Surminimisation of Automata

Victor MARSAULT

Telecom-ParisTech, Paris, France

DLT, Liverpool
2015-07-30

A few starting remarks

- Alphabets are provided with a total order.

In examples: $a < b < c$, $x < y < z$ and $0 < 1 < 2 < 3 < \cdots$

A few starting remarks

- Alphabets are provided with a total order.

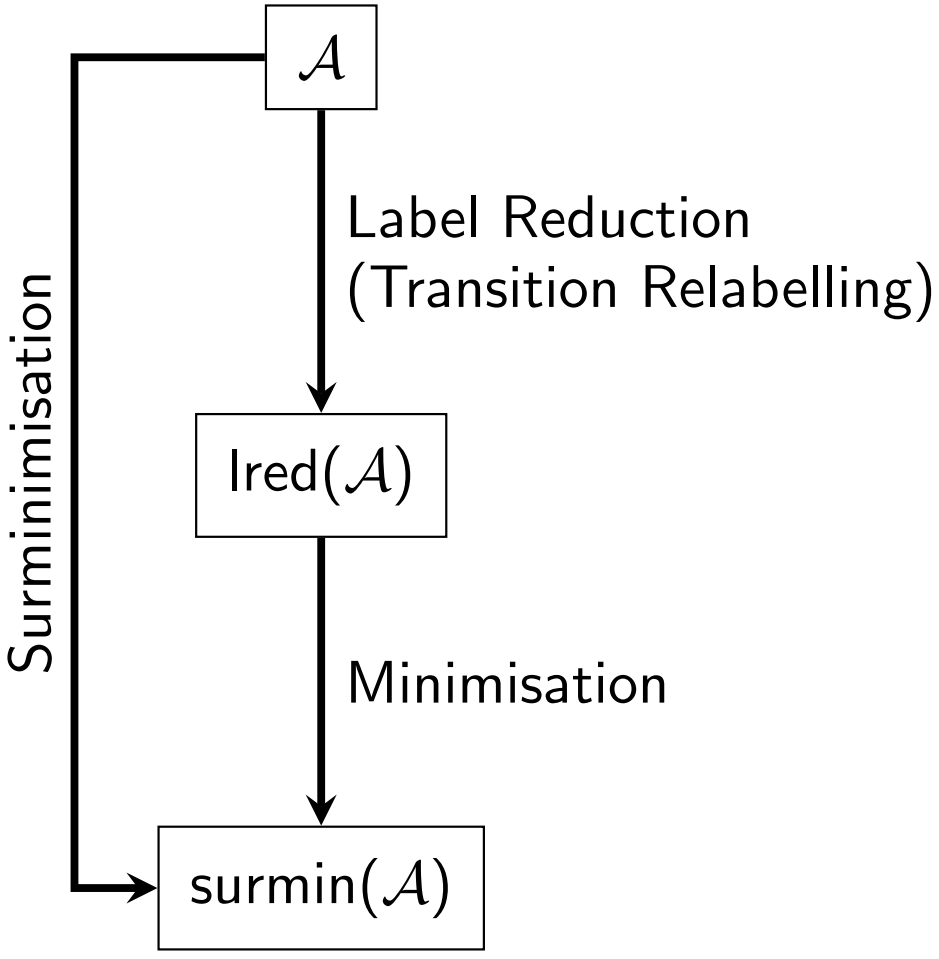
In examples: $a < b < c$, $x < y < z$ and $0 < 1 < 2 < 3 < \cdots$

- Automata are deterministic.
- Automata are trim.

In particular, *minimisation*, *minimal* refers to a trim automaton.

Outline

Automaton Surminimisation



Definition (Label Reduction of Automata)

Automaton $\mathcal{A} = \langle \mathsf{Q}, \mathsf{A}, \delta, \mathsf{i}, \mathsf{F} \rangle$

Its label-reduction: $\mathsf{lred}(\mathcal{A}) = \langle \mathsf{Q}, \mathsf{B}, \delta', \mathsf{i}, \mathsf{F} \rangle$

Definition (Label Reduction of Automata)

Automaton $\mathcal{A} = \langle \mathsf{Q}, \mathsf{A}, \delta, \mathsf{i}, \mathsf{F} \rangle$

Its label-reduction: $\mathsf{lred}(\mathcal{A}) = \langle \mathsf{Q}, \mathsf{B}, \delta', \mathsf{i}, \mathsf{F} \rangle$

Definition (Label Reduction of Automata)

Automaton $\mathcal{A} = \langle \mathrm{Q}, \mathrm{A}, \delta, \mathrm{i}, \mathrm{F} \rangle$

$$\left.\begin{array}{lcl} p & \xrightarrow{a_0} & p_0 \\ p & \xrightarrow{a_1} & p_1 \\ & \vdots & \\ p & \xrightarrow{a_k} & p_k \end{array}\right\} \text{ with } a_0 < a_1 < \cdots < a_k \ .$$

$\Big\downarrow$ Label Reduction

Its label-reduction: $\mathrm{lred}(\mathcal{A}) = \langle \mathrm{Q}, \mathrm{B}, \delta', \mathrm{i}, \mathrm{F} \rangle$

$$\begin{array}{lcl} p & \xrightarrow{0} & p_0 \\ p & \xrightarrow{1} & p_1 \\ & \vdots & \\ p & \xrightarrow{k} & p_k \end{array}$$

Definition (Label Reduction of Automata)

Automaton $\mathcal{A} = \langle \mathrm{Q}, \mathrm{A}, \delta, \mathrm{i}, \mathrm{F} \rangle$

$$\left.\begin{array}{l} p \xrightarrow{a_0} p_0 \\ p \xrightarrow{a_1} p_1 \\ \quad\vdots \\ p \xrightarrow{a_k} p_k \end{array}\right\} \text{ with } a_0 < a_1 < \cdots < a_k \ .$$

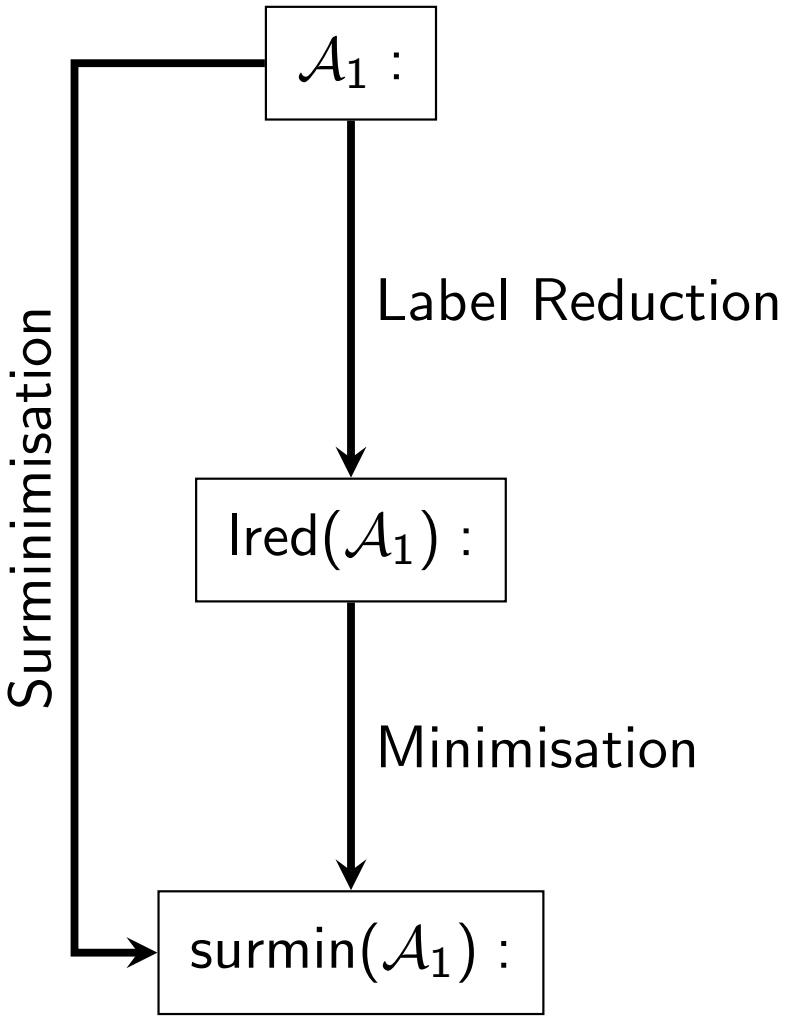
Label Reduction

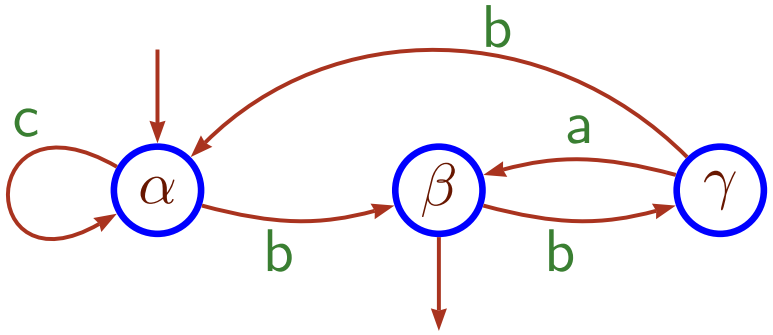
Its label-reduction: $\mathrm{lred}(\mathcal{A}) = \langle \mathrm{Q}, \mathrm{B}, \delta', \mathrm{i}, \mathrm{F} \rangle$

$$\begin{array}{l} p \xrightarrow{0} p_0 \\ p \xrightarrow{1} p_1 \\ \quad\vdots \\ p \xrightarrow{k} p_k \end{array}$$

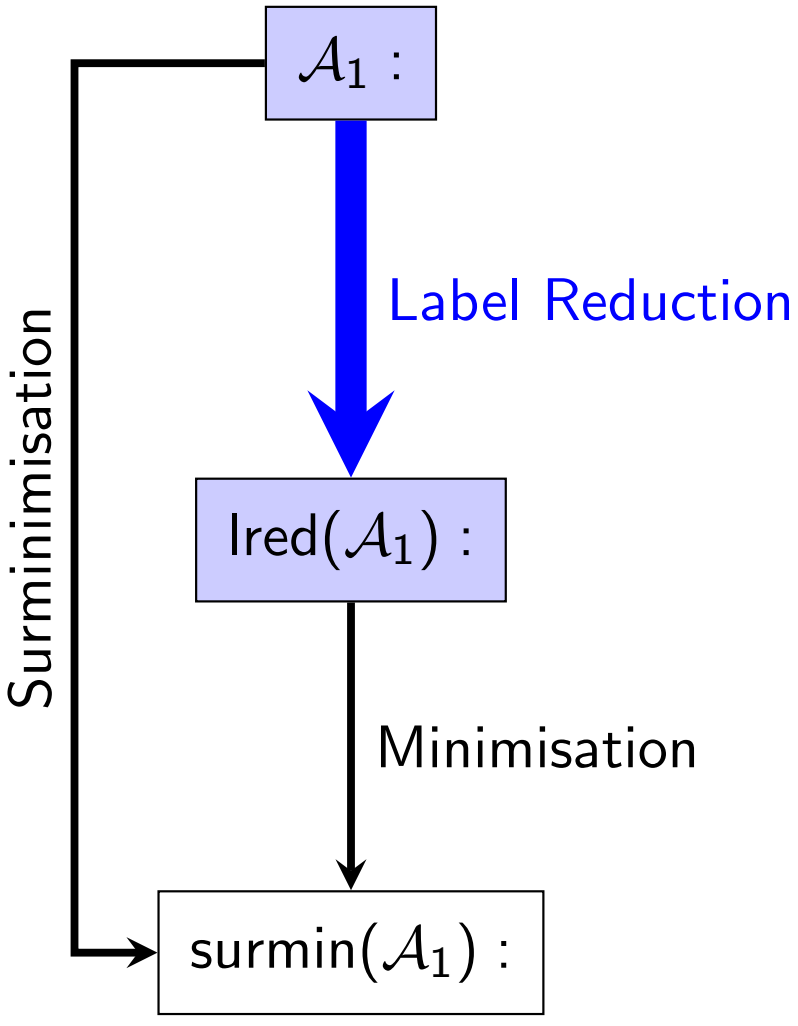
$B = \{0, 1, \cdots (k-1)\}$ where k is the maximal out-degree of the states of $\mathcal{A}$

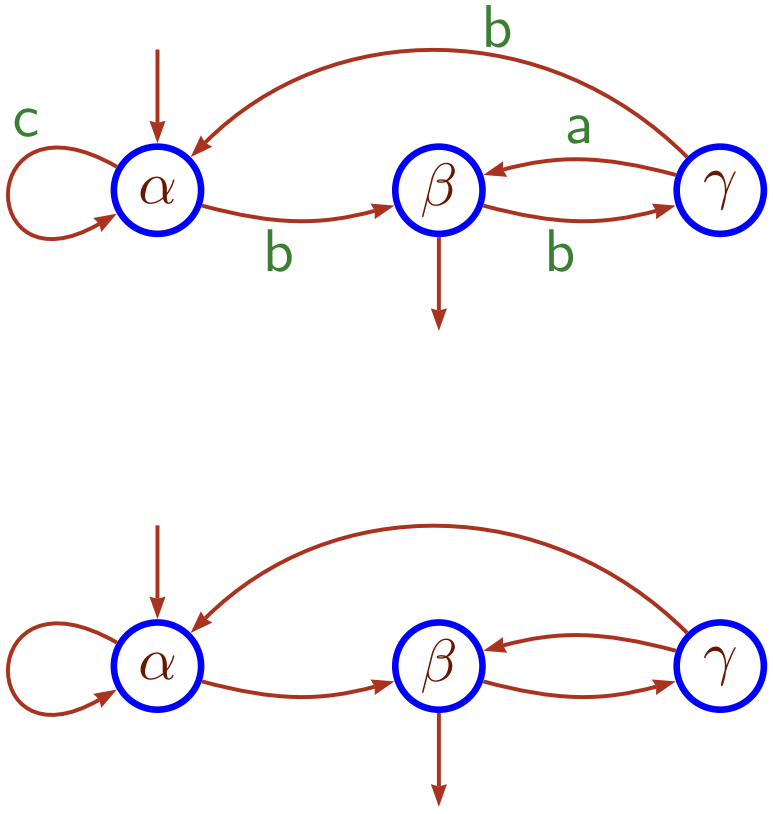
Example of Surminimisation (1)



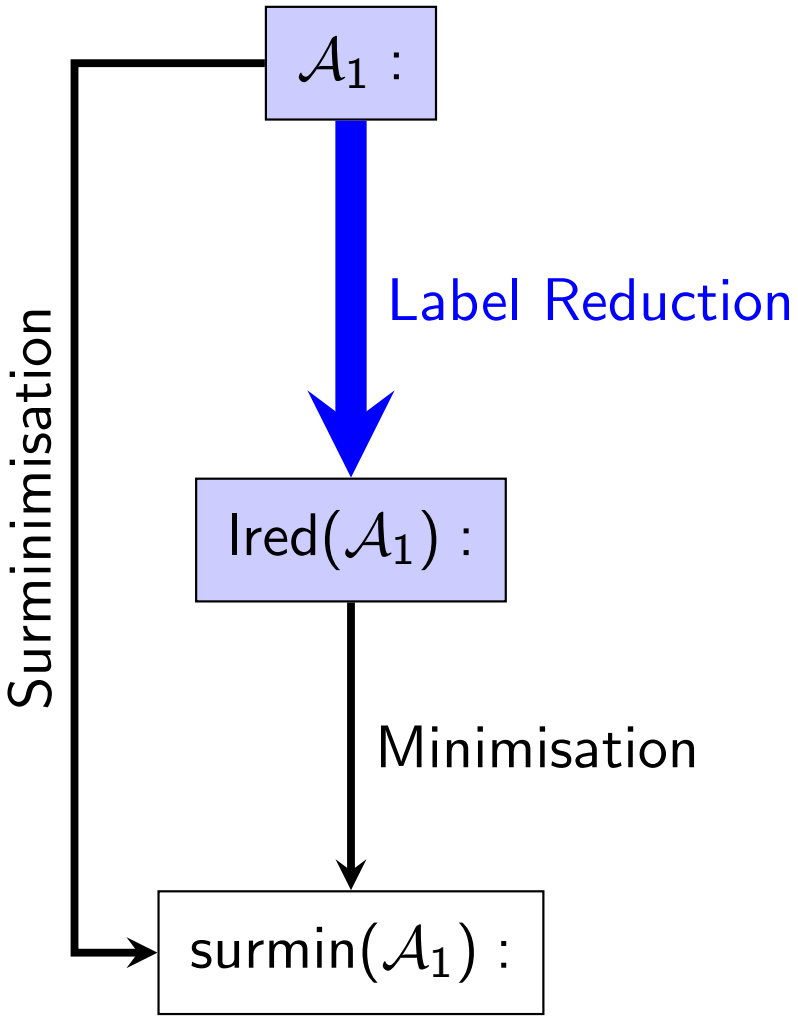


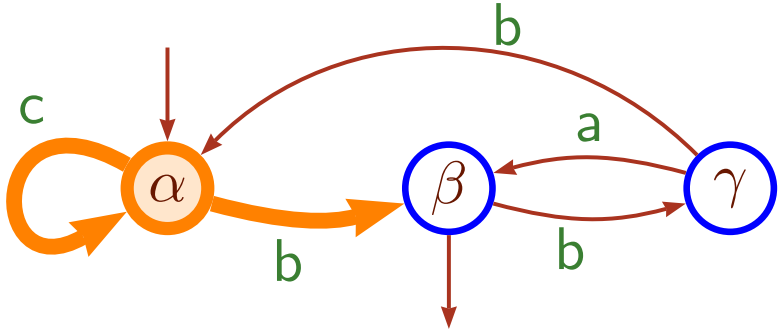
Example of Surminimisation (1)

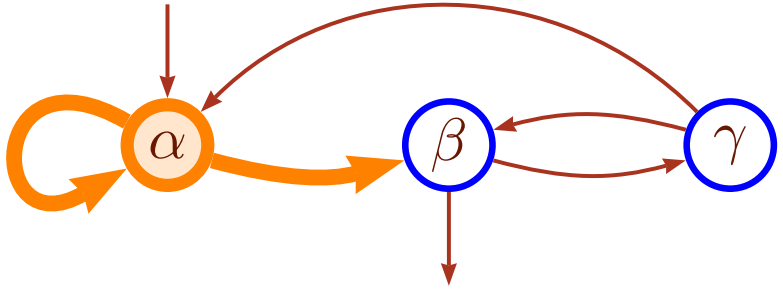




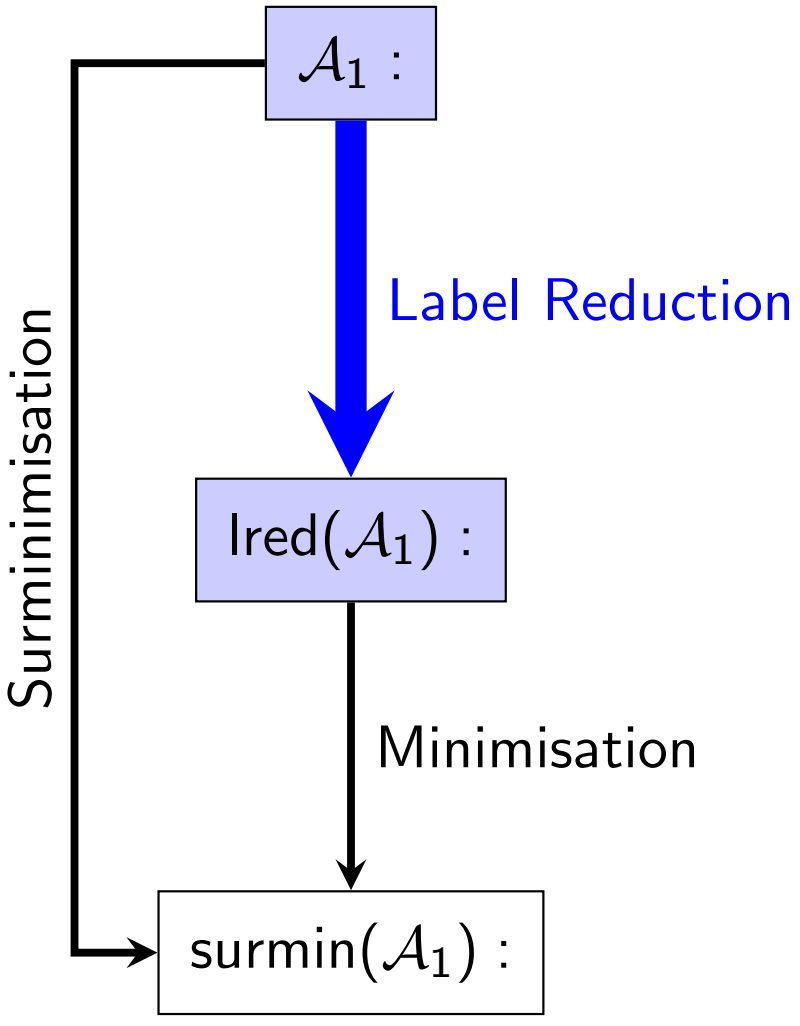
Example of Surminimisation (1)

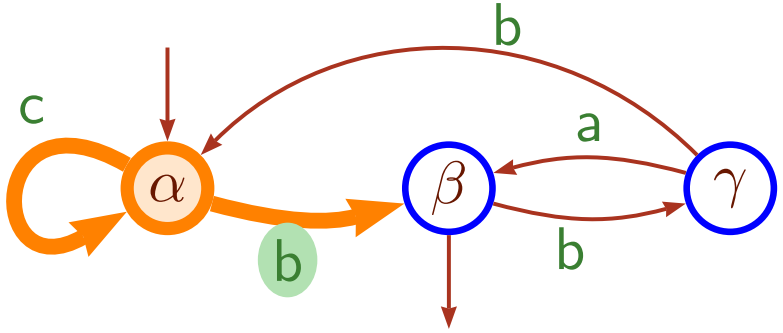


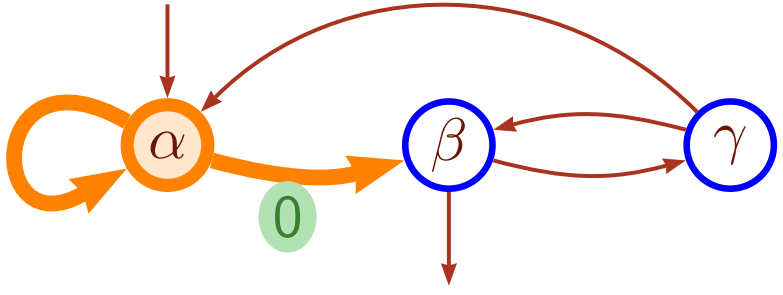




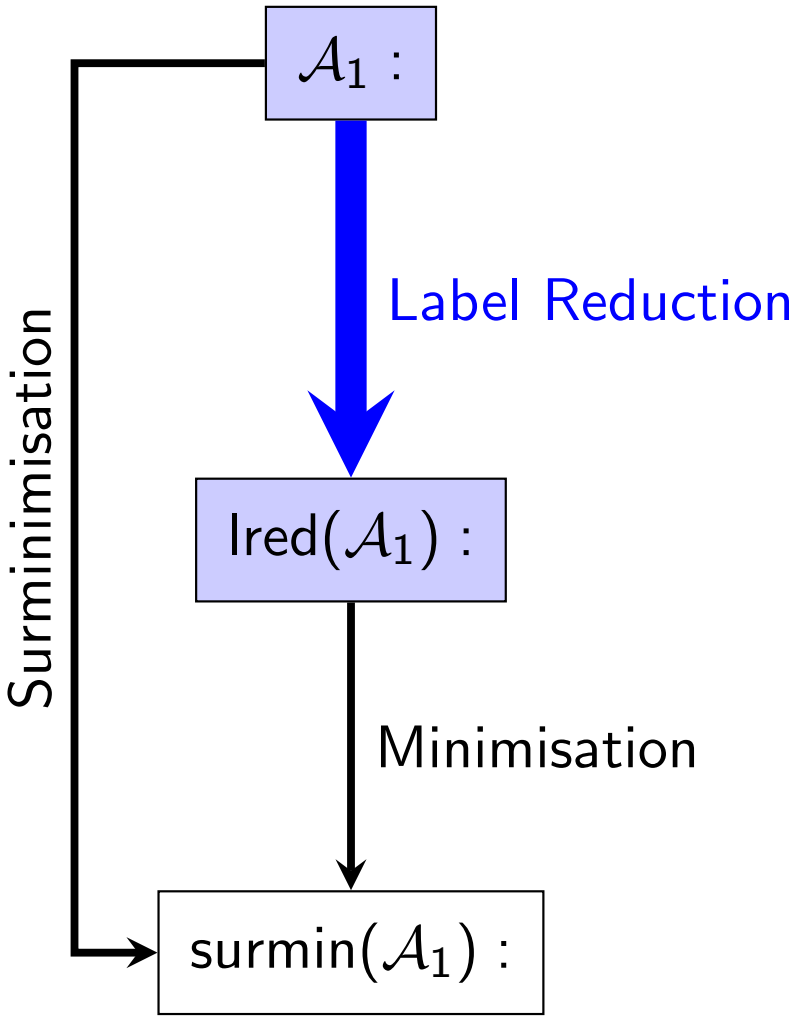
Example of Surminimisation (1)

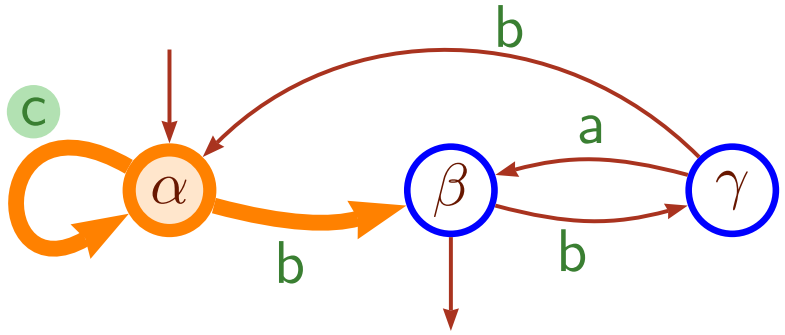


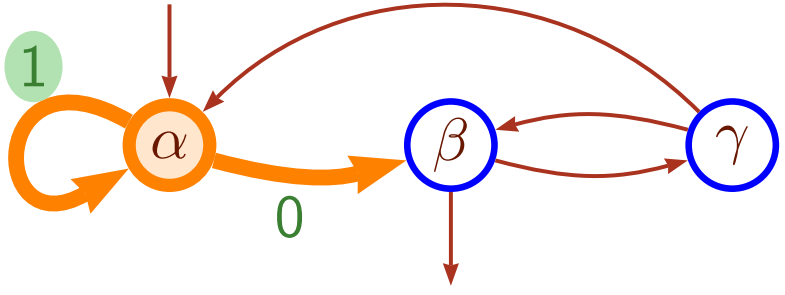




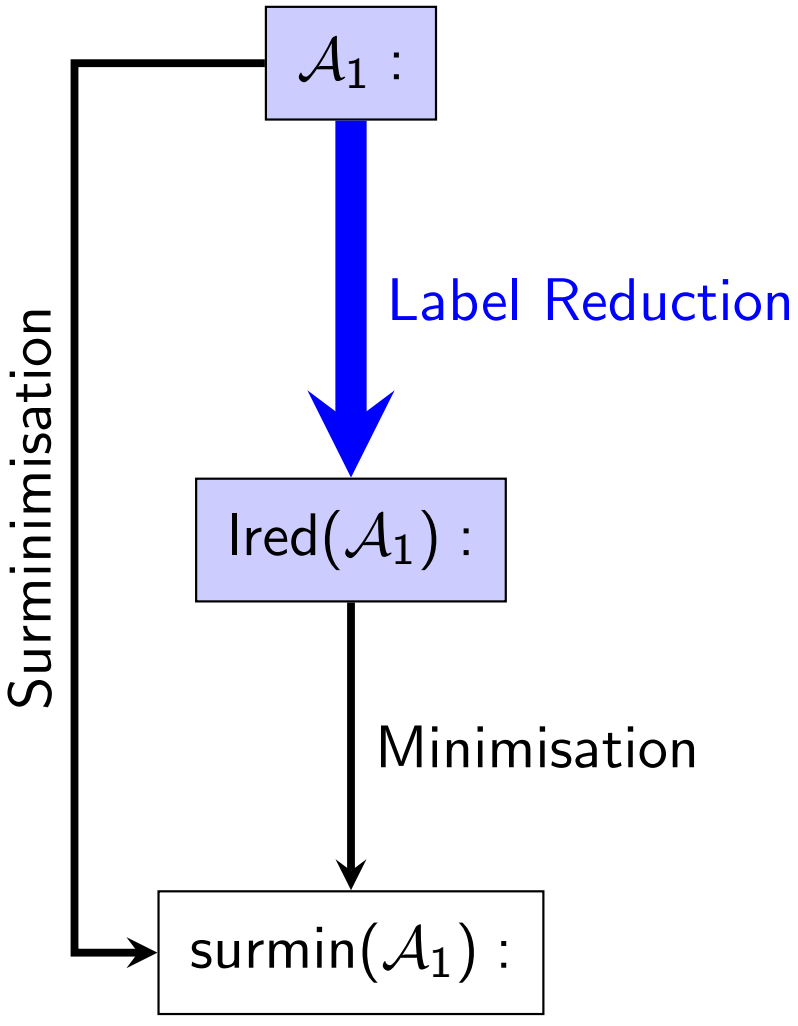
Example of Surminimisation (1)

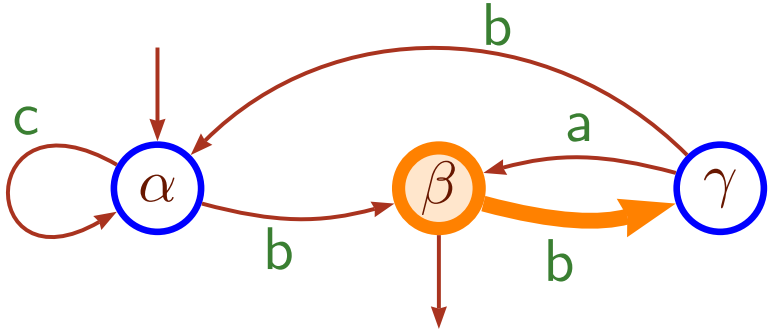


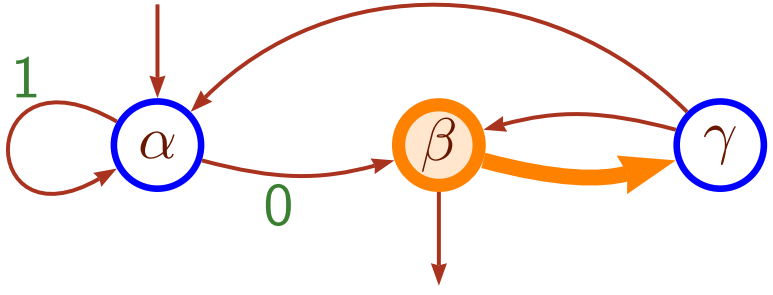




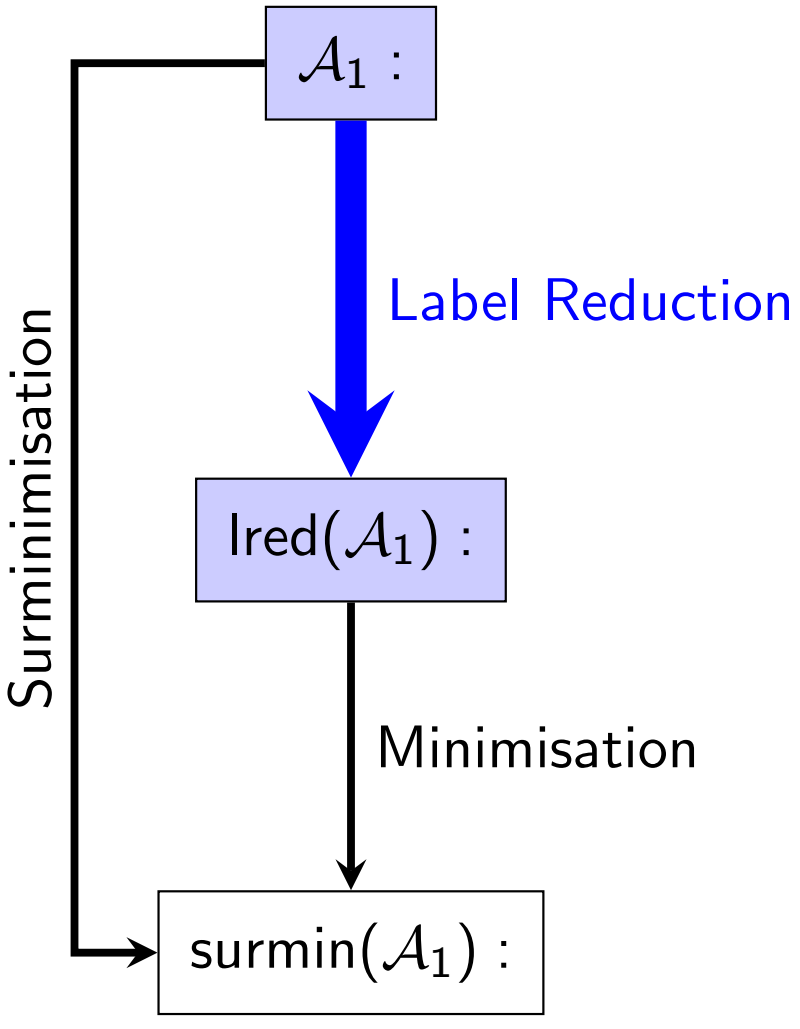
Example of Surminimisation (1)

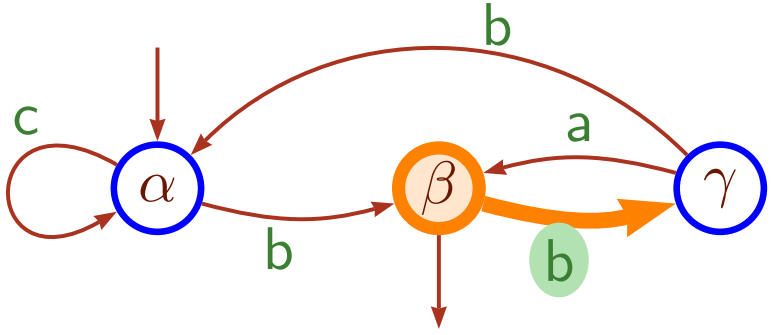


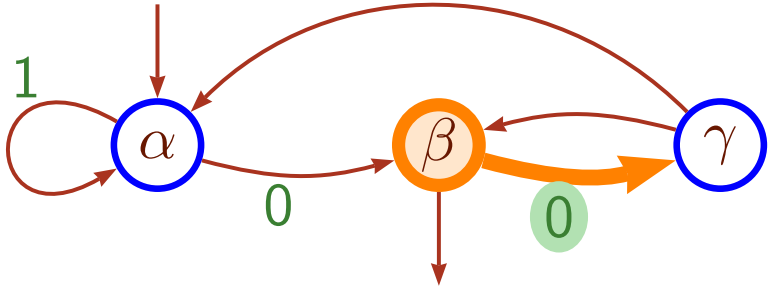




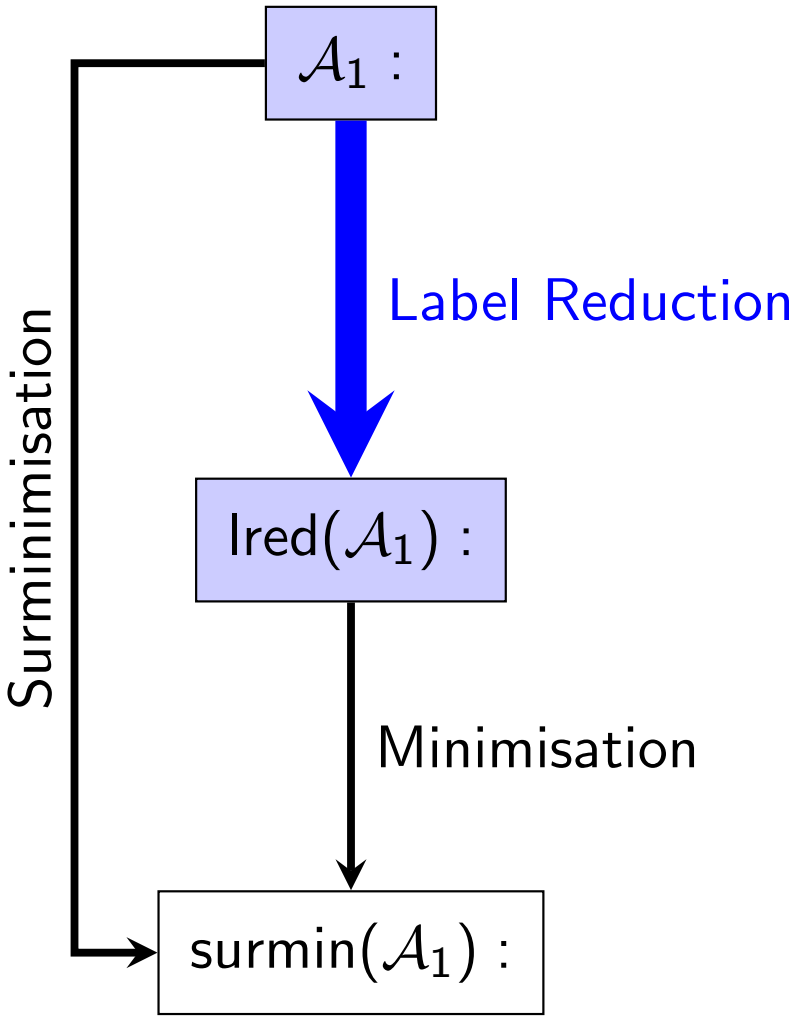
Example of Surminimisation (1)

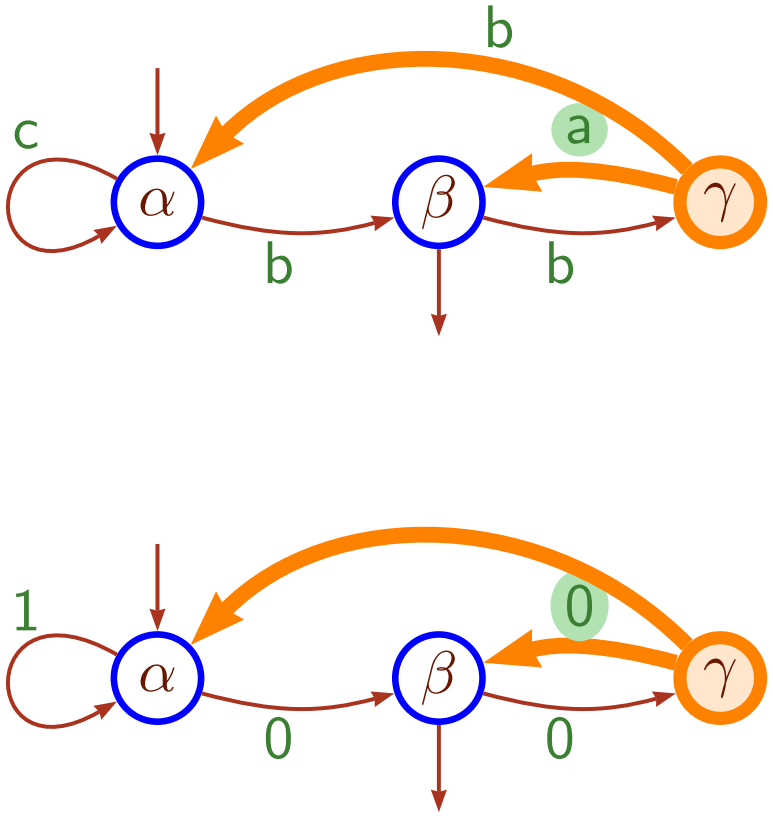




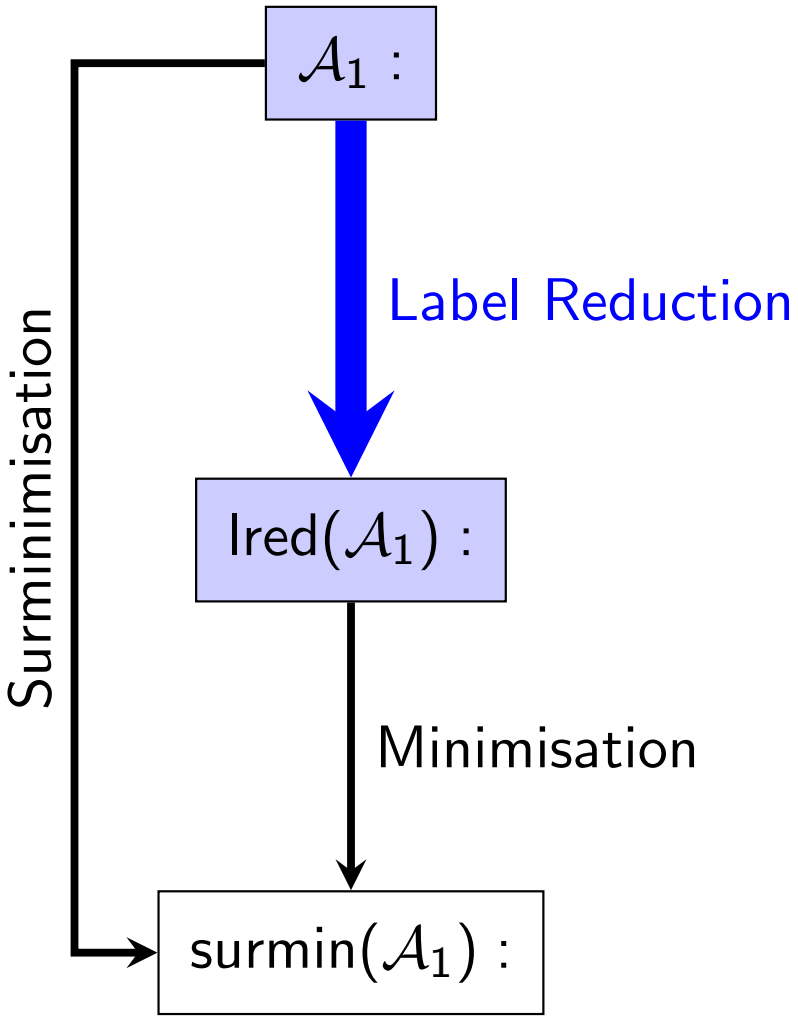


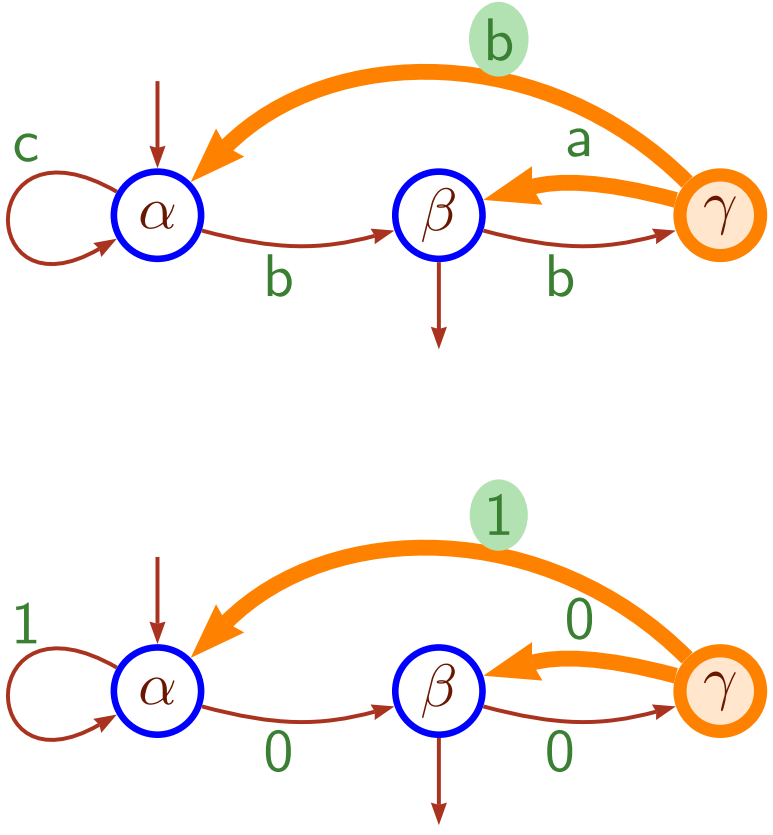
Example of Surminimisation (1)



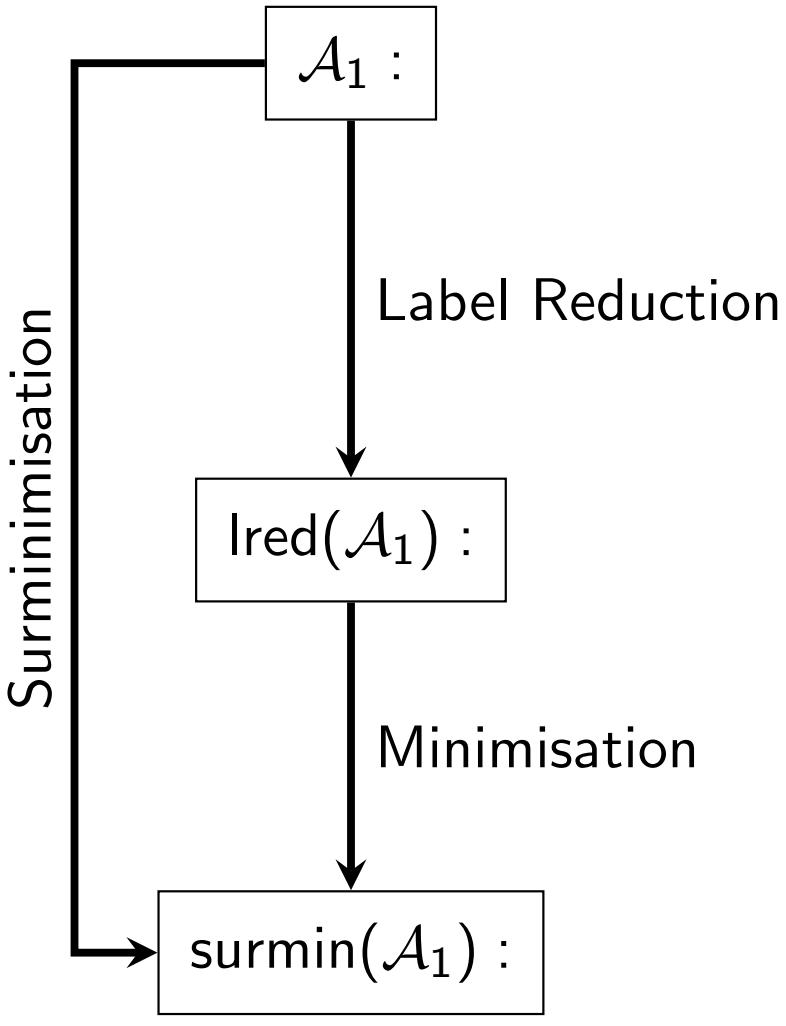


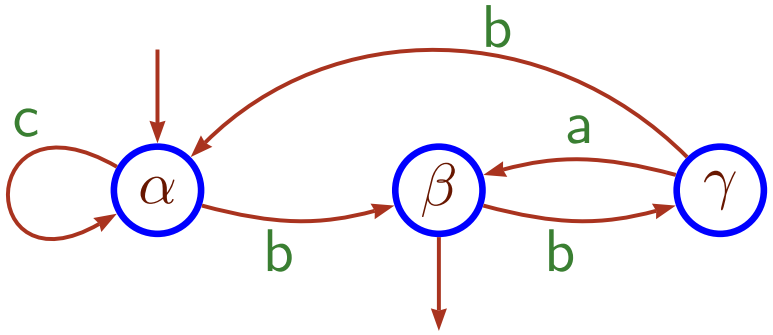
Example of Surminimisation (1)

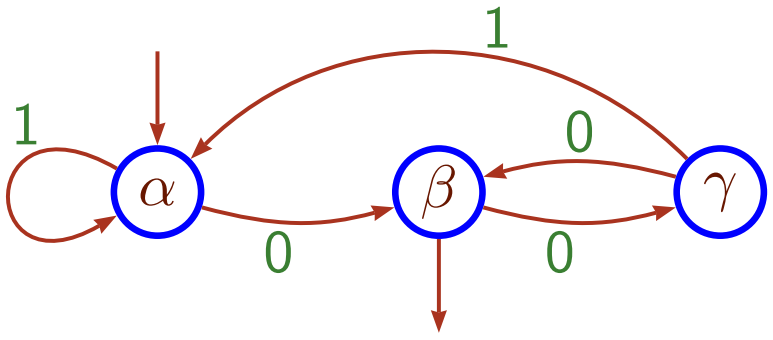




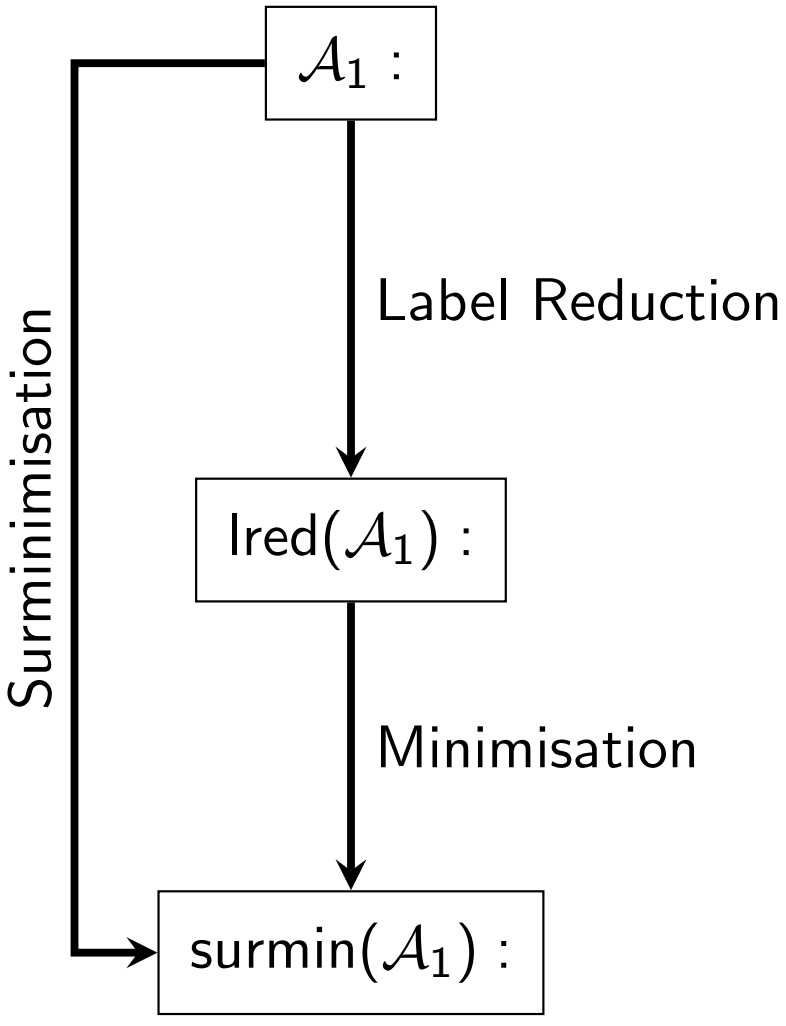
Example of Surminimisation (1)

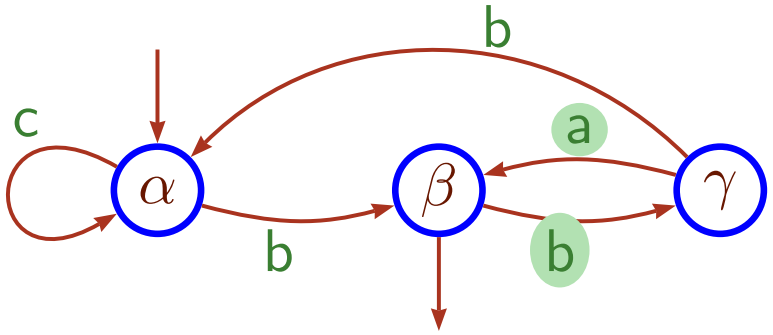


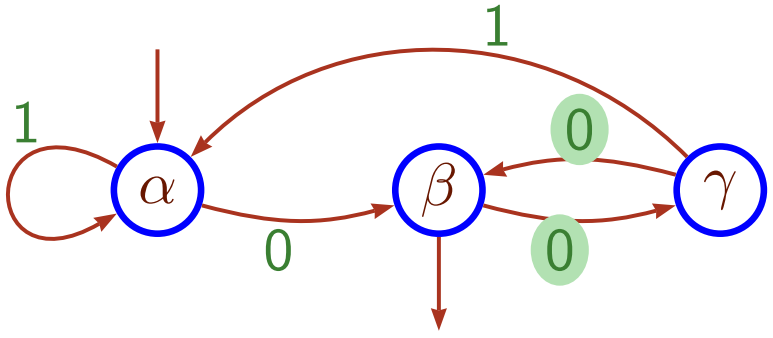




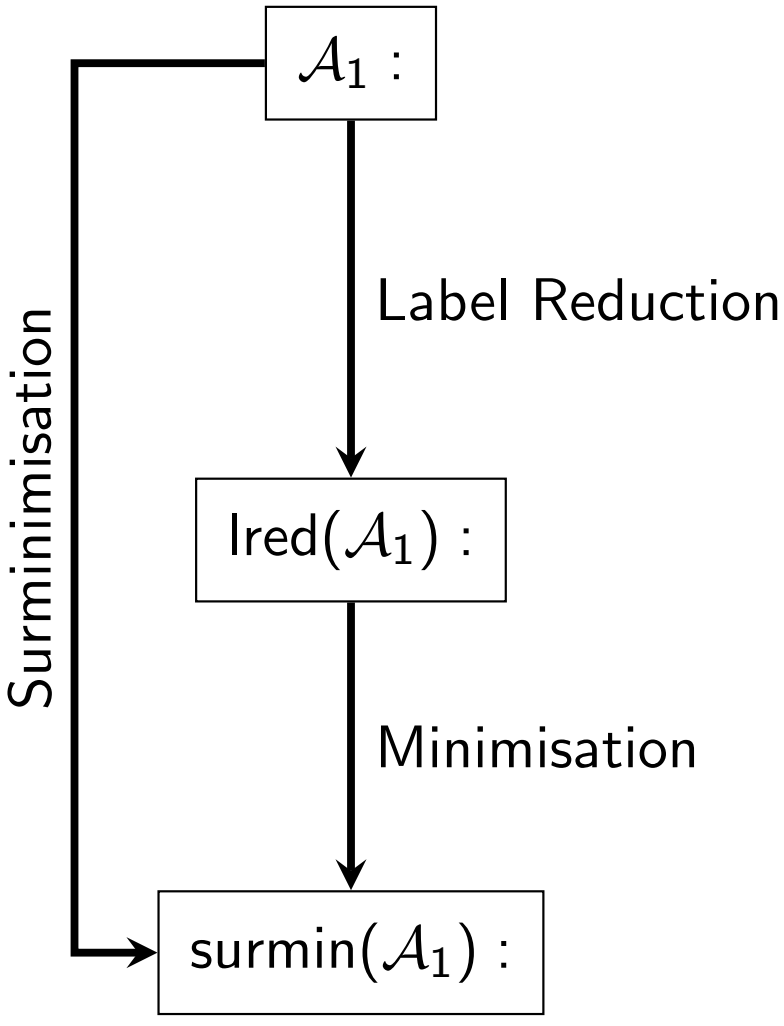
Example of Surminimisation (1)

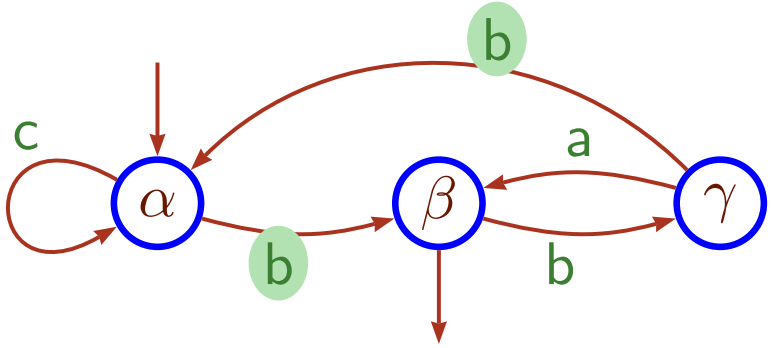


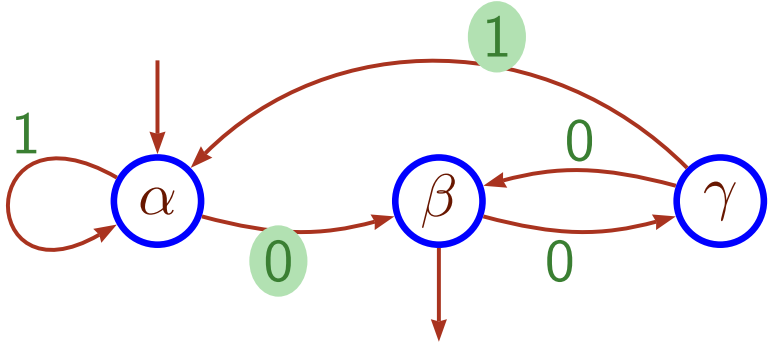




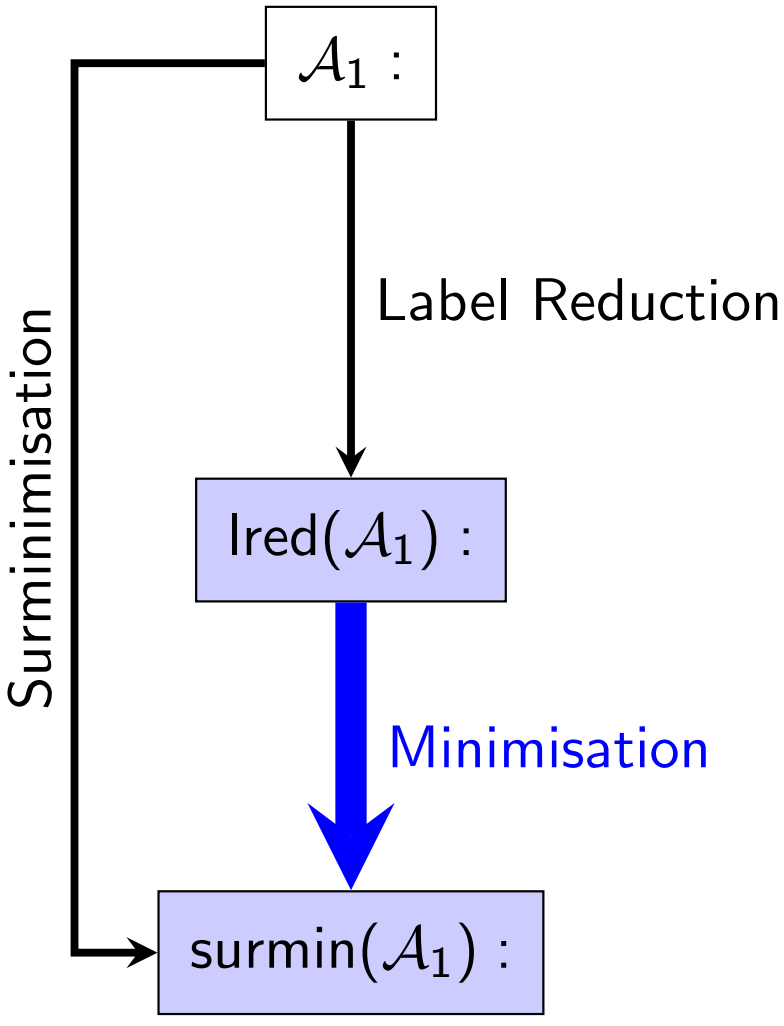
Example of Surminimisation (1)

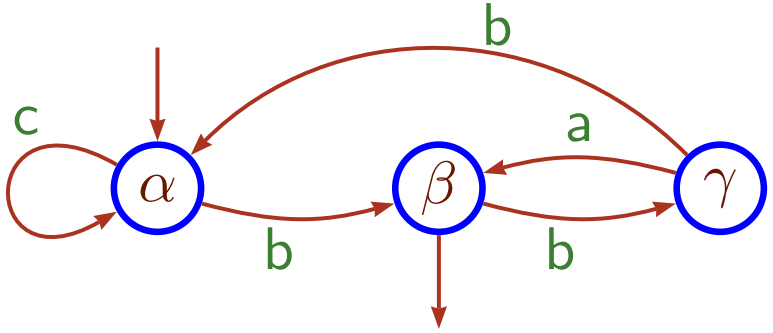


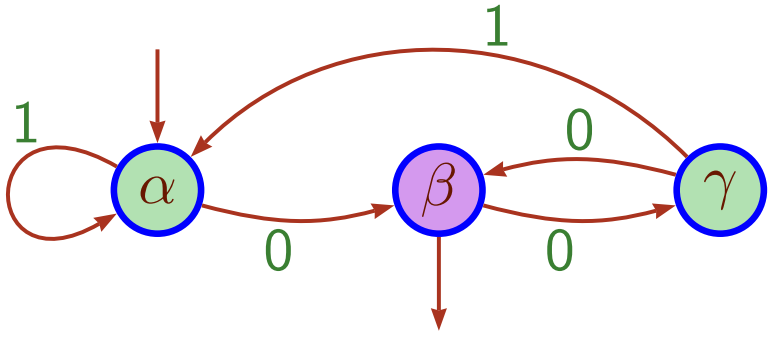


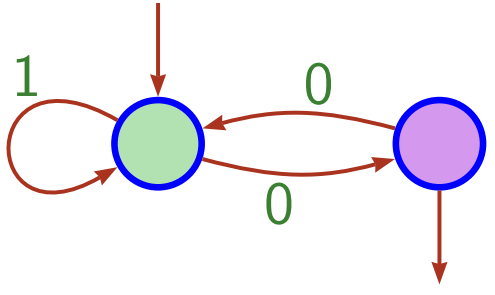


Example of Surminimisation (1)



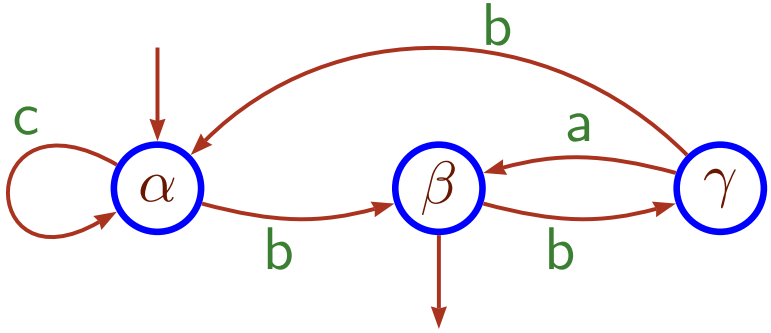






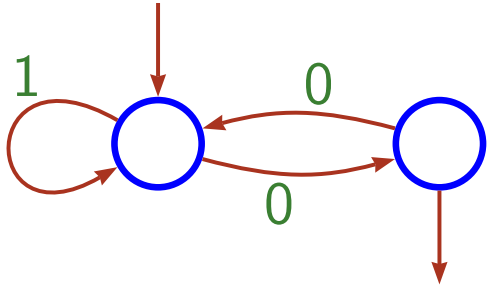
Example of Surminimisation (1)

$\mathcal{A}_1$:

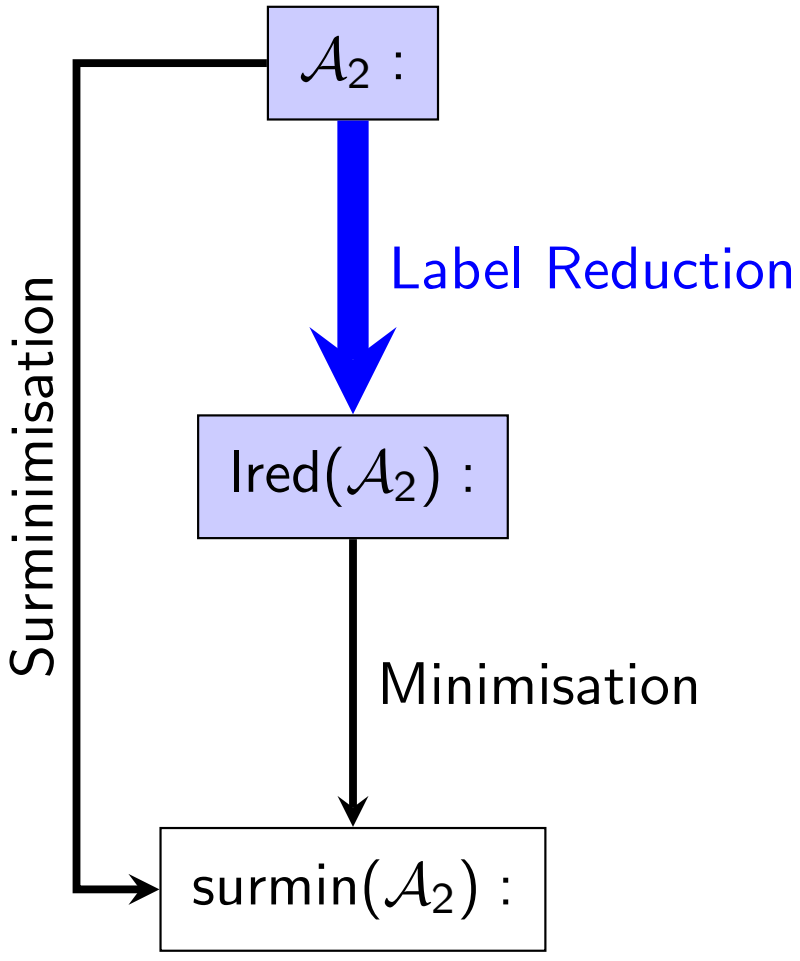


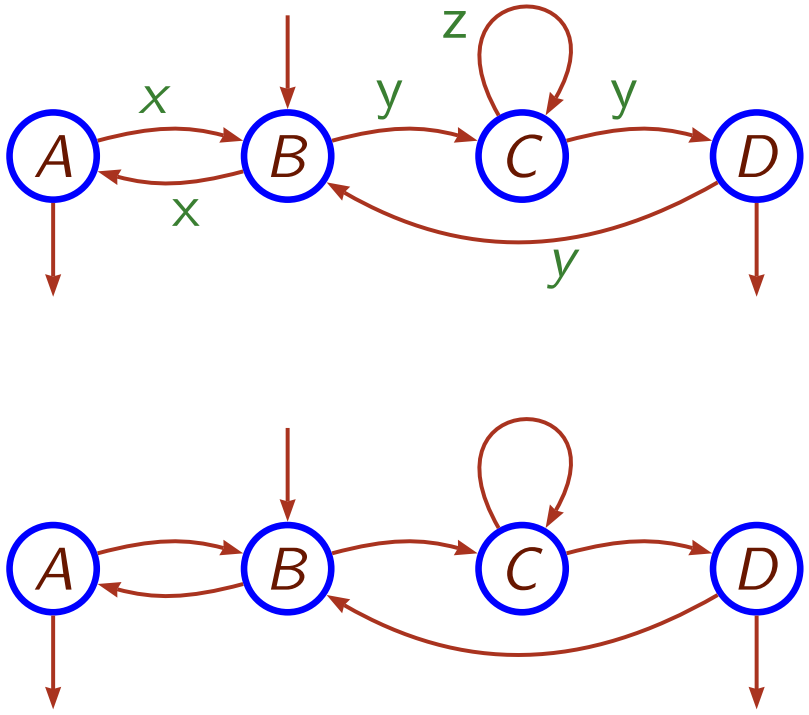
Surminimisation

surmin($\mathcal{A}_1$) :

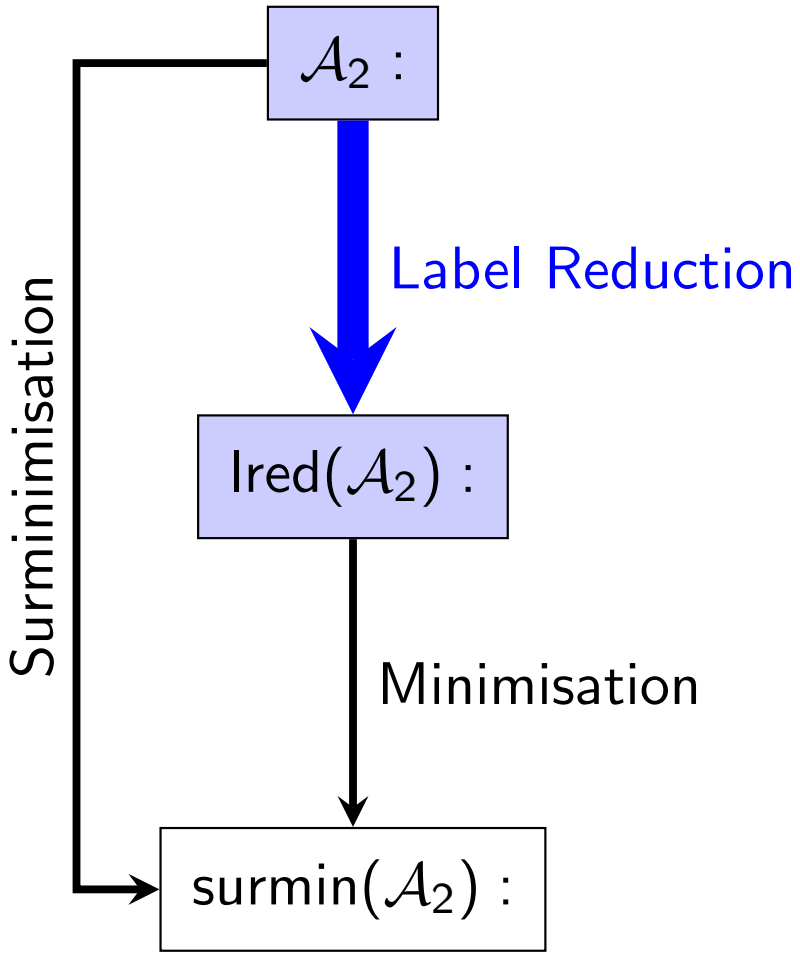


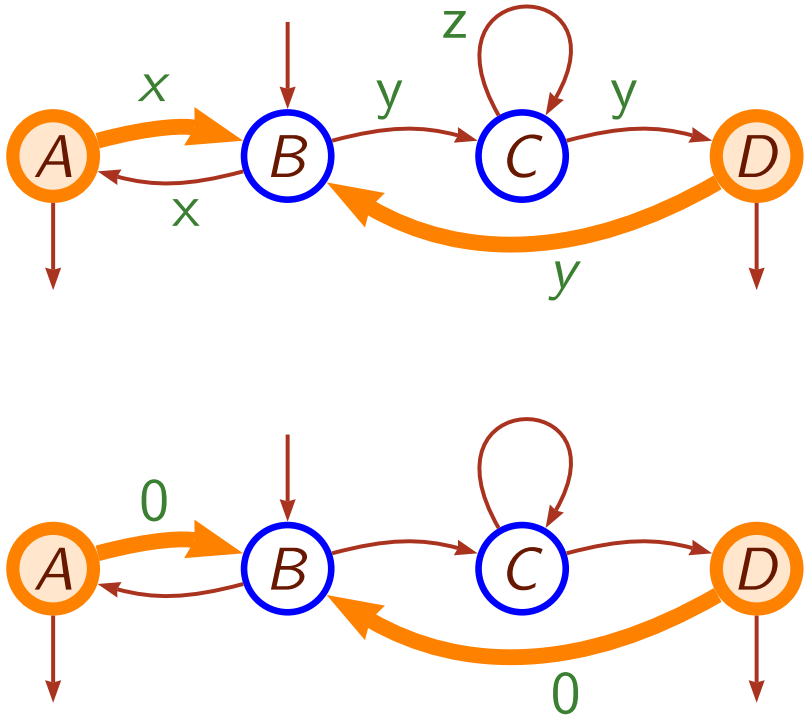
Example of Surminimisation (2)



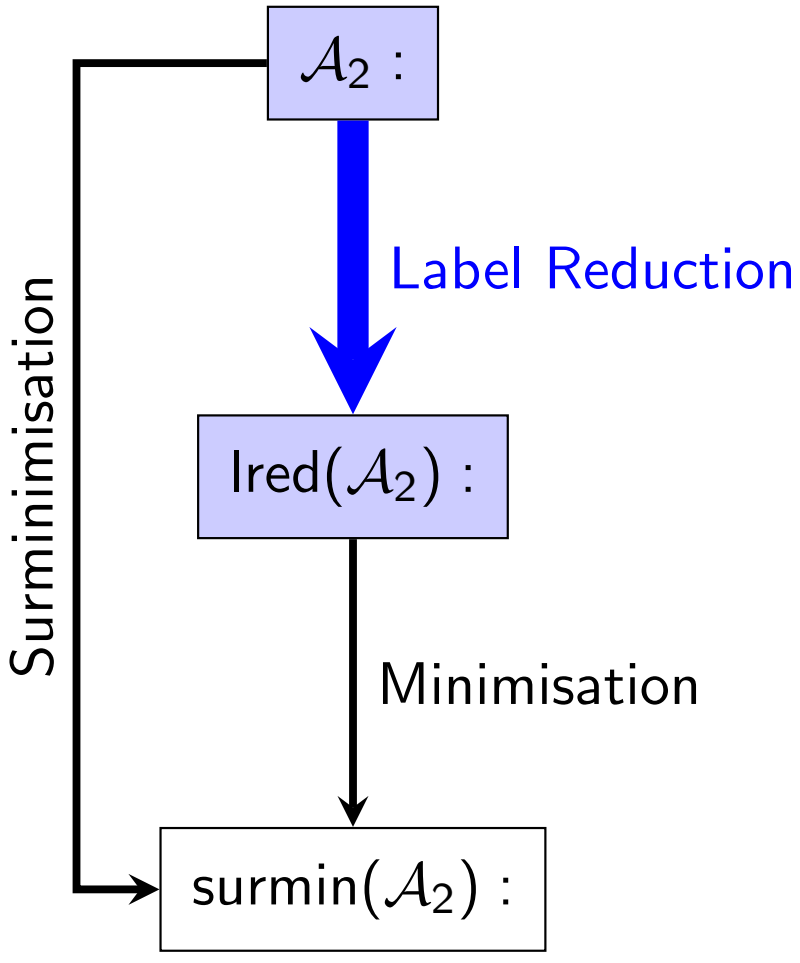


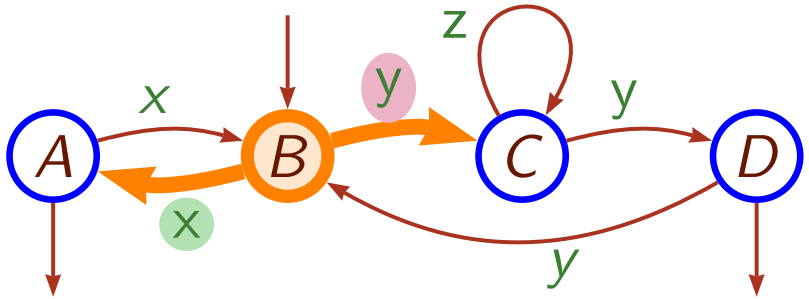
Example of Surminimisation (2)

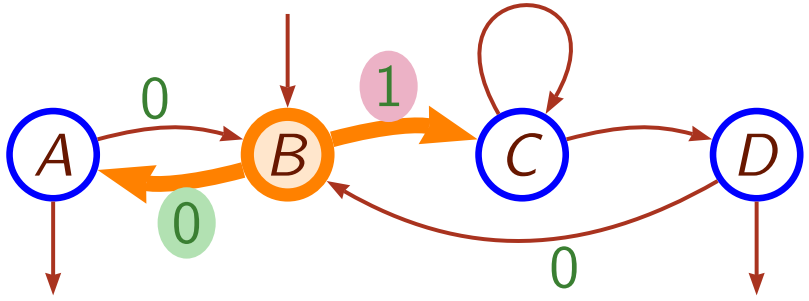




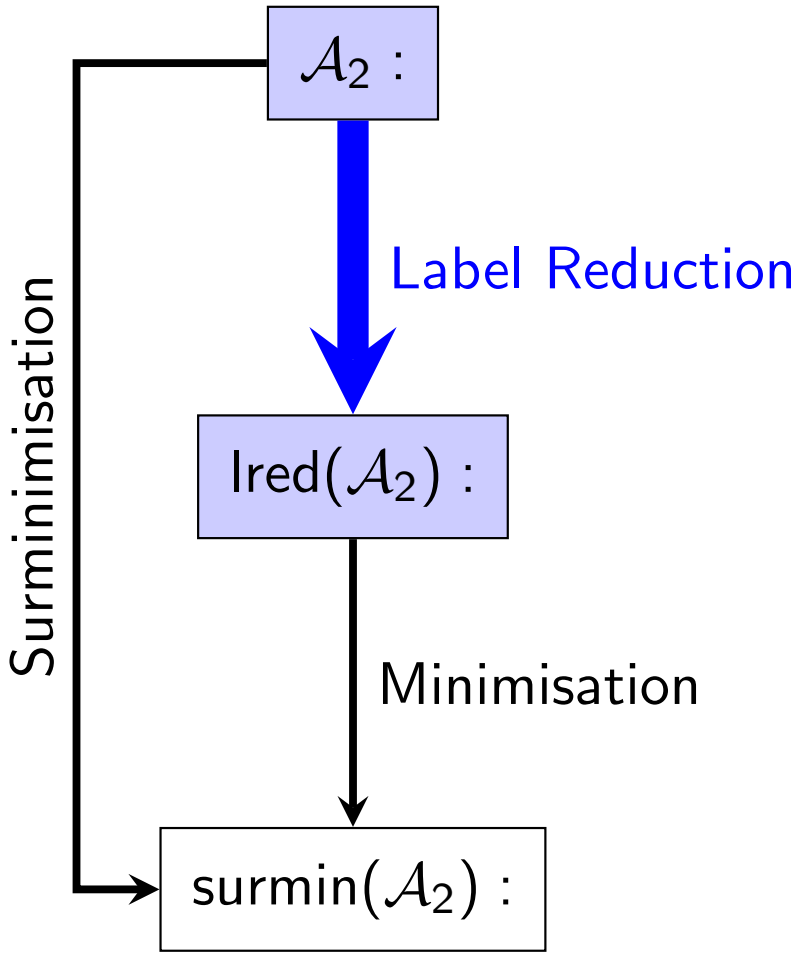
Example of Surminimisation (2)

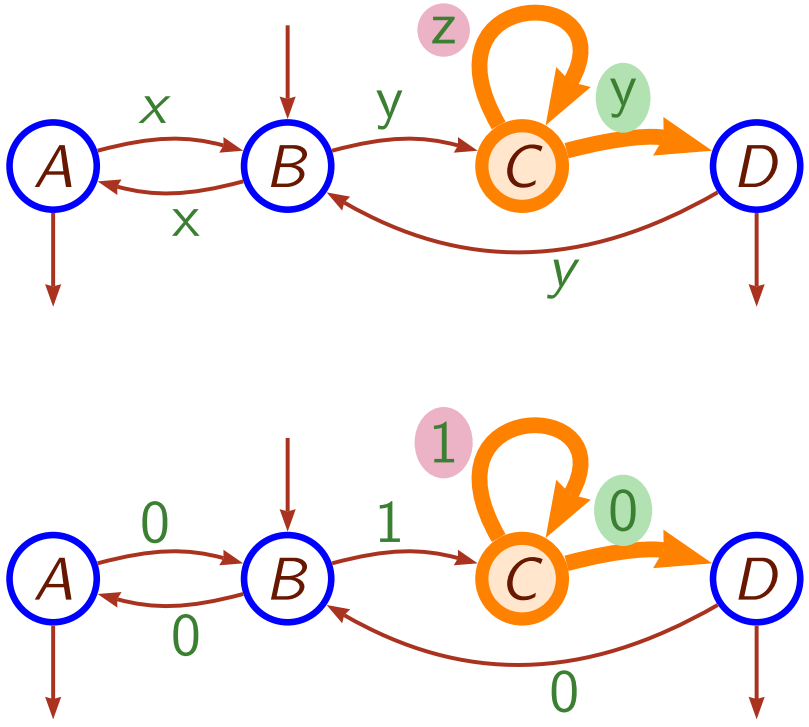




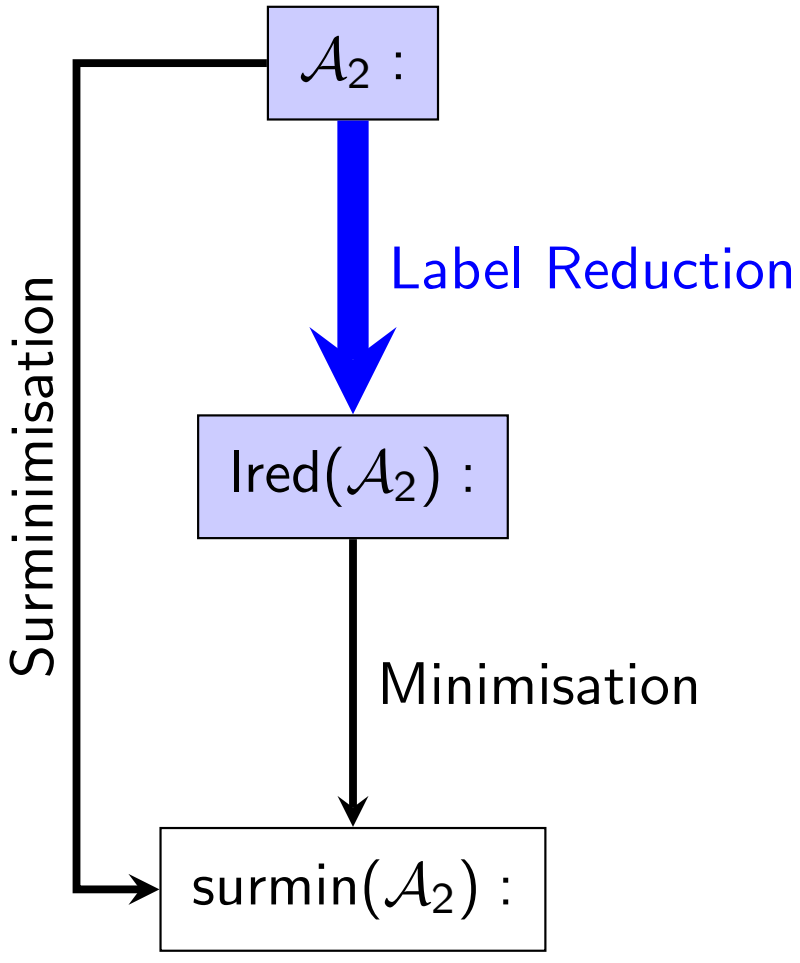


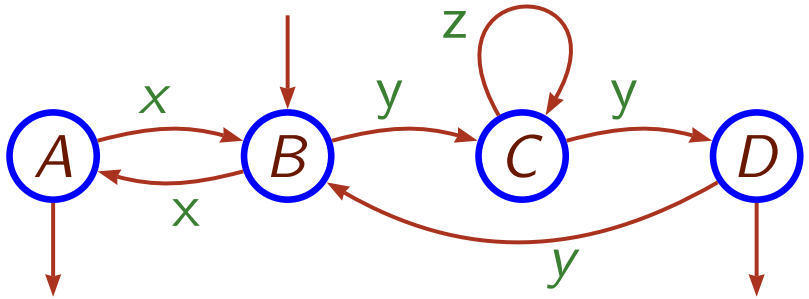
Example of Surminimisation (2)

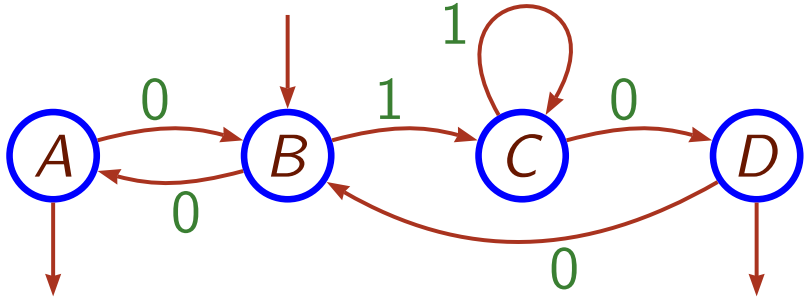




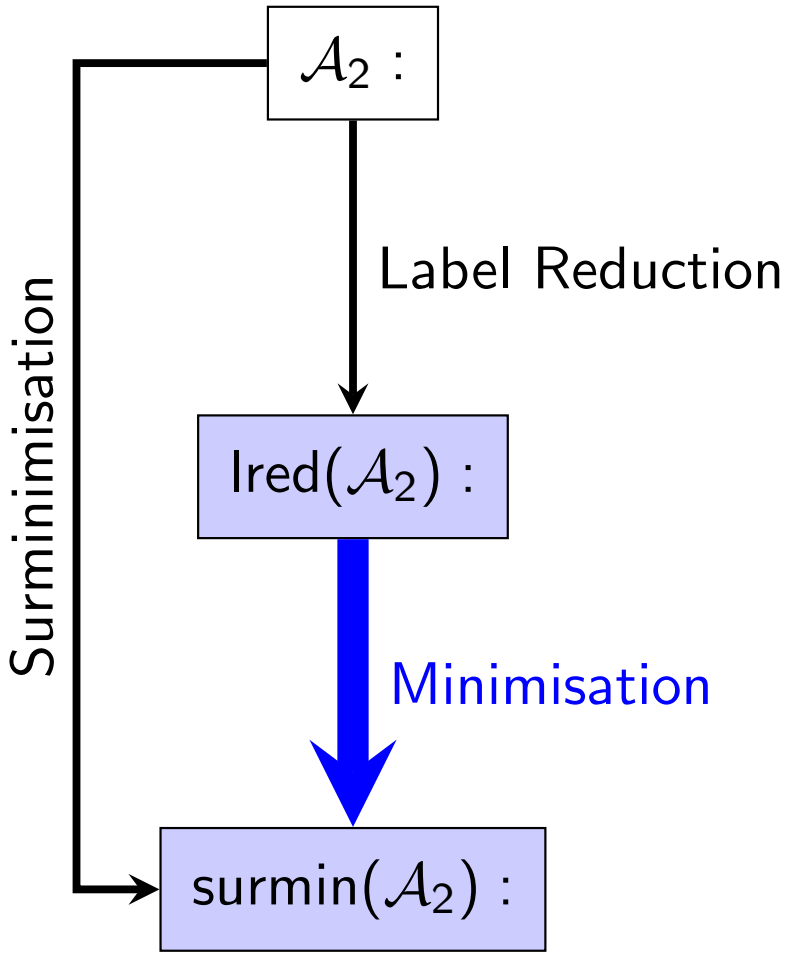
Example of Surminimisation (2)

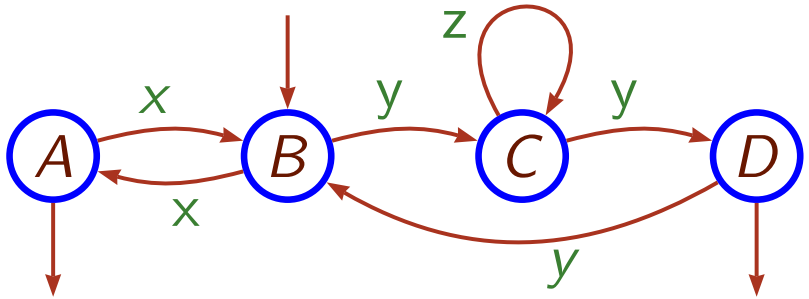


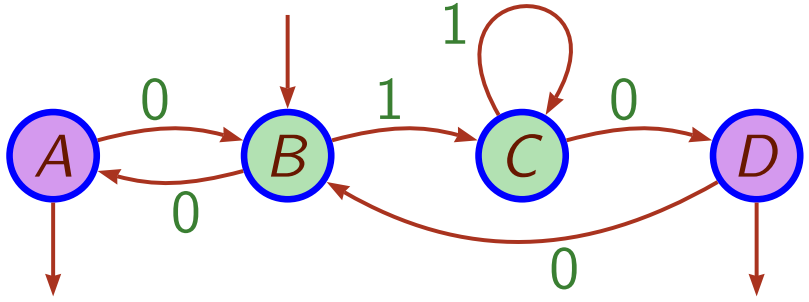


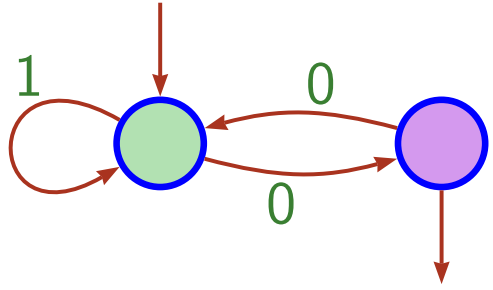


Example of Surminimisation (2)



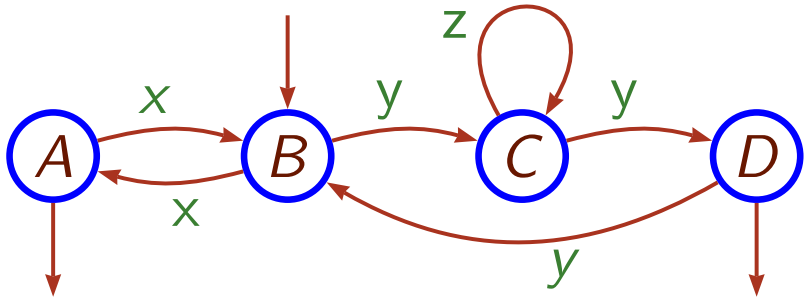






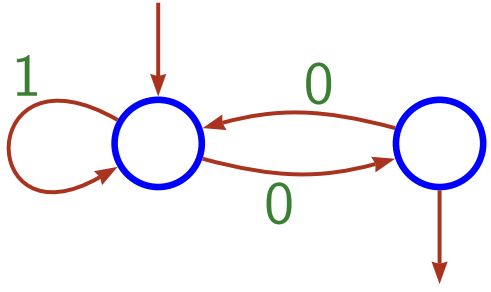
Example of Surminimisation (2)

$\mathcal{A}_2$:



Surminimisation

surmin($\mathcal{A}_2$) :



Properties of Label-Reduction

Lemma

Label-reduction is idempotent.

Commutation Lemma

Label-reduction commutes with automaton morphism.

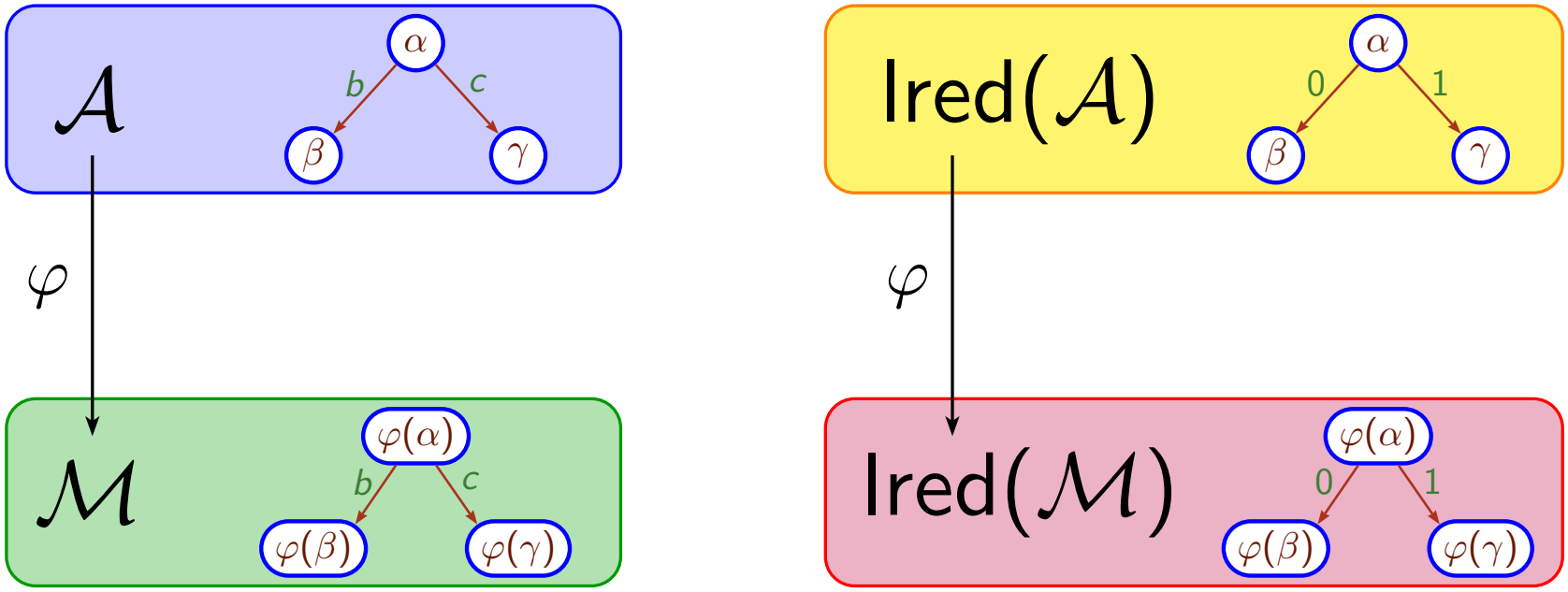
Properties of Label-Reduction

Lemma

Label-reduction is idempotent.

Commutation Lemma

Label-reduction commutes with automaton morphism.



Remarks on Surminimisation

It removes the eventual meaning of letters

$L_3 = 0^*1^*$	*some 0's followed by some 1's*
$\mathsf{lred}(L_3) = 0^* + 0^*10^*$	*words with at most one 1*

In general, "marker" or "matching" symbols disapear.

Remarks on Surminimisation

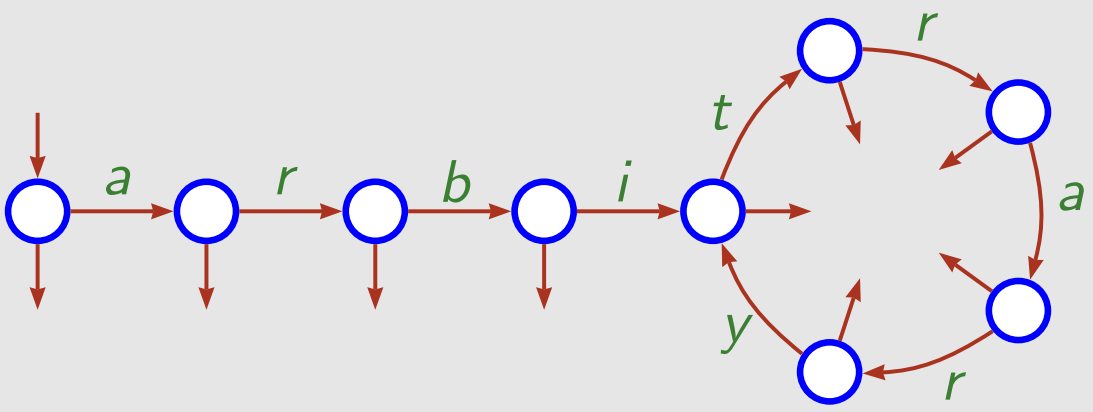
It removes the eventual meaning of letters

$L_3 = 0^*1^*$ *some 0's followed by some 1's*

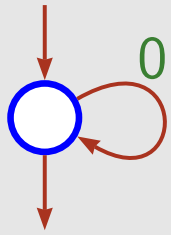
$\mathrm{lred}(L_3) = 0^* + 0^*10^*$ *words with at most one 1*

In general, "marker" or "matching" symbols disapear.

It removes complexity due to an arbitrary choice of letter



$\mathcal{A}_4$



$\mathrm{surmin}(\mathcal{A}_4)$

Remarks on Surminimisation

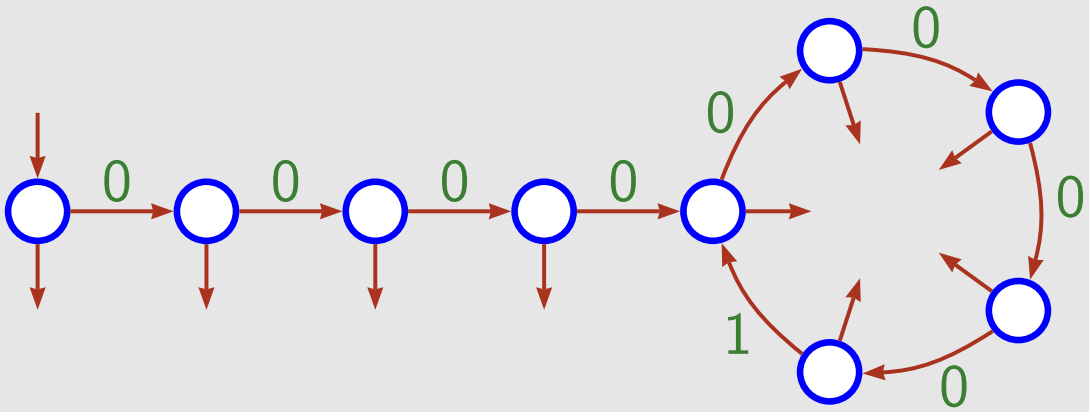
It removes the eventual meaning of letters

$L_3 = 0^*1^*$ *some 0's followed by some 1's*

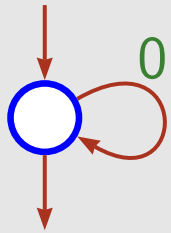
$\mathrm{lred}(L_3) = 0^* + 0^*10^*$ *words with at most one 1*

In general, "marker" or "matching" symbols disapear.

It removes complexity due to an arbitrary choice of letter



$\mathcal{A}_4$



$\mathrm{surmin}(\mathcal{A}_4)$

Outline

T-equivalence (1)

Definition (T-equivalence)

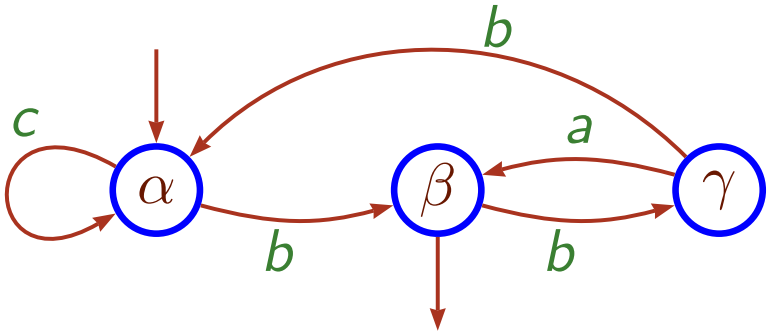
$\mathcal{A}$ and $\mathcal{B}$: two trim DFA

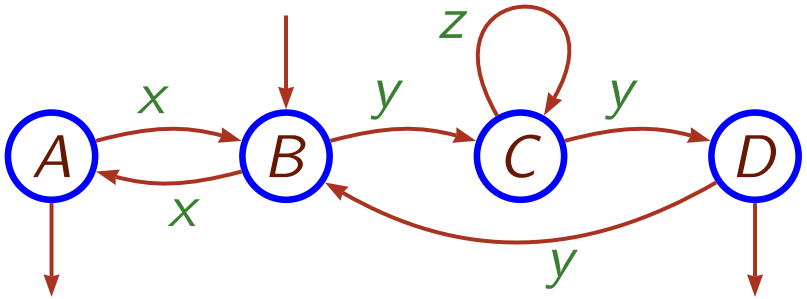
$\mathcal{A} \overset{T}{\sim} \mathcal{B}$ if $\mathsf{lred}(\mathcal{A})$ and $\mathsf{lred}(\mathcal{B})$ accept the same language

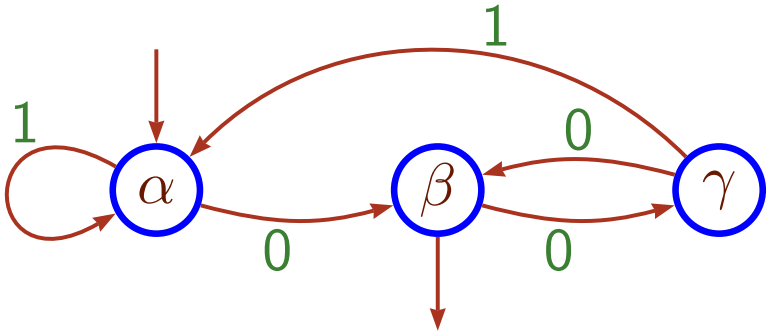
Equivalent definition

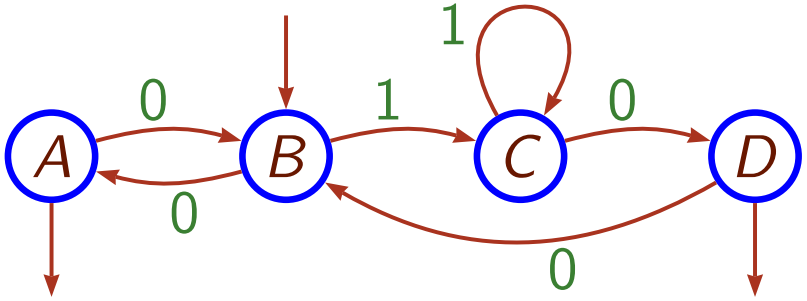
$$\mathcal{A} \overset{T}{\sim} \mathcal{B} \iff \mathsf{surmin}(\mathcal{A}) = \mathsf{surmin}(\mathcal{B})$$

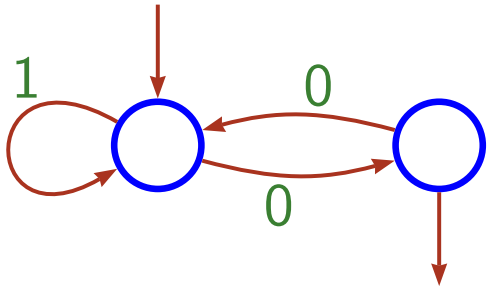
$\mathcal{A}_1$ and $\mathcal{A}_2$ are T-equivalent











T-equivalence (2)

Proposition

$\mathcal{A}$ and $\mathcal{B}$: two **trim** automata.

$$L(\mathcal{A}) = L(\mathcal{B}) \quad \Longrightarrow \quad L(\mathsf{Ired}(\mathcal{A})) = L(\mathsf{Ired}(\mathcal{B}))$$

T-equivalence (2)

Proposition

$\mathcal{A}$ and $\mathcal{B}$: two **trim** automata.

$$L(\mathcal{A}) = L(\mathcal{B}) \quad \Longrightarrow \quad L(\mathsf{Ired}(\mathcal{A})) = L(\mathsf{Ired}(\mathcal{B}))$$

Proof.
Let $\mathcal{M}$ be the minimisation of $\mathcal{A}$ and $\mathcal{B}$.
$\Longrightarrow$ $\exists$ Two morphisms $\phi : \mathcal{A} \to \mathcal{M}$ and $\psi : \mathcal{B} \to \mathcal{M}$.
Apply Commutation Lemma.

T-equivalence (2)

Proposition

$\mathcal{A}$ and $\mathcal{B}$: two **trim** automata.

$$L(\mathcal{A}) = L(\mathcal{B}) \implies L(\mathsf{Ired}(\mathcal{A})) = L(\mathsf{Ired}(\mathcal{B}))$$

Proof.
Let $\mathcal{M}$ be the minimisation of $\mathcal{A}$ and $\mathcal{B}$.
$\implies \exists$ Two morphisms $\phi : \mathcal{A} \to \mathcal{M}$ and $\psi : \mathcal{B} \to \mathcal{M}$.
Apply Commutation Lemma.

Corollary

T-equivalence is a coarser relation than equivalence.

† Two automata are equivalent if they accept the same language.

Representative of T-equivalence class

Proposition

Surminimisation is idempotent.

(Follows from the *commutation Lemma* and the idempotence of label-reduction)

Representative of T-equivalence class

Proposition

Surminimisation is idempotent.

(Follows from the *commutation Lemma* and the idempotence of label-reduction)

Corollary

Each T-equivalence class has a canonical representative.

Invariant of T-equivalence

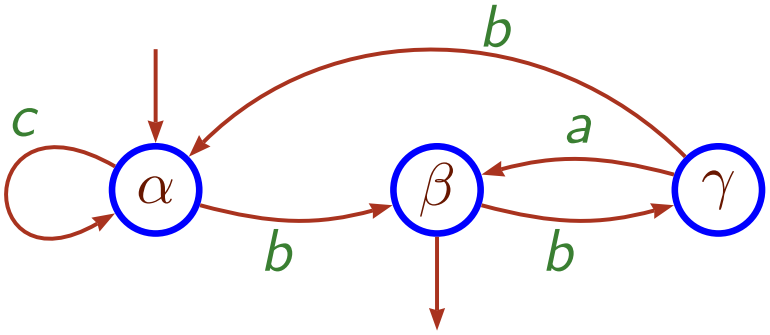
Definition (Automaton Unfolding)

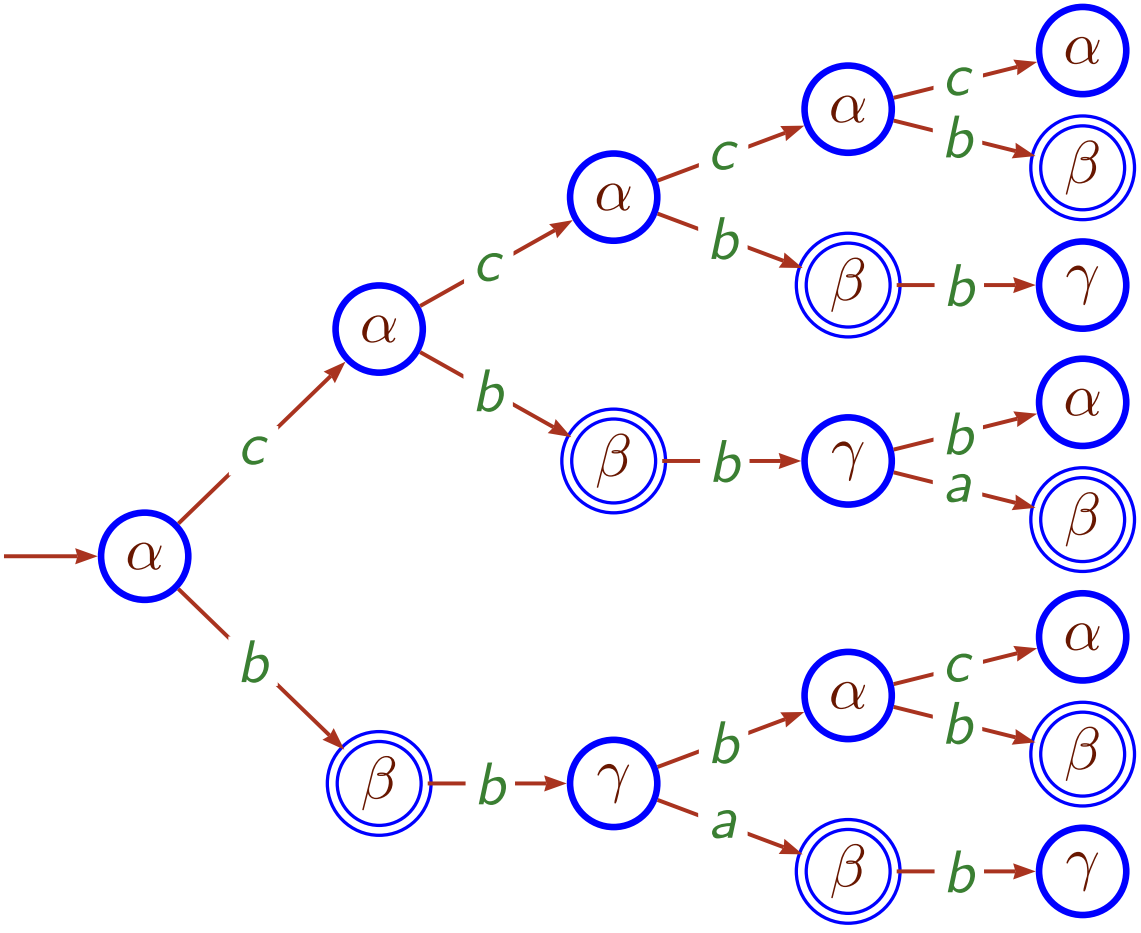
A DFA $\mathcal{A} \xrightarrow{\text{unfolds into}}$ a labelled tree $T_{\mathcal{A}}$.

Runs of $\mathcal{A} \xrightarrow{\text{becomes}}$ branches of $T_{\mathcal{A}}$.

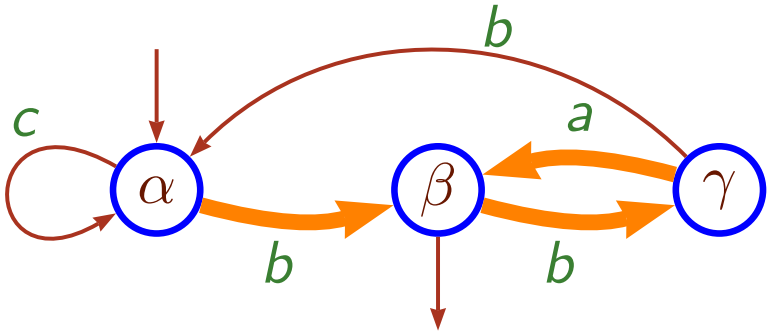
When $\mathcal{A}$ is trim, $T_{\mathcal{A}}$ is also the labelled tree representing $L(\mathcal{A})$.

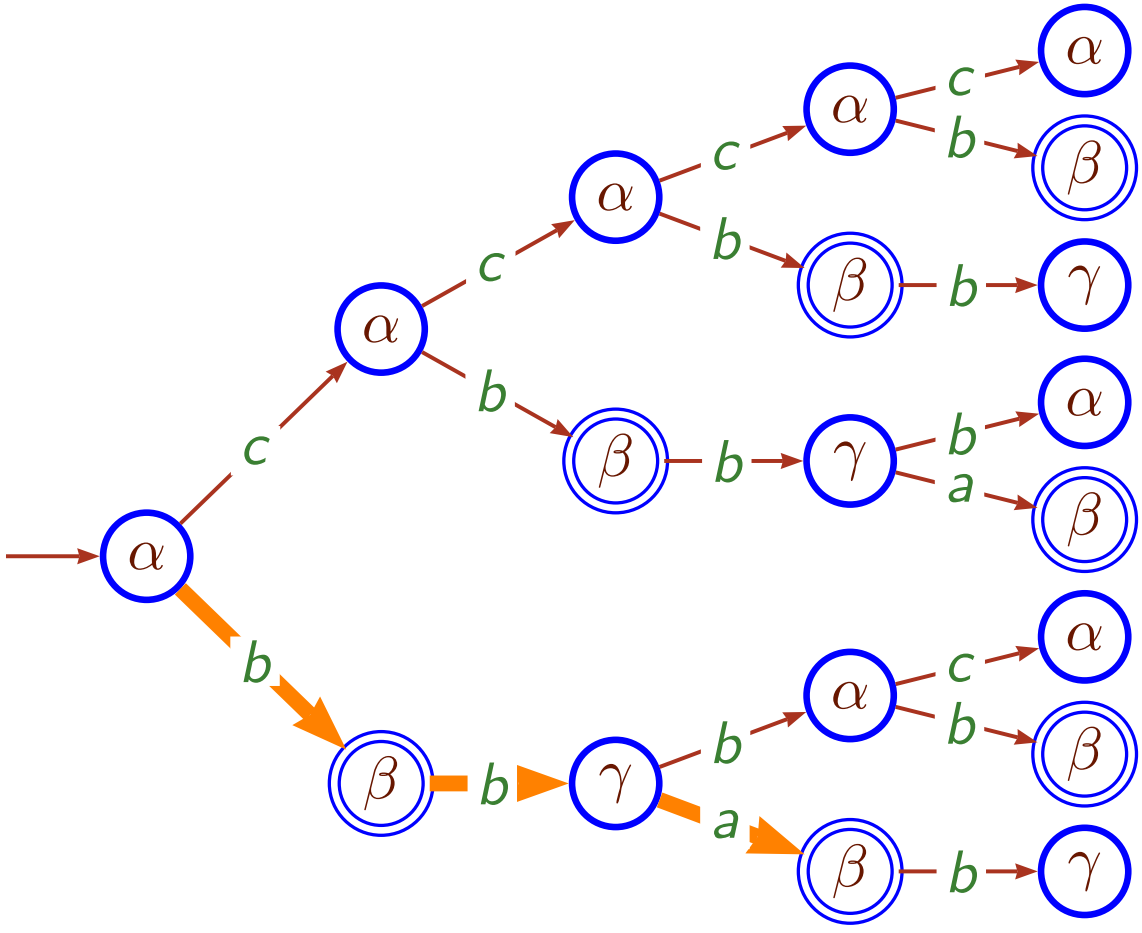
Invariant of T-equivalence – Unfolding of $\mathcal{A}_1$





Invariant of T-equivalence – Unfolding of $\mathcal{A}_1$





Invariant of T-equivalence

Definition (Automaton Unfolding)

A DFA $\mathcal{A} \xrightarrow{\text{unfolds into}}$ a labelled tree $T_{\mathcal{A}}$.

Runs of $\mathcal{A} \xrightarrow{\text{becomes}}$ branches of $T_{\mathcal{A}}$.

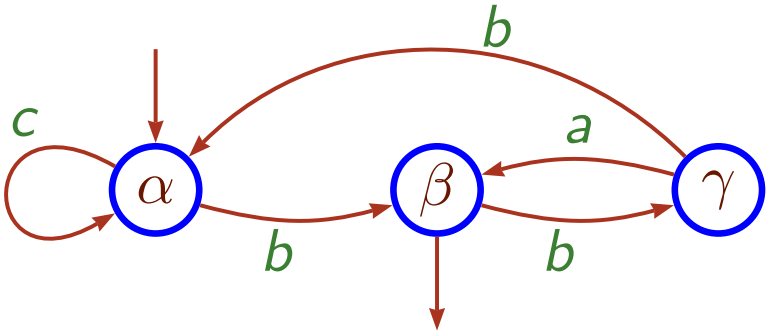
When $\mathcal{A}$ is trim, $T_{\mathcal{A}}$ is also the labelled tree representing $L(\mathcal{A})$.

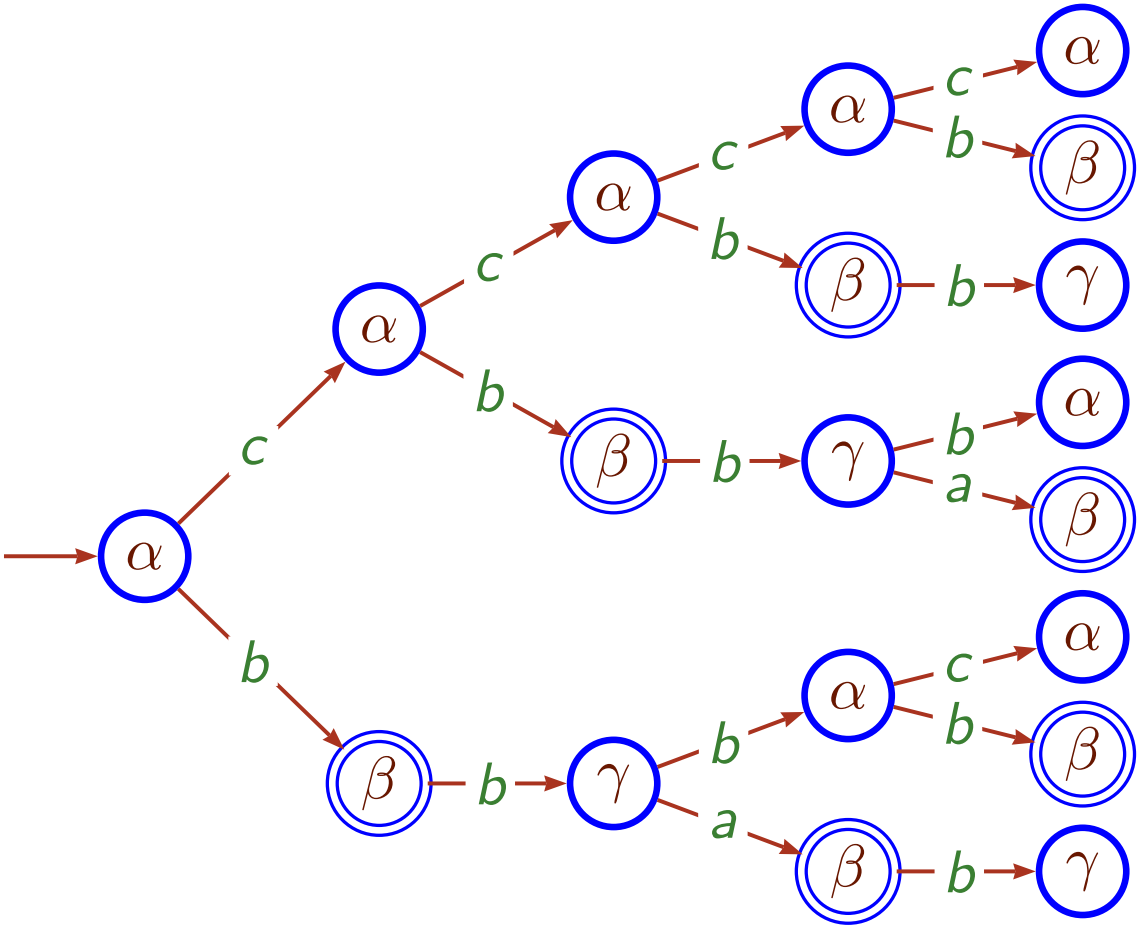
Proposition

$\mathcal{A}$, $\mathcal{B}$: two trim DFA.

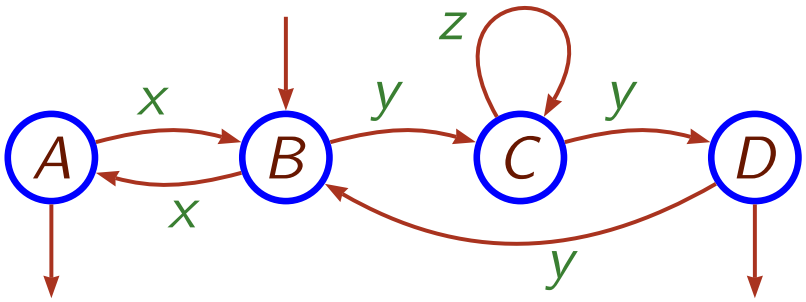
$$\mathcal{A} \overset{T}{\sim} \mathcal{B} \iff T_{\mathcal{A}} \text{ and } T_{\mathcal{B}} \text{ differ only by labelling}$$

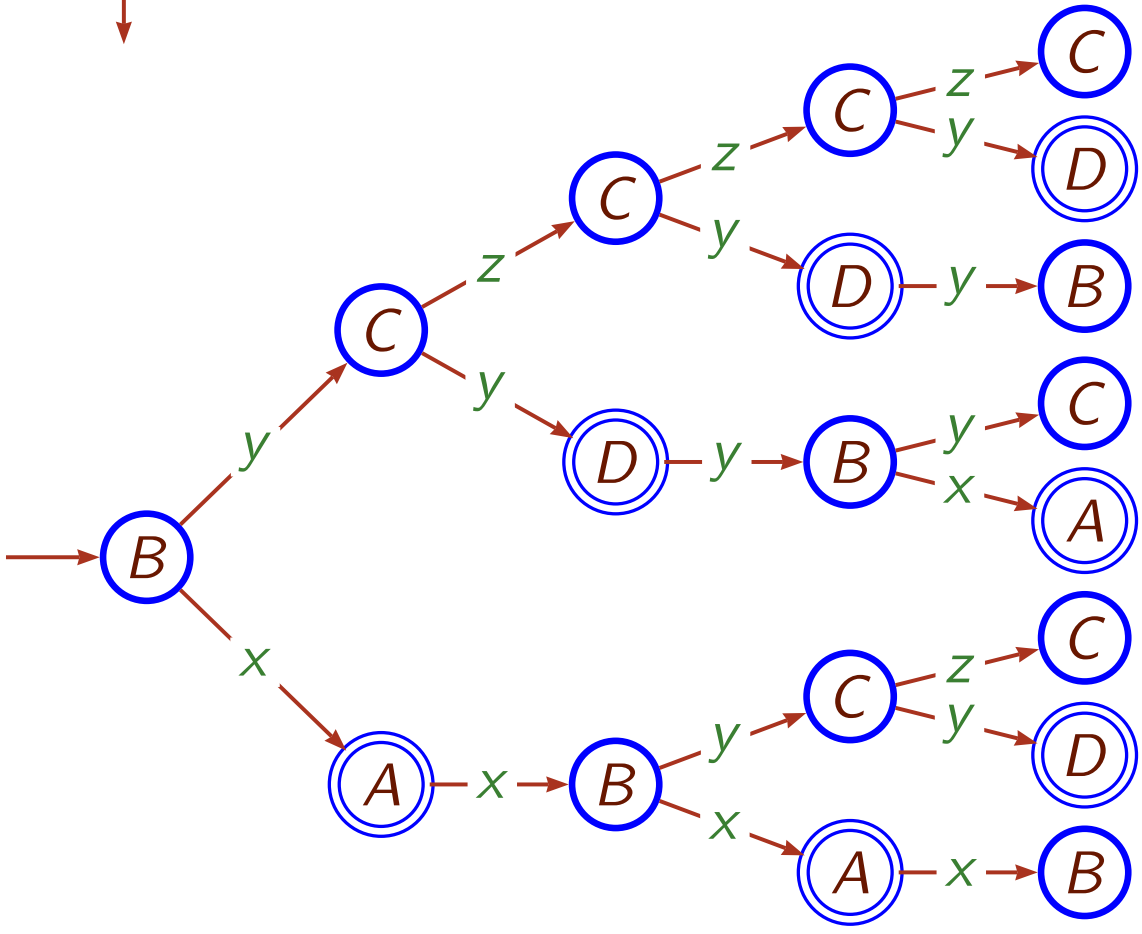
Invariant of T-equivalence – Unfolding of $\mathcal{A}_1$





Invariant of T-equivalence – Unfolding of $\mathcal{A}_2$





Outline

Ordering A^*

Definition (Radix order)

A: an alphabet provided with a total order.

A^* is ordered by the *radix order*:

$$u <_{\text{rad}} v \quad \textit{if} \quad \begin{cases} |u| < |v| \\ \quad \textit{or} \\ |u| = |v| \text{ and } u \text{ is lexicograph. smaller than } v \end{cases}$$

Example : $\epsilon <_{\text{rad}} 20 <_{\text{rad}} 21 <_{\text{rad}} 100$

Abstract Regular Numeration Systems

Definition (ARNS, Lecomte-Rigo 2001)

L: a regular language over A.

An integer n is represented by the word $\langle n \rangle_L$ (in the ARNS L) where $\langle n \rangle_L$ is the $(n+1)$-th word of L (in the radix order).

Definition (ARNS, Lecomte-Rigo 2001)

L: a regular language over A.

An integer n is represented by the word $\langle n \rangle_L$ (in the ARNS L) where $\langle n \rangle_L$ is the $(n+1)$-th word of L (in the radix order).

Example

$L_3 = a^* b^*$
Enumeration of L_3 according to the radix order:

ϵ	a	b	aa	ab	bb	aaa	aab	abb	bbb	$\cdots$
$\langle 0 \rangle$	$\langle 1 \rangle$	$\langle 2 \rangle$	$\langle 3 \rangle$	$\langle 4 \rangle$	$\langle 5 \rangle$	$\langle 6 \rangle$	$\langle 7 \rangle$	$\langle 8 \rangle$	$\langle 9 \rangle$	$\cdots$

Theorem

L, K: two ARNS accepted by two DFA $\mathcal{A}$ and $\mathcal{B}$

$\mathcal{A} \overset{T}{\sim} \mathcal{B} \implies \exists$ Mealy Machine† $\mathcal{A} \boxtimes \mathcal{B}$ realising $\langle n \rangle_L \mapsto \langle n \rangle_K$

† Letter-to-letter and pure sequential transducer.

Theorem

L, K: two ARNS accepted by two DFA $\mathcal{A}$ and $\mathcal{B}$

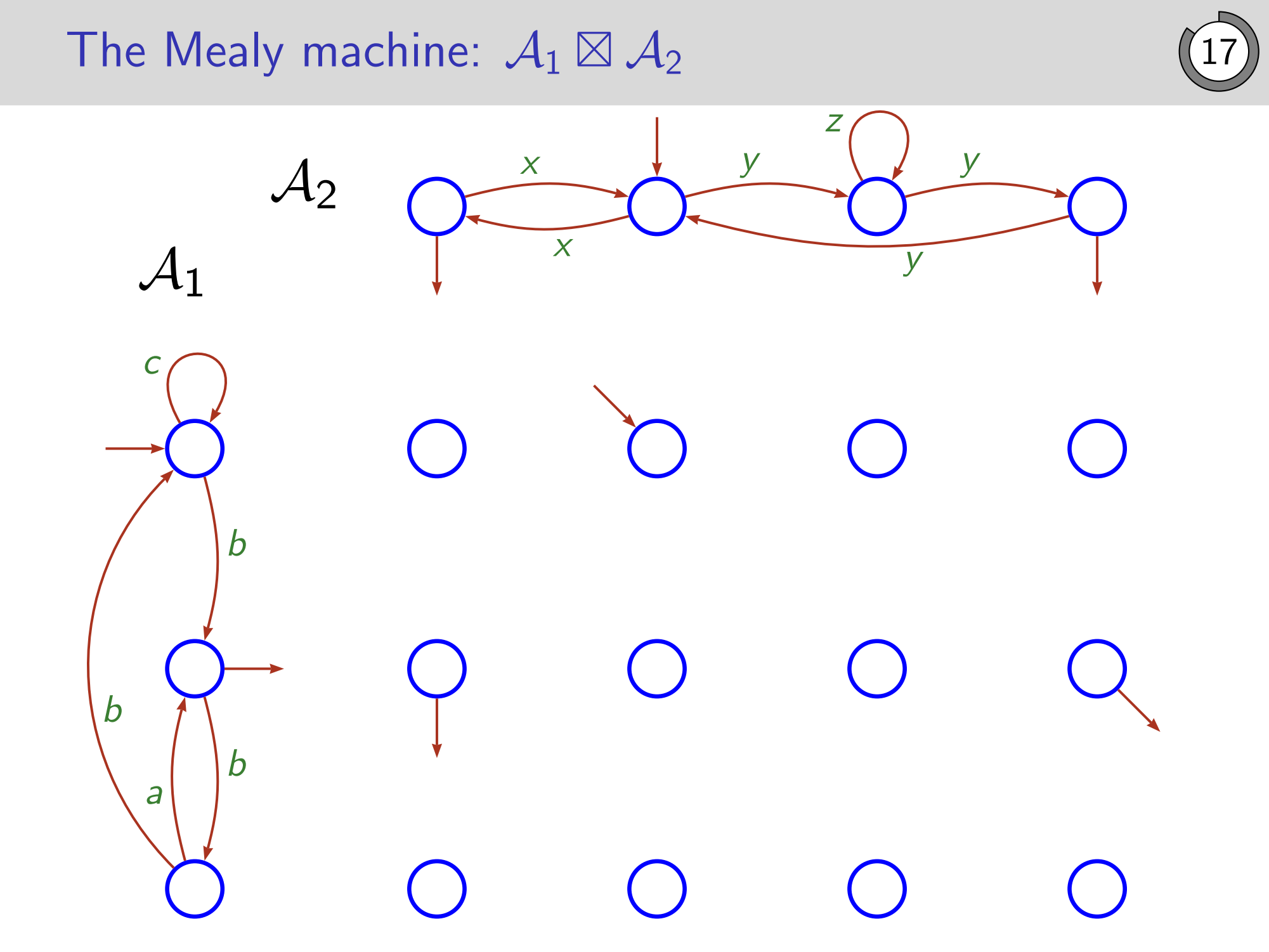
$$\mathcal{A} \overset{T}{\sim} \mathcal{B} \implies \exists \text{ Mealy Machine}^{\dagger}\ \mathcal{A} \boxtimes \mathcal{B} \text{ realising } \langle n \rangle_L \mapsto \langle n \rangle_K$$

$\mathcal{A} \boxtimes \mathcal{B}$ is basically $\mathrm{Ired}(\mathcal{A}) \times \mathrm{Ired}(\mathcal{B})$ where
$(p, p') \xrightarrow{i} (q, q')$ is relabelled by $(p, p') \xrightarrow{a|x} (q, q')$ if

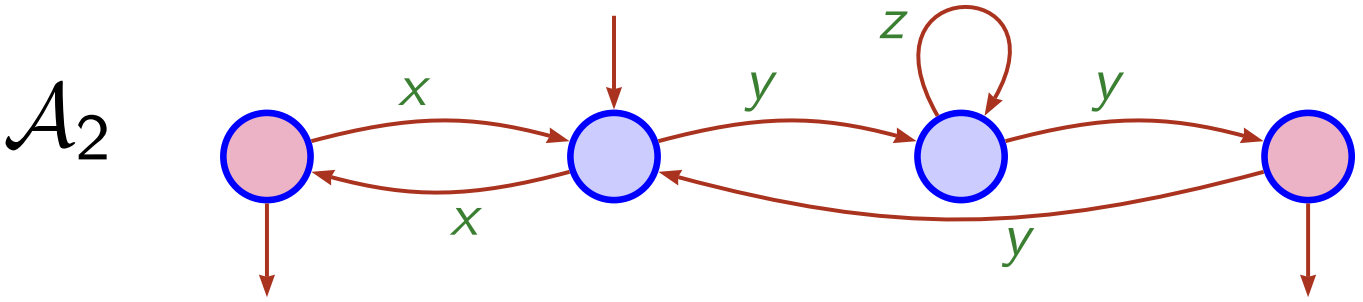
- $p \xrightarrow{a} q$ is the $(i+1)$-th transition going out of p in $\mathcal{A}$.
- $p' \xrightarrow{x} q'$ is the $(i+1)$-th transition going out of p' in $\mathcal{B}$.

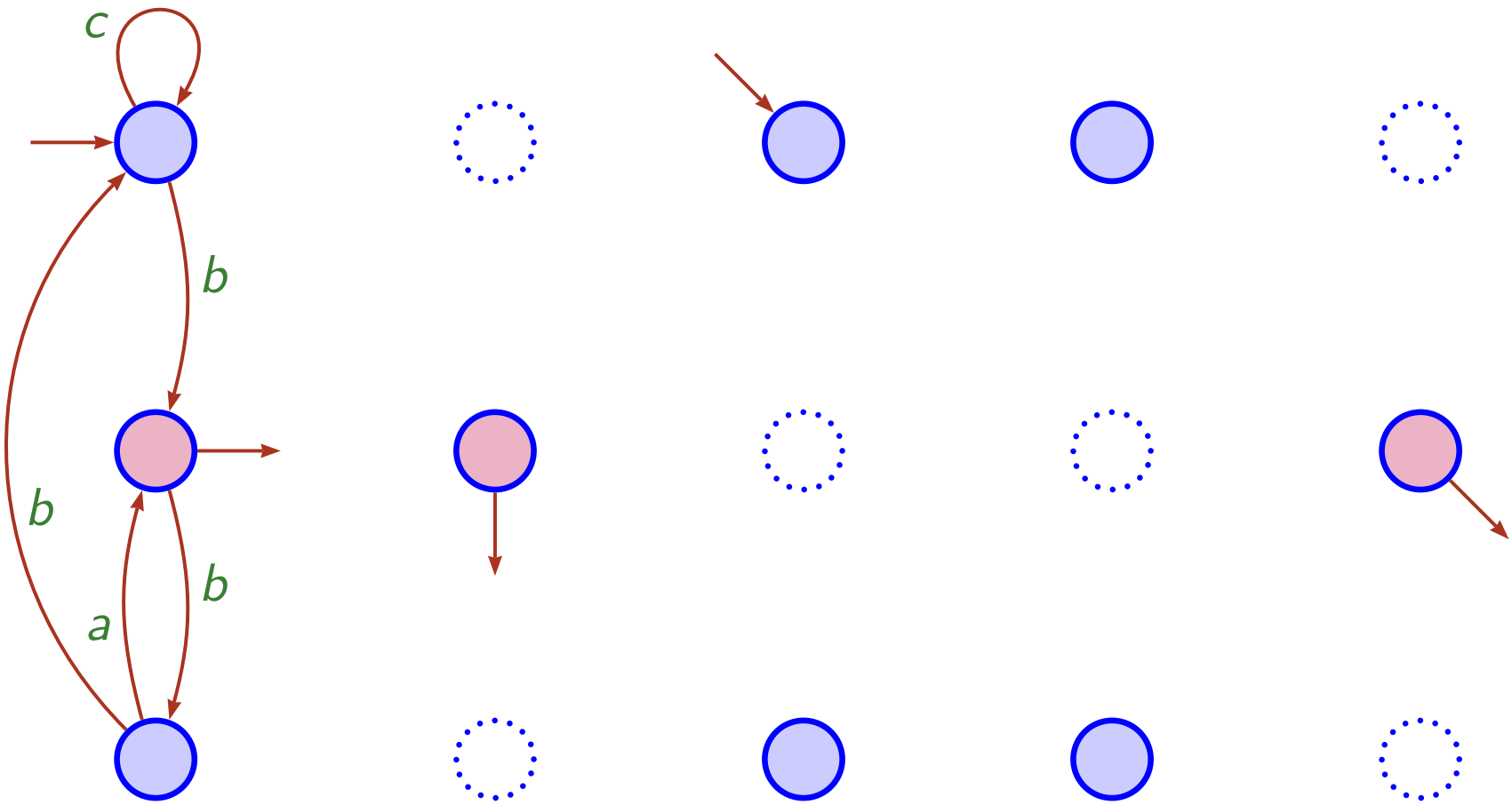
† Letter-to-letter and pure sequential transducer.

The Mealy machine: $\mathcal{A}_1 \boxtimes \mathcal{A}_2$

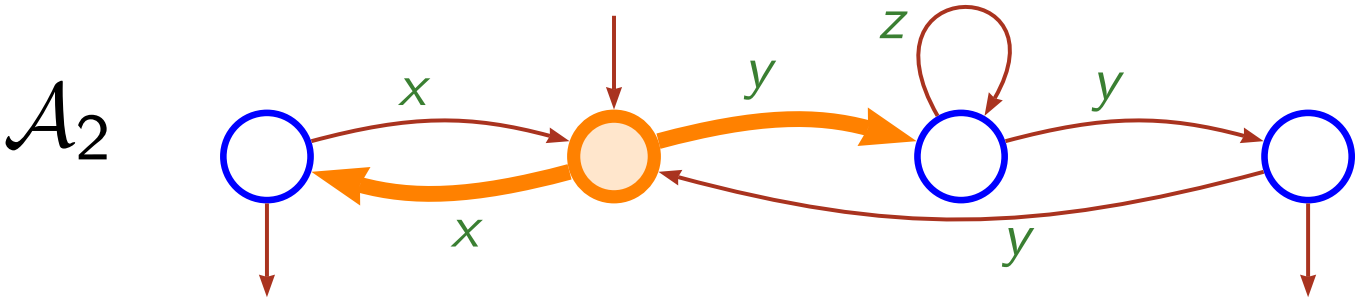


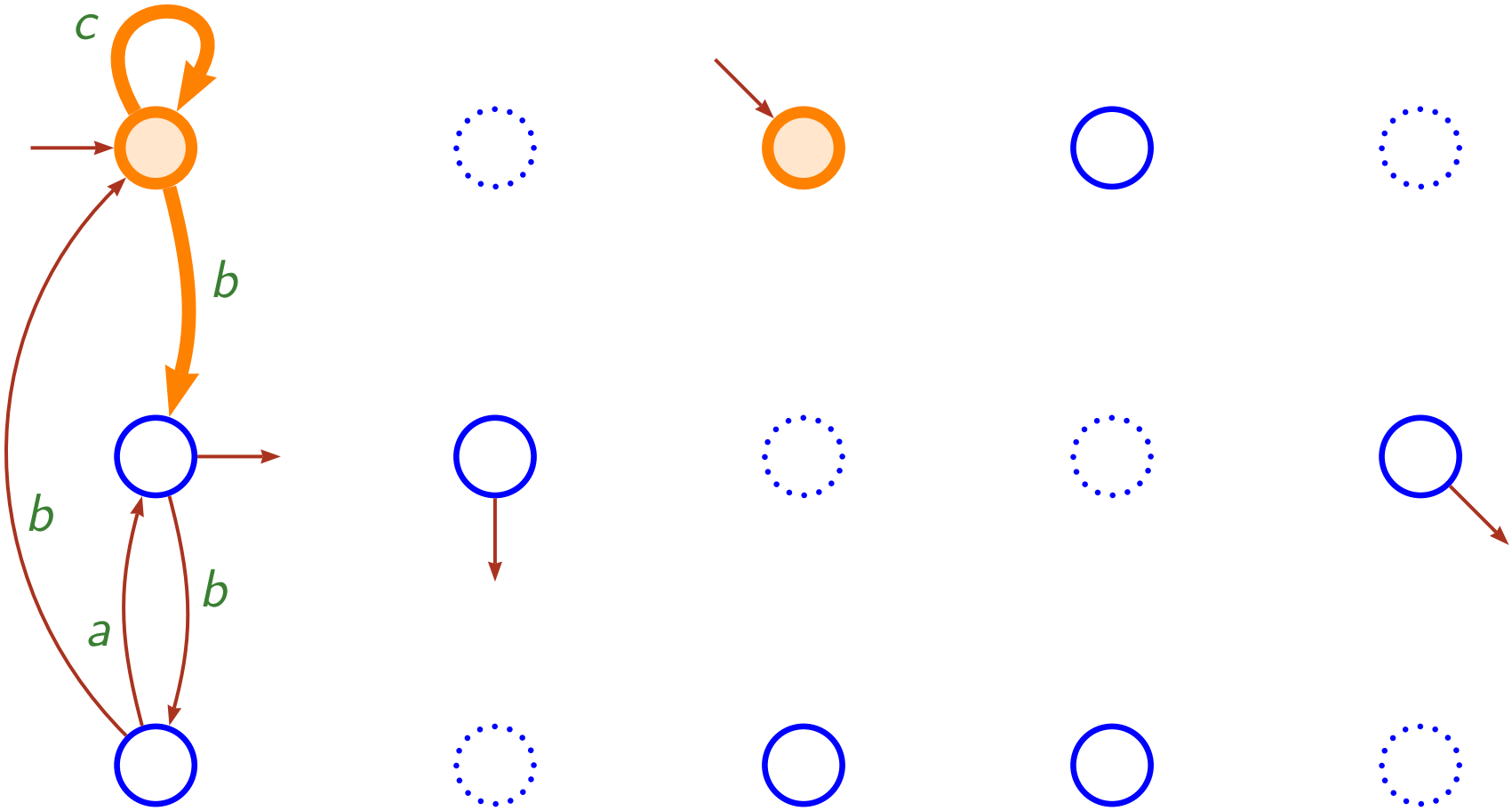
The Mealy machine: $\mathcal{A}_1 \boxtimes \mathcal{A}_2$



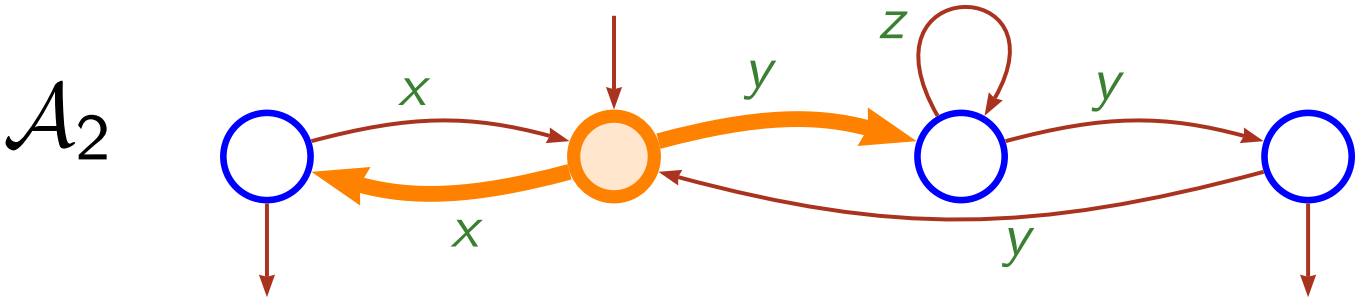


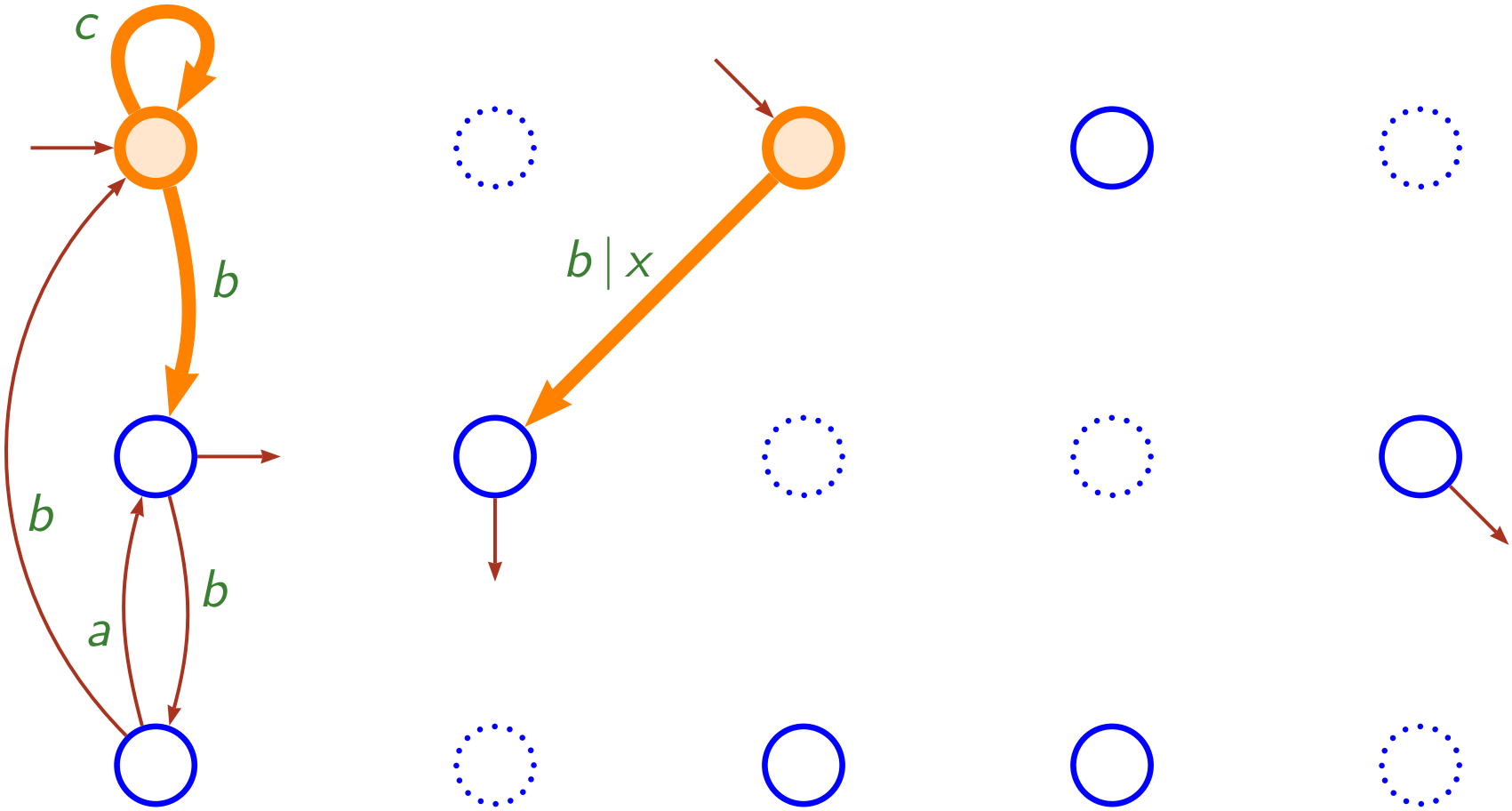
The Mealy machine: $\mathcal{A}_1 \boxtimes \mathcal{A}_2$



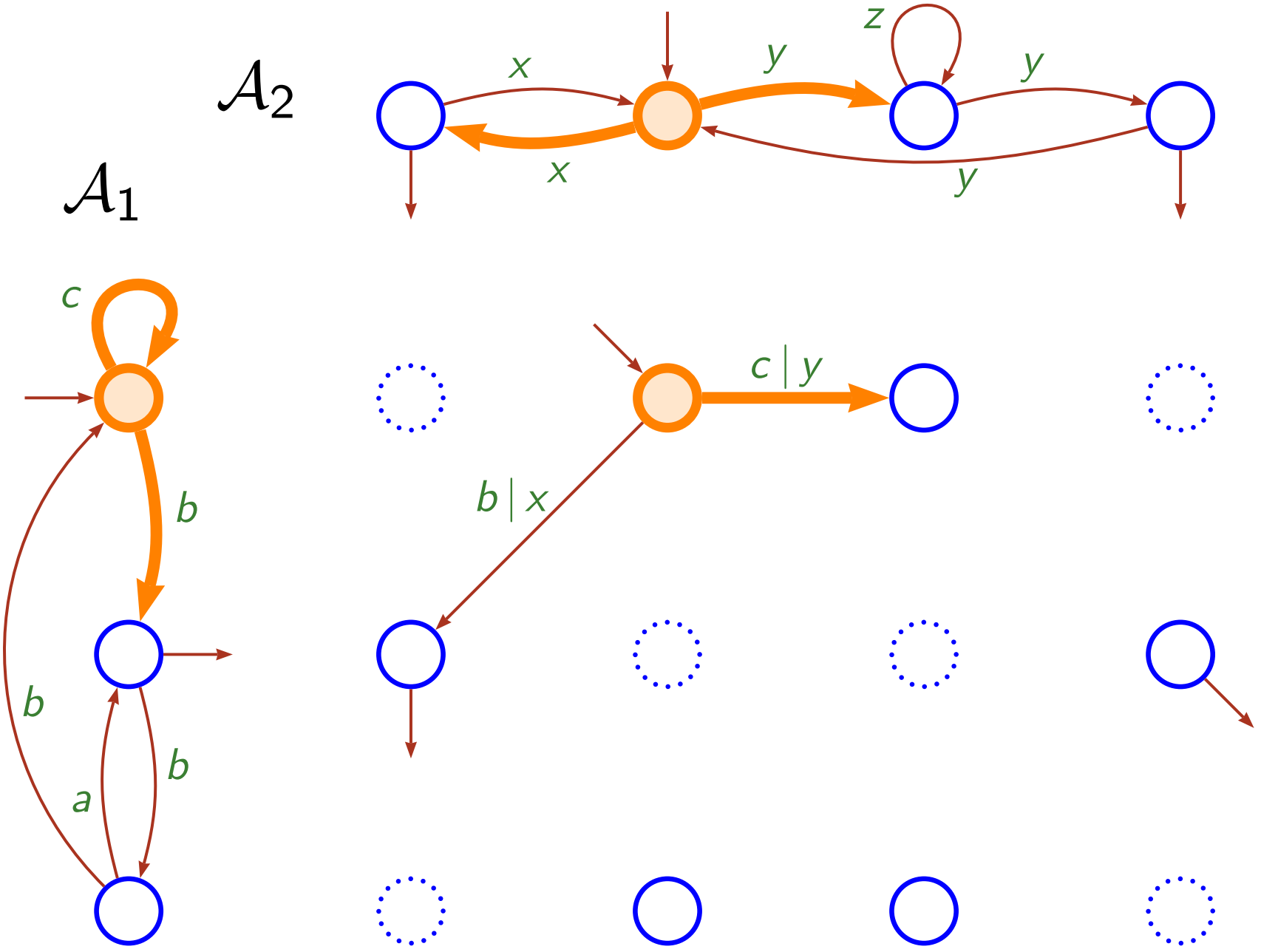


The Mealy machine: $\mathcal{A}_1 \boxtimes \mathcal{A}_2$

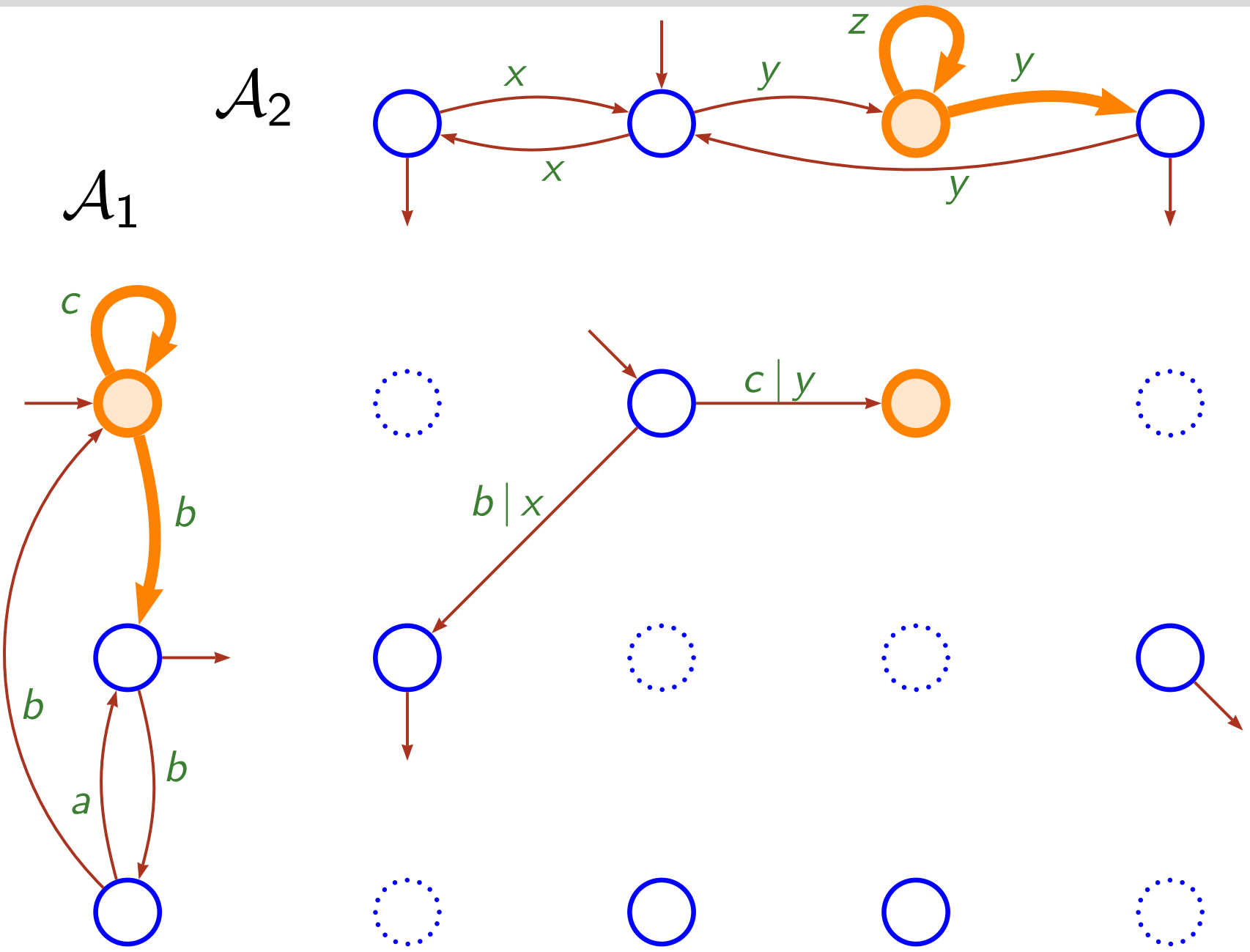




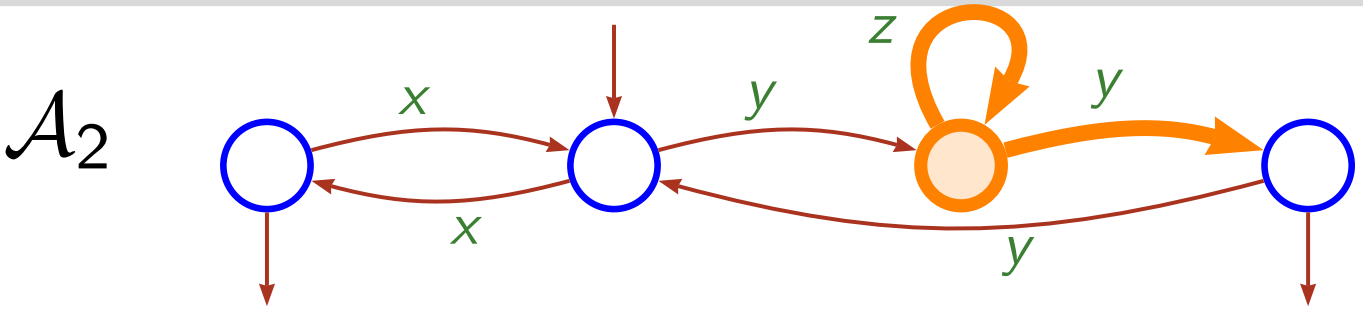
The Mealy machine: $\mathcal{A}_1 \boxtimes \mathcal{A}_2$

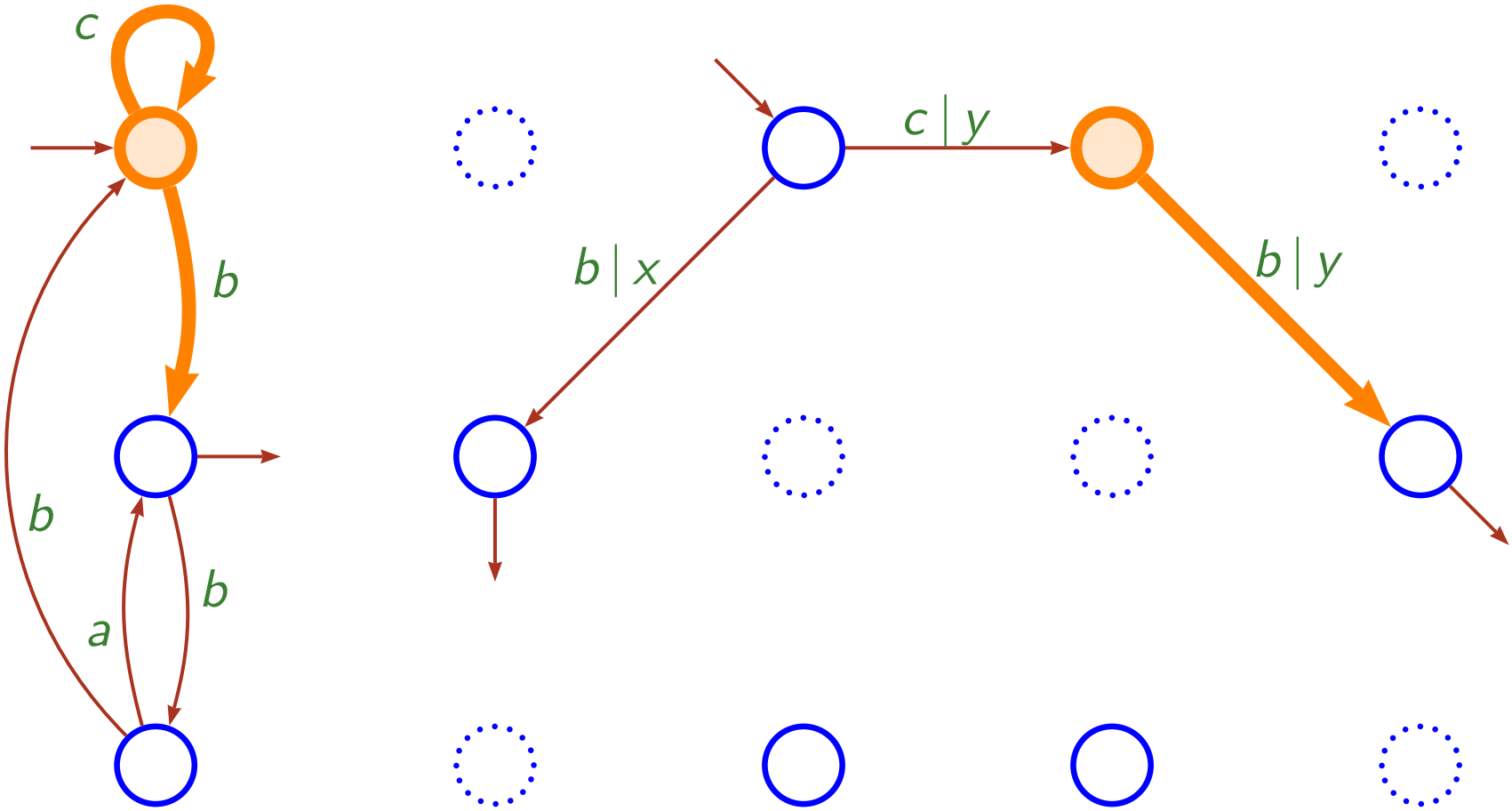


The Mealy machine: $\mathcal{A}_1 \boxtimes \mathcal{A}_2$

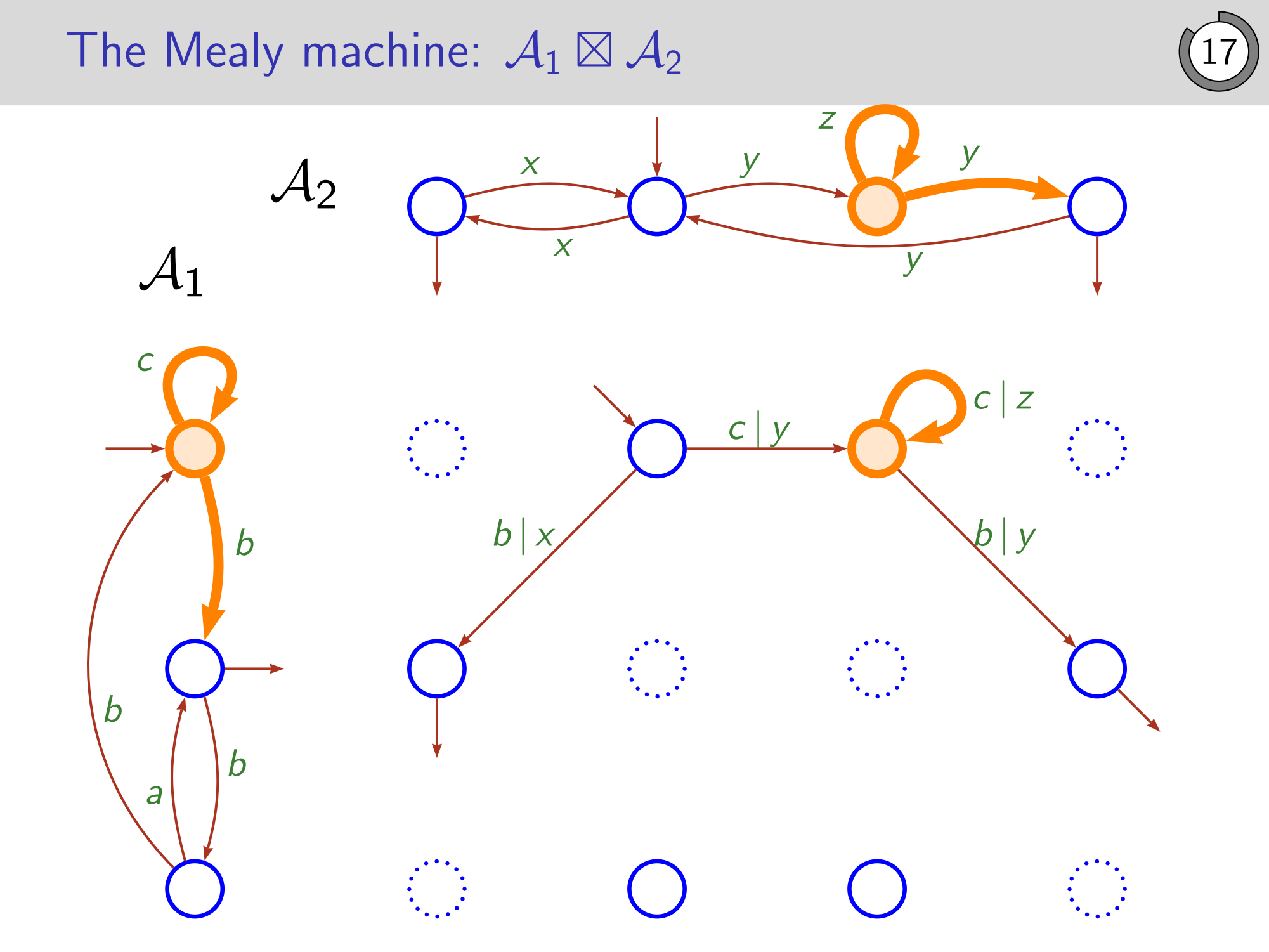


The Mealy machine: $\mathcal{A}_1 \boxtimes \mathcal{A}_2$

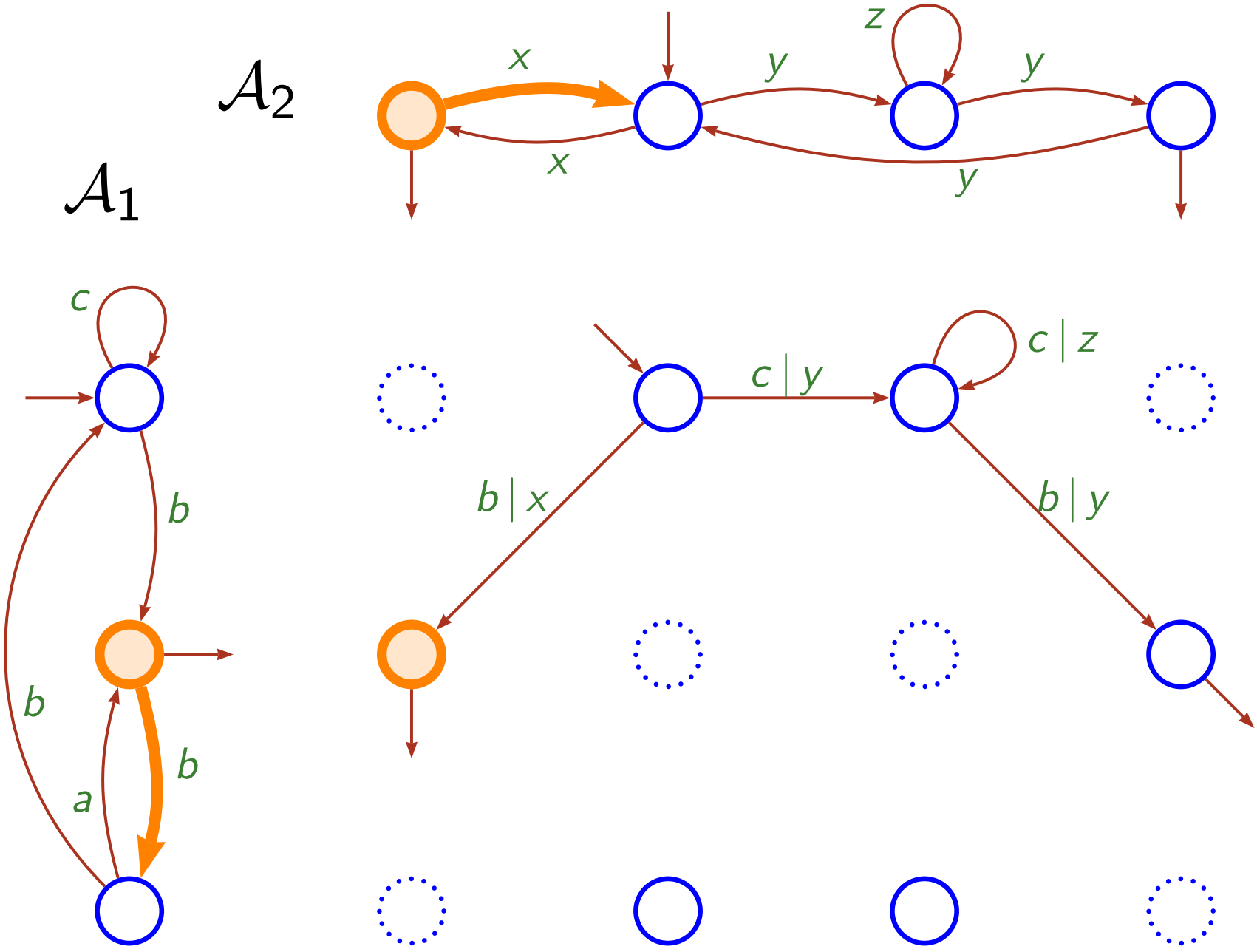




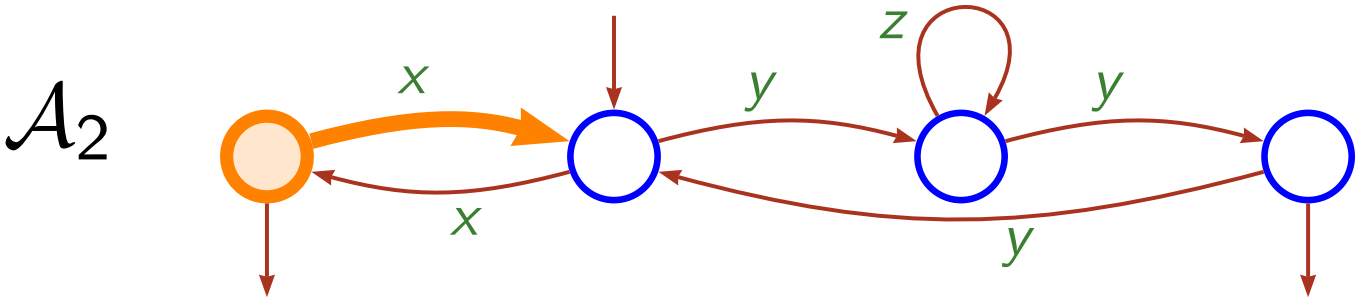
The Mealy machine: $\mathcal{A}_1 \boxtimes \mathcal{A}_2$

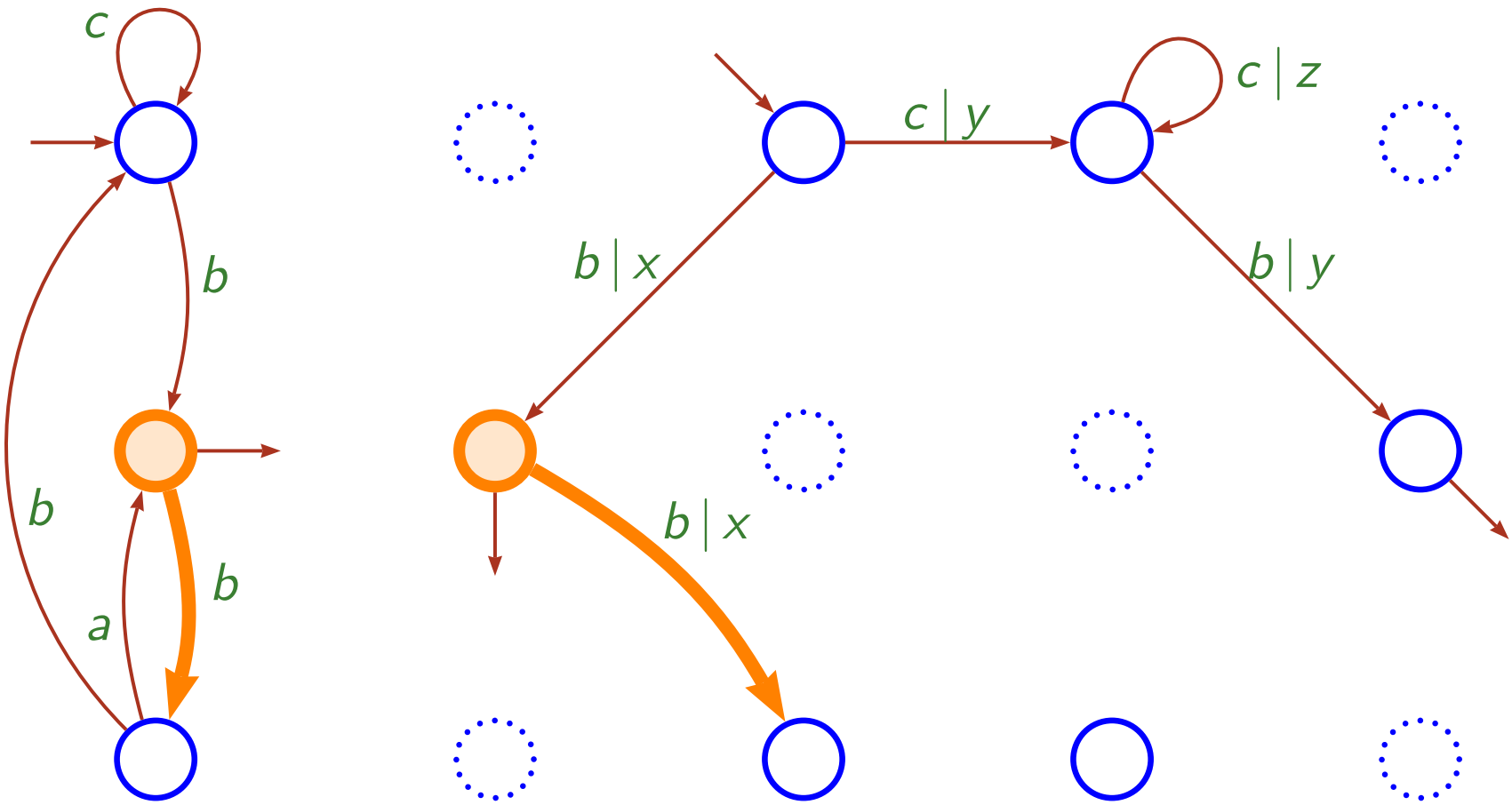


The Mealy machine: $\mathcal{A}_1 \boxtimes \mathcal{A}_2$

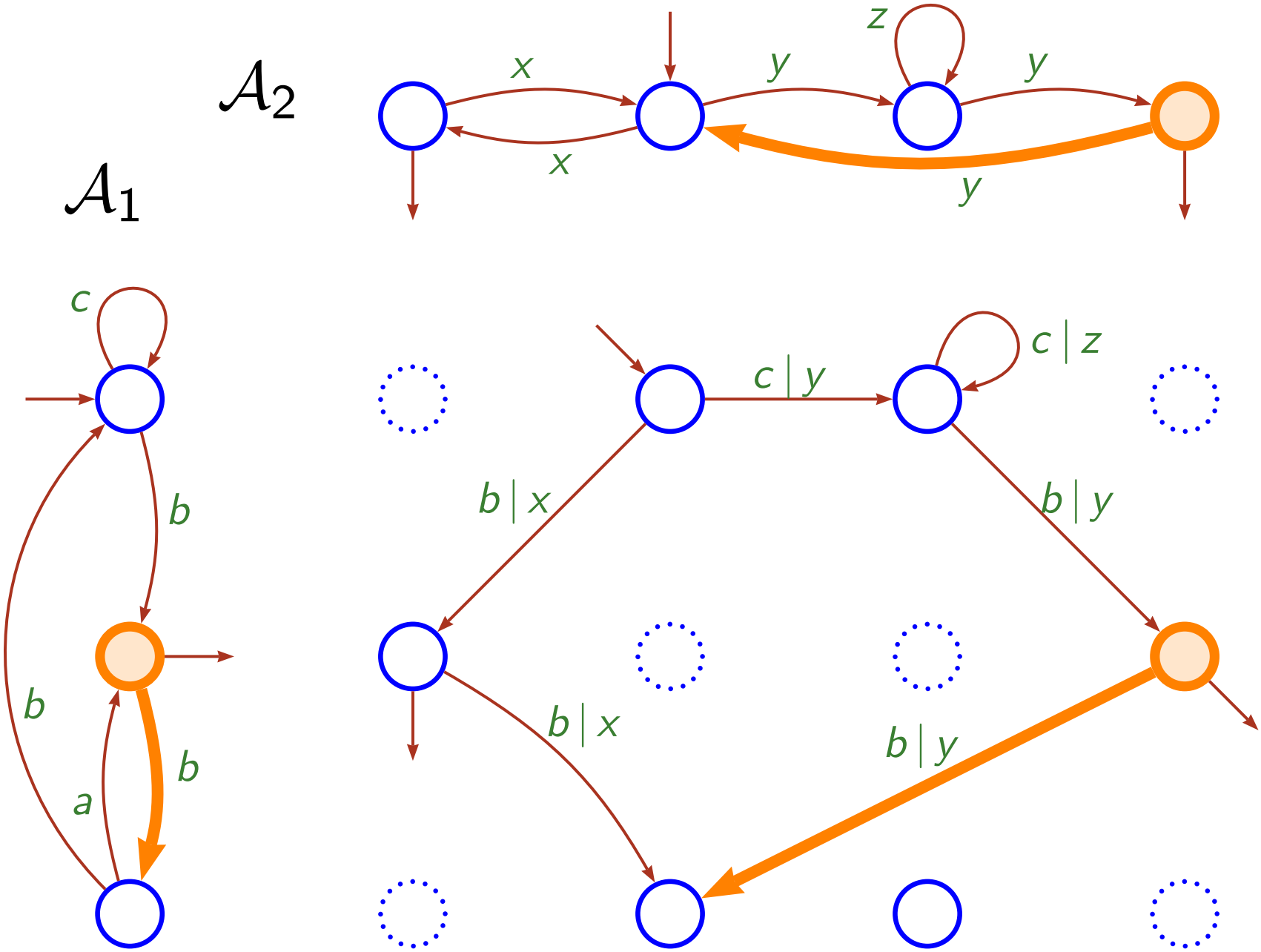


The Mealy machine: $\mathcal{A}_1 \boxtimes \mathcal{A}_2$

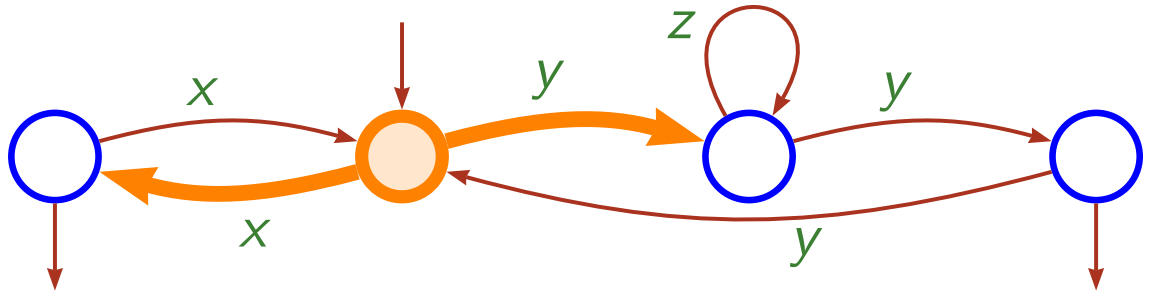


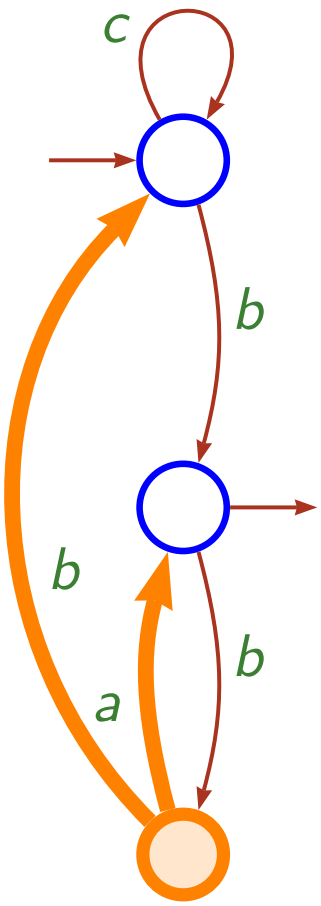


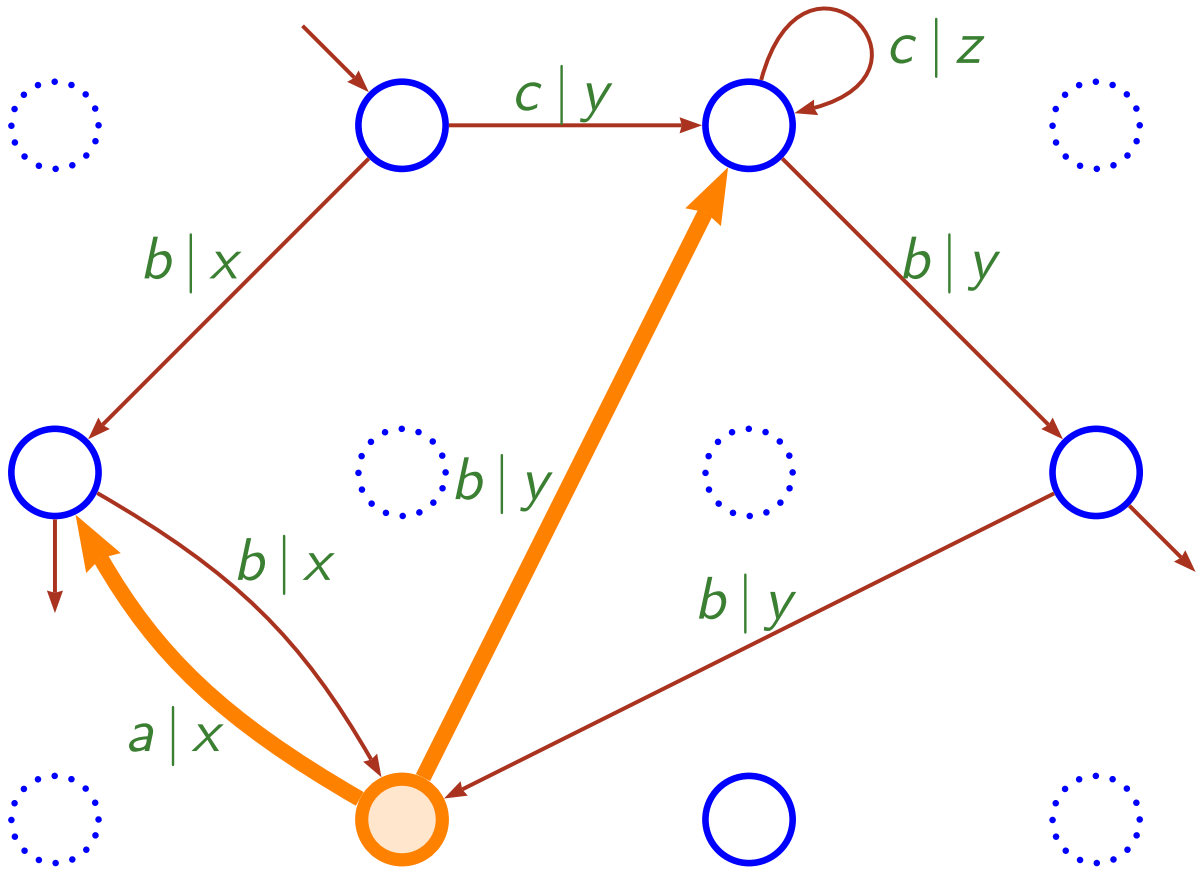
The Mealy machine: $\mathcal{A}_1 \boxtimes \mathcal{A}_2$



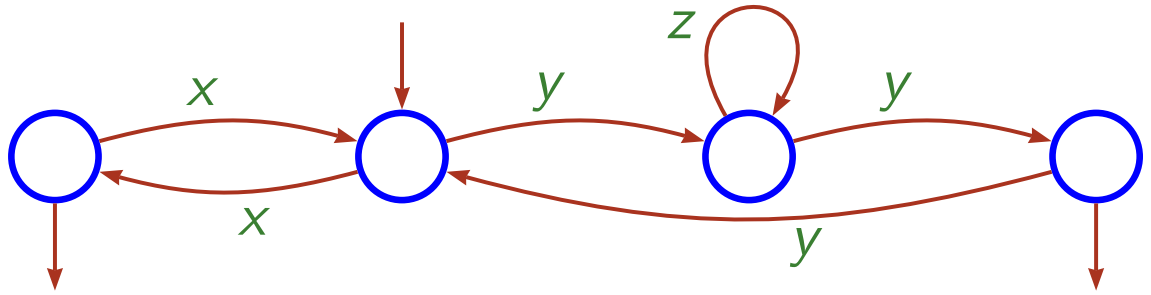
The Mealy machine: $\mathcal{A}_1 \boxtimes \mathcal{A}_2$

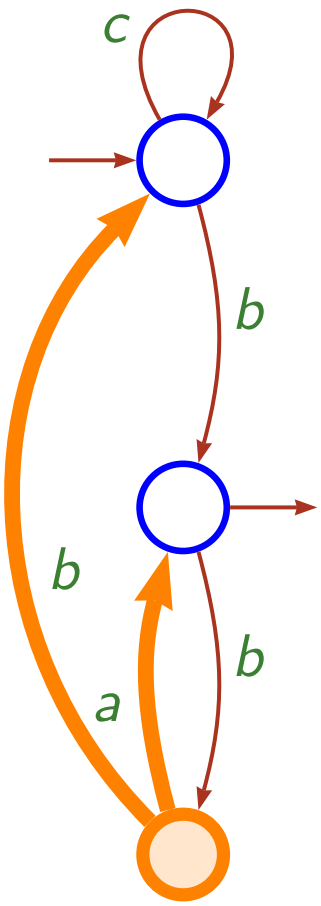


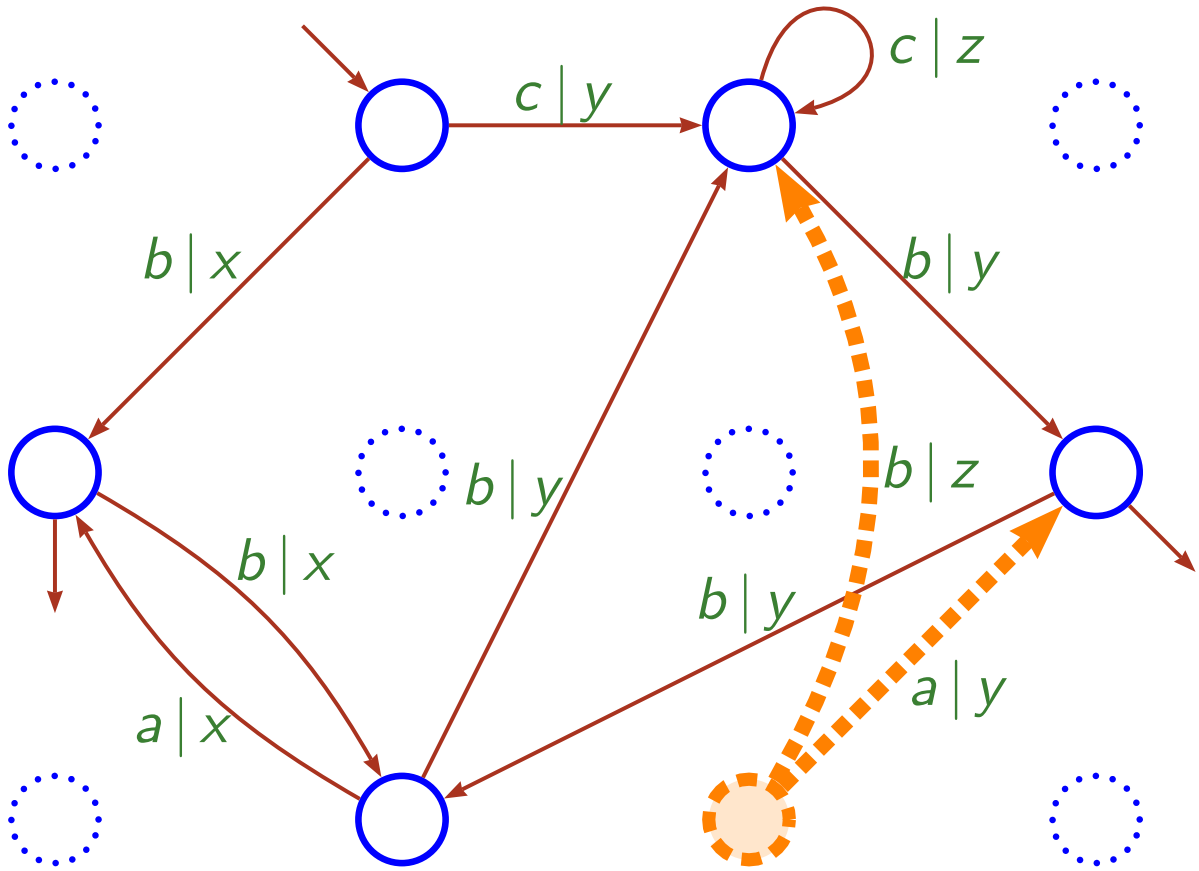




The Mealy machine: $\mathcal{A}_1 \boxtimes \mathcal{A}_2$







Two words on U-systems

Basis: $U = u_0, u_1, u_2, \ldots, u_i, \ldots$

Evaluation function: $\pi(a_k \cdots a_2 a_1 a_0) = \sum_{i=0}^{k} a_i \, u_i$

Representation of n: $\langle n \rangle_U$ is computed by the "greedy algorithm"

Two words on U-systems

Basis: $U = u_0, u_1, u_2, \ldots, u_i, \ldots$

Evaluation function: $\pi(a_k \cdots a_2 a_1 a_0) = \sum_{i=0}^{k} a_i u_i$

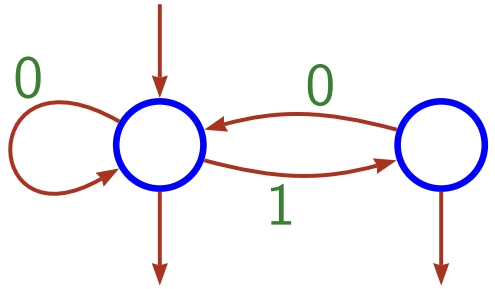
Representation of n: $\langle n \rangle_U$ is computed by the "greedy algorithm"

Example: Fibonacci Numeration System

$F = 1, 2, 3, 5, 8, 13, 21, \cdots$

$F_{(n+2)} = F_{(n+1)} + F_n$

$\langle 30 \rangle_F = 1\,0\,1\,0\,0\,0\,1 \quad (30 = 21 + 8 + 1 = F_6 + F_4 + F_0)$



Automaton accepting $0^* L(F)$

Theorem

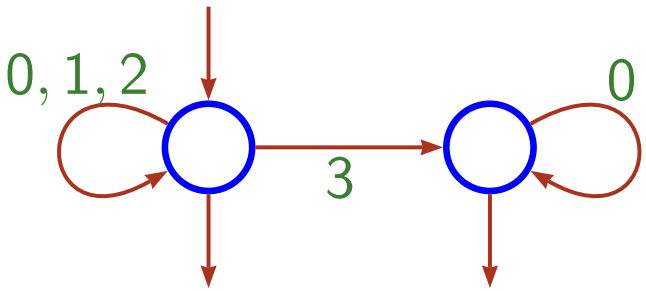
U: a U-system.
$L(U) = \{\langle n \rangle \mid n \in \mathbb{N}\}$.
$\mathcal{A}$: the minimal DFA accepting $0^* L(U)$.

$\mathcal{A}$ is surminimal.

Theorem

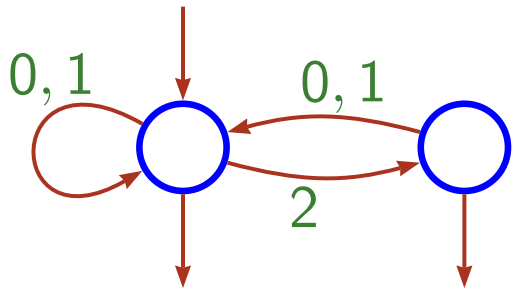
U: a U-system.
$L(U) = \{\langle n \rangle \mid n \in \mathbb{N}\}$.
$\mathcal{A}$: the minimal DFA accepting $0^* L(U)$.

$\mathcal{A}$ is surminimal.



$H = 1,\ 4,\ 13,\ 40,\ 121\ \cdots$

$H_{(n+2)} = 4\,H_{(n+1)} - 3\,H_n$



$D = 1,\ 3,\ 8,\ 22,\ 60\ \cdots$

$D_{(n+2)} = 2\,D_{(n+1)} + 2\,D_n$

Conclusion

Surminimal vs Minimal automaton

- Fewer states
- Fewer letters
- Preserves underlying tree instead of the accepted language.

Conclusion

Surminimal vs Minimal automaton

- Fewer states
- Fewer letters
- Preserves underlying tree instead of the accepted language.

Numeration Systems