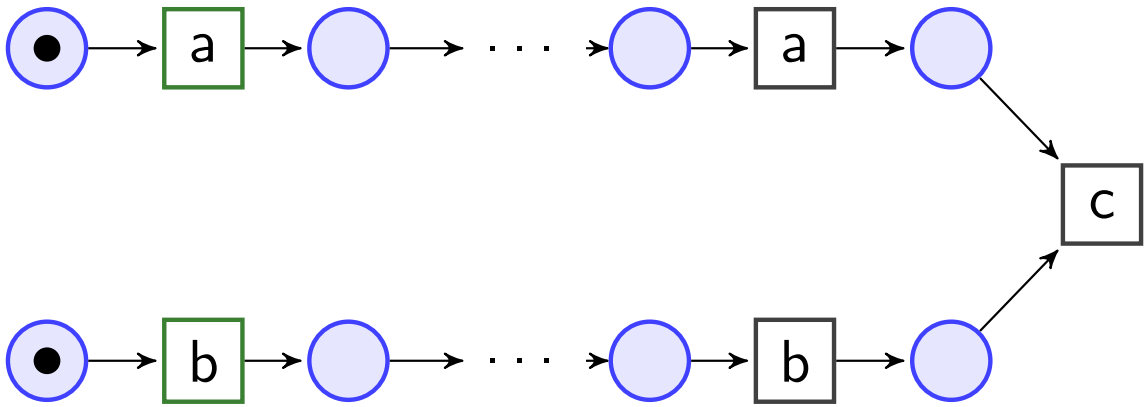
$$L_n = \{wc \mid w \in \{a, b\}^* \ \wedge \ |w|_a = |w|_b = n\}$$



- L_n is finite, thus regular.

Recap: Petri Nets

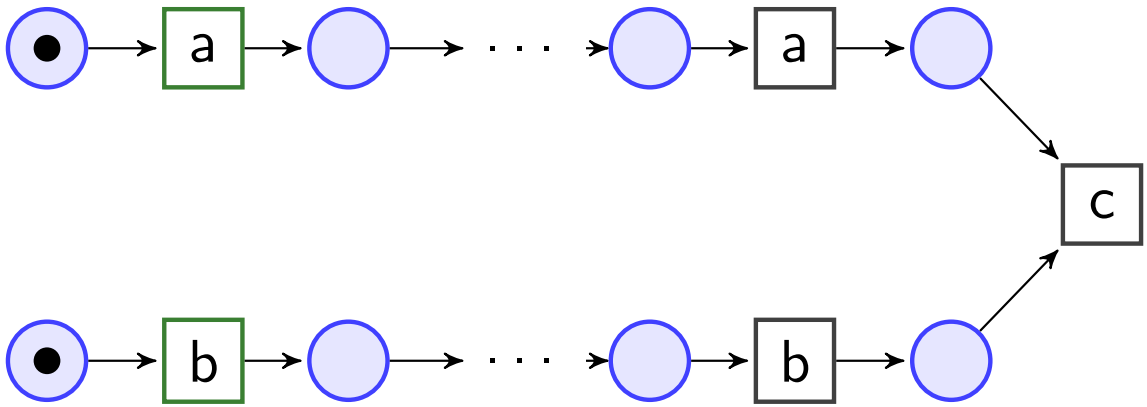
$$L_n = \{wc \mid w \in \{a, b\}^* \ \wedge \ |w|_a = |w|_b = n\}$$



- L_n is finite, thus regular.
- Net has linear size ($O(n^2)$ for NFA)

Recap: Petri Nets

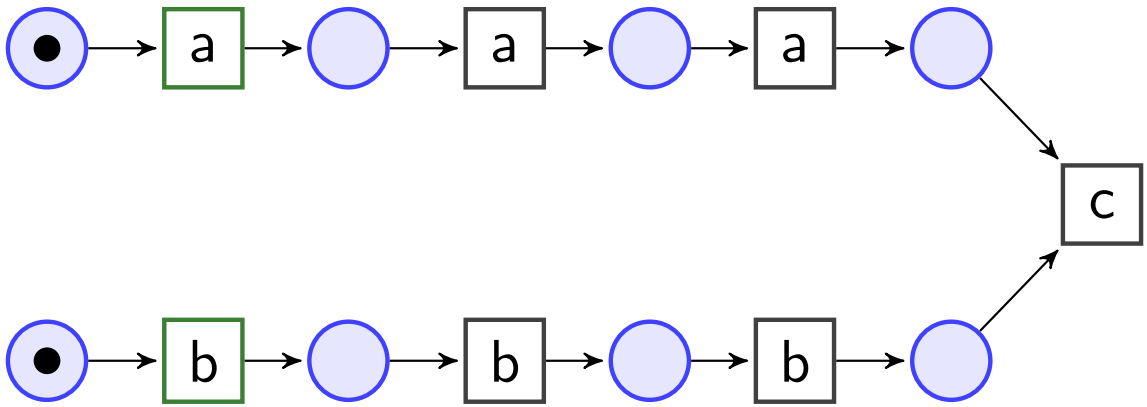
$$L_n = \{wc \mid w \in \{a, b\}^* \ \wedge \ |w|_a = |w|_b = n\}$$



- L_n is finite, thus regular.
- Net has linear size ($O(n^2)$ for NFA)
- Net is 1-safe.

Recap: Petri Nets

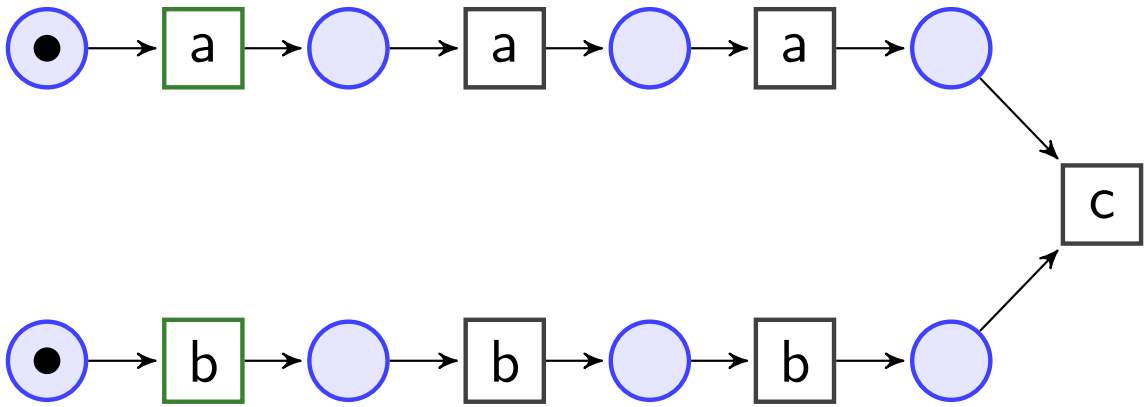
$$L_n = \{wc \mid w \in \{a, b\}^* \ \wedge \ |w|_a = |w|_b = n\}$$



- L_n is finite, thus regular.
- Net has linear size ($O(n^2)$ for *NFA*)
- Net is 1-safe.
- The Net is *deterministic*.

Recap: Petri Nets

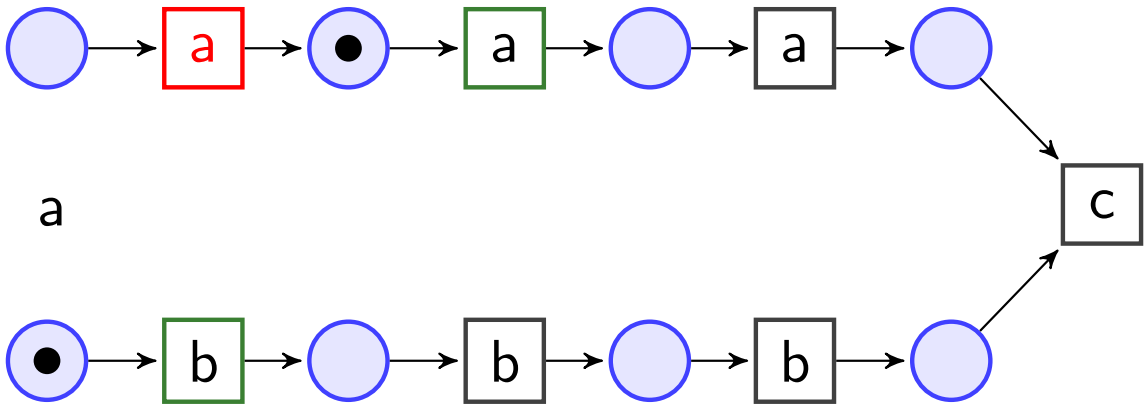
$$L_n = \{wc \mid w \in \{a, b\}^* \ \wedge \ |w|_a = |w|_b = n\}$$



- L_n is finite, thus regular.
- Net has linear size ($O(n^2)$ for *NFA*)
- Net is 1-safe.
- The Net is *deterministic*. (orly!?)

Recap: Petri Nets

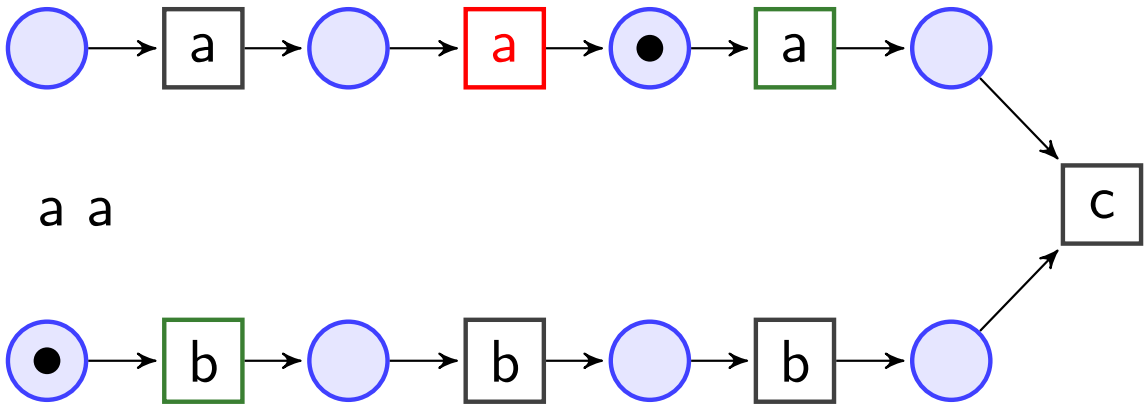
$$L_n = \{wc \mid w \in \{a,b\}^* \ \wedge \ |w|_a = |w|_b = n\}$$



- L_n is finite, thus regular.
- Net has linear size ($O(n^2)$ for *NFA*)
- Net is 1-safe.
- The Net is *deterministic*.

Recap: Petri Nets

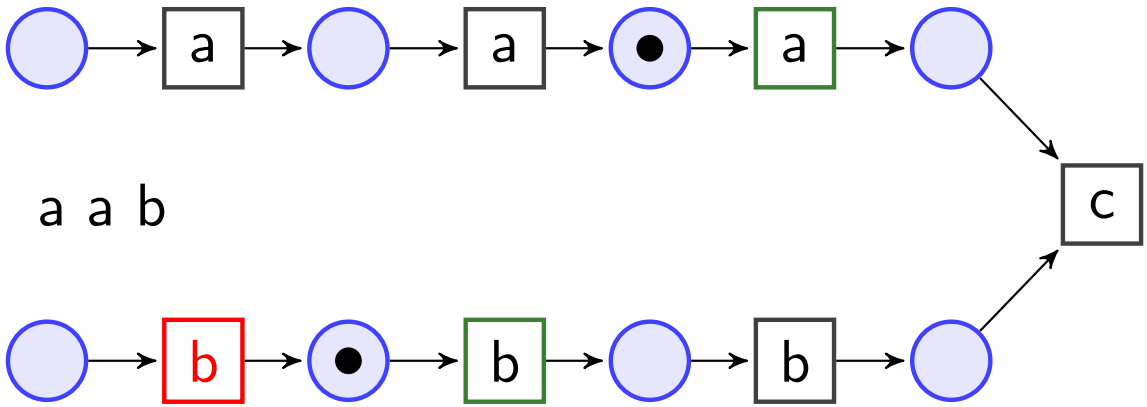
$$L_n = \{wc \mid w \in \{a, b\}^* \ \wedge \ |w|_a = |w|_b = n\}$$



- L_n is finite, thus regular.
- Net has linear size ($O(n^2)$ for *NFA*)
- Net is 1-safe.
- The Net is *deterministic*.

Recap: Petri Nets

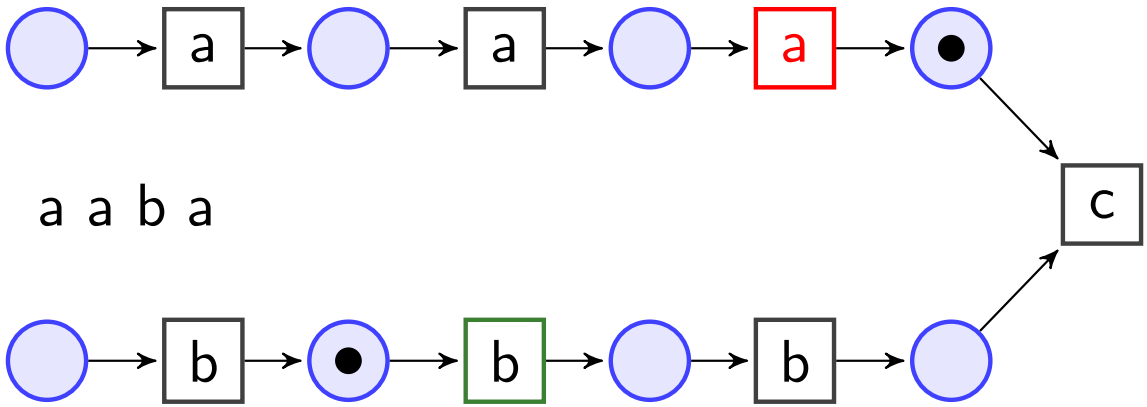
$$L_n = \{wc \mid w \in \{a, b\}^* \ \wedge \ |w|_a = |w|_b = n\}$$



- L_n is finite, thus regular.
- Net has linear size ($O(n^2)$ for *NFA*)
- Net is 1-safe.
- The Net is *deterministic*.

Recap: Petri Nets

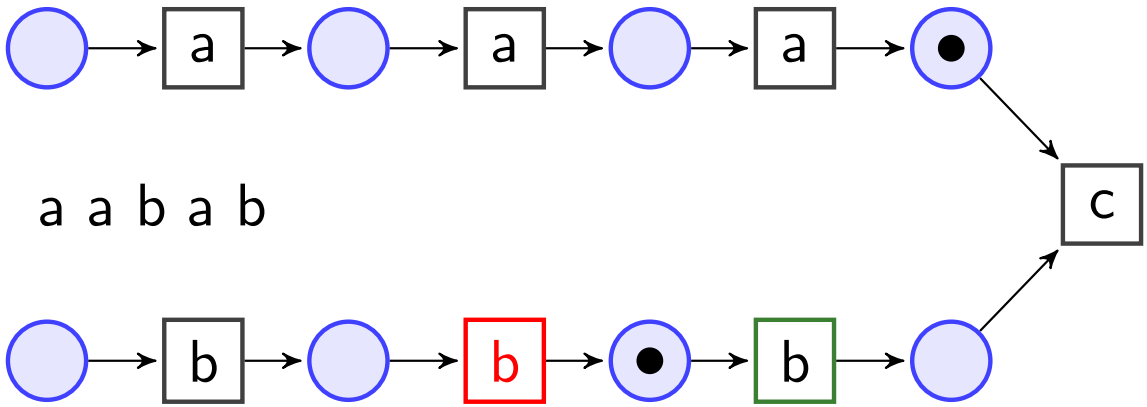
$$L_n = \{wc \mid w \in \{a, b\}^* \ \wedge \ |w|_a = |w|_b = n\}$$



- L_n is finite, thus regular.
- Net has linear size ($O(n^2)$ for *NFA*)
- Net is 1-safe.
- The Net is *deterministic*.

Recap: Petri Nets

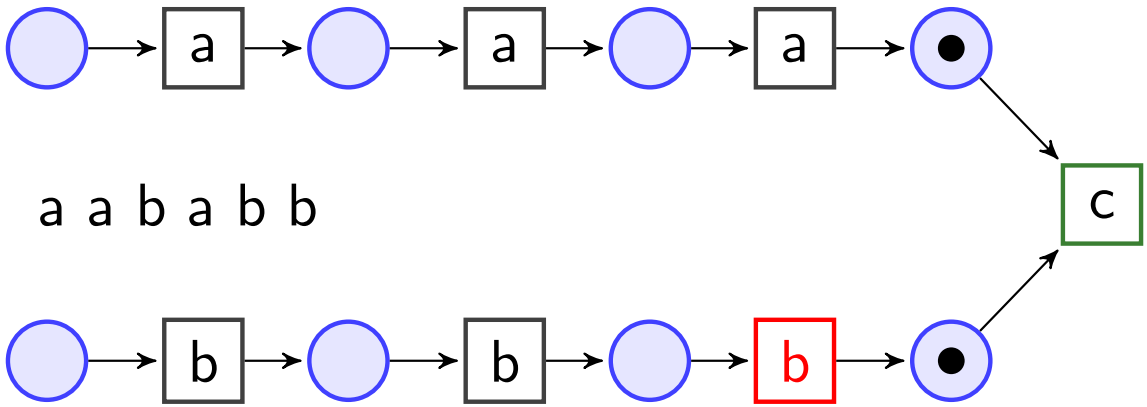
$$L_n = \{wc \mid w \in \{a, b\}^* \ \wedge \ |w|_a = |w|_b = n\}$$



- L_n is finite, thus regular.
- Net has linear size ($O(n^2)$ for *NFA*)
- Net is 1-safe.
- The Net is *deterministic*.

Recap: Petri Nets

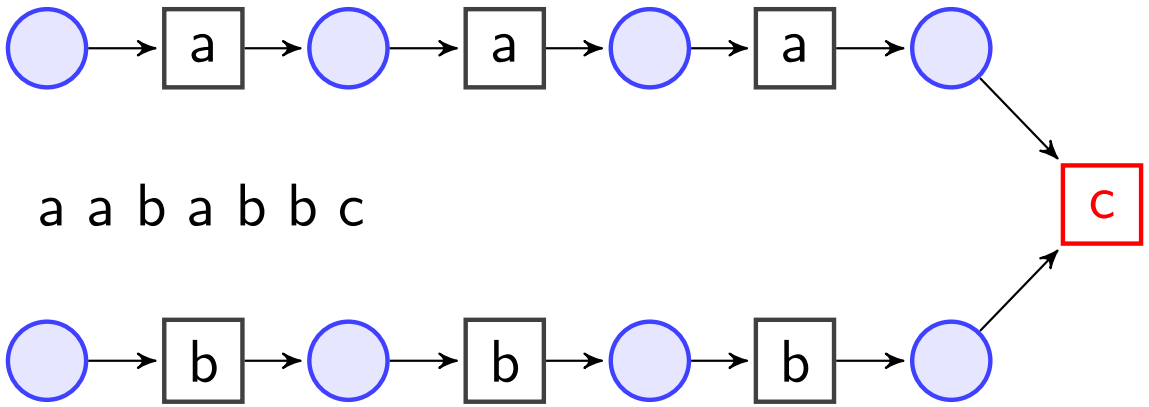
$$L_n = \{wc \mid w \in \{a,b\}^* \ \wedge \ |w|_a = |w|_b = n\}$$



- L_n is finite, thus regular.
- Net has linear size ($O(n^2)$ for *NFA*)
- Net is 1-safe.
- The Net is *deterministic*.

Recap: Petri Nets

$$L_n = \{wc \mid w \in \{a, b\}^* \ \wedge \ |w|_a = |w|_b = n\}$$



- L_n is finite, thus regular.
- Net has linear size ($O(n^2)$ for *NFA*)
- Net is 1-safe.
- The Net is *deterministic*.

Recap: Petri Nets

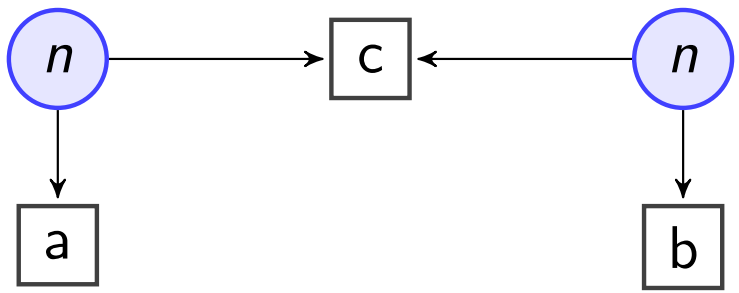
Can Petri Nets make it better ?

Recap: Petri Nets

Can Petri Nets make it better ?

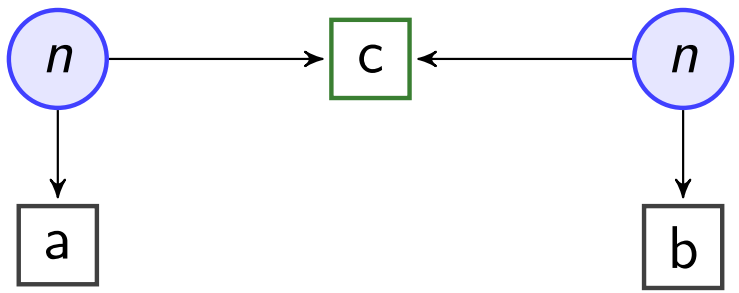
Recap: Petri Nets

Can Petri Nets make it better ?



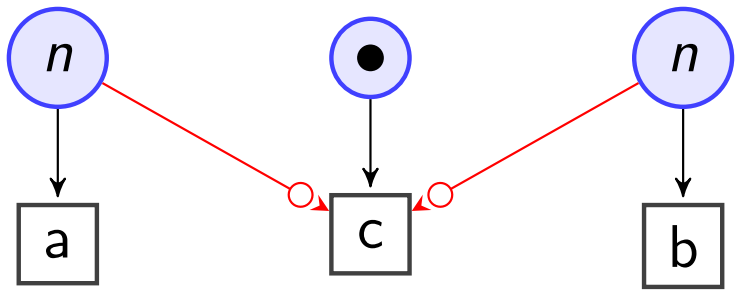
Recap: Petri Nets

Can Petri Nets make it better ?



Recap: Petri Nets

Can Petri Nets make it better ?

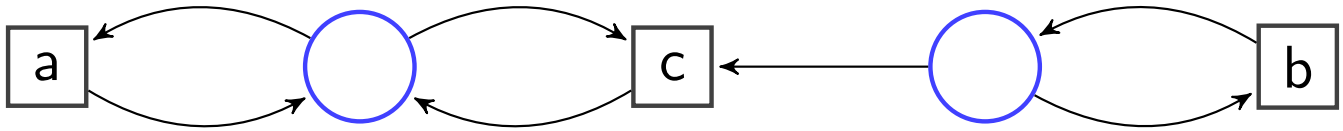


An inhibitor arc imposes the precondition that the transition may only fire when the place is empty; this allows arbitrary computations on numbers of tokens to be expressed, which makes the formalism Turing complete.[1]

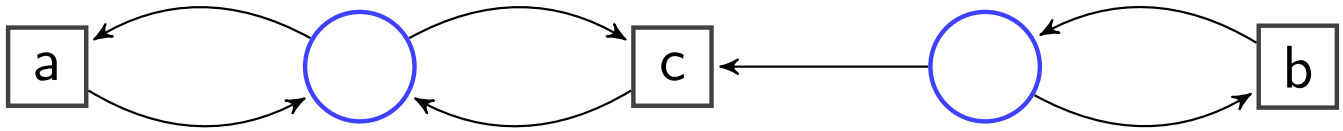
[1] http://en.wikipedia.org/wiki/Petri_net

Content

A Finite Asynchronous Automaton accepting L_n



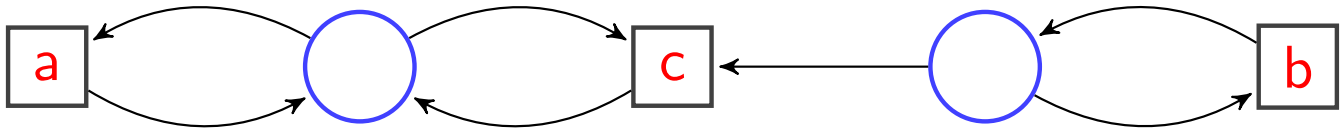
A Finite Asynchronous Automaton accepting L_n



A *Signature*:

$$\sigma = (\Sigma, R, \mathcal{E})$$

A Finite Asynchronous Automaton accepting L_n

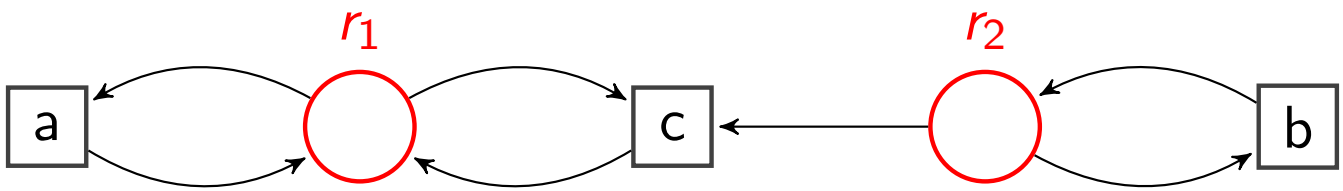


A *Signature*:

$$\sigma = (\Sigma, R, \mathcal{E})$$

- Σ is an Alphabet (the set of Actions)

A Finite Asynchronous Automaton accepting L_n

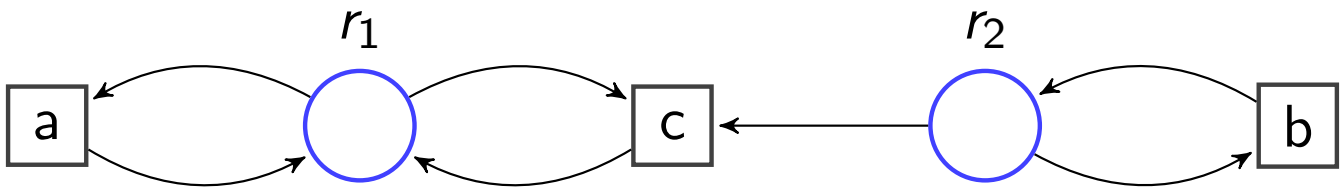


A *Signature*:

$$\sigma = (\Sigma, R, \mathcal{E})$$

- Σ is an Alphabet (the set of Actions)
- R is the Set of Registers

A Finite Asynchronous Automaton accepting L_n

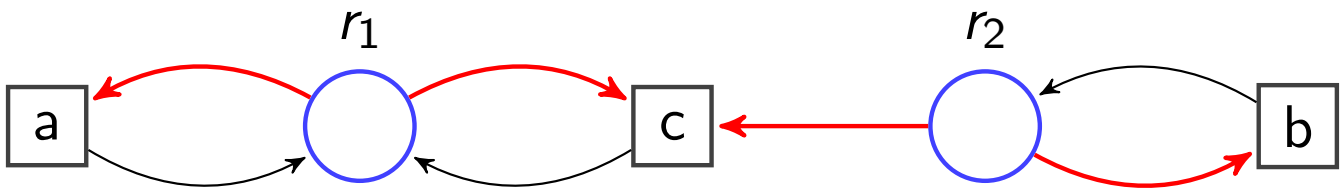


A *Signature*:

$$\sigma = (\Sigma, R, \mathcal{E})$$

- Σ is an Alphabet (the set of Actions)
- R is the Set of Registers
- $\mathcal{E} \subseteq R \times \Sigma \cup \Sigma \times R$ is the connection relation.

A Finite Asynchronous Automaton accepting L_n

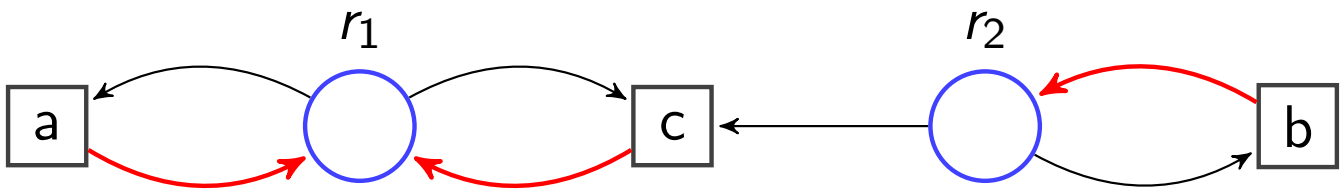


A *Signature*:

$$\sigma = (\Sigma, R, \mathcal{E})$$

- Σ is an Alphabet (the set of Actions)
- R is the Set of Registers
- $\mathcal{E} \subseteq R \times \Sigma \cup \Sigma \times R$ is the connection relation.

A Finite Asynchronous Automaton accepting L_n

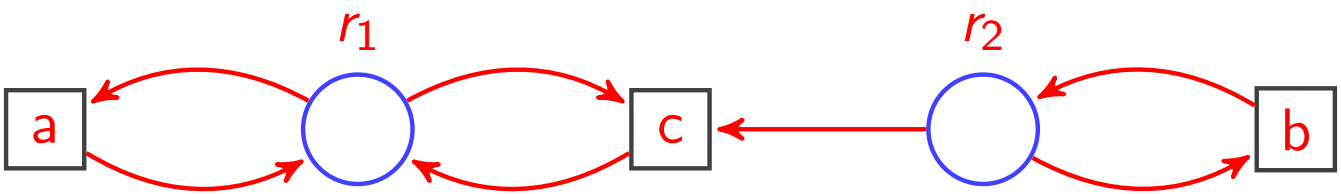


A *Signature*:

$$\sigma = (\Sigma, R, \mathcal{E})$$

- Σ is an Alphabet (the set of Actions)
- R is the Set of Registers
- $\mathcal{E} \subseteq R \times \Sigma \cup \Sigma \times R$ is the connection relation.

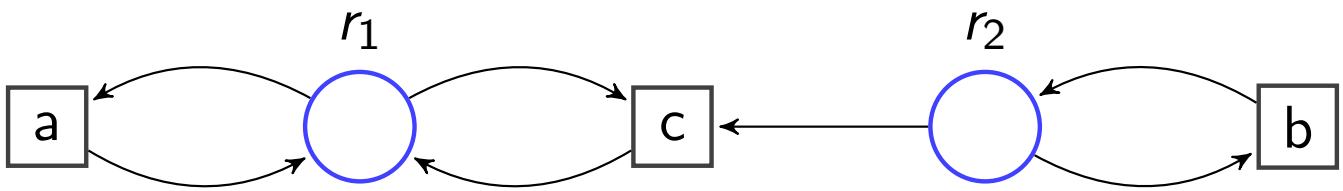
A Finite Asynchronous Automaton accepting L_n



Finite Asynchronous Transition System (fat-system):

$$\tau = (\underbrace{\Sigma, R, \mathcal{E}}_{\sigma}, X, \Delta)$$

A Finite Asynchronous Automaton accepting L_n

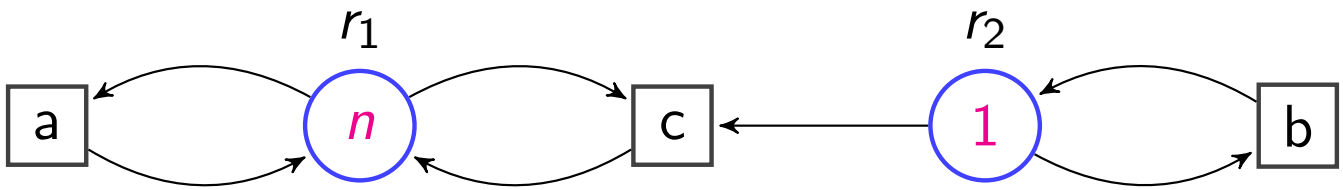


Finite Asynchronous Transition System (fat-system):

$$\tau = (\Sigma, R, \mathcal{E}, X, \Delta)$$

- X is the a finite set of values
- $X = \mathbb{N}_{\leq n} \cup \{s\}$

A Finite Asynchronous Automaton accepting L_n



Finite Asynchronous Transition System (fat-system):

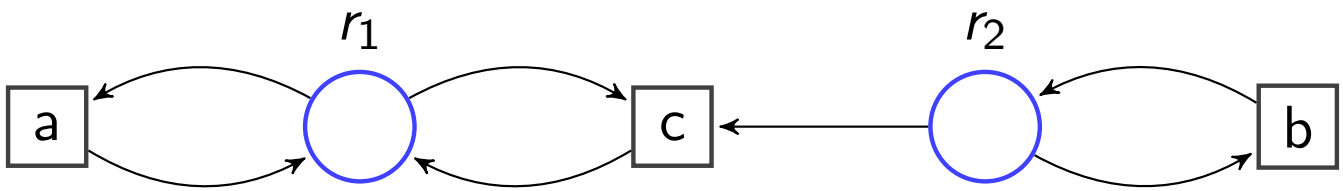
$$\tau = (\Sigma, R, \mathcal{E}, X, \Delta)$$

- X is the a finite set of values
- $X = \mathbb{N}_{\leq n} \cup \{s\}$

Global states:

$$S = X^R$$

A Finite Asynchronous Automaton accepting L_n



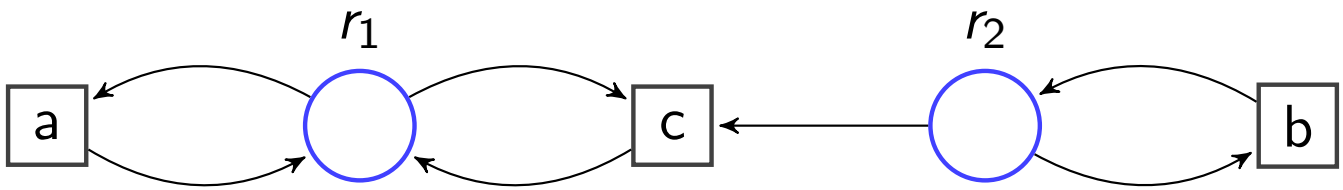
Finite Asynchronous Transition System (fat-system):

$$\tau = (\Sigma, R, \mathcal{E}, X, \Delta)$$

- X is the a finite set of values
- $\Delta = \{\delta_a \mid a \in \Sigma\}$ is a family of transition Relations

- $X = \mathbb{N}_{\leq n} \cup \{s\}$
- $\Delta = \{\delta_a, \delta_b, \delta_c\}$

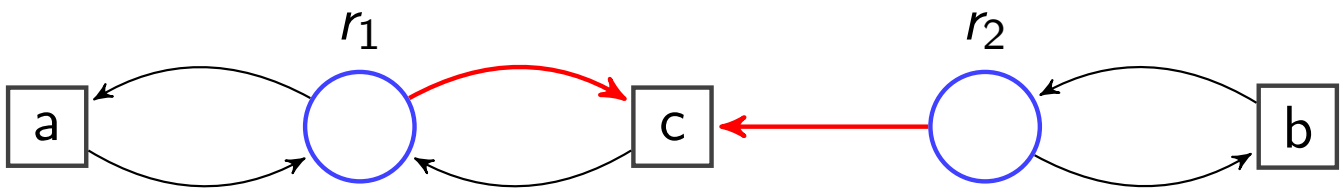
A Finite Asynchronous Automaton accepting L_n



Local transitions:

$$\delta_a \subseteq X^{\mathcal{E}a} \times X^{a\mathcal{E}}$$

A Finite Asynchronous Automaton accepting L_n



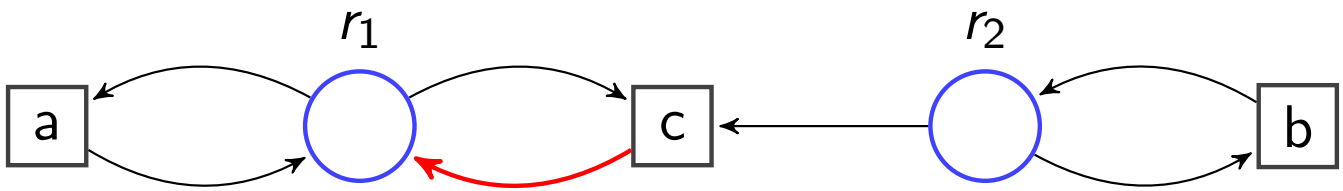
δ_c

r_1	r_2	r_1
0	0	s

Local transitions:

$$\delta_a \subseteq X^{\mathcal{E}a} \times X^{a\mathcal{E}}$$

A Finite Asynchronous Automaton accepting L_n



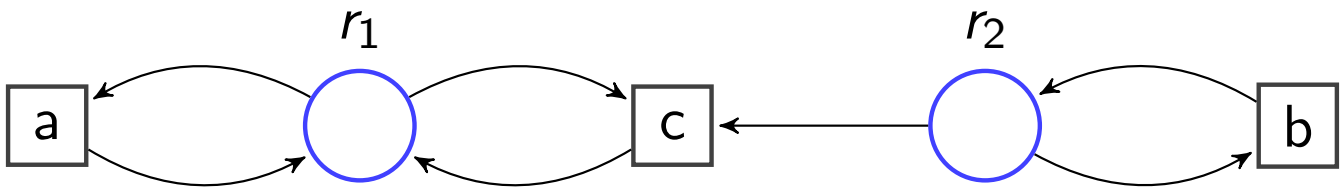
δ_c

r_1	r_2	r_1
0	0	s

Local transitions:

$$\delta_a \subseteq X^{\mathcal{E}a} \times X^{a\mathcal{E}}$$

A Finite Asynchronous Automaton accepting L_n



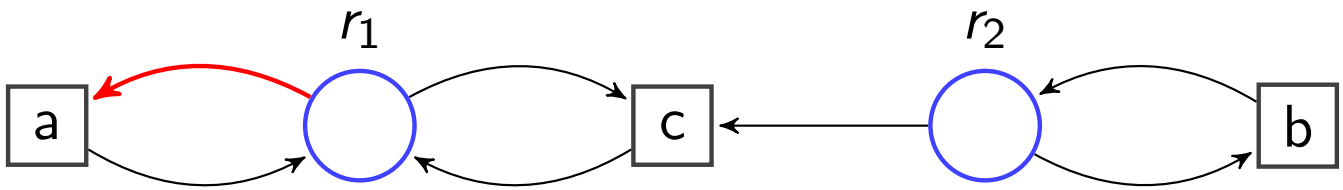
δ_c

r_1	r_2	r_1
0	0	s

Local transitions:

$$\delta_a \subseteq X^{\mathcal{E}a} \times X^{a\mathcal{E}}$$

A Finite Asynchronous Automaton accepting L_n



δ_a

r_1	r_1
i	$i-1$

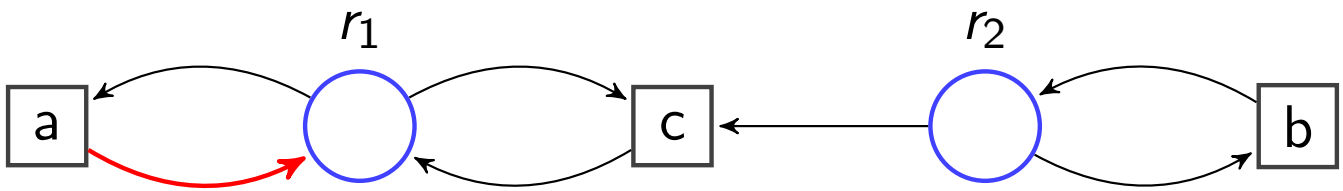
δ_c

r_1	r_2	r_1
0	0	s

Local transitions:

$$\delta_a \subseteq X^{\mathcal{E}a} \times X^{a\mathcal{E}}$$

A Finite Asynchronous Automaton accepting L_n



δ_a

r_1	r_1
i	$i-1$

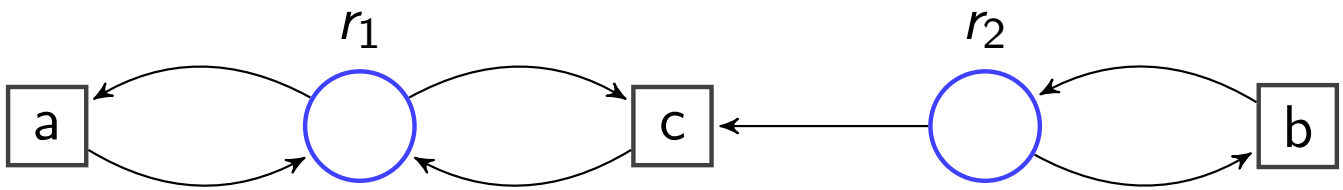
δ_c

r_1	r_2	r_1
0	0	s

Local transitions:

$$\delta_a \subseteq X^{\mathcal{E}a} \times X^{a\mathcal{E}}$$

A Finite Asynchronous Automaton accepting L_n



δ_a

r_1	r_1
i	$i-1$

δ_c

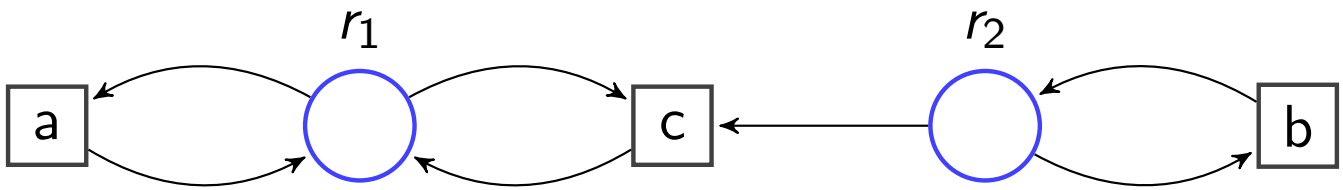
r_1	r_2	r_1
0	0	s

$0 < i \leq n$

Local transitions:

$$\delta_a \subseteq X^{\mathcal{E}a} \times X^{a\mathcal{E}}$$

A Finite Asynchronous Automaton accepting L_n



δ_a

r_1	r_1
i	$i-1$

δ_c

r_1	r_2	r_1
0	0	s

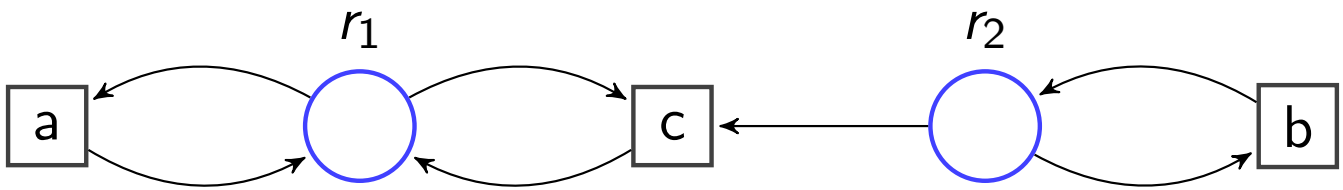
δ_b

r_2	r_2
i	$i-1$

Local transitions:

$$\delta_a \subseteq X^{\mathcal{E}a} \times X^{a\mathcal{E}}$$

A Finite Asynchronous Automaton accepting L_n



δ_a

r_1	r_1
i	$i-1$

δ_c

r_1	r_2	r_1
0	0	s

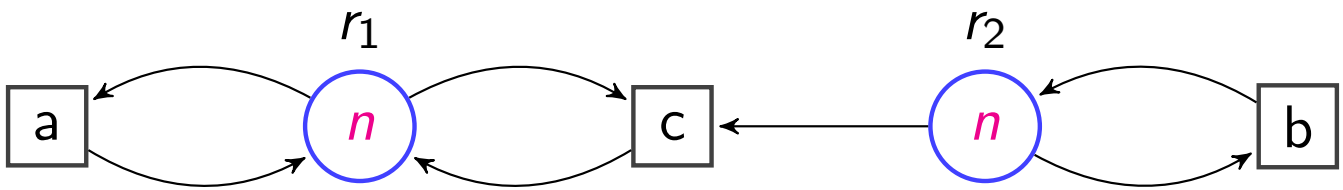
δ_b

r_2	r_2
i	$i-1$

Finite Asynchronous Automaton (FAA):

$$\mathcal{A} = (\underbrace{\Sigma, R, \mathcal{E}, X, \Delta}_{\text{fat-system}}, I, F)$$

A Finite Asynchronous Automaton accepting L_n



δ_a

r_1	r_1
i	$i-1$

δ_c

r_1	r_2	r_1
0	0	s

δ_b

r_2	r_2
i	$i-1$

Finite Asynchronous Automaton (FAA):

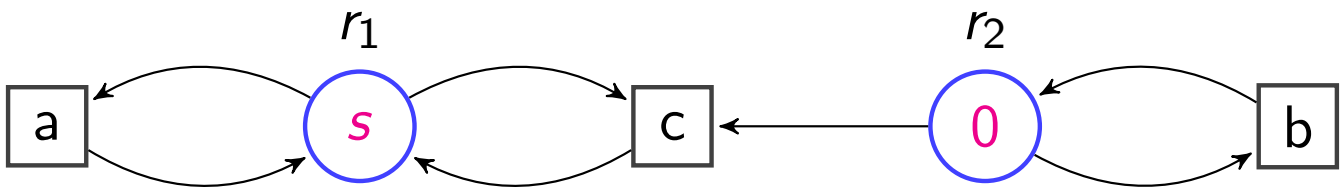
$$\mathcal{A} = (\Sigma, R, \mathcal{E}, X, \Delta, I, F)$$

$$I = \{f_I\}$$

$$f_I(r_1) = n$$

$$f_I(r_2) = n$$

A Finite Asynchronous Automaton accepting L_n



δ_a

r_1	r_1
i	$i-1$

δ_c

r_1	r_2	r_1
0	0	s

δ_b

r_2	r_2
i	$i-1$

Finite Asynchronous Automaton (FAA):

$$\mathcal{A} = (\Sigma, R, \mathcal{E}, X, \Delta, I, F)$$

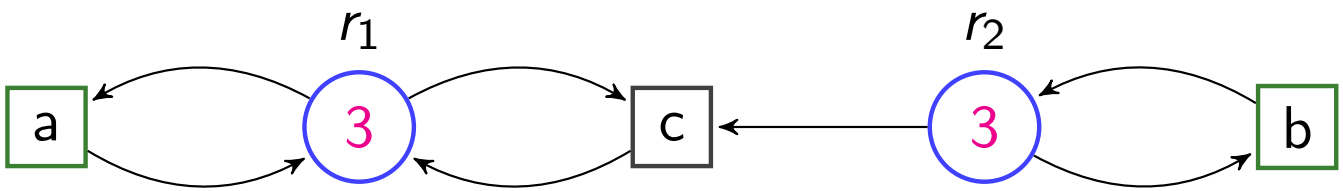
$$F = \{f_F\}$$

$$f_F(r_1) = s$$

$$f_F(r_2) = 0$$

A Finite Asynchronous Automaton accepting L_3

Example Run for $aababbc \in L_3$



δ_a

r_1	r_1
3	2
2	1
1	0

δ_c

r_1	r_2	r_1
0	0	s

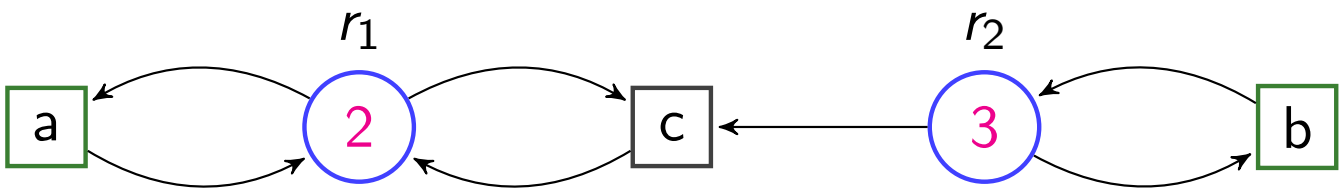
δ_b

r_2	r_2
3	2
2	1
1	0

a a b a b b c

A Finite Asynchronous Automaton accepting L_3

Example Run for $aababbc \in L_3$



δ_a

r_1	r_1
3	2
2	1
1	0

δ_c

r_1	r_2	r_1
0	0	s

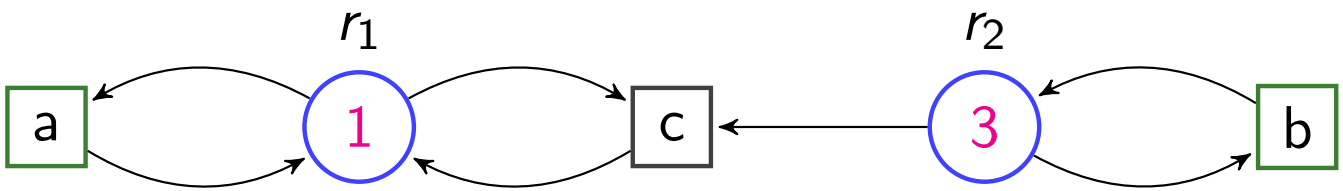
δ_b

r_2	r_2
3	2
2	1
1	0

a a b a b b c

A Finite Asynchronous Automaton accepting L_3

Example Run for $aababbc \in L_3$



δ_a

r_1	r_1
3	2
2	1
1	0

δ_c

r_1	r_2	r_1
0	0	s

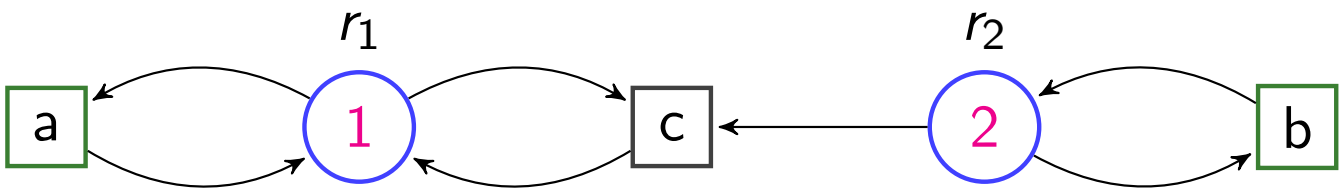
δ_b

r_2	r_2
3	2
2	1
1	0

a a b a b b c

A Finite Asynchronous Automaton accepting L_3

Example Run for $aababbc \in L_3$



δ_a

r_1	r_1
3	2
2	1
1	0

δ_c

r_1	r_2	r_1
0	0	s

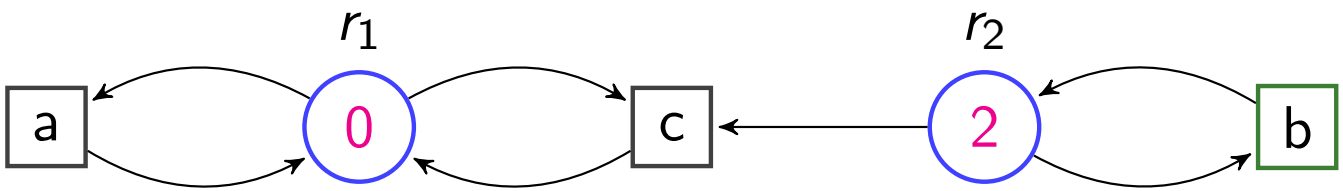
δ_b

r_2	r_2
3	2
2	1
1	0

a a b a b b c

A Finite Asynchronous Automaton accepting L_3

Example Run for $aababbc \in L_3$



δ_a

r_1	r_1
3	2
2	1
1	0

δ_c

r_1	r_2	r_1
0	0	s

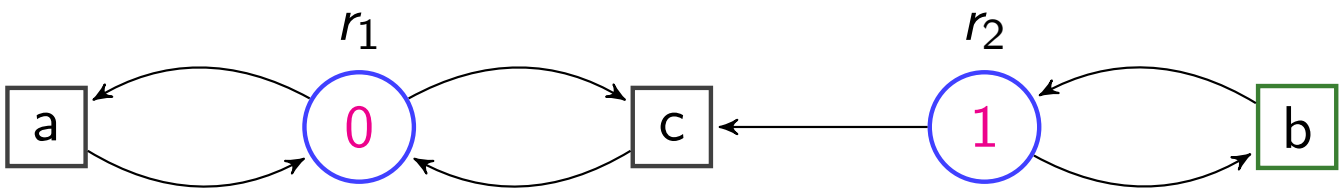
δ_b

r_2	r_2
3	2
2	1
1	0

a a b a b b c

A Finite Asynchronous Automaton accepting L_3

Example Run for $aababbc \in L_3$



δ_a

r_1	r_1
3	2
2	1
1	0

δ_c

r_1	r_2	r_1
0	0	s

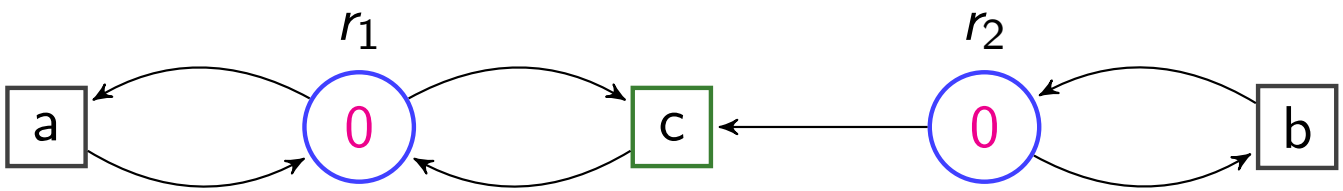
δ_b

r_2	r_2
3	2
2	1
1	0

a a b a b b c

A Finite Asynchronous Automaton accepting L_3

Example Run for $aababbc \in L_3$



δ_a

r_1	r_1
3	2
2	1
1	0

δ_c

r_1	r_2	r_1
0	0	s

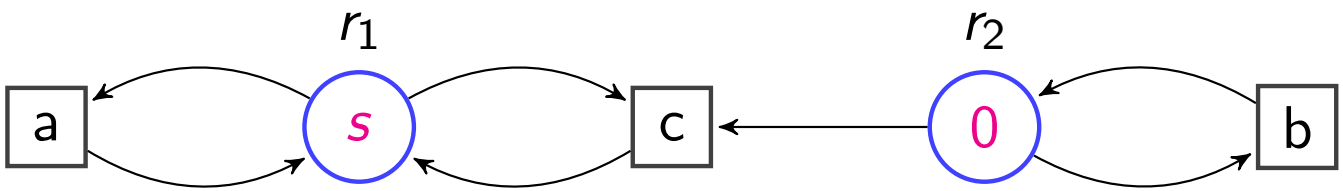
δ_b

r_2	r_2
3	2
2	1
1	0

a a b a b b c

A Finite Asynchronous Automaton accepting L_3

Example Run for $aababbc \in L_3$



δ_a

r_1	r_1
3	2
2	1
1	0

δ_c

r_1	r_2	r_1
0	0	s

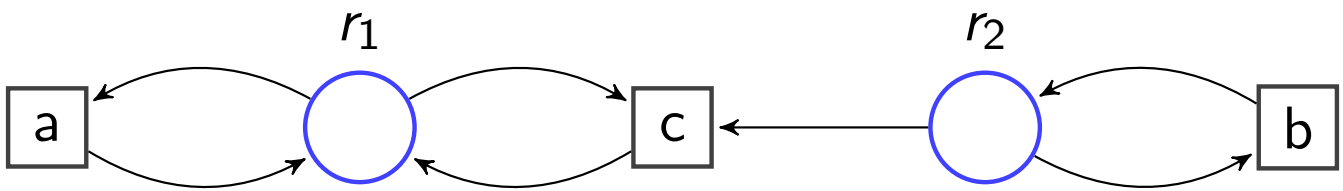
δ_b

r_2	r_2
3	2
2	1
1	0

a a b a b b c

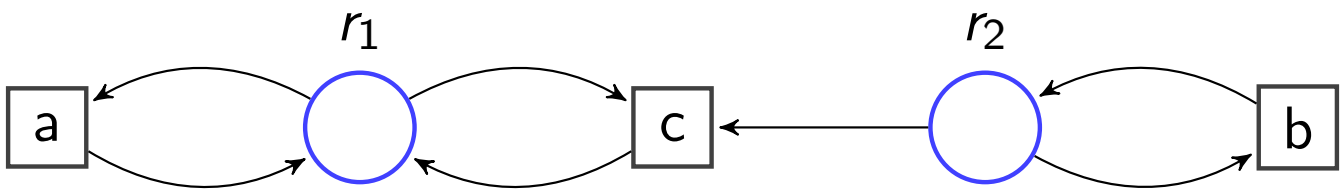
Independence

A closer look at the signature



Independence

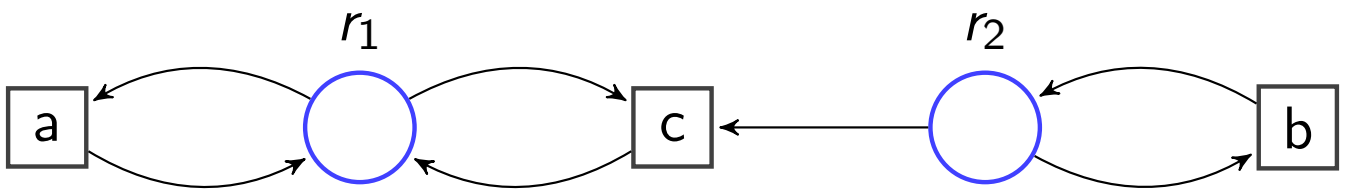
A closer look at the signature



When are two Actions $a, b \in \Sigma$ **independent** ?

Independence

A closer look at the signature

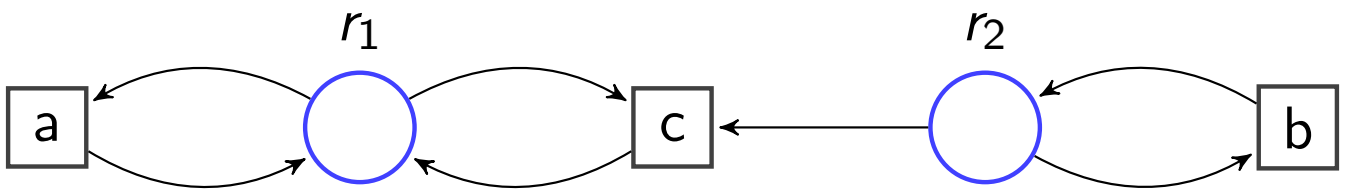


When are two Actions $a, b \in \Sigma$ **independent** ?

What does independent mean?

Independence

A closer look at the signature

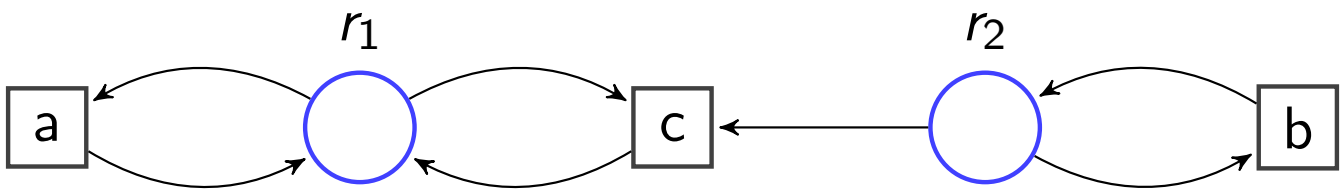


When are two Actions $a, b \in \Sigma$ **independent** ?

In a signature σ two Actions a, b are called independent,
if σ's structure doesn't allow a fat-system to distinguish ab and ba.

Independence

A closer look at the signature

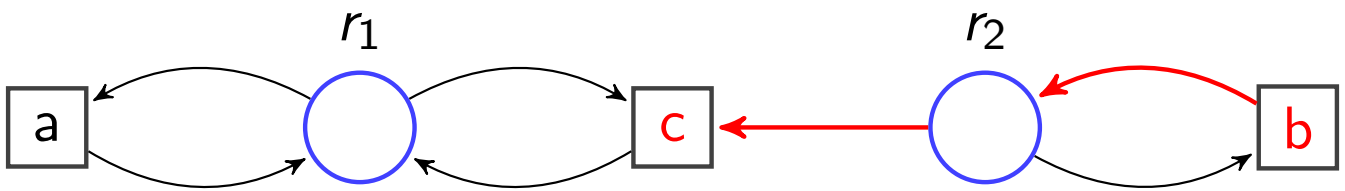


When are two Actions $a, b \in \Sigma$ **independent** ?

In a signature σ two Actions a, b are called independent,
if for all fat-systems with σ a and b can be executed in any order
without altering the result of the computation.

Independence

A closer look at the signature

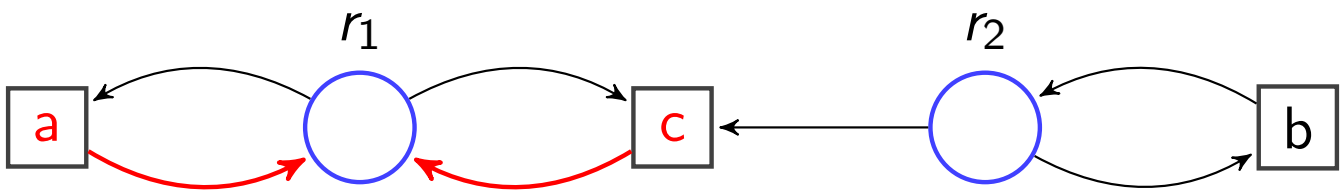


$C_\sigma = \mathcal{E}^2 \cap \Sigma^2$

$a \to \bigcirc \to b$

Independence

A closer look at the signature



$$C_\sigma = \mathcal{E}^2 \cap \Sigma^2$$

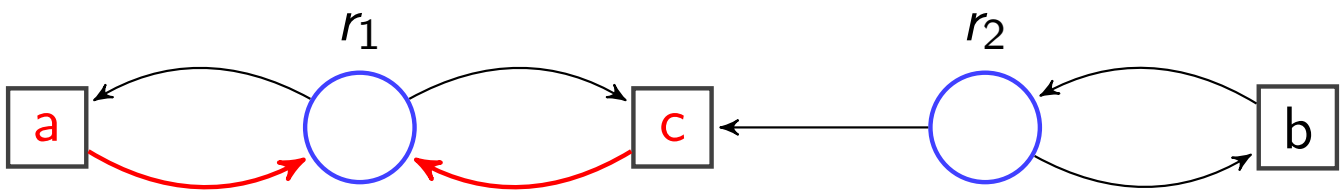
$$W_\sigma = \mathcal{E} \times \mathcal{E}^{-1} \cap \Sigma^2$$

$$a \to \bigcirc \to b$$

$$a \to \bigcirc \leftarrow b$$

Independence

A closer look at the signature



$$C_\sigma = \mathcal{E}^2 \cap \Sigma^2$$

$$W_\sigma = \mathcal{E} \times \mathcal{E}^{-1} \cap \Sigma^2$$

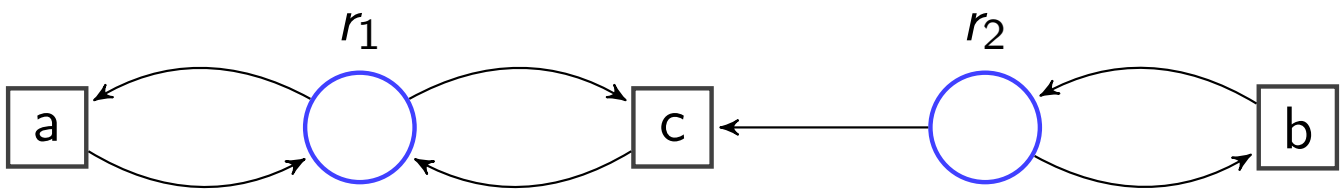
$$a \to \bigcirc \to b$$

$$a \to \bigcirc \leftarrow b$$

Whiteboard Example

Independence

A closer look at the signature



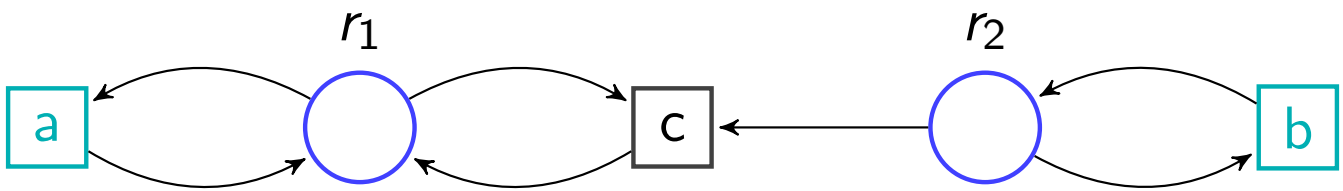
$$C_\sigma = \mathcal{E}^2 \cap \Sigma^2 \qquad a \to \bigcirc \to b$$

$$W_\sigma = \mathcal{E} \times \mathcal{E}^{-1} \cap \Sigma^2 \qquad a \to \bigcirc \leftarrow b$$

$$D_\sigma = C_\sigma \cup C_\sigma^{-1} \cup W_\sigma$$

Independence

A closer look at the signature



$$C_\sigma = \mathcal{E}^2 \cap \Sigma^2$$

$$W_\sigma = \mathcal{E} \times \mathcal{E}^{-1} \cap \Sigma^2$$

$$D_\sigma = C_\sigma \cup C_\sigma^{-1} \cup W_\sigma$$

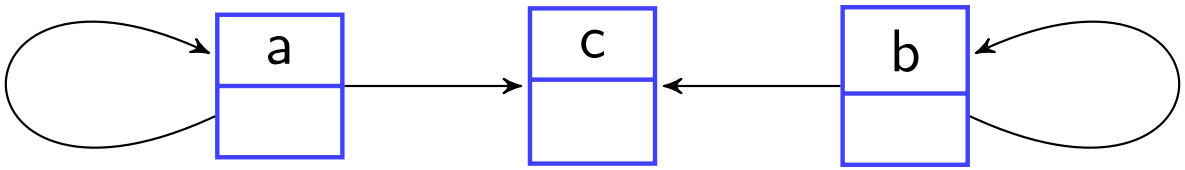
$$I_\sigma = \Sigma^2 \setminus D_\sigma$$

$a \to \bigcirc \to b$

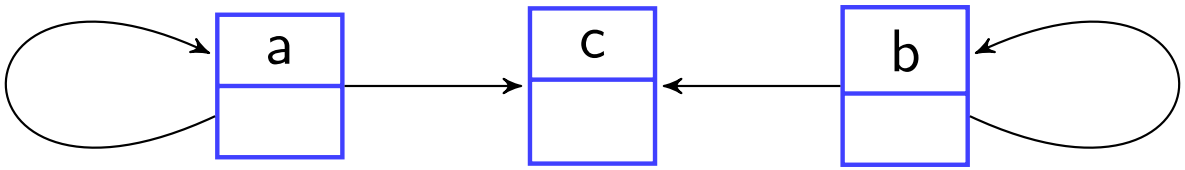
$a \to \bigcirc \leftarrow b$

Content

A particularly simple signature



A particularly simple signature

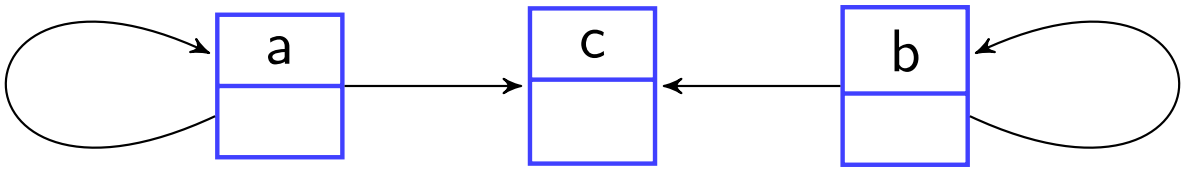


signature of a fat-system:

$$\sigma = (\Sigma, R, \mathcal{E})$$

Where $\mathcal{E} \subseteq R \times \Sigma \cup \Sigma \times R$

A particularly simple signature

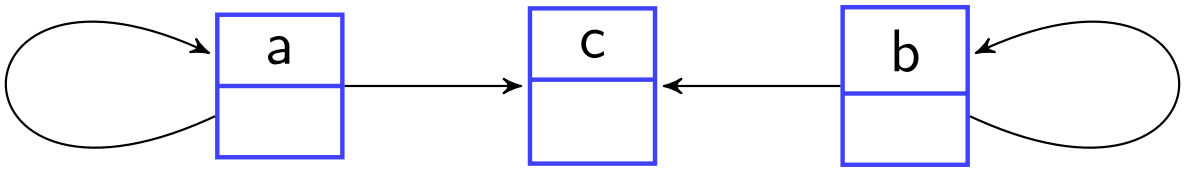


New Type of signature:

$$\sigma = (\Sigma, C)$$

Where $C \subseteq \Sigma^2$

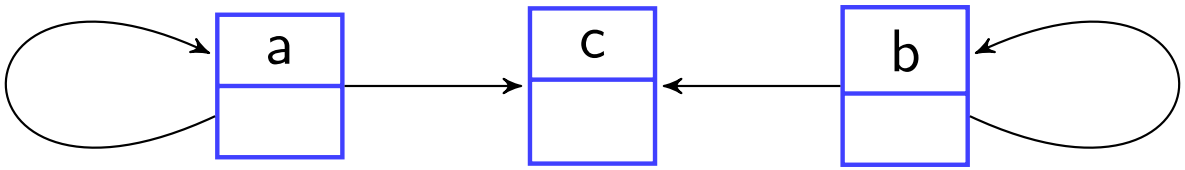
A particularly simple signature



Finite Asynchronous Cellular Transition System (fact-system):

$$\tau = (\Sigma, C, X, \Delta)$$

A particularly simple signature



Finite Asynchronous Cellular Transition System (fact-system):

$$\tau = (\Sigma, C, X, \Delta)$$

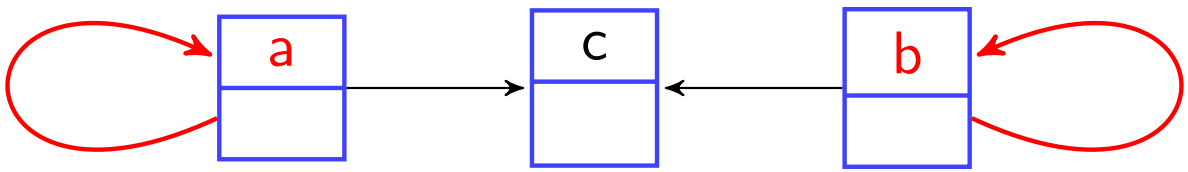
Where

$$\Delta = \{\delta_a \mid a \in \Sigma\}$$

And for $a \in \Sigma$

$$\delta_a \subseteq X^{Ca} \times X$$

A particularly simple signature



Finite Asynchronous Cellular Transition System (fact-system):

$$\tau = (\Sigma, C, X, \Delta)$$

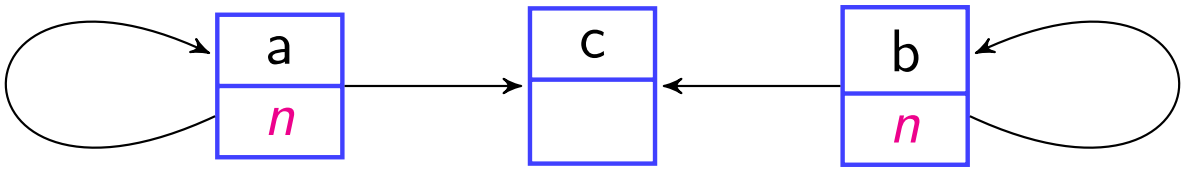
Where

$$\Delta = \{\delta_a \mid a \in \Sigma\}$$

And for $a \in \Sigma$

$$\delta_a \subseteq X^{Ca} \times X$$

A particularly simple signature



Finite Asynchronous Cellular Transition System (fact-system):

$$\tau = (\Sigma, C, X, \Delta)$$

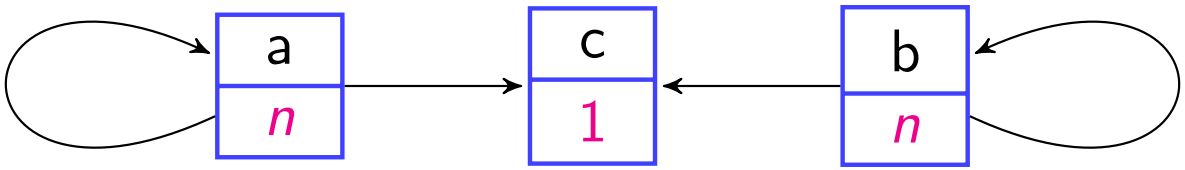
Where

$$\Delta = \{\delta_a \mid a \in \Sigma\}$$

And for $a \in \Sigma$

$$\delta_a \subseteq X^{Ca} \times X$$

A particularly simple signature



Finite Asynchronous Cellular Transition System (fact-system):

$$\tau = (\Sigma, C, X, \Delta)$$

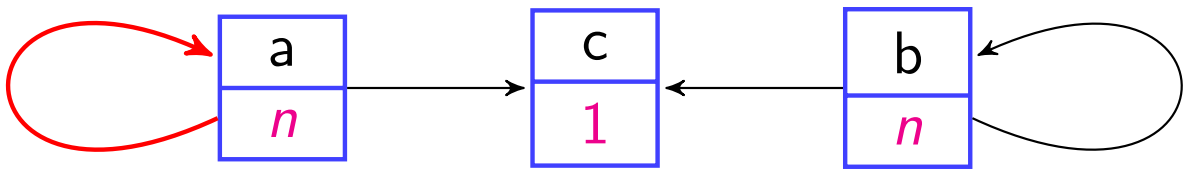
Where

$$\Delta = \{\delta_a \mid a \in \Sigma\}$$

And for $a \in \Sigma$

$$\delta_a \subseteq X^{Ca} \times X$$

A particularly simple signature



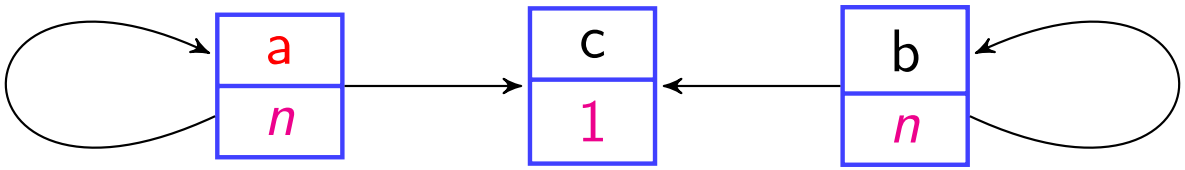
δ_a

a	a
i	$i-1$

δ_b

b	b
i	$i-1$

A particularly simple signature



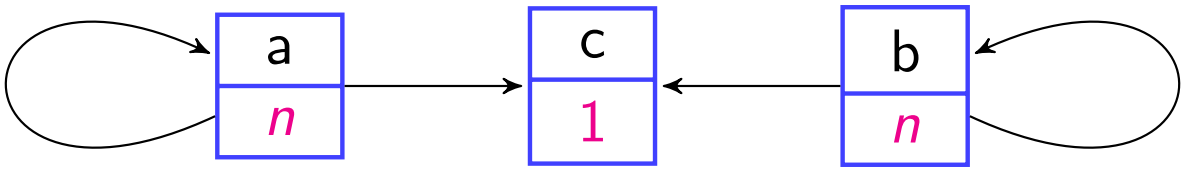
δ_a

a	a
i	$i-1$

δ_b

b	b
i	$i-1$

A particularly simple signature



δ_a

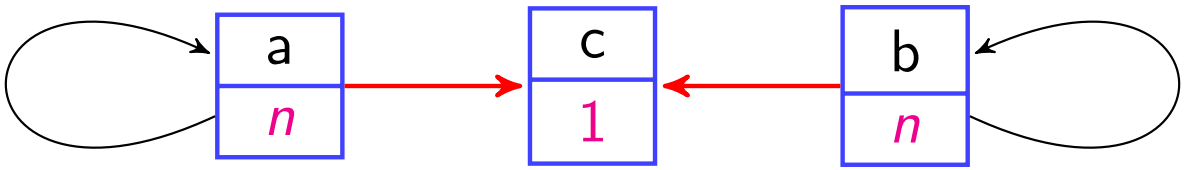
a	a
i	$i-1$

δ_b

b	b
i	$i-1$

$0 < i \leq n$

A particularly simple signature



δ_a

a	a
i	$i-1$

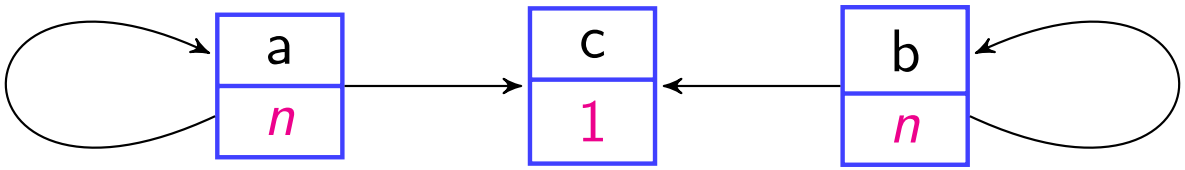
δ_c

a	b	c
0	0	0

δ_b

b	b
i	$i-1$

A particularly simple signature



δ_a

a	a
i	$i-1$

δ_c

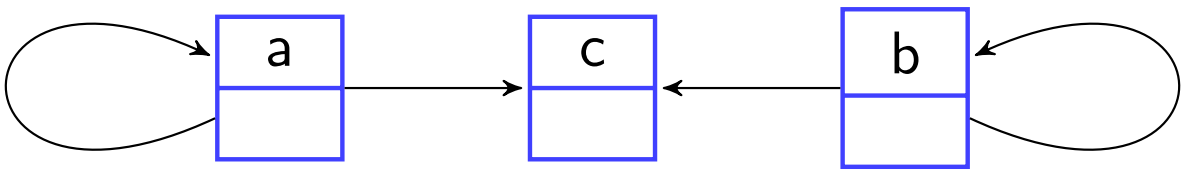
a	b	c
0	0	0

δ_b

b	b
i	$i-1$

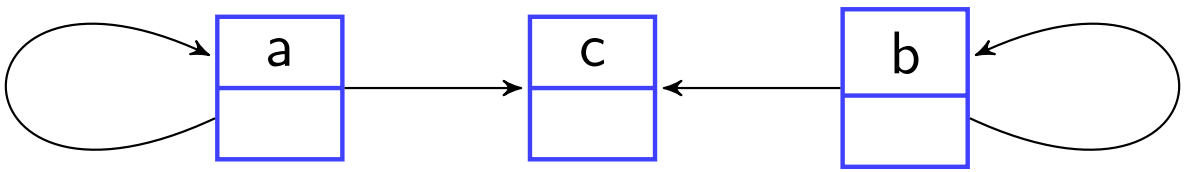
Independence

no writing conflicts



Independence

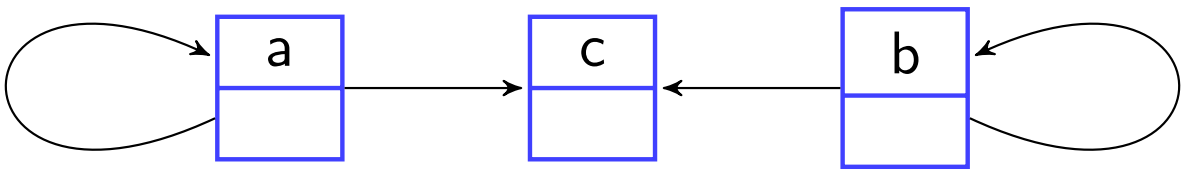
no writing conflicts



When are two Actions $a, b \in \Sigma$ **dependent** ?

Independence

no writing conflicts

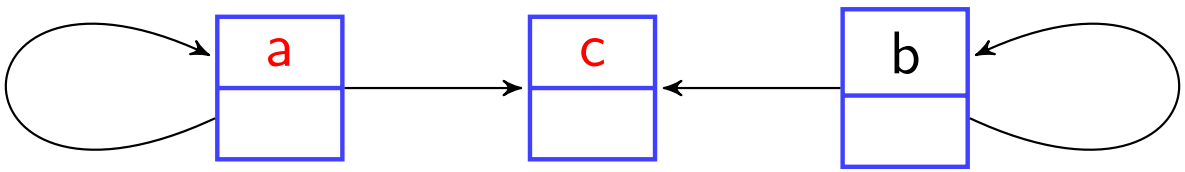


When are two Actions $a, b \in \Sigma$ **dependent** ?

In a signature σ two Actions a, b are called independent,
if σ's structure doesn't allow a fat-system to distinguish ab and ba.

Independence

no writing conflicts



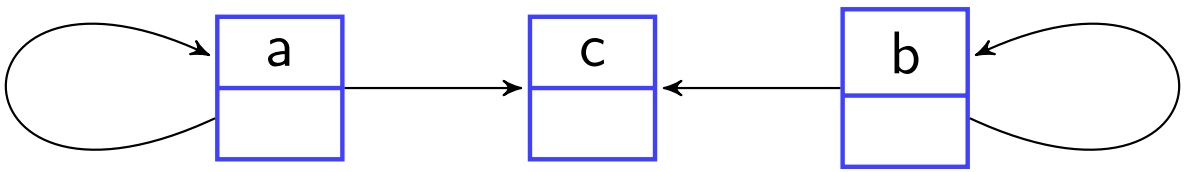
$$\sigma = (\Sigma, C)$$

$C_\sigma = C$

$(a \to \bigcirc \to b)$

Independence

no writing conflicts



$$\sigma = (\Sigma, C)$$

$C_\sigma = C$

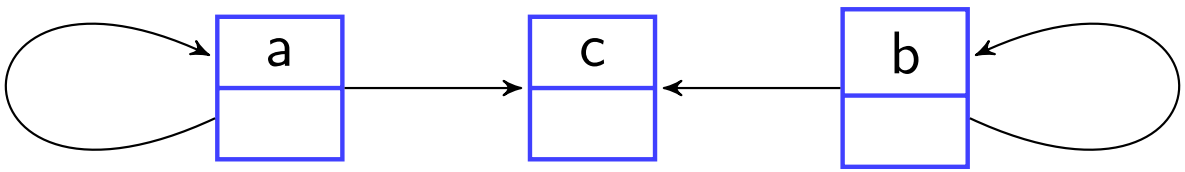
$W_\sigma = id_\Sigma$

$(a \to \bigcirc \to b)$

$(a \to \bigcirc \leftarrow b)$

Independence

no writing conflicts



$$\sigma = (\Sigma, C)$$

$C_\sigma = C$

$W_\sigma = id_\Sigma$

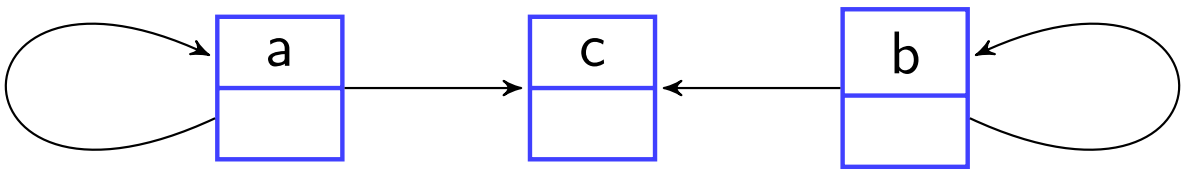
$D_\sigma = C_\sigma \cup C_\sigma^{-1} \cup W_\sigma$

$(a \to \bigcirc \to b)$

$(a \to \bigcirc \leftarrow b)$

Independence

no writing conflicts



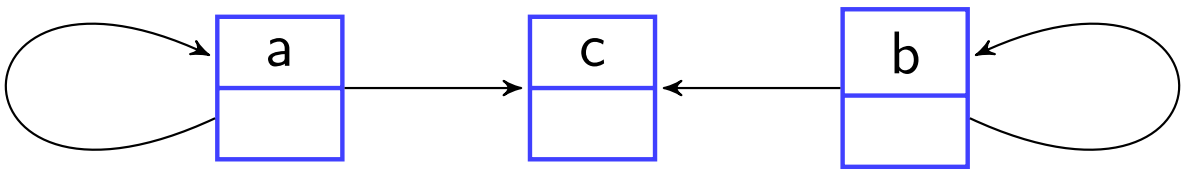
$$\sigma = (\Sigma, C)$$

$$C_\sigma = C$$
$$W_\sigma = id_\Sigma$$
$$D_\sigma = C_\sigma \cup C_\sigma^{-1} \cup W_\sigma$$
$$I_\sigma = \Sigma^2 \setminus D_\sigma$$

$$(a \to \bigcirc \to b)$$
$$(a \to \bigcirc \leftarrow b)$$

Independence

no writing conflicts



$$\sigma = (\Sigma, C)$$

$$C_\sigma = C \qquad (a \to \bigcirc \to b)$$

$$W_\sigma = id_\Sigma \qquad (a \to \bigcirc \leftarrow b)$$

$$D_\sigma = C_\sigma \cup C_\sigma^{-1} \cup W_\sigma$$

$$I_\sigma = \Sigma^2 \setminus D_\sigma \qquad \{(a, b) \mid a, b \text{ not adjacent}\}$$

Simulations

Are fact-systems regular?

Simulations

NDT $\rightarrow$ fact

Proof.

Let $\tau = (\Sigma, Q, \delta)$ be a regular finite transition system.
WLOG let $Q = \{q_0, \ldots, q_{k-1}\}$.

Simulations

NDT $\rightarrow$ fact

Proof.

Let $\tau = (\Sigma, Q, \delta)$ be a regular finite transition system.
WLOG let $Q = \{q_0, \ldots, q_{k-1}\}$.

Simulations

NDT $\rightarrow$ fact

Proof.

Let $\tau = (\Sigma, Q, \delta)$ be a regular finite transition system.
WLOG let $Q = \{q_0, \ldots, q_{k-1}\}$.

Simulations

NDT $\rightarrow$ fact

Proof.

Let $\tau = (\Sigma, Q, \delta)$ be a regular finite transition system.
WLOG let $Q = \{q_0, \ldots, q_{k-1}\}$.

$\tau_C = (\Sigma, C, X, \Delta)$ s.t.:

Simulations

NDT $\rightarrow$ fact

Proof.

Let $\tau = (\Sigma, Q, \delta)$ be a regular finite transition system.
WLOG let $Q = \{q_0, \ldots, q_{k-1}\}$.

$\tau_C = (\Sigma, C, X, \Delta)$ s.t.:

- $C = \Sigma^2$

Simulations

NDT $\rightarrow$ fact

Proof.

Let $\tau = (\Sigma, Q, \delta)$ be a regular finite transition system.
WLOG let $Q = \{q_0, \ldots, q_{k-1}\}$.

$\tau_C = (\Sigma, C, X, \Delta)$ s.t.:

- $C = \Sigma^2$
- $X = \{0, \ldots, k-1\}$

Simulations

NDT $\rightarrow$ fact

Proof.

Let $\tau = (\Sigma, Q, \delta)$ be a regular finite transition system.
WLOG let $Q = \{q_0, \ldots, q_{k-1}\}$.

$\tau_C = (\Sigma, C, X, \Delta)$ s.t.:

- $C = \Sigma^2$
- $X = \{0, \ldots, k-1\}$
- $\Delta = \bigcup_{a \in \Sigma} \{\delta_a\}$

Simulations

NDT $\rightarrow$ fact

Proof.

Let $\tau = (\Sigma, Q, \delta)$ be a regular finite transition system.
WLOG let $Q = \{q_0, \ldots, q_{k-1}\}$.

let $a \in \Sigma, \beta \in X^\Sigma$ and $x \in X$.

$\tau_C = (\Sigma, C, X, \Delta)$ s.t.: $(\beta, x) \in \delta_a$

- $C = \Sigma^2$
- $X = \{0, \ldots, k-1\}$
- $\Delta = \bigcup_{a \in \Sigma} \{\delta_a\}$

Simulations

NDT $\rightarrow$ fact

Proof.

Let $\tau = (\Sigma, Q, \delta)$ be a regular finite transition system.
WLOG let $Q = \{q_0, \ldots, q_{k-1}\}$.

let $a \in \Sigma, \beta \in X^\Sigma$ and $x \in X$.

$\tau_C = (\Sigma, C, X, \Delta)$ s.t.: $(\beta, x) \in \delta_a \Leftrightarrow \exists s, s' :$

- $C = \Sigma^2$
- $X = \{0, \ldots, k-1\}$
- $\Delta = \bigcup_{a \in \Sigma} \{\delta_a\}$

Simulations

NDT $\rightarrow$ fact

Proof.

Let $\tau = (\Sigma, Q, \delta)$ be a regular finite transition system.
WLOG let $Q = \{q_0, \ldots, q_{k-1}\}$.

$\tau_C = (\Sigma, C, X, \Delta)$ s.t.:

- $C = \Sigma^2$
- $X = \{0, \ldots, k-1\}$
- $\Delta = \bigcup_{a \in \Sigma} \{\delta_a\}$

let $a \in \Sigma, \beta \in X^\Sigma$ and $x \in X$.

$(\beta, x) \in \delta_a \Leftrightarrow \exists s, s' :$

$(q_s, a, q_{s'}) \in \delta$

Simulations

NDT $\rightarrow$ fact

Proof.

Let $\tau = (\Sigma, Q, \delta)$ be a regular finite transition system.
WLOG let $Q = \{q_0, \ldots, q_{k-1}\}$.

$\tau_C = (\Sigma, C, X, \Delta)$ s.t.:

- $C = \Sigma^2$
- $X = \{0, \ldots, k-1\}$
- $\Delta = \bigcup_{a \in \Sigma} \{\delta_a\}$

let $a \in \Sigma, \beta \in X^\Sigma$ and $x \in X$.

$(\beta, x) \in \delta_a \Leftrightarrow \exists s, s' :$

$$(q_s, a, q_{s'}) \in \delta$$
$$\wedge \sum_{b \in \Sigma} \beta(b) \mod k = s$$

Simulations

NDT $\rightarrow$ fact

Proof.

Let $\tau = (\Sigma, Q, \delta)$ be a regular finite transition system.
WLOG let $Q = \{q_0, \ldots, q_{k-1}\}$.

$\tau_C = (\Sigma, C, X, \Delta)$ s.t.:

- $C = \Sigma^2$
- $X = \{0, \ldots, k-1\}$
- $\Delta = \bigcup_{a \in \Sigma} \{\delta_a\}$

let $a \in \Sigma, \beta \in X^\Sigma$ and $x \in X$.

$(\beta, x) \in \delta_a \Leftrightarrow \exists s, s' :$

$$(q_s, a, q_{s'}) \in \delta$$
$$\wedge \sum_{b \in \Sigma} \beta(b) \mod k = s$$
$$\wedge s - \beta(a) + x \mod k = s'$$

Simulations

NDT $\rightarrow$ fact

Proof.

Let $\tau = (\Sigma, Q, \delta)$ be a regular finite transition system.
WLOG let $Q = \{q_0, \ldots, q_{k-1}\}$.

$\tau_C = (\Sigma, C, X, \Delta)$ s.t.:

- $C = \Sigma^2$
- $X = \{0, \ldots, k-1\}$
- $\Delta = \bigcup_{a \in \Sigma} \{\delta_a\}$

let $a \in \Sigma, \beta \in X^\Sigma$ and $x \in X$.

$(\beta, x) \in \delta_a \Leftrightarrow \exists s, s' :$

$$(q_s, a, q_{s'}) \in \delta$$
$$\wedge \sum_{b \in \Sigma} \beta(b) \mod k = s$$
$$\wedge s - \beta(a) + x \mod k = s'$$

τ_C simulates τ. $\square$

Simulations

fat $\rightarrow$ fact

Can a finite asynchronous cellular automaton (faca) simulate a faa?

Simulations

fat $\rightarrow$ fact

Can a finite asynchronous cellular automaton (faca) simulate a faa?

– both accept regular languages.

Simulations

fat $\rightarrow$ fact

Can a finite asynchronous cellular automaton (faca) simulate a faa?

– both accept regular languages.

What would be an interesting simulation?

Simulations

fat $\rightarrow$ fact

Can a finite asynchronous cellular automaton (faca) simulate a faa?

– both accept regular languages.

What would be an interesting simulation?

– One that maintains independence.

Simulations

fat $\rightarrow$ fact

How can we assure that all independence is maintained?

Simulations

fat $\rightarrow$ fact

How can we assure that all independence is maintained?

Try not to interfere with the signature:

Fat signature
$\sigma = (\Sigma, R, \mathcal{E})$

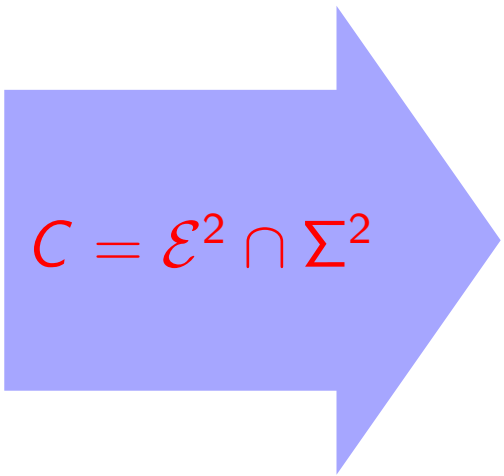
Simulations

fat $\rightarrow$ fact

How can we assure that all independence is maintained?

Try not to interfere with the signature:

Fat signature
$\sigma = (\Sigma, R, \mathcal{E})$



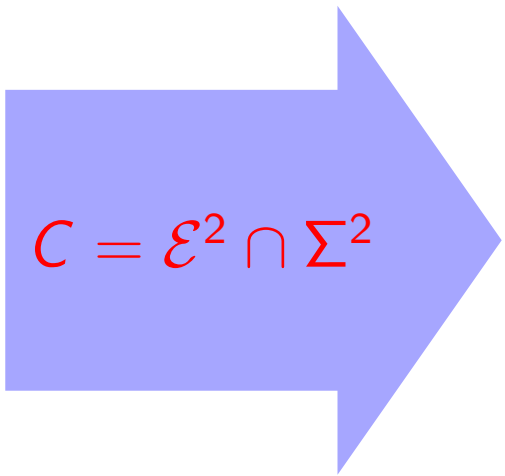
Simulations

fat $\rightarrow$ fact

How can we assure that all independence is maintained?

Try not to interfere with the signature:

Fat signature
$\sigma = (\Sigma, R, \mathcal{E})$



Fact signature
$\sigma' = (\Sigma, C)$

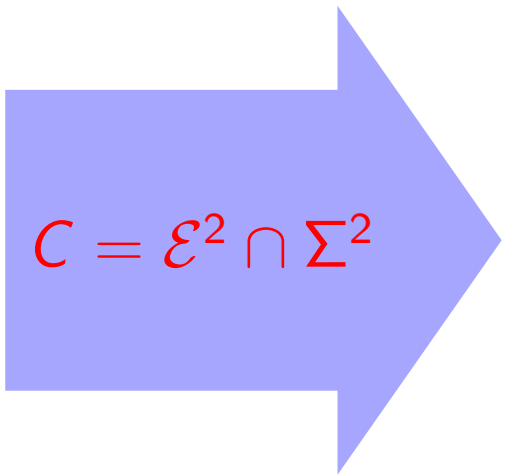
Simulations

fat $\rightarrow$ fact

How can we assure that all independence is maintained?

Try not to interfere with the signature:

Fat signature
$\sigma = (\Sigma, R, \mathcal{E})$



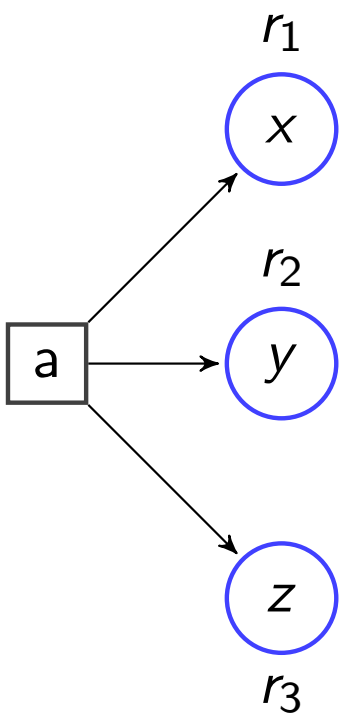
Fact signature
$\sigma' = (\Sigma, C)$

σ induces C

Simulations

fat $\rightarrow$ fact

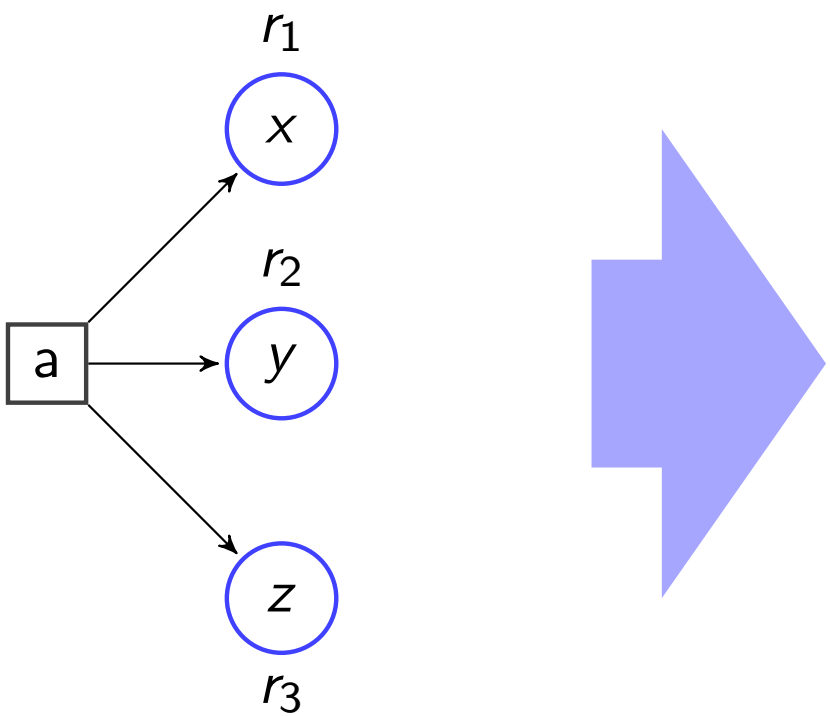
If there are no writing-conflicts



Simulations

fat $\rightarrow$ fact

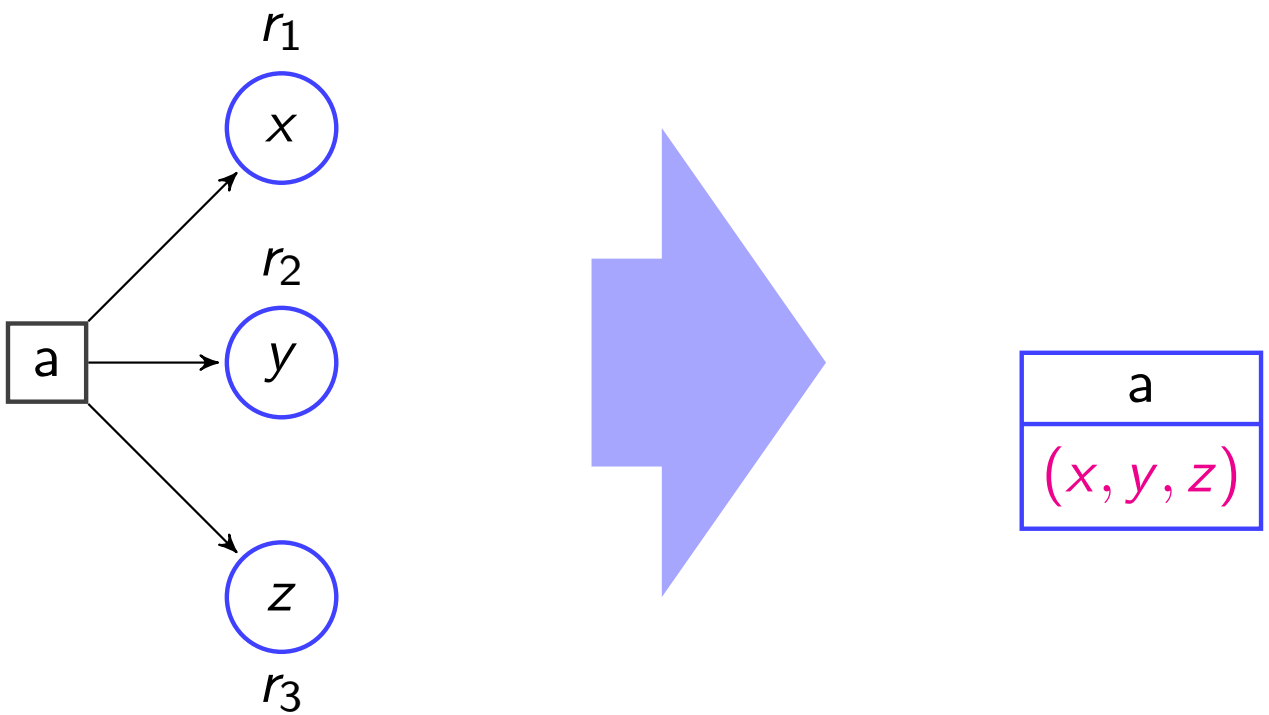
If there are no writing-conflicts



Simulations

fat $\rightarrow$ fact

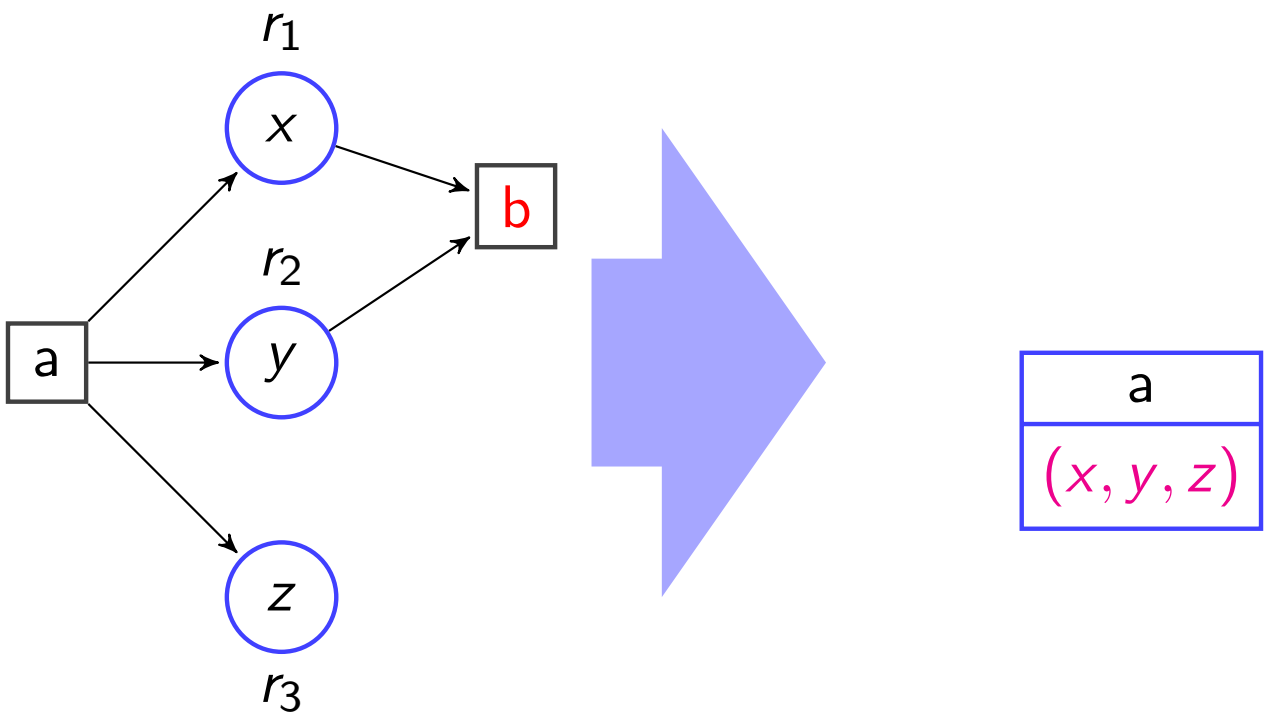
If there are no writing-conflicts



Simulations

fat $\rightarrow$ fact

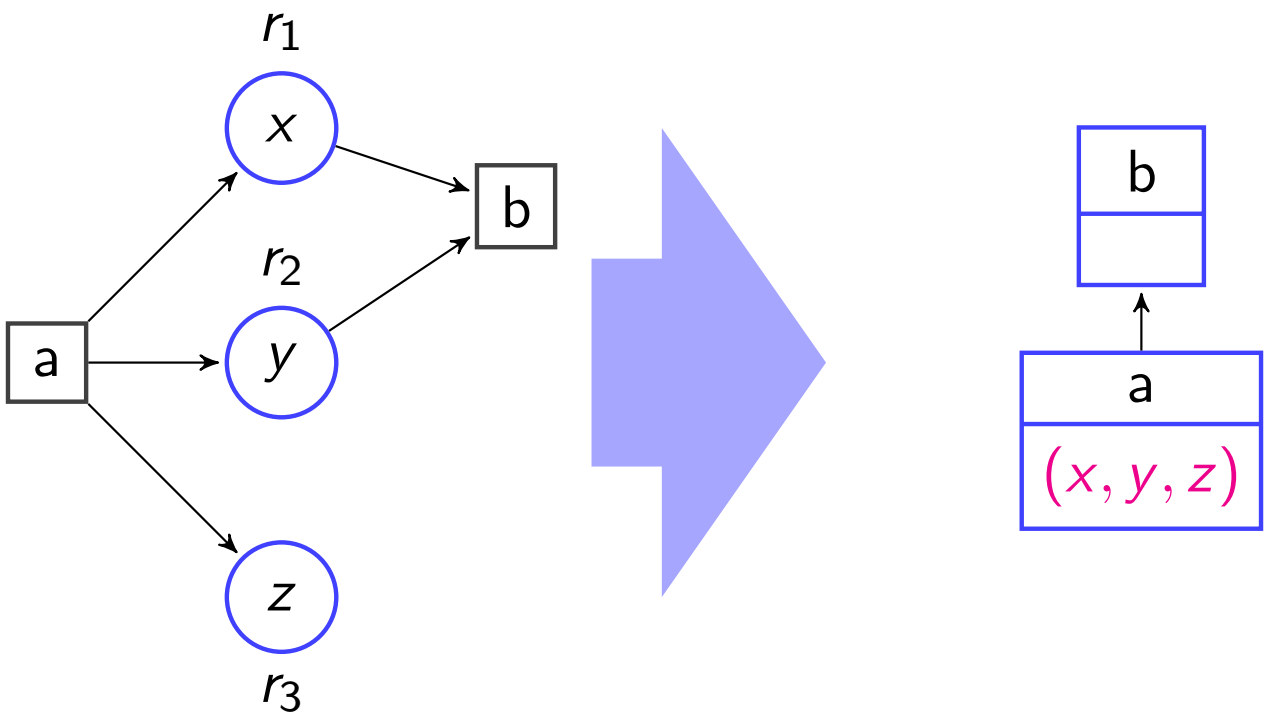
If there are no writing-conflicts



Simulations

fat $\rightarrow$ fact

If there are no writing-conflicts



Simulations

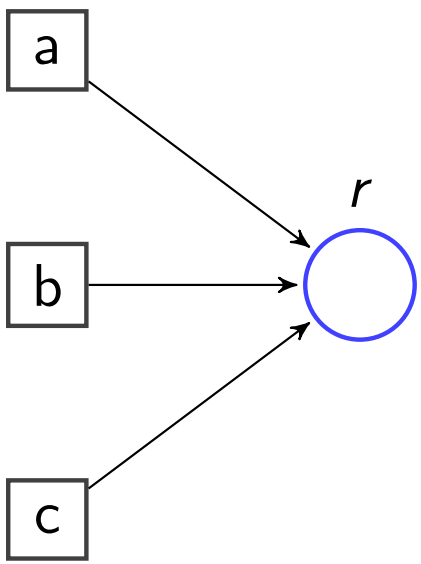
fat $\rightarrow$ fact

Writing-conflict

Simulations

fat $\rightarrow$ fact

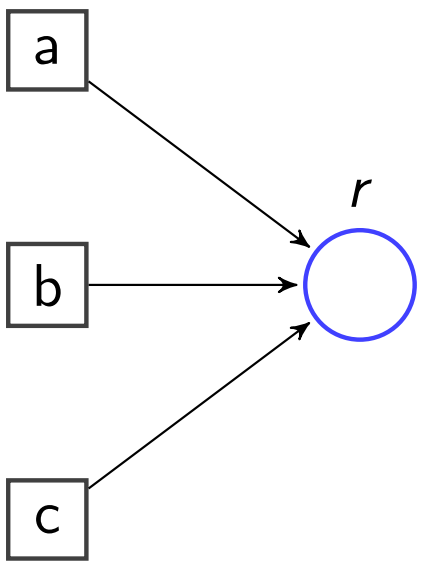
Writing-conflict

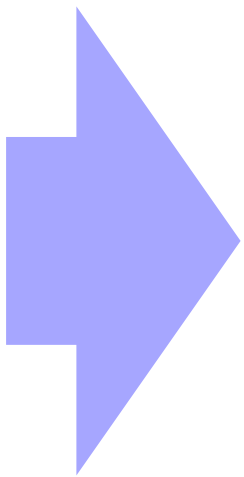


Simulations

fat $\rightarrow$ fact

Writing-conflict

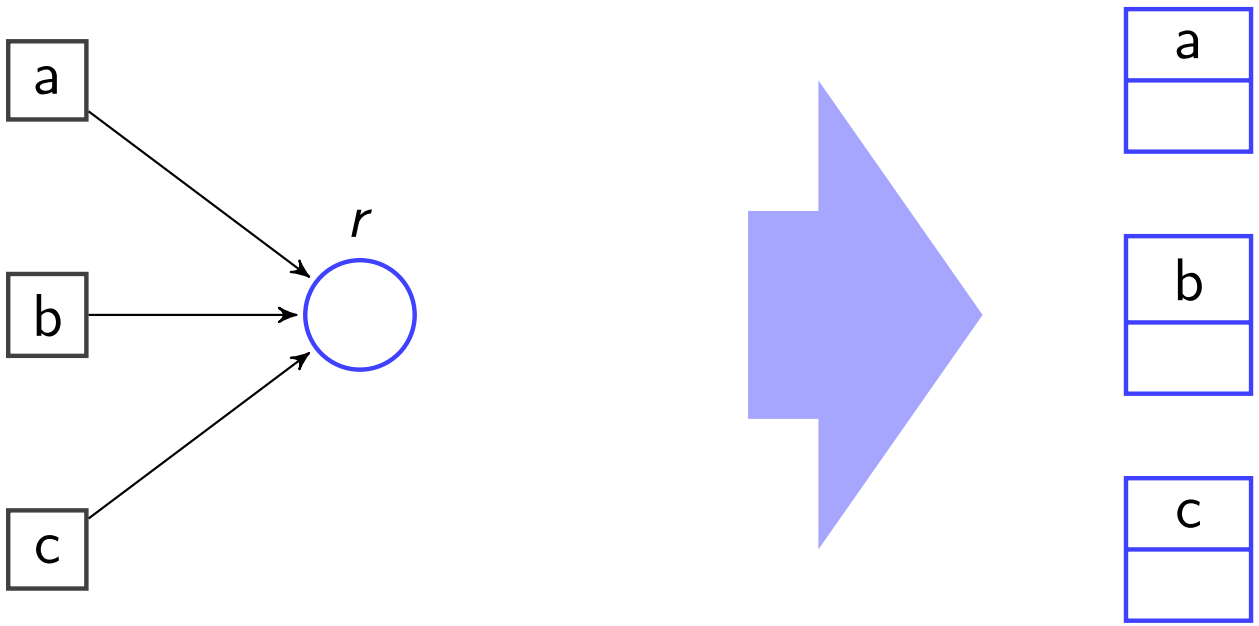




Simulations

fat $\rightarrow$ fact

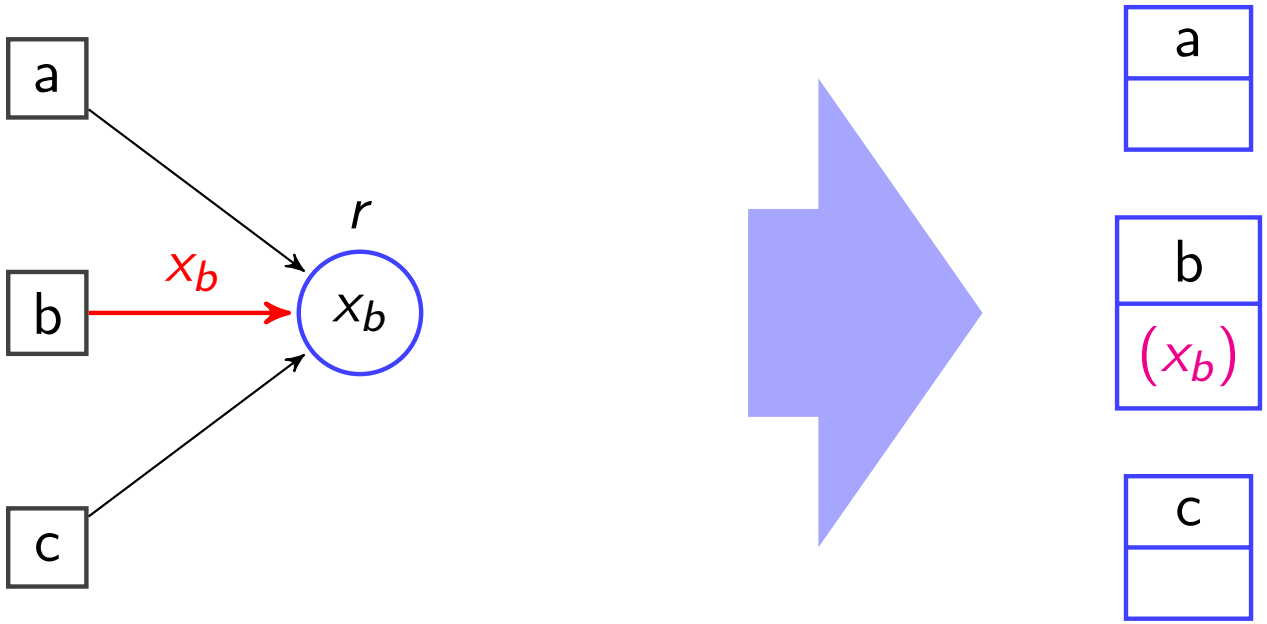
Writing-conflict



Simulations

fat $\rightarrow$ fact

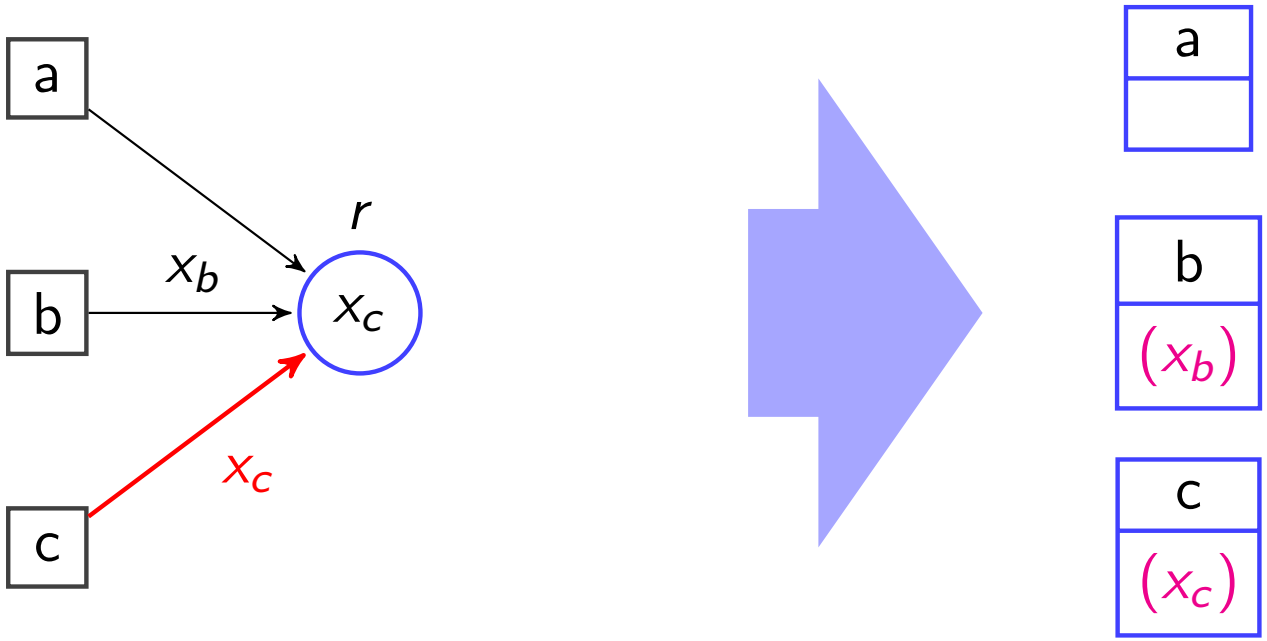
Writing-conflict



Simulations

fat $\rightarrow$ fact

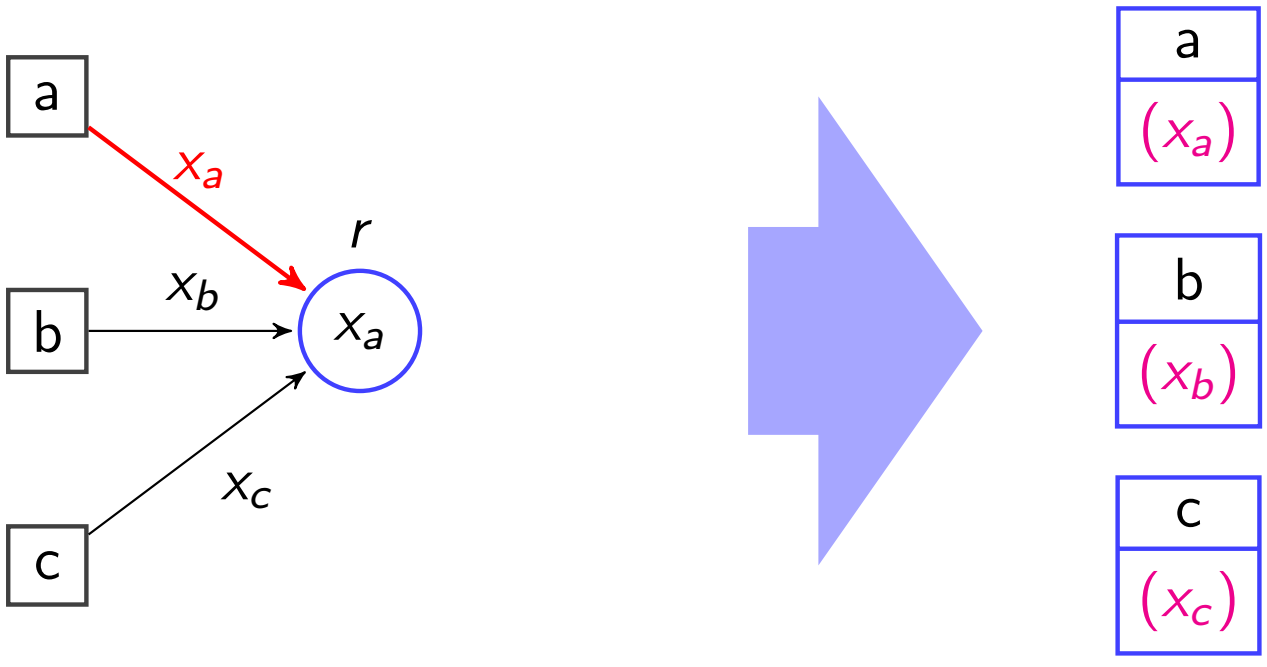
Writing-conflict



Simulations

fat $\rightarrow$ fact

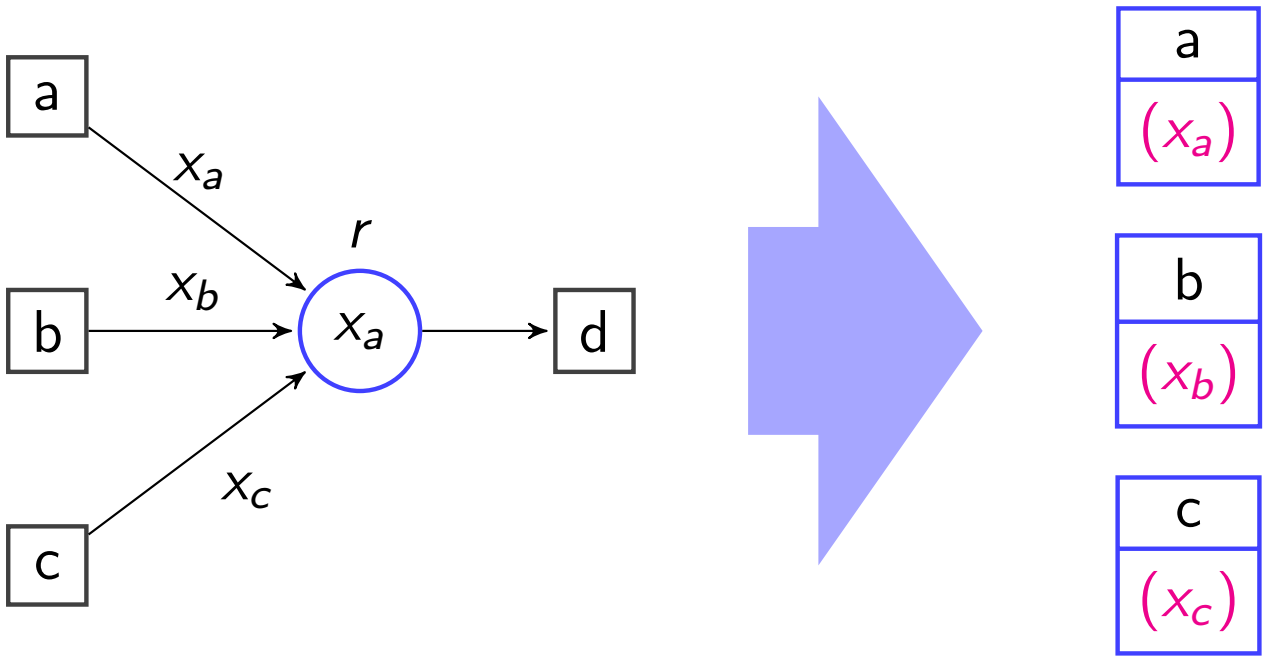
Writing-conflict



Simulations

fat $\rightarrow$ fact

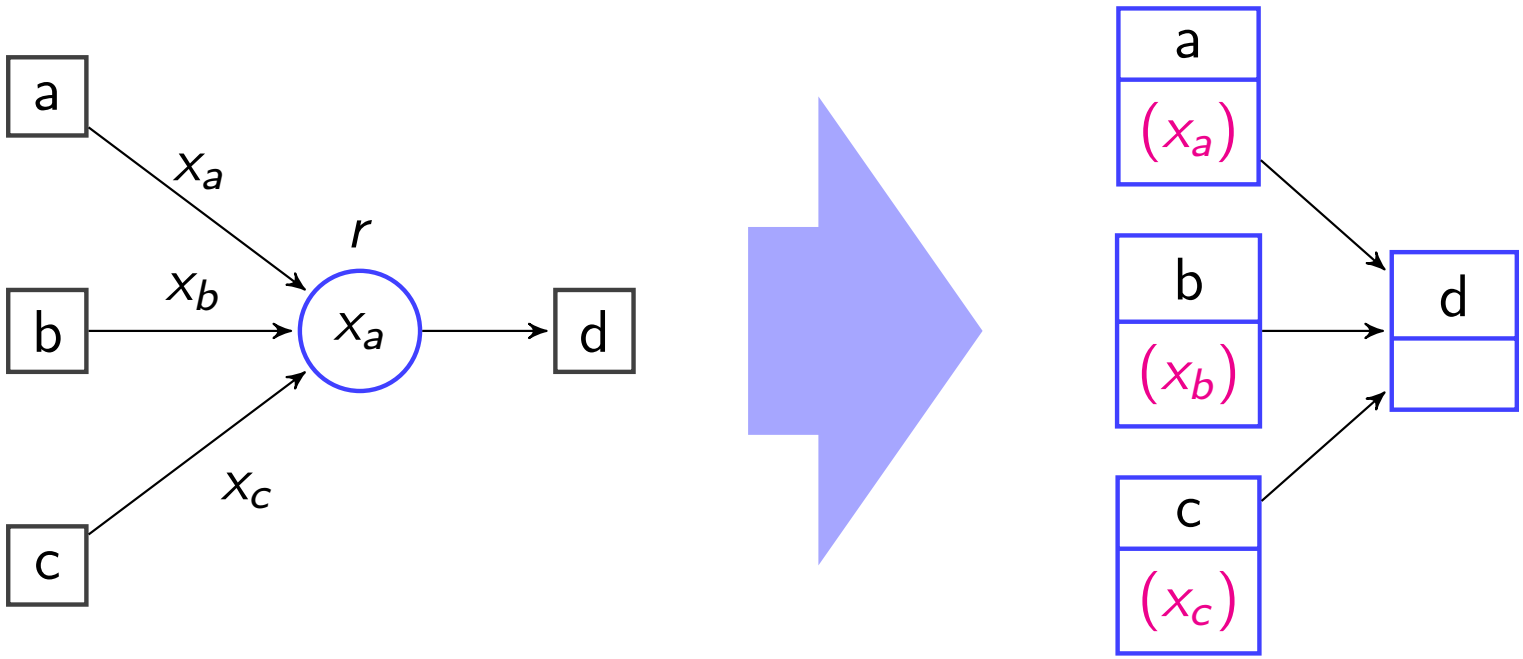
Writing-conflict



Simulations

fat $\rightarrow$ fact

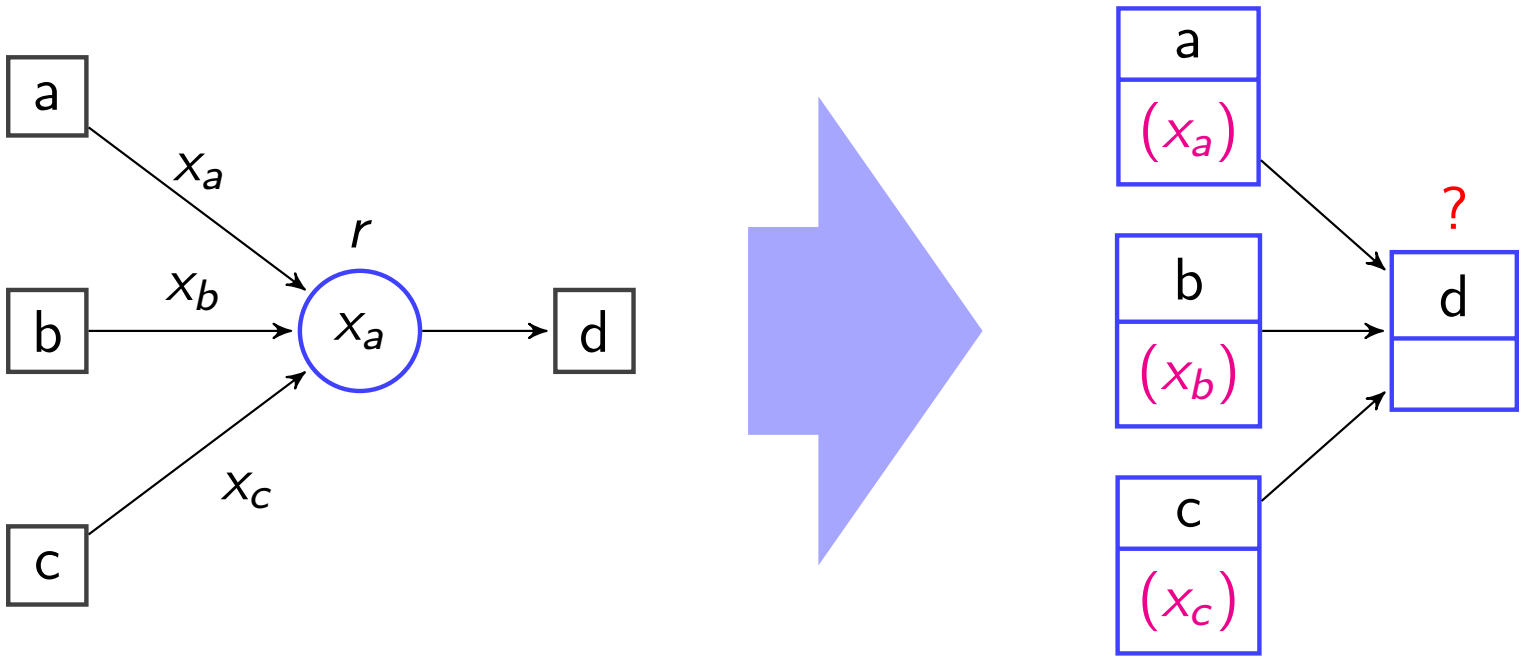
Writing-conflict



Simulations

fat $\rightarrow$ fact

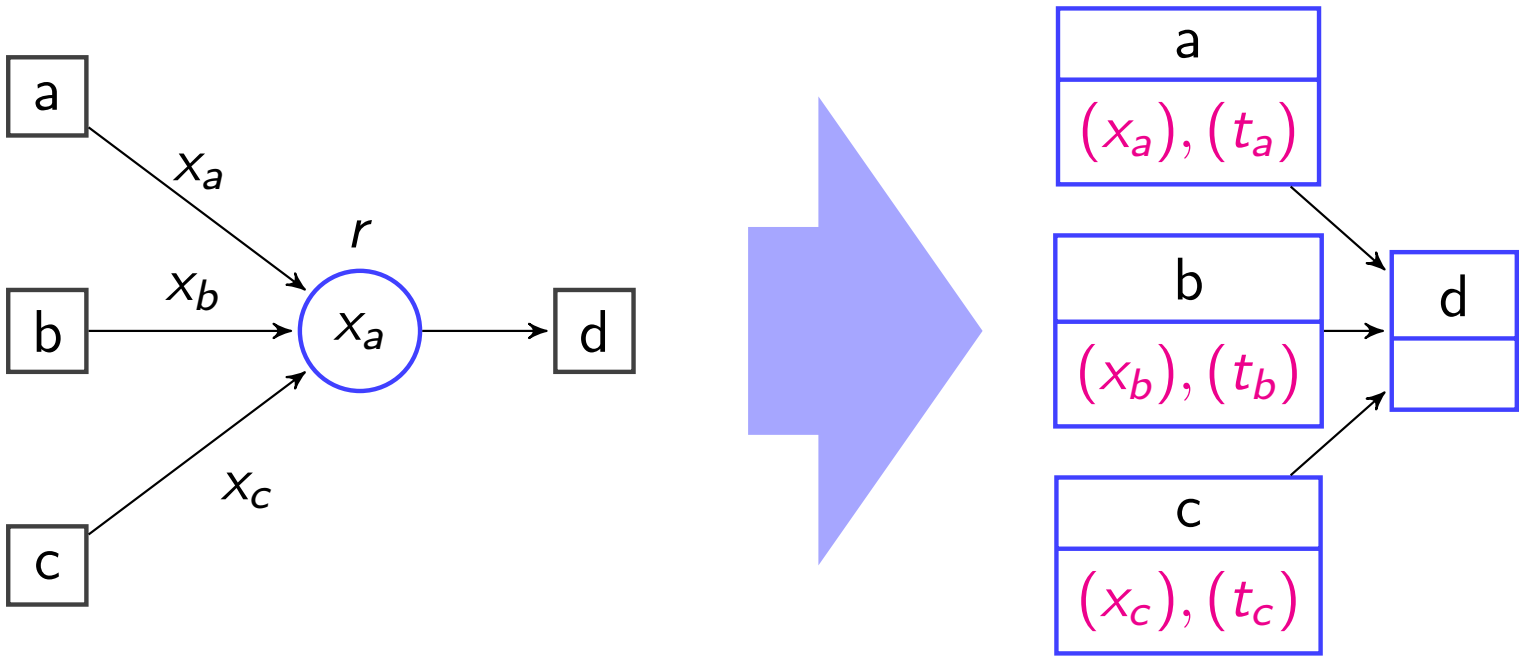
Writing-conflict



Simulations

fat $\rightarrow$ fact

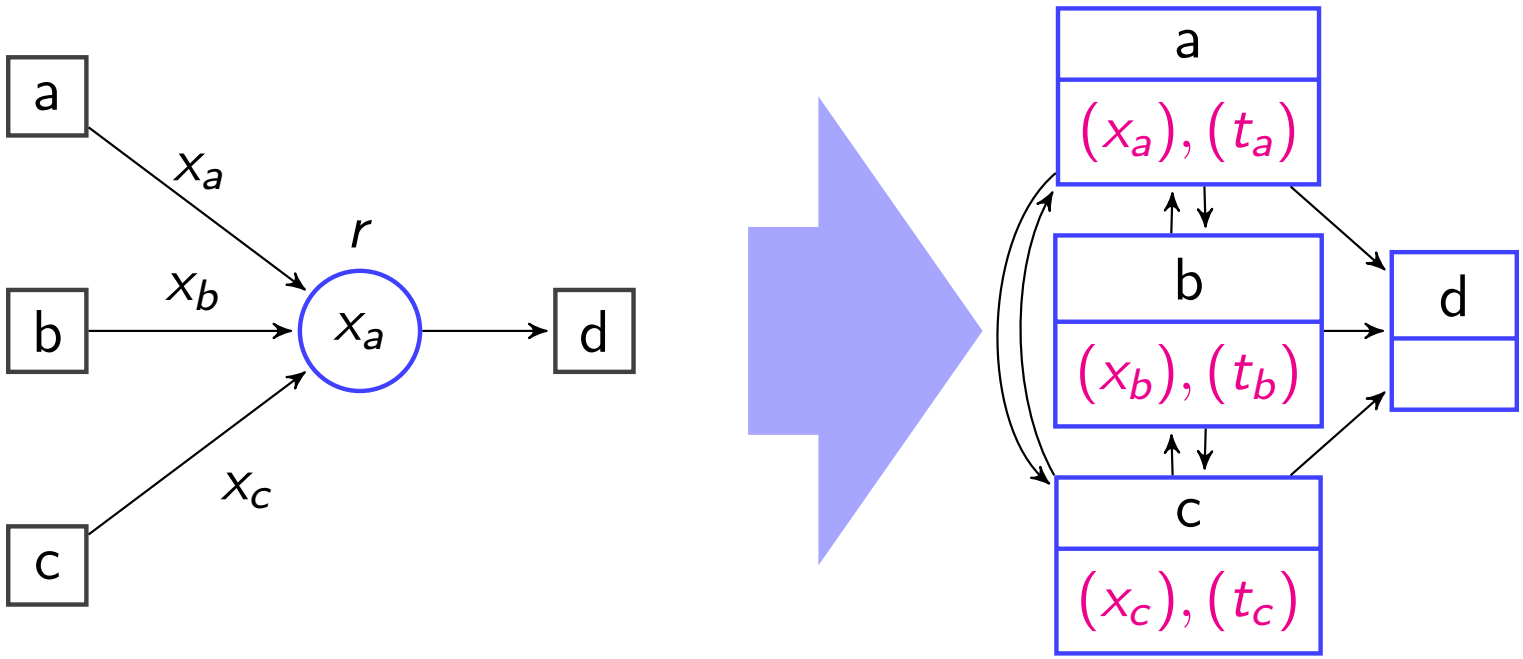
Add a time-stamp



Simulations

fat $\rightarrow$ fact

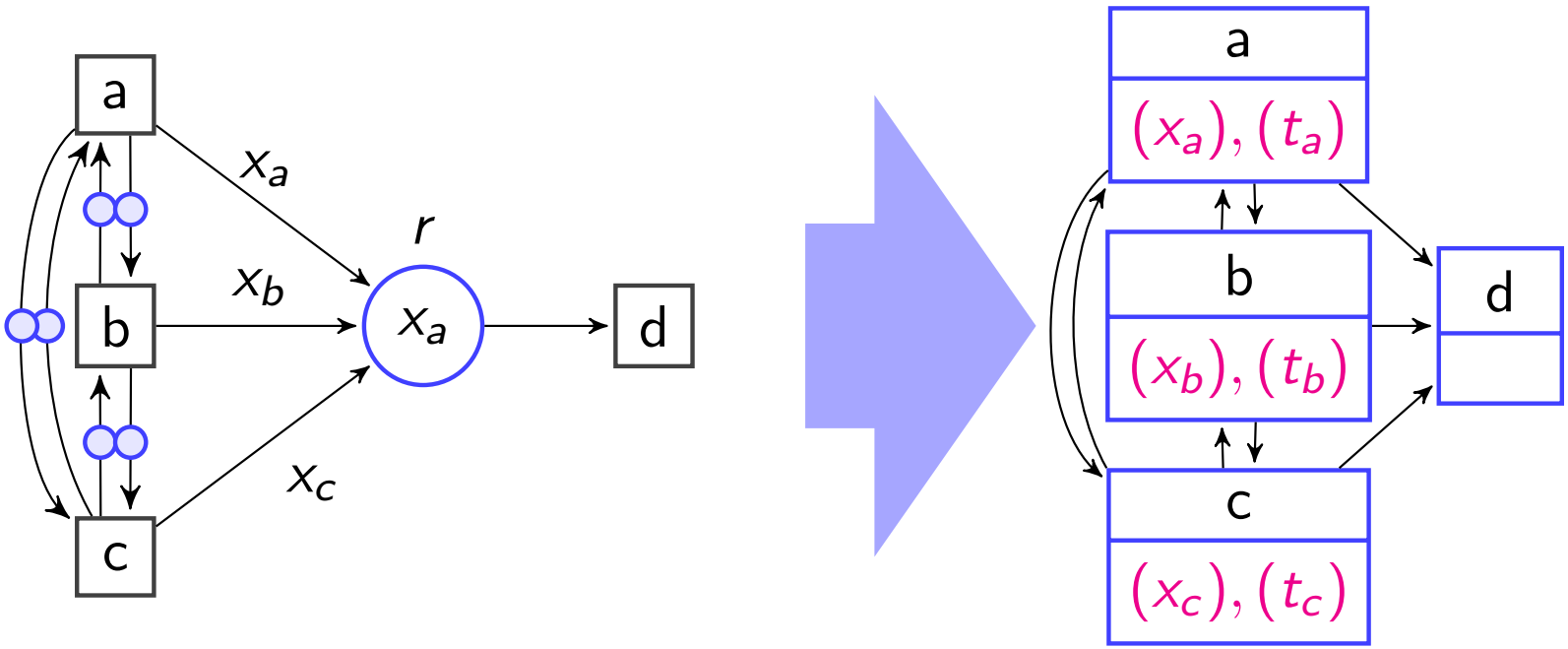
Add a time-stamp



Simulations

fat $\rightarrow$ fact

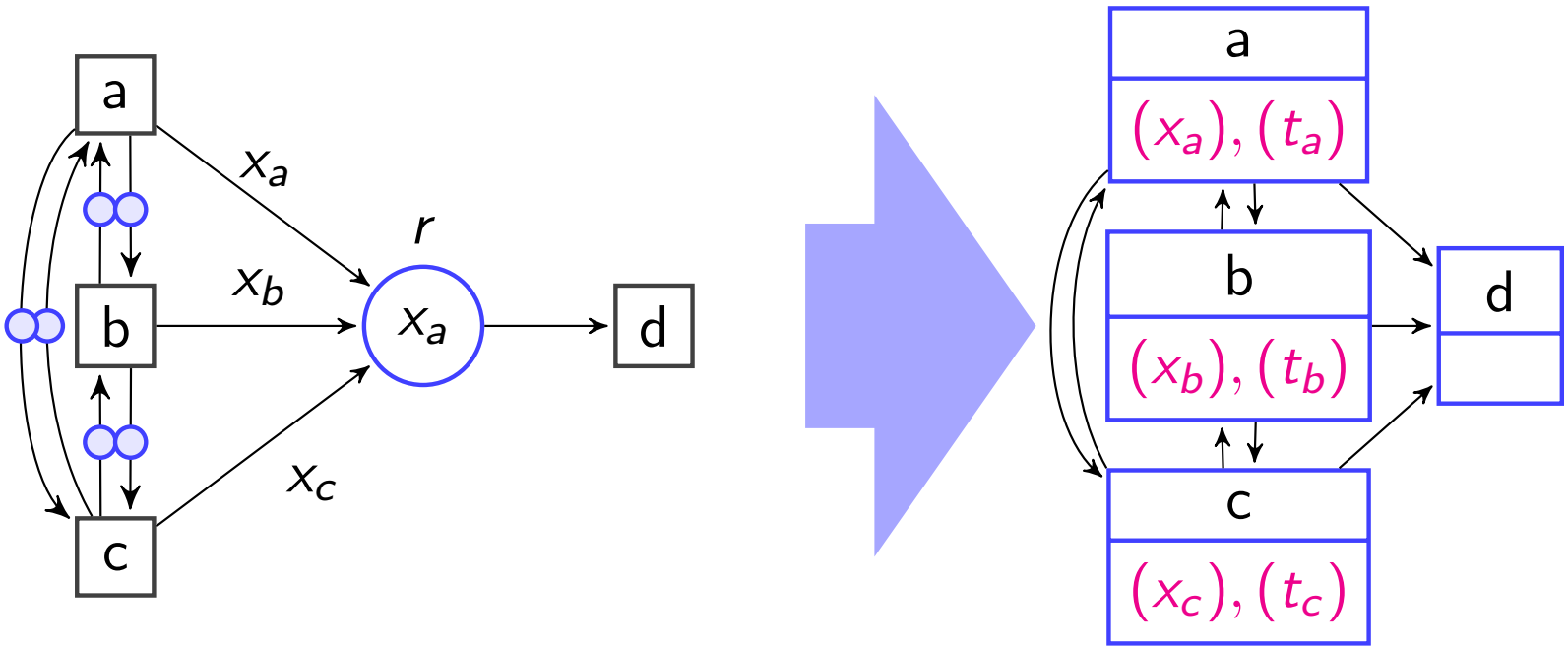
Add a time-stamp



Simulations

fat $\rightarrow$ fact

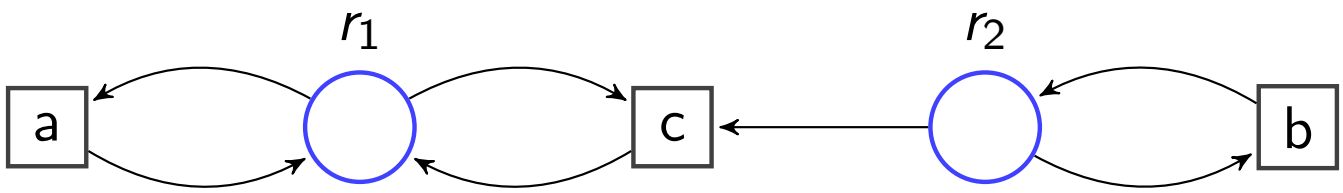
Add a time-stamp



This writing conflict is ω-redundant.

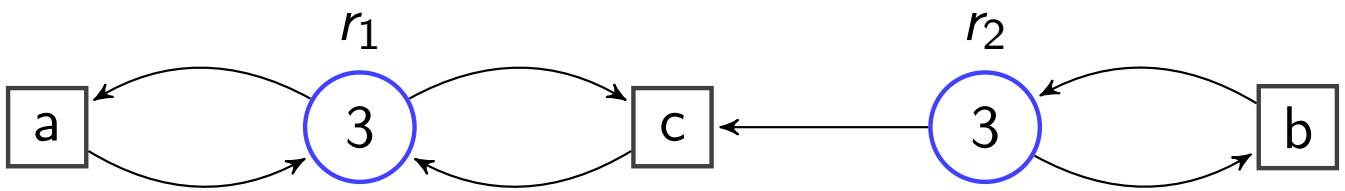
Simulations

fat $\rightarrow$ fact: Example



Simulations

fat $\rightarrow$ fact: Example



δ_a

r_1	r_1
i	$i-1$

δ_c

r_1	r_2	r_1
0	0	s

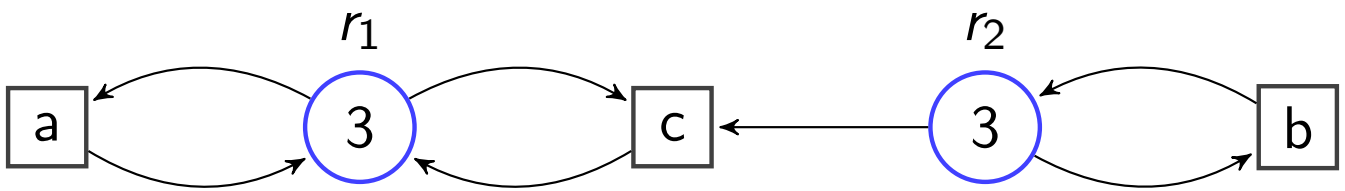
δ_b

r_2	r_2
i	$i-1$

$0 < i \leq 3$

Simulations

fat $\rightarrow$ fact: Example



δ_a

r_1	r_1
i	$i-1$

δ_c

r_1	r_2	r_1
0	0	s

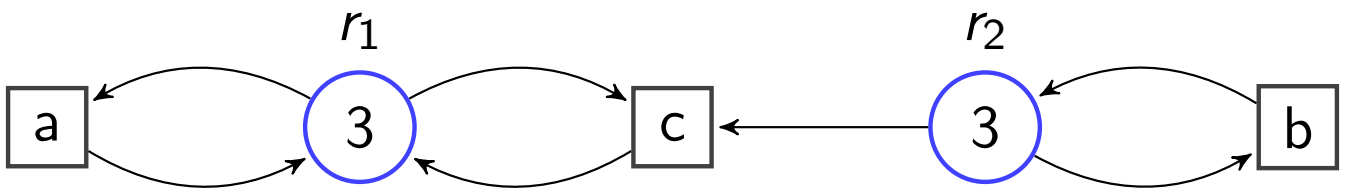
δ_b

r_2	r_2
i	$i-1$

$0 < i \leq 3$

Simulations

fat $\rightarrow$ fact: Example



δ_a

r_1	r_1
i	$i-1$

δ_c

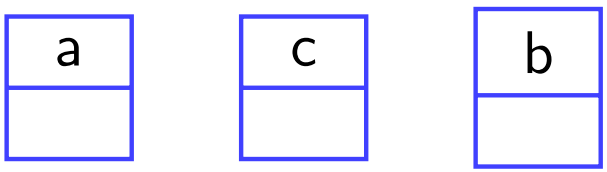
r_1	r_2	r_1
0	0	s

δ_b

r_2	r_2
i	$i-1$

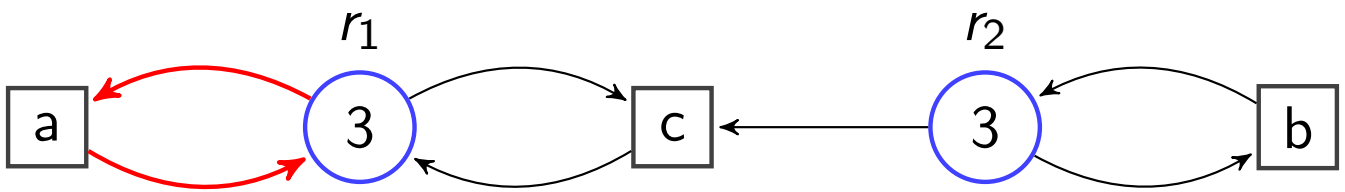
$0 < i \leq 3$

$$\Sigma = \Sigma_{fat}$$



Simulations

fat $\rightarrow$ fact: Example



δ_a

r_1	r_1
i	$i-1$

δ_c

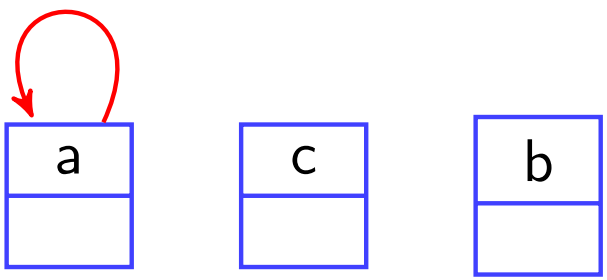
r_1	r_2	r_1
0	0	s

δ_b

r_2	r_2
i	$i-1$

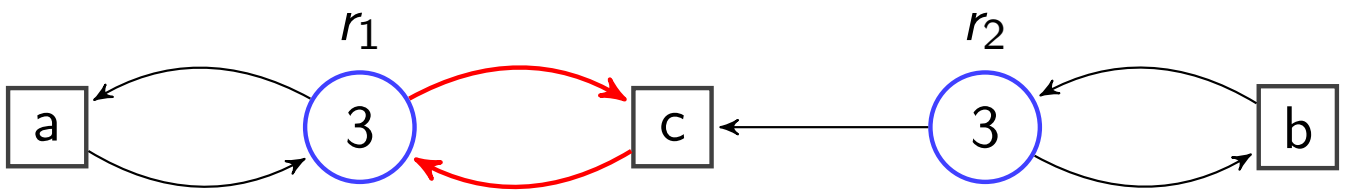
$0 < i \leq 3$

$$C = \mathcal{E}^2_{fat} \cap \Sigma^2$$



Simulations

fat $\rightarrow$ fact: Example



δ_a

r_1	r_1
i	$i-1$

δ_c

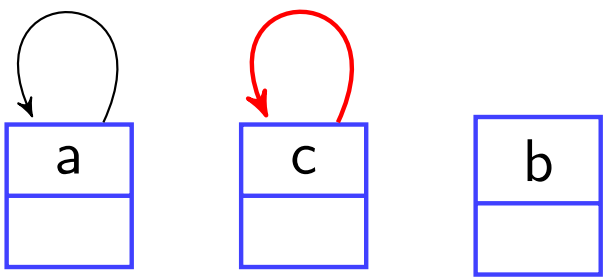
r_1	r_2	r_1
0	0	s

δ_b

r_2	r_2
i	$i-1$

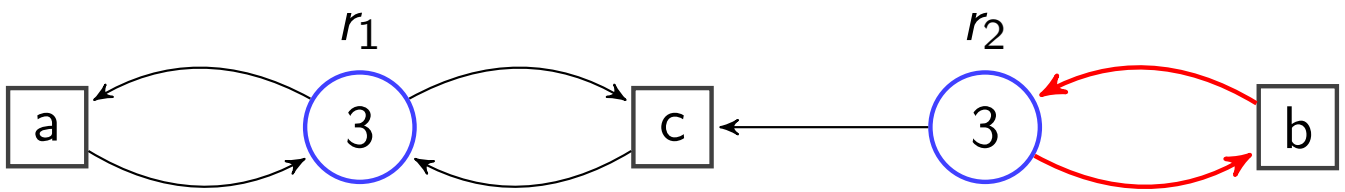
$0 < i \leq 3$

$$C = \mathcal{E}^2_{fat} \cap \Sigma^2$$



Simulations

fat $\rightarrow$ fact: Example



δ_a

r_1	r_1
i	$i-1$

δ_c

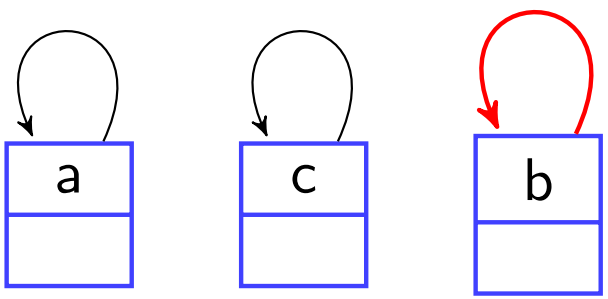
r_1	r_2	r_1
0	0	s

δ_b

r_2	r_2
i	$i-1$

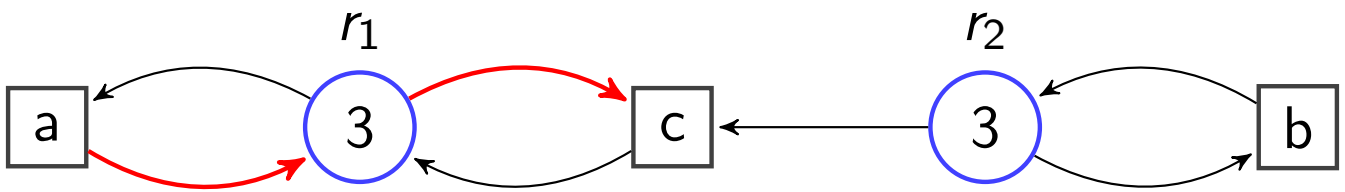
$0 < i \leq 3$

$$C = \mathcal{E}^2_{fat} \cap \Sigma^2$$



Simulations

fat $\rightarrow$ fact: Example



δ_a

r_1	r_1
i	$i-1$

δ_c

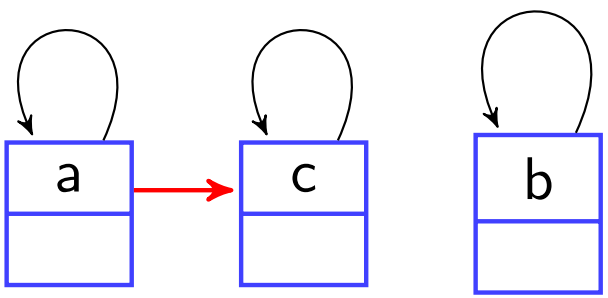
r_1	r_2	r_1
0	0	s

δ_b

r_2	r_2
i	$i-1$

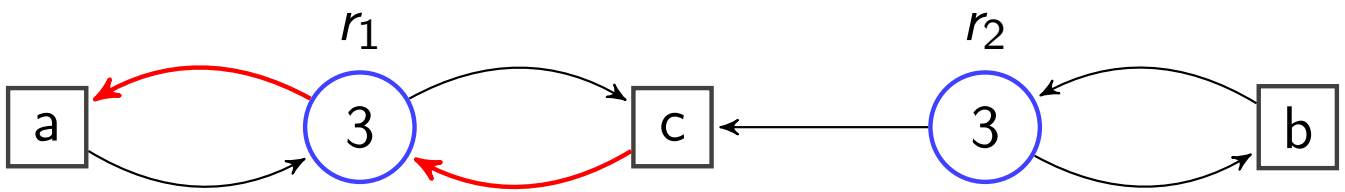
$0 < i \leq 3$

$$C = \mathcal{E}^2_{fat} \cap \Sigma^2$$



Simulations

fat $\rightarrow$ fact: Example



δ_a

r_1	r_1
i	$i-1$

δ_c

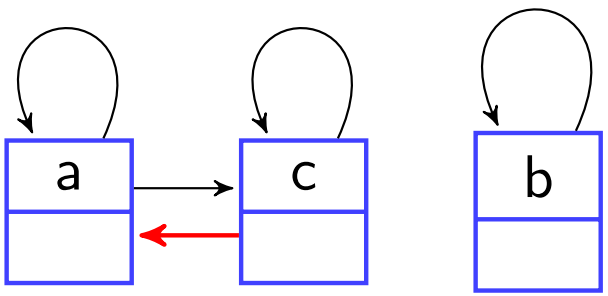
r_1	r_2	r_1
0	0	s

δ_b

r_2	r_2
i	$i-1$

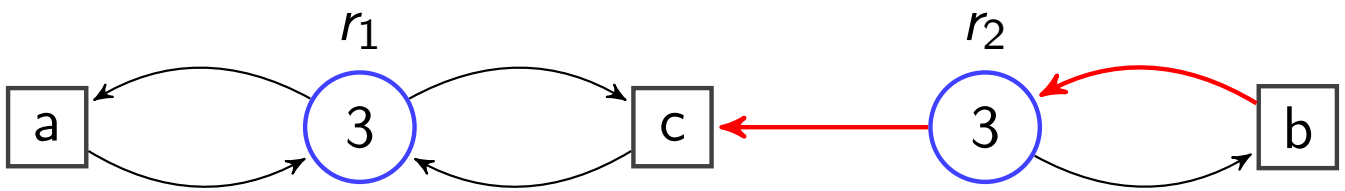
$0 < i \leq 3$

$$C = \mathcal{E}^2_{fat} \cap \Sigma^2$$



Simulations

fat $\rightarrow$ fact: Example



δ_a

r_1	r_1
i	$i-1$

δ_c

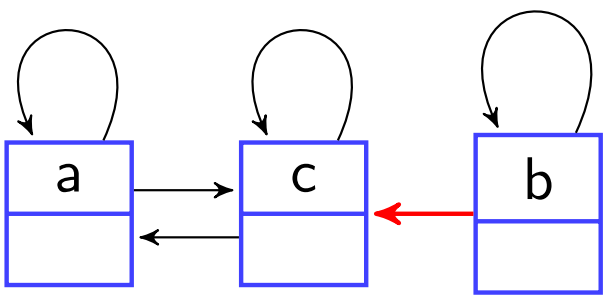
r_1	r_2	r_1
0	0	s

δ_b

r_2	r_2
i	$i-1$

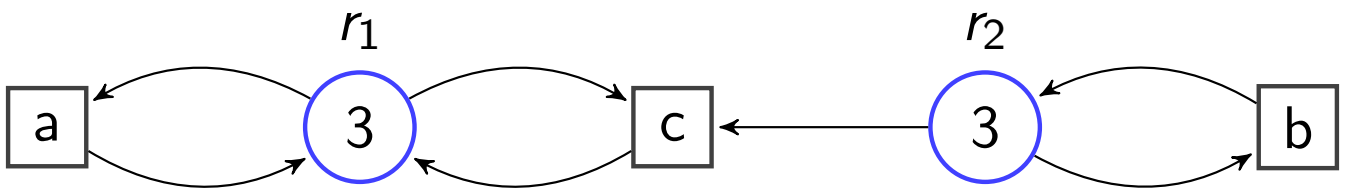
$0 < i \leq 3$

$$C = \mathcal{E}^2_{fat} \cap \Sigma^2$$



Simulations

fat $\rightarrow$ fact: Example



δ_a

r_1	r_1
i	$i-1$

δ_c

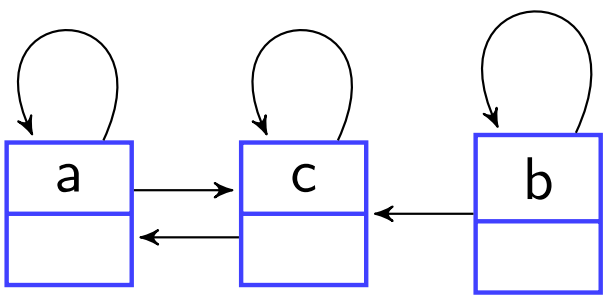
r_1	r_2	r_1
0	0	s

δ_b

r_2	r_2
i	$i-1$

$0 < i \leq 3$

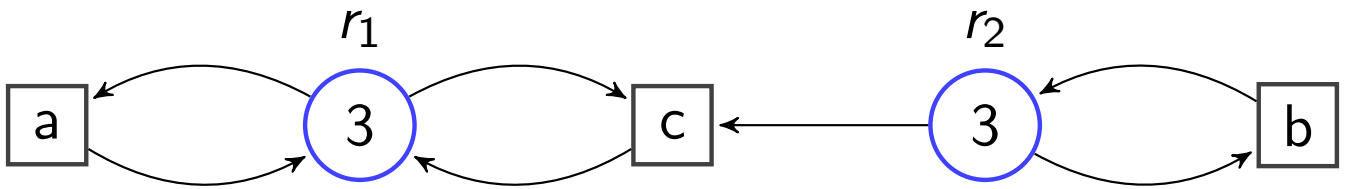
$$X = \bigcup_{a \in \Sigma} X_a$$



$X = \bigcup_{a \in \Sigma} X_a$

Simulations

fat $\rightarrow$ fact: Example



δ_a

r_1	r_1
i	$i-1$

δ_c

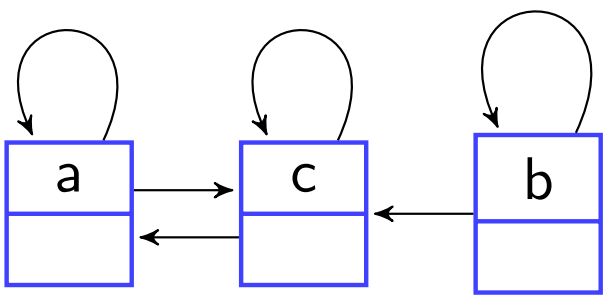
r_1	r_2	r_1
0	0	s

δ_b

r_2	r_2
i	$i-1$

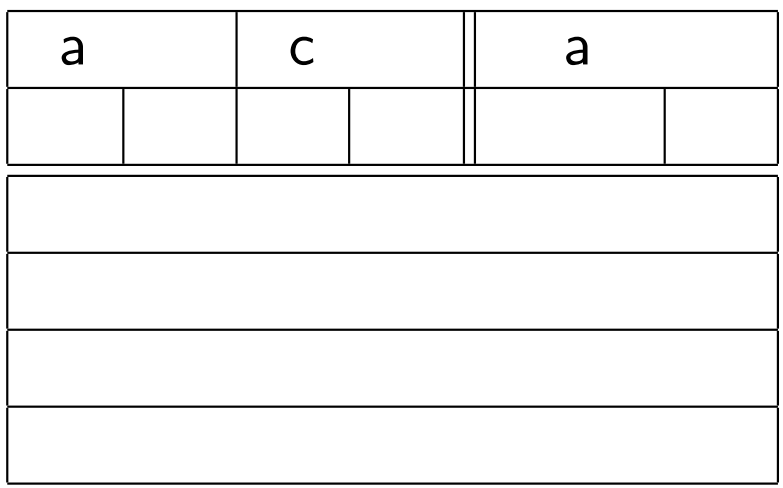
$0 < i \leq 3$

$$\delta_a \subseteq \prod_{\alpha \in Ca} X_\alpha \times X_a$$



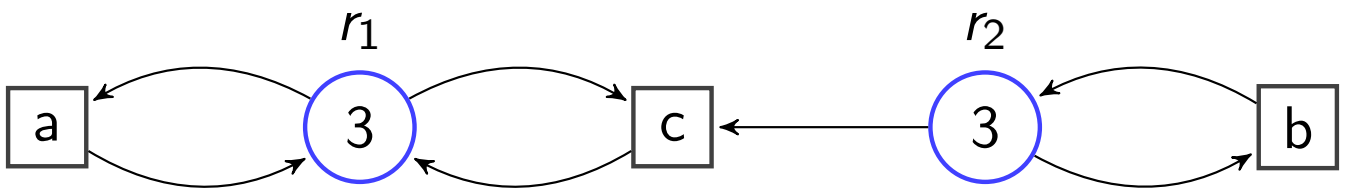
$$X = \bigcup_{a \in \Sigma} X_a$$

δ_a:



Simulations

fat $\rightarrow$ fact: Example



δ_a

r_1	r_1
i	$i-1$

δ_c

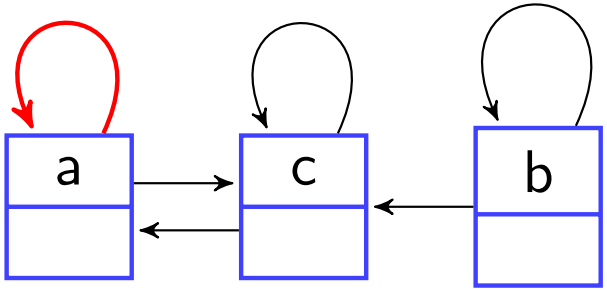
r_1	r_2	r_1
0	0	s

δ_b

r_2	r_2
i	$i-1$

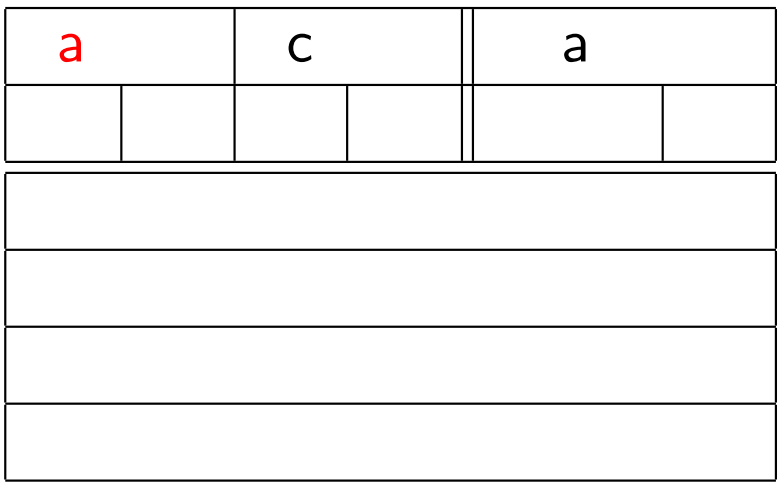
$0 < i \leq 3$

$$\delta_a \subseteq \prod_{\alpha \in Ca} X_\alpha \times X_a$$



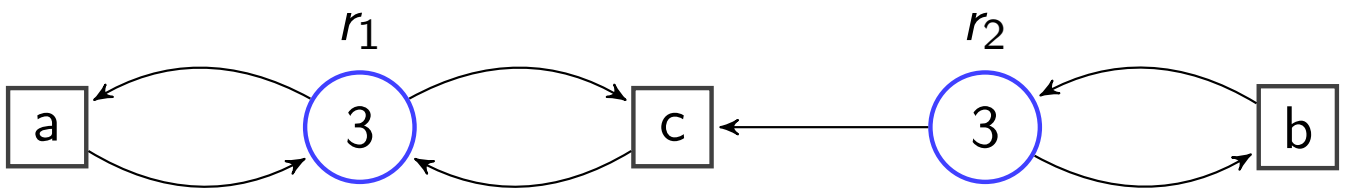
$$X = \bigcup_{a \in \Sigma} X_a$$

δ_a:



Simulations

fat $\rightarrow$ fact: Example



δ_a

r_1	r_1
i	$i-1$

δ_c

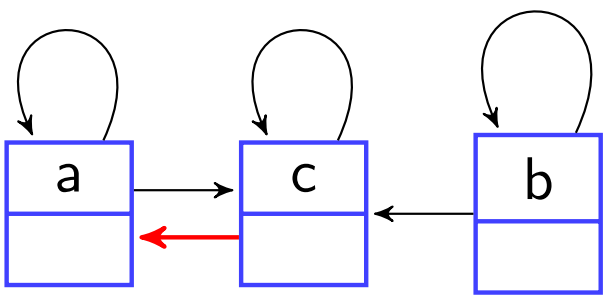
r_1	r_2	r_1
0	0	s

δ_b

r_2	r_2
i	$i-1$

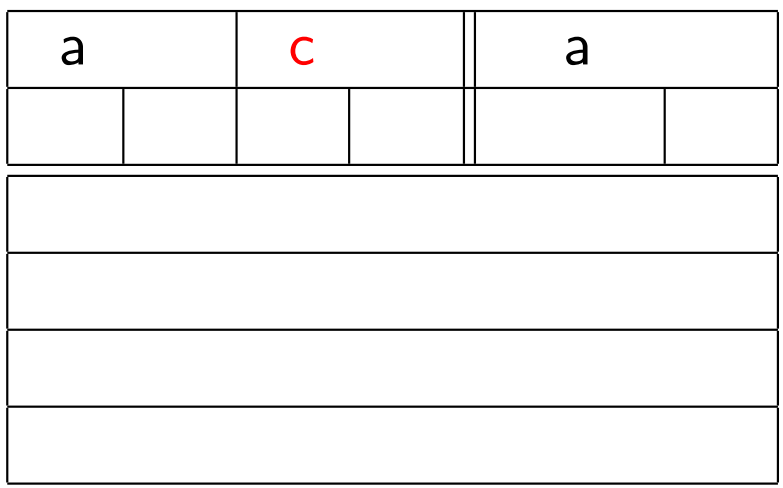
$0 < i \leq 3$

$$\delta_a \subseteq \prod_{\alpha \in Ca} X_\alpha \times X_a$$



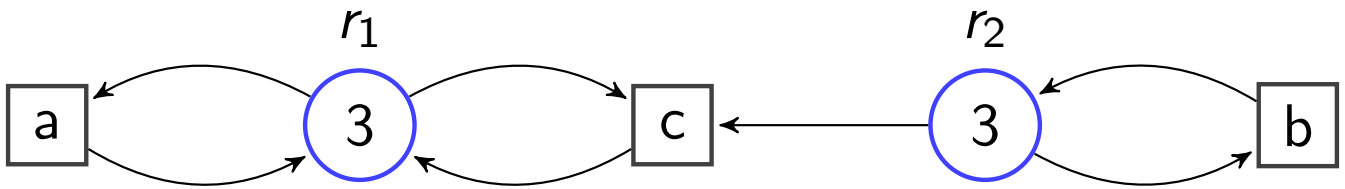
$$X = \bigcup_{a \in \Sigma} X_a$$

δ_a:



Simulations

fat $\rightarrow$ fact: Example



δ_a

r_1	r_1
i	$i-1$

δ_c

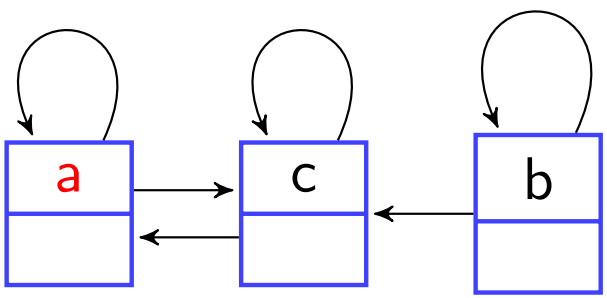
r_1	r_2	r_1
0	0	s

δ_b

r_2	r_2
i	$i-1$

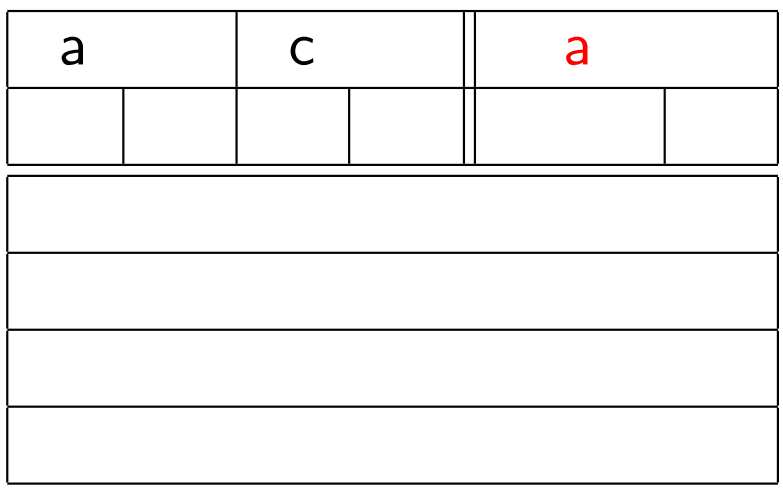
$0 < i \leq 3$

$$\delta_a \subseteq \prod_{\alpha \in Ca} X_\alpha \times X_a$$



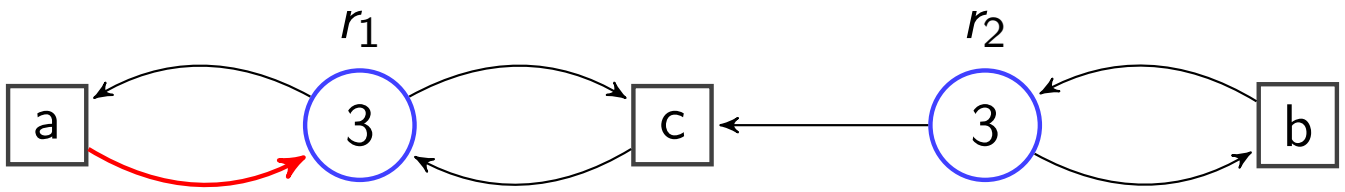
$$X = \bigcup_{a \in \Sigma} X_a$$

δ_a:



Simulations

fat $\rightarrow$ fact: Example



δ_a

r_1	r_1
i	$i-1$

δ_c

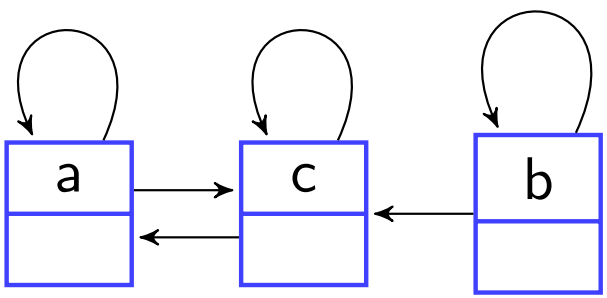
r_1	r_2	r_1
0	0	s

δ_b

r_2	r_2
i	$i-1$

$0 < i \leq 3$

$$\delta_a \subseteq \prod_{\alpha \in Ca} X_\alpha \times X_a$$



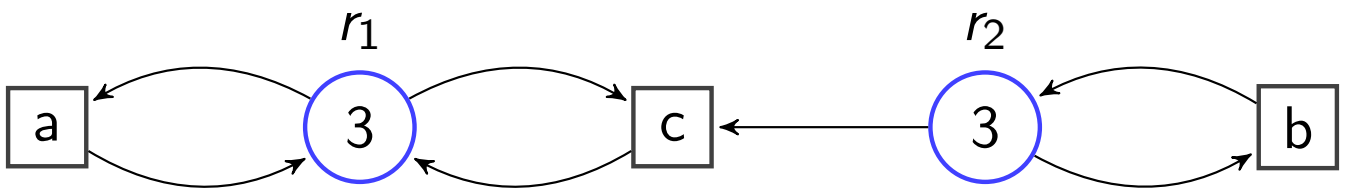
$$X = \bigcup_{a \in \Sigma} X_a$$

δ_a:

a		c			a	
r_1					r_1	

Simulations

fat $\rightarrow$ fact: Example



δ_a

r_1	r_1
i	$i-1$

δ_c

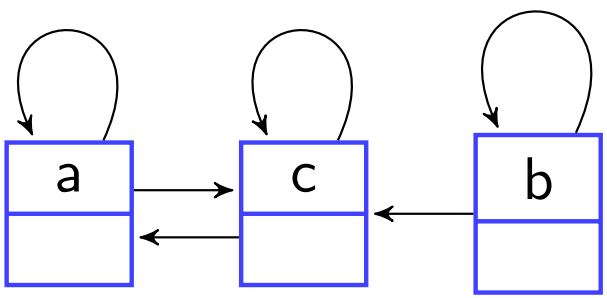
r_1	r_2	r_1
0	0	s

δ_b

r_2	r_2
i	$i-1$

$0 < i \leq 3$

$$\delta_a \subseteq \prod_{\alpha \in Ca} X_\alpha \times X_a$$



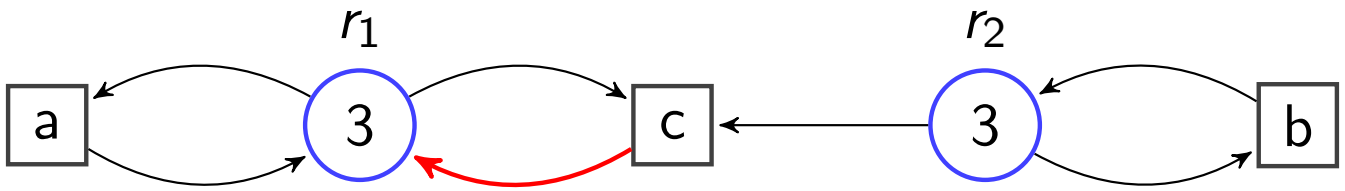
$$X = \bigcup_{a \in \Sigma} X_a$$

δ_a:

a		c		a	
r_1	t_a			r_1	t_a

Simulations

fat $\rightarrow$ fact: Example



δ_a

r_1	r_1
i	$i-1$

δ_c

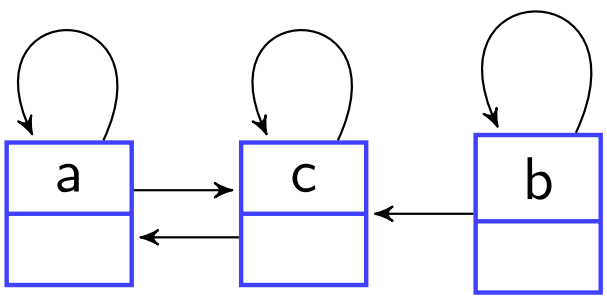
r_1	r_2	r_1
0	0	s

δ_b

r_2	r_2
i	$i-1$

$0 < i \leq 3$

$$\delta_a \subseteq \prod_{\alpha \in Ca} X_\alpha \times X_a$$



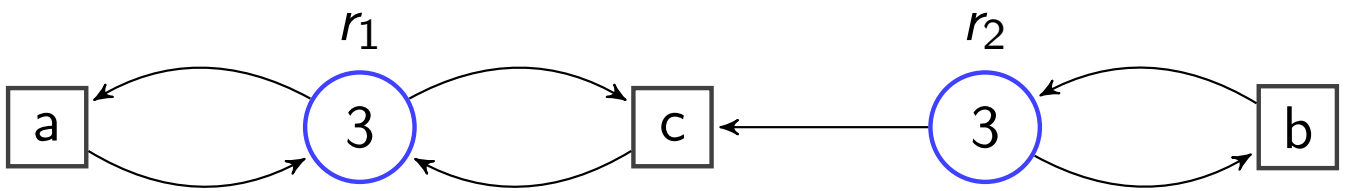
$$X = \bigcup_{a \in \Sigma} X_a$$

δ_a:

a		c		a	
r_1	t_a	r_1		r_1	t_a

Simulations

fat $\rightarrow$ fact: Example



δ_a

r_1	r_1
i	$i-1$

δ_c

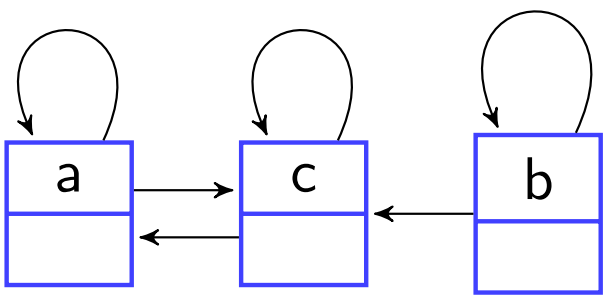
r_1	r_2	r_1
0	0	s

δ_b

r_2	r_2
i	$i-1$

$0 < i \leq 3$

$$\delta_a \subseteq \prod_{\alpha \in Ca} X_\alpha \times X_a$$



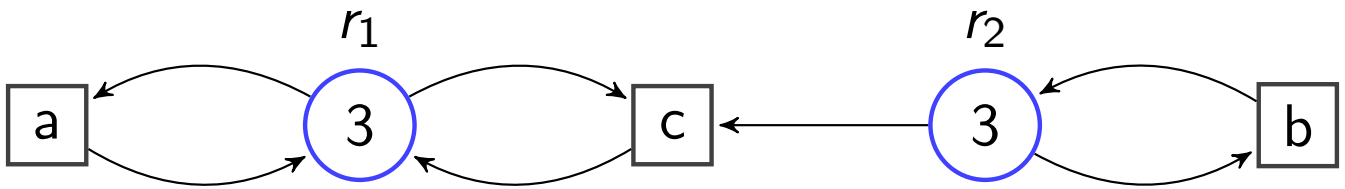
$$X = \bigcup_{a \in \Sigma} X_a$$

δ_a:

a		c		a	
r_1	t_a	r_1	t_c	r_1	t_a

Simulations

fat $\rightarrow$ fact: Example



δ_a

r_1	r_1
i	$i-1$

δ_c

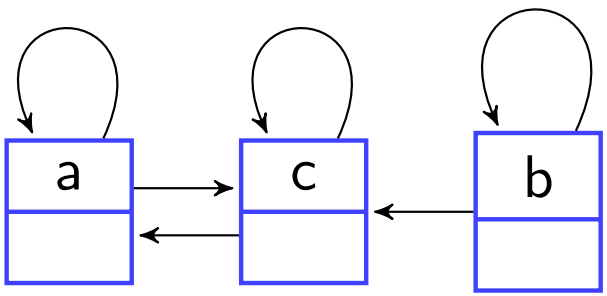
r_1	r_2	r_1
0	0	s

δ_b

r_2	r_2
i	$i-1$

$0 < i \leq 3$

$$\delta_a \subseteq \prod_{\alpha \in Ca} X_\alpha \times X_a$$



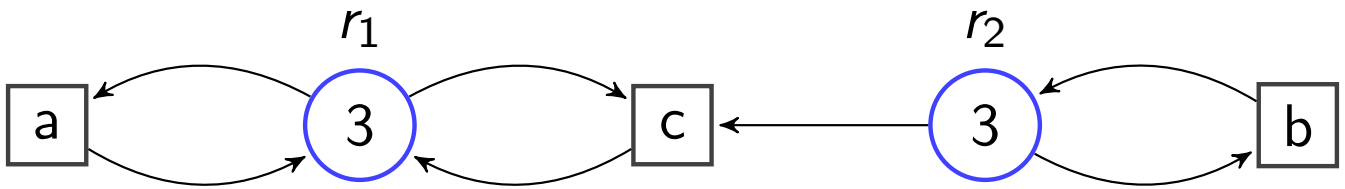
$$X = \bigcup_{a \in \Sigma} X_a$$

δ_a:

a (0)		c		a (0)	
r_1	t_a	r_1	t_c	r_1	t_a

Simulations

fat $\rightarrow$ fact: Example



δ_a

r_1	r_1
i	$i-1$

δ_c

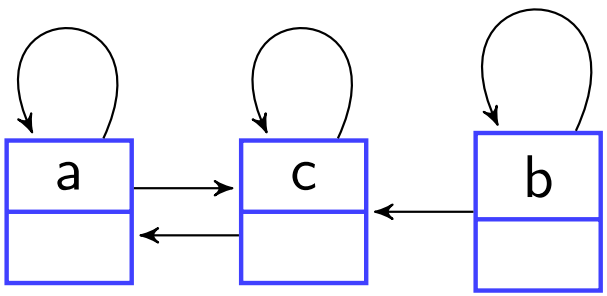
r_1	r_2	r_1
0	0	s

δ_b

r_2	r_2
i	$i-1$

$0 < i \leq 3$

$$\delta_a \subseteq \prod_{\alpha \in Ca} X_\alpha \times X_a$$



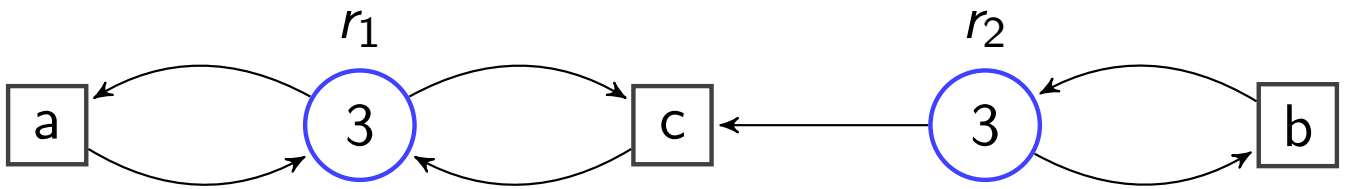
$$X = \bigcup_{a \in \Sigma} X_a$$

δ_a:

a (0)		c (1)		a (0)	
r_1	t_a	r_1	t_c	r_1	t_a

Simulations

fat $\rightarrow$ fact: Example



δ_a

r_1	r_1
i	$i-1$

δ_c

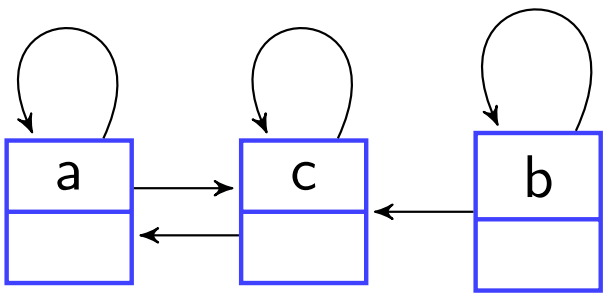
r_1	r_2	r_1
0	0	s

δ_b

r_2	r_2
i	$i-1$

$0 < i \leq 3$

$$\delta_a \subseteq \prod_{\alpha \in Ca} X_\alpha \times X_a$$



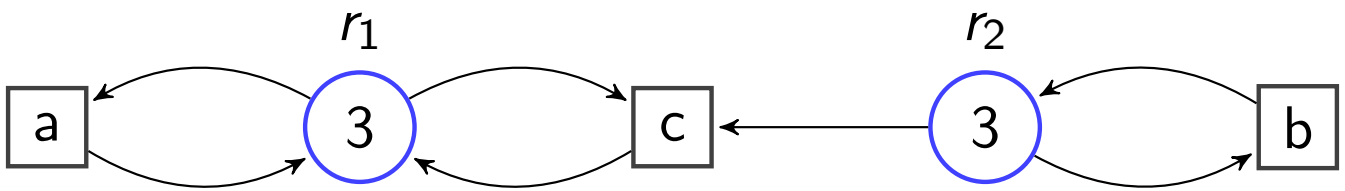
$$X = \bigcup_{a \in \Sigma} X_a$$

δ_a:

a (0)		c (1)		a (0)	
r_1	t_a	r_1	t_c	r_1	t_a
i	0	*	0	$i-1$	0

Simulations

fat $\rightarrow$ fact: Example



δ_a

r_1	r_1
i	$i-1$

δ_c

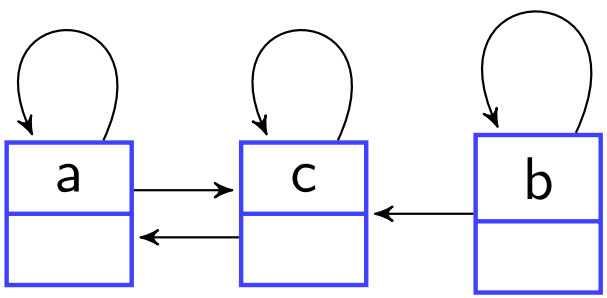
r_1	r_2	r_1
0	0	s

δ_b

r_2	r_2
i	$i-1$

$0 < i \leq 3$

$$\delta_a \subseteq \prod_{\alpha \in Ca} X_\alpha \times X_a$$



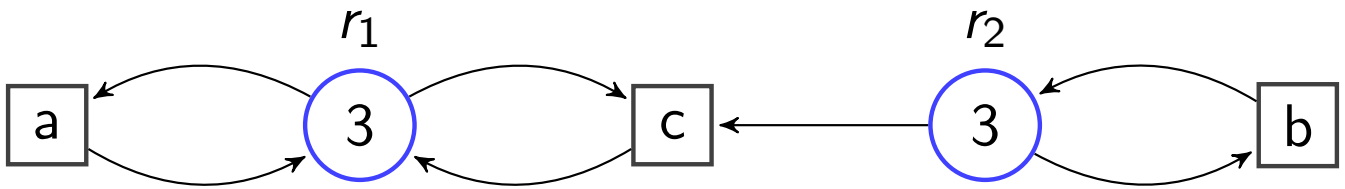
$$X = \bigcup_{a \in \Sigma} X_a$$

δ_a:

a (0)		c (1)		a (0)	
r_1	t_a	r_1	t_c	r_1	t_a
i	0	*	0	$i-1$	0

Simulations

fat $\rightarrow$ fact: Example



δ_a

r_1	r_1
i	$i-1$

δ_c

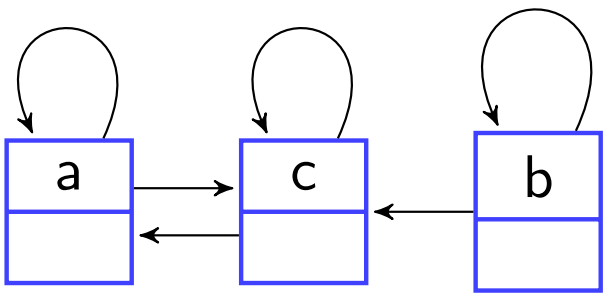
r_1	r_2	r_1
0	0	s

δ_b

r_2	r_2
i	$i-1$

$0 < i \leq 3$

$$\delta_a \subseteq \prod_{\alpha \in Ca} X_\alpha \times X_a$$



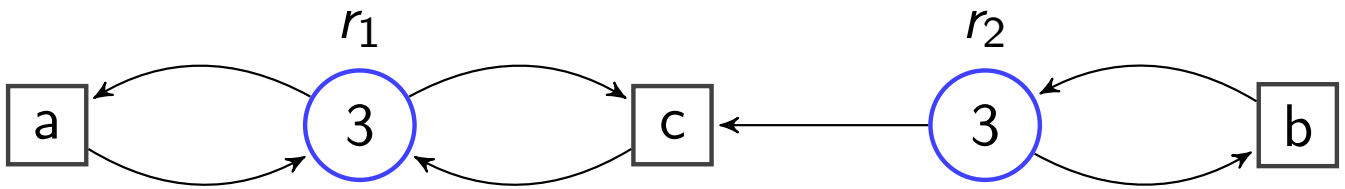
$$X = \bigcup_{a \in \Sigma} X_a$$

δ_a:

a (0)		c (1)		a (0)	
r_1	t_a	r_1	t_c	r_1	t_a
i	0	*	0	$i-1$	0
i	1	*	1	$i-1$	1

Simulations

fat $\rightarrow$ fact: Example



δ_a

r_1	r_1
i	$i-1$

δ_c

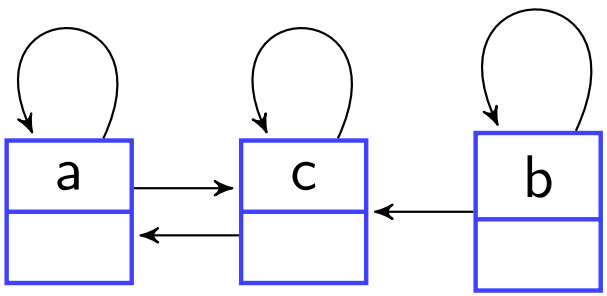
r_1	r_2	r_1
0	0	s

δ_b

r_2	r_2
i	$i-1$

$0 < i \leq 3$

$$\delta_a \subseteq \prod_{\alpha \in Ca} X_\alpha \times X_a$$



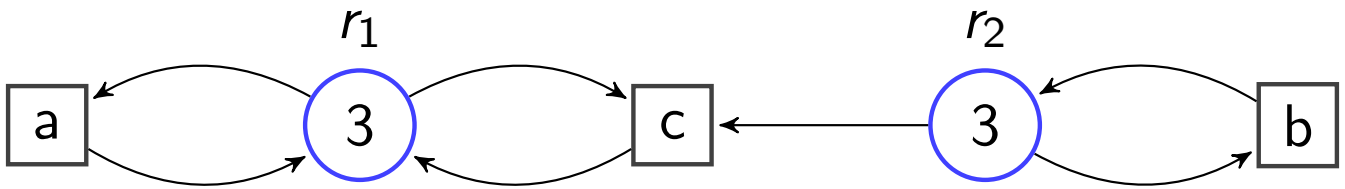
$$X = \bigcup_{a \in \Sigma} X_a$$

δ_a:

a (0)		c (1)		a (0)	
r_1	t_a	r_1	t_c	r_1	t_a
i	0	*	0	$i-1$	0
i	1	*	1	$i-1$	1

Simulations

fat $\rightarrow$ fact: Example



δ_a

r_1	r_1
i	$i-1$

δ_c

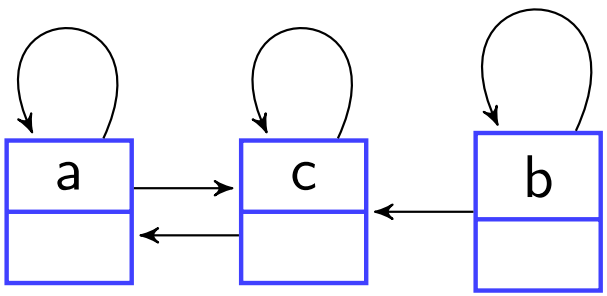
r_1	r_2	r_1
0	0	s

δ_b

r_2	r_2
i	$i-1$

$0 < i \leq 3$

$$\delta_a \subseteq \prod_{\alpha \in Ca} X_\alpha \times X_a$$



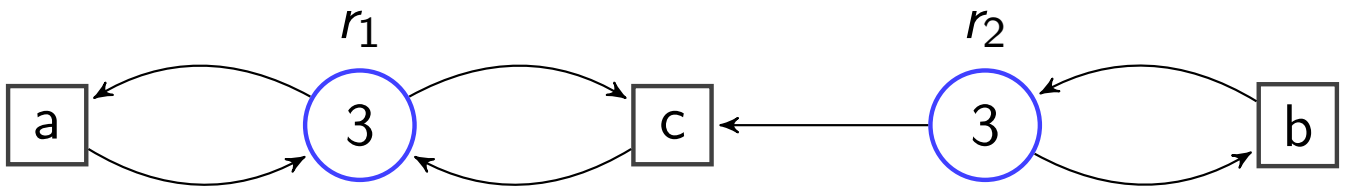
$$X = \bigcup_{a \in \Sigma} X_a$$

δ_a:

a (0)		c (1)		a (0)	
r_1	t_a	r_1	t_c	r_1	t_a
i	0	*	0	$i-1$	0
i	1	*	1	$i-1$	1
*	0	i	1	$i-1$	1

Simulations

fat $\rightarrow$ fact: Example



δ_a

r_1	r_1
i	$i-1$

δ_c

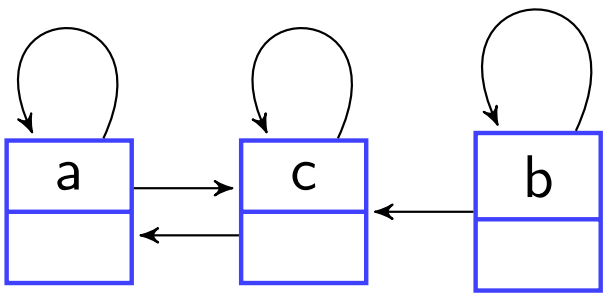
r_1	r_2	r_1
0	0	s

δ_b

r_2	r_2
i	$i-1$

$0 < i \leq 3$

$$\delta_a \subseteq \prod_{\alpha \in Ca} X_\alpha \times X_a$$



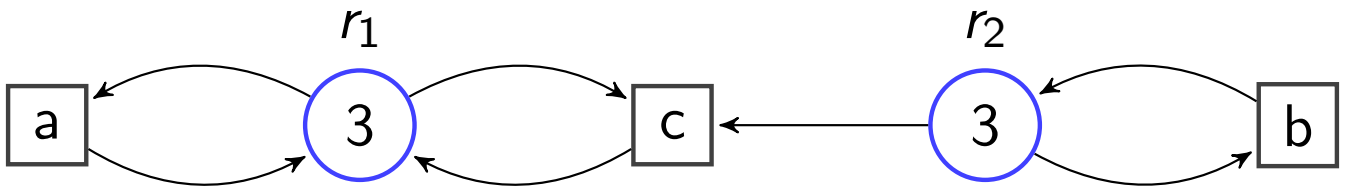
$$X = \bigcup_{a \in \Sigma} X_a$$

δ_a:

a (0)		c (1)		a (0)	
r_1	t_a	r_1	t_c	r_1	t_a
i	0	*	0	$i-1$	0
i	1	*	1	$i-1$	1
*	0	i	1	$i-1$	1

Simulations

fat $\rightarrow$ fact: Example



δ_a

r_1	r_1
i	$i-1$

δ_c

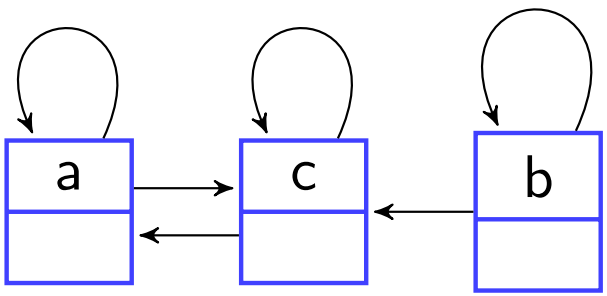
r_1	r_2	r_1
0	0	s

δ_b

r_2	r_2
i	$i-1$

$0 < i \leq 3$

$$\delta_a \subseteq \prod_{\alpha \in Ca} X_\alpha \times X_a$$



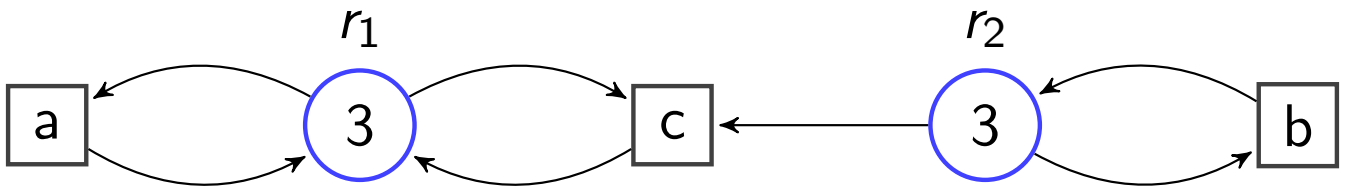
$$X = \bigcup_{a \in \Sigma} X_a$$

δ_a:

a (0)		c (1)		a (0)	
r_1	t_a	r_1	t_c	r_1	t_a
i	0	*	0	$i-1$	0
i	1	*	1	$i-1$	1
*	0	i	1	$i-1$	1
*	1	i	0	$i-1$	0

Simulations

fat $\rightarrow$ fact: Example



δ_a

r_1	r_1
i	$i-1$

δ_c

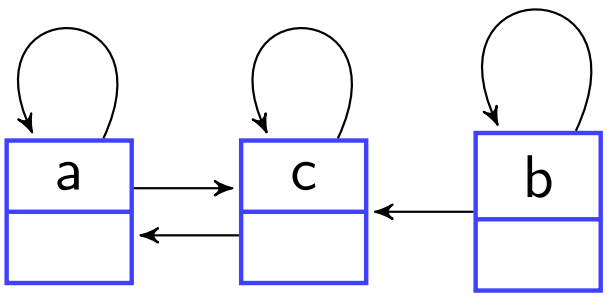
r_1	r_2	r_1
0	0	s

δ_b

r_2	r_2
i	$i-1$

$0 < i \leq 3$

$$\delta_a \subseteq \prod_{\alpha \in Ca} X_\alpha \times X_a$$



$$X = \bigcup_{a \in \Sigma} X_a$$

δ_a:

a (0)		c (1)		a (0)	
r_1	t_a	r_1	t_c	r_1	t_a
i	0	*	0	$i-1$	0
i	1	*	1	$i-1$	1
*	0	i	1	$i-1$	1
*	1	i	0	$i-1$	0

Simulations

fact $\rightarrow$ fat

Does the other direction work too?

Simulations

fact $\rightarrow$ fat

Does the other direction work too?

Let $\tau = (\Sigma, C, X, \Delta)$ be a fact-system and σ an ω-redundant signature that induces C.

Simulations

fact $\rightarrow$ fat

Does the other direction work too?

Let $\tau = (\Sigma, C, X, \Delta)$ be a fact-system and σ an ω-redundant signature that induces C.

Then there exists a fat-system $\tau' = (\Sigma, R, \mathcal{E}, X', \Delta')$ over σ covering τ.

Content

Languages

Finite Asynchronous Automaton (FAA):

$$\mathcal{A} = (\underbrace{\Sigma, R, \mathcal{E}}_{\sigma}, X, \Delta, I, F)$$

Languages

Finite Asynchronous Automaton (FAA):

$$\mathcal{A} = (\underbrace{\Sigma, R, \mathcal{E}}_{\sigma}, X, \Delta, I, F)$$

$$L(\mathcal{A}) = \{w \in \Sigma^* \mid \exists s_0 \in I, s_f \in F : s_f \in \Delta(s_0, w)\}$$

Languages

Finite Asynchronous Automaton (FAA):

$$\mathcal{A} = (\underbrace{\Sigma, R, \mathcal{E}}_{\sigma}, X, \Delta, I, F)$$

$$L(\mathcal{A}) = \{w \in \Sigma^* \mid \exists s_0 \in I, s_f \in F : s_f \in \Delta(s_0, w)\}$$

$\mathcal{L}_\sigma^d :=$ class of languages recognized by deterministic faa over the signature σ

Languages

Finite Asynchronous Automaton (FAA):

$$\mathcal{A} = (\underbrace{\Sigma, R, \mathcal{E}}_{\sigma}, X, \Delta, I, F)$$

$$L(\mathcal{A}) = \{w \in \Sigma^* \mid \exists s_0 \in I, s_f \in F : s_f \in \Delta(s_0, w)\}$$

$\mathcal{L}_\sigma^d :=$ class of languages recognized by deterministic faa over the signature σ

$\mathcal{L}_\sigma^n :=$ class of languages recognized by non-deterministic faa over the signature σ

Languages

Finite Asynchronous Automaton (FAA):

$$\mathcal{A} = (\underbrace{\Sigma, R, \mathcal{E}}_{\sigma}, X, \Delta, I, F)$$

$$L(\mathcal{A}) = \{w \in \Sigma^* \mid \exists s_0 \in I, s_f \in F : s_f \in \Delta(s_0, w)\}$$

$\mathcal{L}_\sigma^d :=$ class of languages recognized by deterministic faa over the signature σ

$\mathcal{L}_\sigma^n :=$ class of languages recognized by non-deterministic faa over the signature σ

Defined analogously for FACA

Languages

Let

- $\sigma_1 = (\Sigma, R, \mathcal{E})$ be an ω-redundant signature of a fat-system

Languages

Let

- $\sigma_1 = (\Sigma, R, \mathcal{E})$ be an ω-redundant signature of a fat-system
- $\sigma_2 = (\Sigma, C)$ be the induced signature of a fact-system $(C = \mathcal{E}^2 \cap \Sigma^2)$

Languages

Let

- $\sigma_1 = (\Sigma, R, \mathcal{E})$ be an ω-redundant signature of a fat-system
- $\sigma_2 = (\Sigma, C)$ be the induced signature of a fact-system $(C = \mathcal{E}^2 \cap \Sigma^2)$

Then

$$\mathcal{L}^d_{\sigma_1} = \mathcal{L}^d_{\sigma_2} \qquad \text{and} \qquad \mathcal{L}^n_{\sigma_1} = \mathcal{L}^n_{\sigma_2}$$

Languages

Let

- $\sigma_1 = (\Sigma, R, \mathcal{E})$ be an ω-redundant signature of a fat-system
- $\sigma_2 = (\Sigma, C)$ be the induced signature of a fact-system ($C = \mathcal{E}^2 \cap \Sigma^2$)

Then

$$\mathcal{L}^d_{\sigma_1} = \mathcal{L}^d_{\sigma_2} \qquad \text{and} \qquad \mathcal{L}^n_{\sigma_1} = \mathcal{L}^n_{\sigma_2}$$

Proof.

Simulation fat$\leftrightarrow$fact plus a little magic regarding initial/final states and languages. $\square$

Languages

Let

- $\sigma_1 = (\Sigma, R, \mathcal{E})$ be an ω-redundant signature of a fat-system
- $\sigma_2 = (\Sigma, C)$ be the induced signature of a fact-system ($C = \mathcal{E}^2 \cap \Sigma^2$)

Then

$$\mathcal{L}^d_{\sigma_1} = \mathcal{L}^d_{\sigma_2} \qquad \text{and} \qquad \mathcal{L}^n_{\sigma_1} = \mathcal{L}^n_{\sigma_2}$$

Proof.

Simulation fat$\leftrightarrow$fact plus a little magic regarding initial/final states and languages. □

In the sequel we restrain our attention to languages recognized by faca.

Languages

For each faca $\mathcal{A} = (\Sigma, C, X, \Delta, I, F)$ there exists a faca $\mathcal{A}'$ over the same signature $\sigma = (\Sigma, C)$ with only one initial state s.t. $L(\mathcal{A}) = L(\mathcal{A}')$.

Languages

For each faca $\mathcal{A} = (\Sigma, C, X, \Delta, I, F)$ there exists a faca $\mathcal{A}'$ over the same signature $\sigma = (\Sigma, C)$ with only one initial state s.t. $L(\mathcal{A}) = L(\mathcal{A}')$.

Proof.

boring

□

Languages

For each faca $\mathcal{A} = (\Sigma, C, X, \Delta, I, F)$ there exists a faca $\mathcal{A}'$ over the same signature $\sigma = (\Sigma, C)$ with only one initial state s.t. $L(\mathcal{A}) = L(\mathcal{A}')$.

Proof.

boring □

In the sequel wlog we assume that there is only one initial state.

Languages

Let $\sigma_1 = (\Sigma_1, C_1)$ and $\sigma_2 = (\Sigma_2, C_2)$ be two signatures, such that

Languages

Let $\sigma_1 = (\Sigma_1, C_1)$ and $\sigma_2 = (\Sigma_2, C_2)$ be two signatures, such that

$$\Sigma_1 \subseteq \Sigma_2 \qquad \text{and} \qquad C_1 = C_2 \cap \Sigma_1^2$$

Languages

Let $\sigma_1 = (\Sigma_1, C_1)$ and $\sigma_2 = (\Sigma_2, C_2)$ be two signatures, such that

$$\Sigma_1 \subseteq \Sigma_2 \qquad \text{and} \qquad C_1 = C_2 \cap \Sigma_1^2$$

Then for each language $L \subseteq \Sigma_1^*$:

Languages

Let $\sigma_1 = (\Sigma_1, C_1)$ and $\sigma_2 = (\Sigma_2, C_2)$ be two signatures, such that

$$\Sigma_1 \subseteq \Sigma_2 \qquad \text{and} \qquad C_1 = C_2 \cap \Sigma_1^2$$

Then for each language $L \subseteq \Sigma_1^*$: $\qquad L \in \mathcal{L}^n_{\sigma_1} \quad \Leftrightarrow \quad L \in \mathcal{L}^n_{\sigma_2}$

Languages

Let $\sigma_1 = (\Sigma_1, C_1)$ and $\sigma_2 = (\Sigma_2, C_2)$ be two signatures, such that

$$\Sigma_1 \subseteq \Sigma_2 \qquad \text{and} \qquad C_1 = C_2 \cap \Sigma_1^2$$

Then for each language $L \subseteq \Sigma_1^*$: $\qquad L \in \mathcal{L}^d_{\sigma_1} \quad \Leftrightarrow \quad L \in \mathcal{L}^d_{\sigma_2}$

Languages

Let $\sigma_1 = (\Sigma_1, C_1)$ and $\sigma_2 = (\Sigma_2, C_2)$ be two signatures, such that

$$\Sigma_1 \subseteq \Sigma_2 \qquad \text{and} \qquad C_1 = C_2 \cap \Sigma_1^2$$

Then for each language $L \subseteq \Sigma_1^*$: $\qquad L \in \mathcal{L}^u_{\sigma_1} \quad \Leftrightarrow \quad L \in \mathcal{L}^u_{\sigma_2}$

proof idea.

$\Rightarrow$: Let $\mathcal{A}_1$ be a faca over σ_1. Construct $\mathcal{A}_2$ over σ_2 such that it works like $\mathcal{A}_1$ for Actions in Σ_1. For Actions in $\Sigma_2 \setminus \Sigma_1$ it goes into a state such that a final state cannot be reached anymore.

Languages

Let $\sigma_1 = (\Sigma_1, C_1)$ and $\sigma_2 = (\Sigma_2, C_2)$ be two signatures, such that

$$\Sigma_1 \subseteq \Sigma_2 \qquad \text{and} \qquad C_1 = C_2 \cap \Sigma_1^2$$

Then for each language $L \subseteq \Sigma_1^*$: $\quad L \in \mathcal{L}^u_{\sigma_1} \quad \Leftrightarrow \quad L \in \mathcal{L}^u_{\sigma_2}$

proof idea.

$\Rightarrow$: Let $\mathcal{A}_1$ be a faca over σ_1. Construct $\mathcal{A}_2$ over σ_2 such that it works like $\mathcal{A}_1$ for Actions in Σ_1. For Actions in $\Sigma_2 \setminus \Sigma_1$ it goes into a state such that a final state cannot be reached anymore.

$\Leftarrow$: Let $\mathcal{A}_2$ be a faca over σ_2 with initial state s_0^2, final states F_2. Construct $\mathcal{A}_1$ over σ_1 with the same values X that works like $\mathcal{A}_2$ for Actions in Σ_1. Let s be a final state, then $s_{|\Sigma_2 \setminus \Sigma_1} = s^2_{0|\Sigma_2 \setminus \Sigma_1}$. Pick $F_1 = \{s_{|\Sigma_1} \mid s \in F_2\}$. □

Languages

Lemma: Let $C_1 \subseteq C_2 \subseteq \Sigma^2$ and $\sigma_i = (\Sigma, C_i), i = 1, 2$. Then

Languages

Lemma: Let $C_1 \subseteq C_2 \subseteq \Sigma^2$ and $\sigma_i = (\Sigma, C_i), i = 1, 2$. Then

$$\mathcal{L}^d_{\sigma_1} \subseteq \mathcal{L}^d_{\sigma_2} \quad \text{and} \quad \mathcal{L}^n_{\sigma_1} \subseteq \mathcal{L}^n_{\sigma_2}$$

Languages

Lemma: Let $C_1 \subseteq C_2 \subseteq \Sigma^2$ and $\sigma_i = (\Sigma, C_i), i = 1, 2$. Then

$$\mathcal{L}^d_{\sigma_1} \subseteq \mathcal{L}^d_{\sigma_2} \quad \text{and} \quad \mathcal{L}^n_{\sigma_1} \subseteq \mathcal{L}^n_{\sigma_2}$$

Proof idea.

Given $\mathcal{A}_1$ faca over σ_1, construct $\mathcal{A}_2$ over σ_2 s.t. $L(\mathcal{A}_1) = L(\mathcal{A}_2)$:

Languages

Lemma: Let $C_1 \subseteq C_2 \subseteq \Sigma^2$ and $\sigma_i = (\Sigma, C_i), i = 1, 2$. Then

$$\mathcal{L}^d_{\sigma_1} \subseteq \mathcal{L}^d_{\sigma_2} \quad \text{and} \quad \mathcal{L}^n_{\sigma_1} \subseteq \mathcal{L}^n_{\sigma_2}$$

Proof idea.

Given $\mathcal{A}_1$ faca over σ_1, construct $\mathcal{A}_2$ over σ_2 s.t. $L(\mathcal{A}_1) = L(\mathcal{A}_2)$:

Let $a \in \Sigma, s \in X^{C_2 a}, x \in X$ then

$$(s, x) \in \delta^2_a \in \Delta_2 \qquad \textit{iff} \qquad (s_{|C_1 a}, x) \in \delta^1_a \in \Delta_1$$

□

Languages

Theorem (Hierarchy Theorem)

$C_1, C_2 \subseteq \Sigma^2$ *and* $\sigma_i = (\Sigma, C_i), i = 1, 2$. *Then*

Languages

Theorem (Hierarchy Theorem)

$C_1, C_2 \subseteq \Sigma^2$ and $\sigma_i = (\Sigma, C_i), i = 1, 2$. Then

$$\mathcal{L}^d_{\sigma_1} \subseteq \mathcal{L}^d_{\sigma_2} \quad \textit{iff} \quad C_1 \subseteq C_2$$

Languages

Theorem (Hierarchy Theorem)

$C_1, C_2 \subseteq \Sigma^2$ and $\sigma_i = (\Sigma, C_i), i = 1, 2$. Then

$$\mathcal{L}^d_{\sigma_1} \subseteq \mathcal{L}^d_{\sigma_2} \quad \textit{iff} \quad C_1 \subseteq C_2$$

Proof.

$\Leftarrow$: given by the Lemma.

Languages

Theorem (Hierarchy Theorem)

$C_1, C_2 \subseteq \Sigma^2$ and $\sigma_i = (\Sigma, C_i), i = 1, 2$. Then

$$\mathcal{L}^d_{\sigma_1} \subseteq \mathcal{L}^d_{\sigma_2} \quad \textit{iff} \quad C_1 \subseteq C_2$$

Proof.

$\Leftarrow$: given by the Lemma.

$\Rightarrow$: blurp.

Determinization

Theorem (Zielonka's Theorem)

Let $\sigma = (\Sigma, C)$. If C is symmetric and reflexive then

Determinization

Theorem (Zielonka's Theorem)

Let $\sigma = (\Sigma, C)$. If C is symmetric and reflexive then

$$\mathcal{L}_\sigma^d = \mathcal{L}_\sigma^n$$

Complexity

- indirect proof construction

Complexity

- indirect proof construction
- triple-exponential blow-up in size

Complexity

- indirect proof construction
- triple-exponential blow-up in size
- direct determinization shown in [KMS94]

Complexity

- indirect proof construction
- triple-exponential blow-up in size
- direct determinization shown in [KMS94]
- "only" double-exponential

Complexity

- indirect proof construction
- triple-exponential blow-up in size
- direct determinization shown in [KMS94]
- "only" double-exponential $\leftarrow$ essentially optimal

Sources


V. Diekert and G. Rozenberg.
The Book of Traces.
World Scientific, 1995.


Nils Klarlund, Madhavan Mukund, and Milind A. Sohoni.
Determinizing asynchronous automata.
In *Proceedings of the 21st International Colloquium on Automata, Languages and Programming*, ICALP '94, pages 130–141, London, UK, UK, 1994. Springer-Verlag.


Antoni Mazurkiewicz.
Concurrent program schemes and their interpretations.
Technical report, DAIMI Aarhus University, 1977.
Report PB 78.


Wieslaw Zielonka.
Notes on finite asynchronous automata.
ITA, 21(2):99–135, 1987.