

Hard combinatorics

Denis Béchet, Université de Nantes / CNRS LINA

Sylvain Lippi, Université de Nice / CNRS I3S

Hard interaction

- A model of computation as...

- Turing machines
- The λ -calculus
- Cellular automata

- but...

Hard interaction

- A model of computation as...

- Turing machines
- The λ -calculus
- Cellular automata

- but...

- **unlike** Turing machines, computation is distributed
- **unlike** the λ -calculus, reduction steps are local
- **unlike** cellular automata, computation is asynchronous

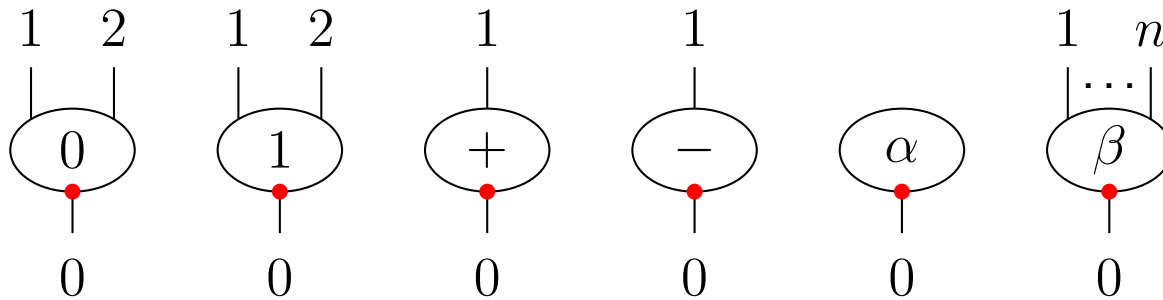
What is hard interaction ?

- A set Σ of **symbols** and their arities. $(O, 2), (1, 2), (+, 1), (-, 1), (\alpha, 0), (\beta, n)$

What is hard interaction ?

- A set Σ of **symbols** and their arities. $(O, 2), (1, 2), (+, 1), (-, 1), (\alpha, 0), (\beta, n)$

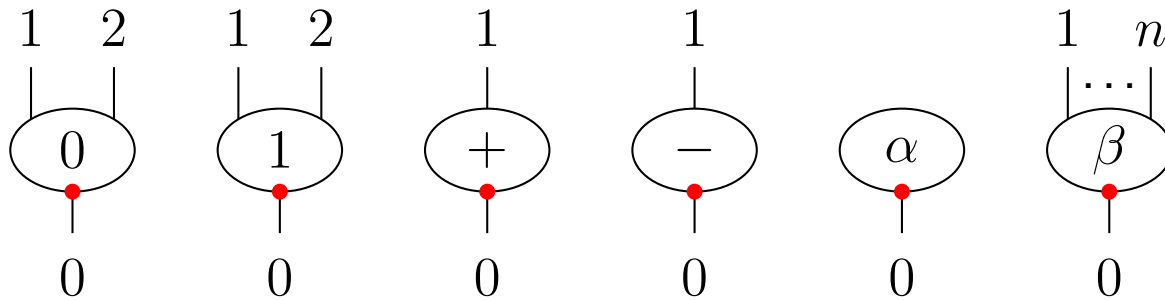
- Occurrences of symbols are **Cells**.



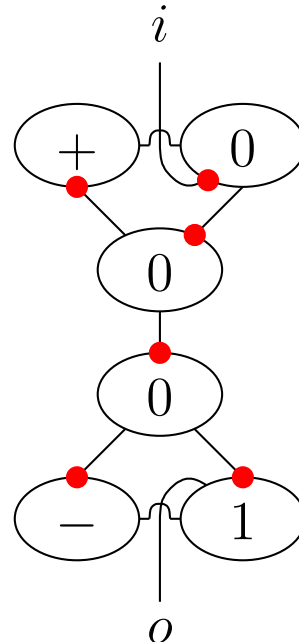
What is hard interaction ?

- A set Σ of **symbols** and their arities. $(0, 2), (1, 2), (+, 1), (-, 1), (\alpha, 0), (\beta, n)$

- Occurrences of symbols are **Cells**.



- A **net** is a set of cells and free ports where all the ports are connected pairwise.



What is a hard system ?

Example :

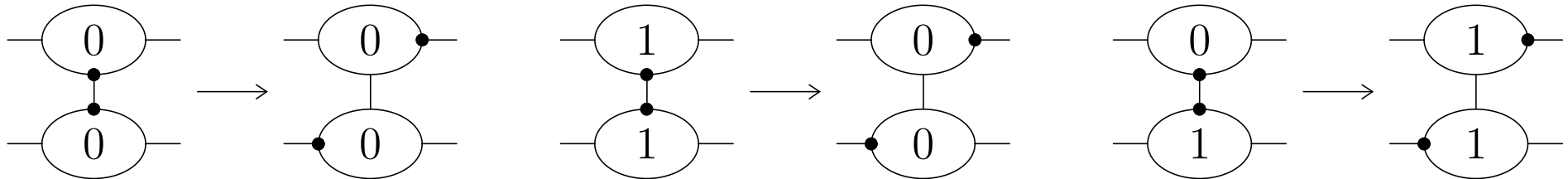
1. A set of symbols $\Sigma = \{ (0, 2) , (1, 2) \}$

What is a hard system ?

Example :

1. A set of symbols $\Sigma = \{ (0, 2) , (1, 2) \}$

2. A set R of **interaction rules** (one for each pair of symbols **at most**)



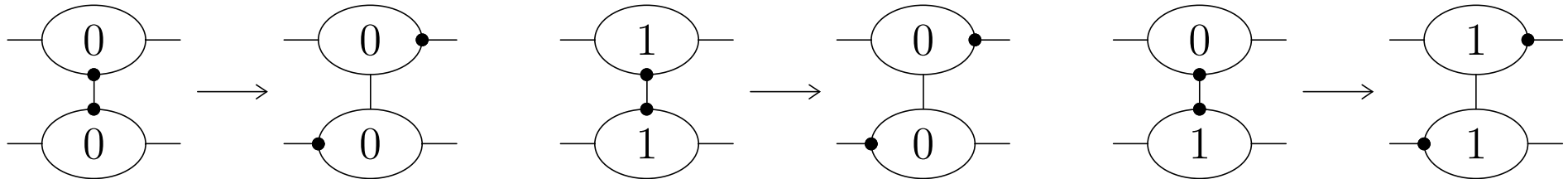
- The left member is an active pair.
- The geometry of the net is invariant.

What is a hard system ?

Example :

1. A set of symbols $\Sigma = \{ (0, 2) , (1, 2) \}$

2. A set R of **interaction rules** (one for each pair of symbols **at most**)



- The left member is an active pair.
- The geometry of the net is invariant.

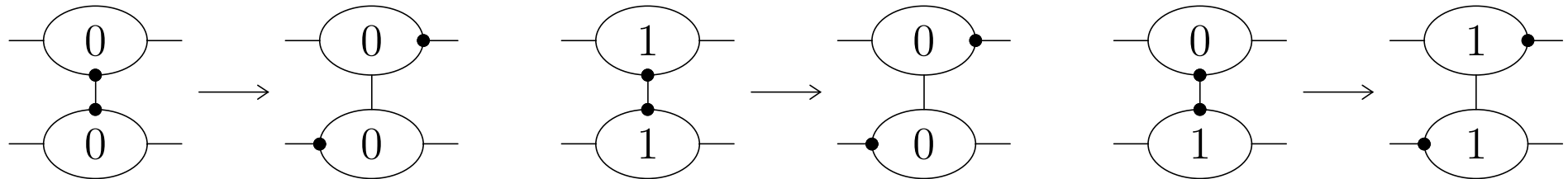
$$\begin{aligned}
 R : \Sigma \times \Sigma &\rightarrow \Sigma \times \mathbb{N} \\
 (0, 0) &\mapsto (0, 1) \\
 (1, 1) &\mapsto (0, 1) \\
 (0, 1) &\mapsto (1, 1) \\
 (1, 0) &\mapsto (1, 1)
 \end{aligned}$$

What is a hard system ?

Example :

1. A set of symbols $\Sigma = \{ (0, 2) , (1, 2) \}$

2. A set R of **interaction rules** (one for each pair of symbols **at most**)



- The left member is an active pair.
- The geometry of the net is invariant.

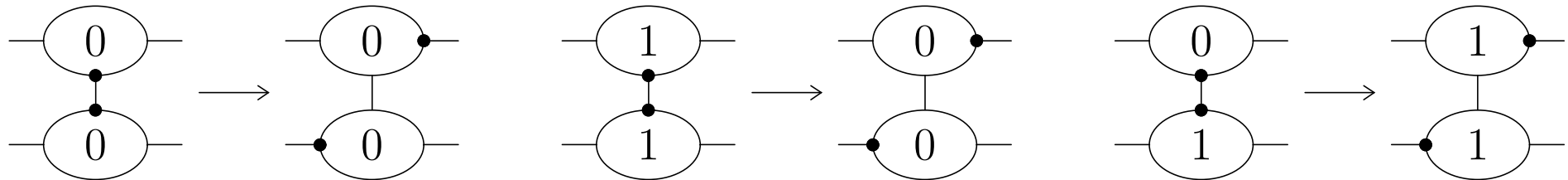
$$R : \Sigma \times \Sigma \rightarrow \Sigma \times \mathbb{N}$$

$$(x, y) \mapsto (x + y, 1) \quad \text{where} \quad x, y \in \{0, 1\}$$

What is a hard system ?

Example :

1. A set of symbols $\Sigma = \{ (0, 2) , (1, 2) \}$
2. A set R of **interaction rules** (one for each pair of symbols **at most**)



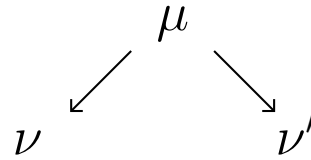
- The left member is an active pair.
- The geometry of the net is invariant.

$$R : \Sigma \times \Sigma \rightarrow \Sigma \times \mathbb{N}$$

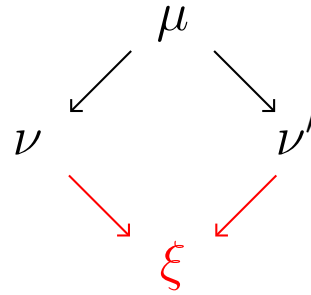
$$(x, y) \mapsto (x + y, 1) \quad \text{where } x, y \in \{0, 1\}$$

Remark. A Turing Machine can be translated into a hard system.

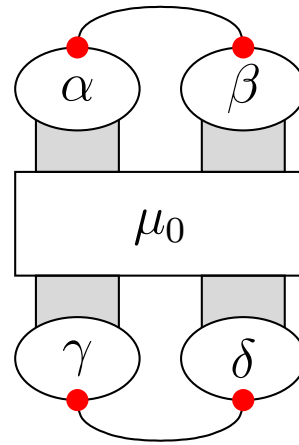
Strong confluence property



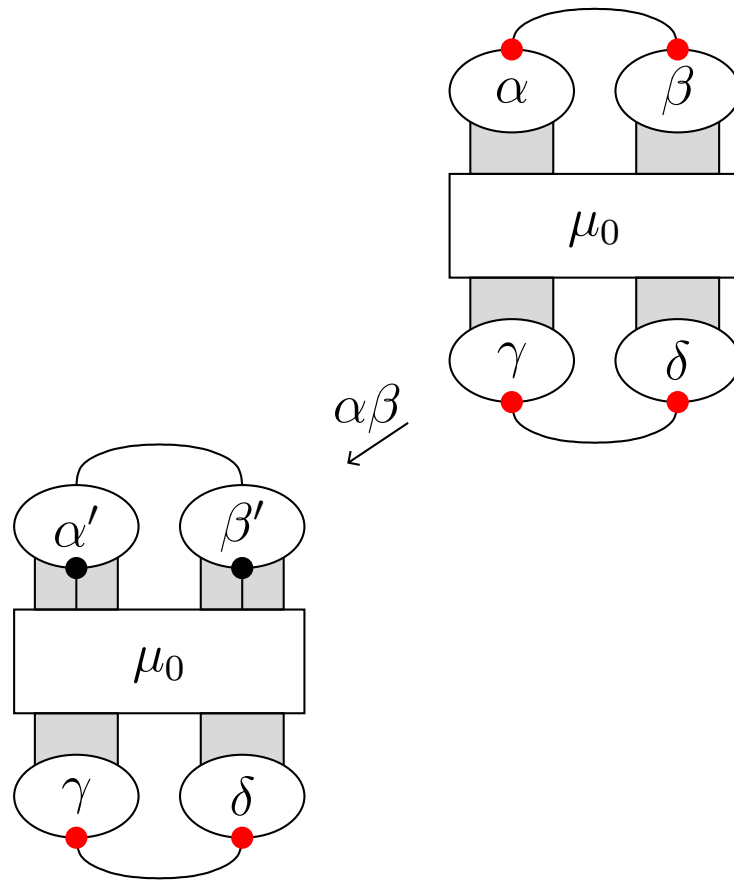
Strong confluence property



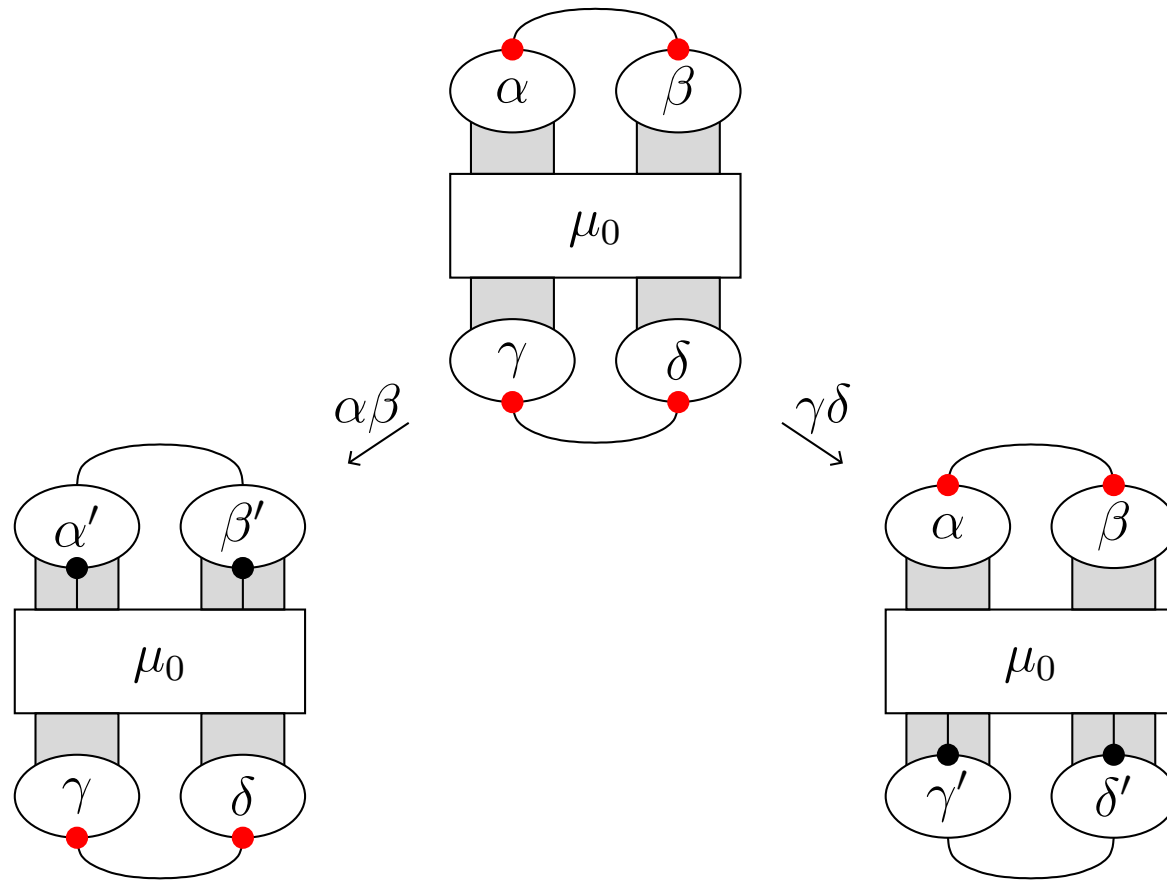
Strong confluence property



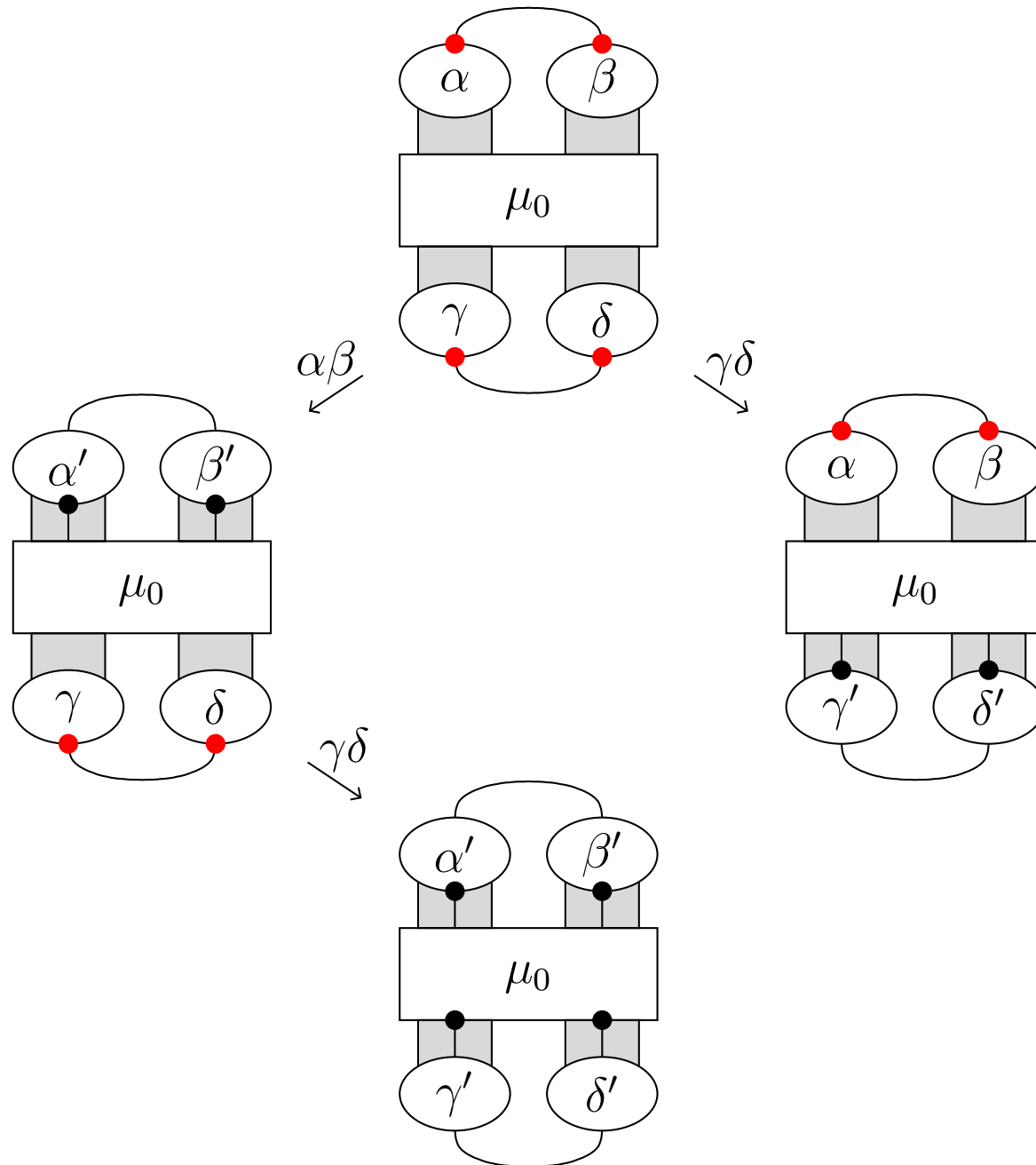
Strong confluence property



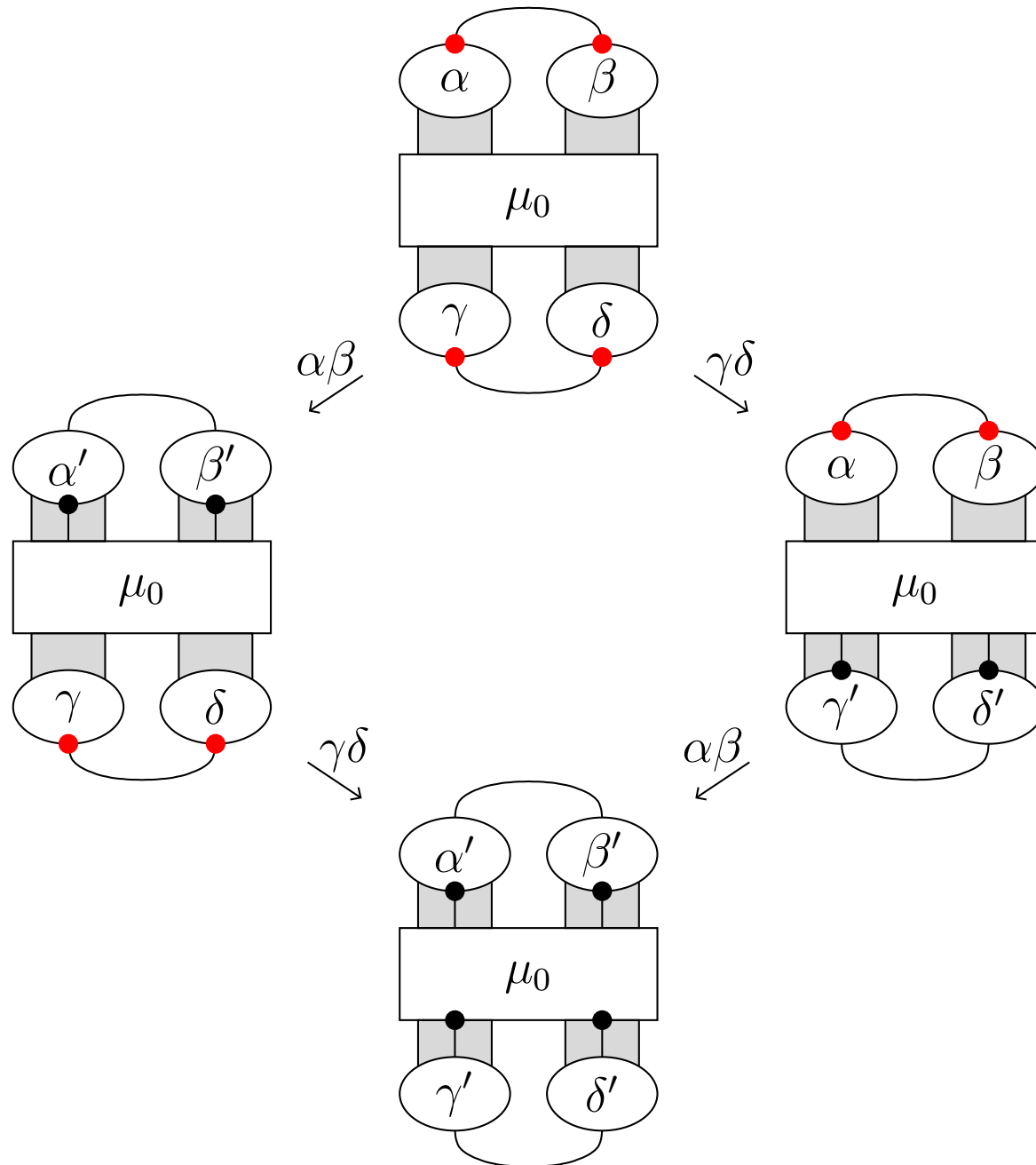
Strong confluence property



Strong confluence property

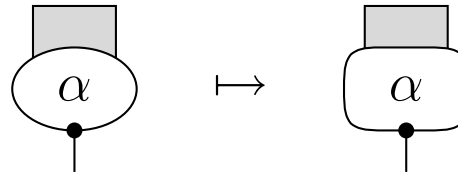


Strong confluence property



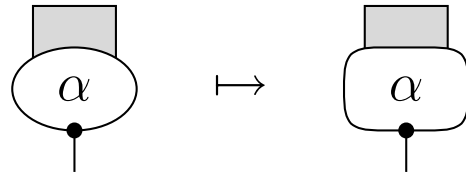
Translation of (Σ, R) into (Σ', R')

- Cells of Σ are translated into nets of Σ' ,

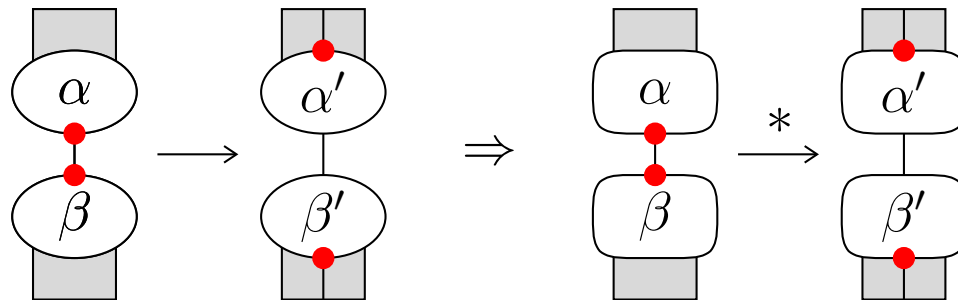


Translation of (Σ, R) into (Σ', R')

- Cells of Σ are translated into nets of Σ' ,



- The translation is compatible with reduction,



Remark. Translations extend to nets obviously.

Example, translation into a binary hard system (1)

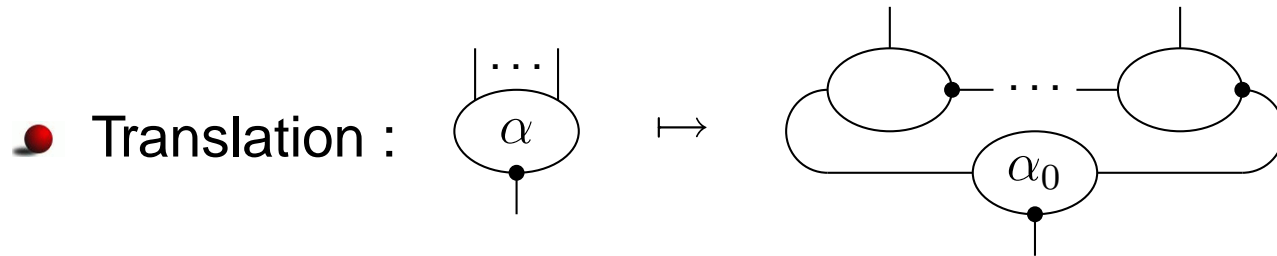
- For any symbol α of arity n we introduce $n + 1$ (binary) symbols $\alpha_0, \alpha_1, \dots, \alpha_n$

Example, translation into a binary hard system (1)

- For any symbol α of arity n we introduce $n + 1$ (binary) symbols $\alpha_0, \alpha_1, \dots, \alpha_n$
- We introduce a (binary) “void” symbol

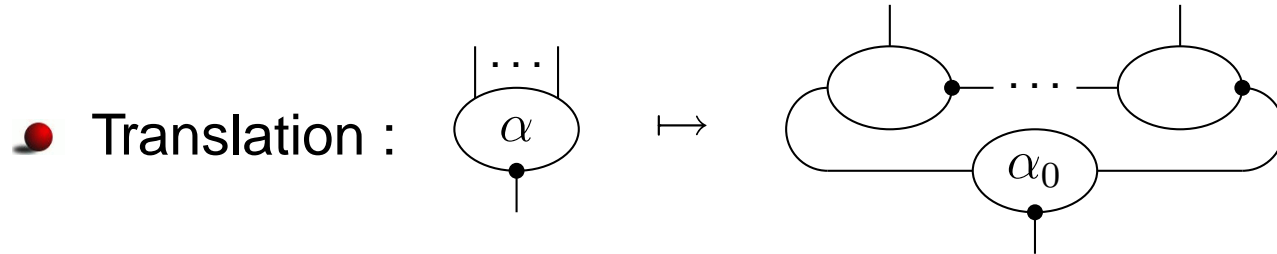
Example, translation into a binary hard system (1)

- For any symbol α of arity n we introduce $n + 1$ (binary) symbols $\alpha_0, \alpha_1, \dots, \alpha_n$
- We introduce a (binary) “void” symbol



Example, translation into a binary hard system (1)

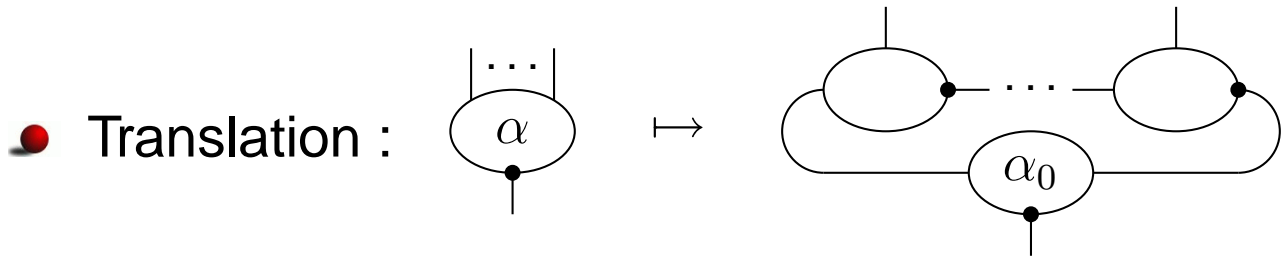
- For any symbol α of arity n we introduce $n + 1$ (binary) symbols $\alpha_0, \alpha_1, \dots, \alpha_n$
- We introduce a (binary) “void” symbol



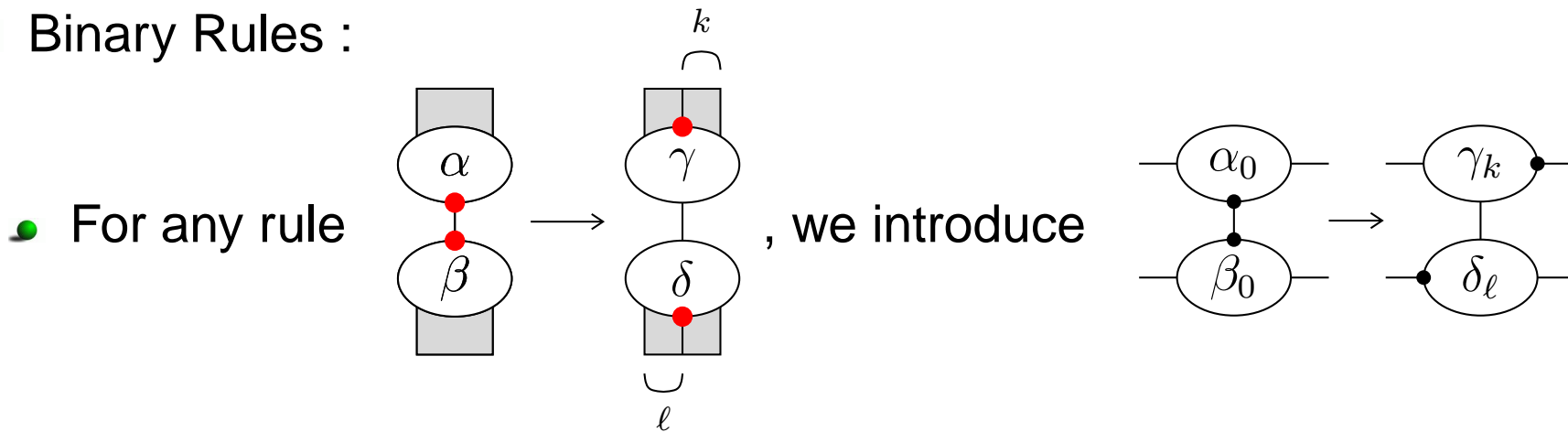
- Binary Rules :

Example, translation into a binary hard system (1)

- For any symbol α of arity n we introduce $n + 1$ (binary) symbols $\alpha_0, \alpha_1, \dots, \alpha_n$
- We introduce a (binary) “void” symbol

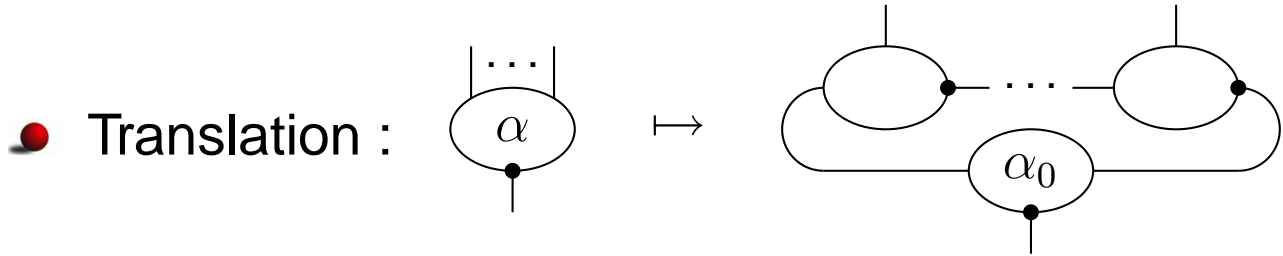


- Binary Rules :

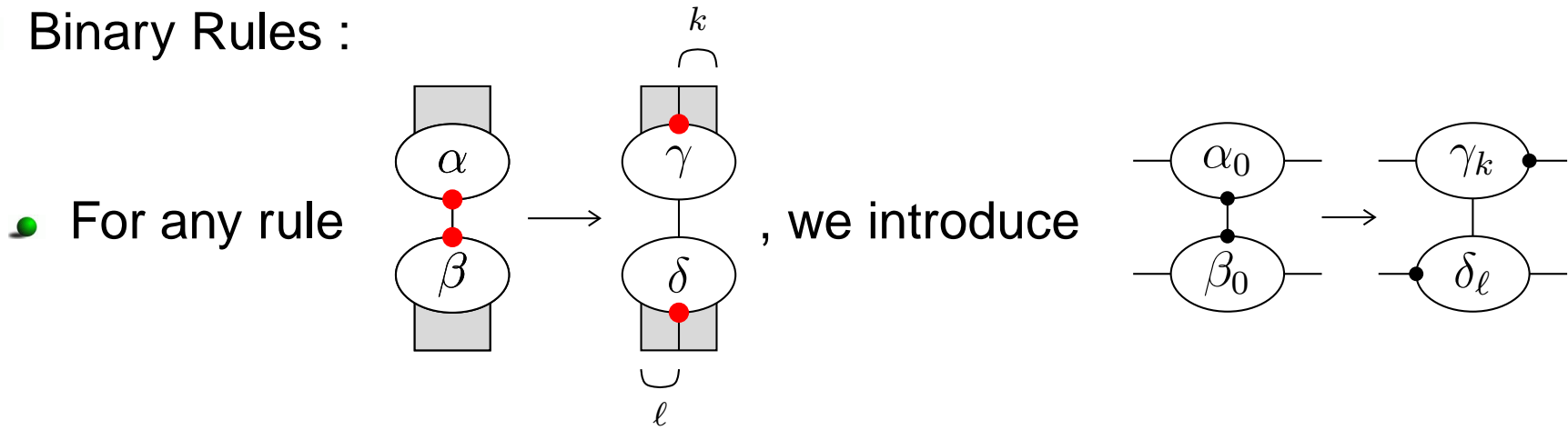


Example, translation into a binary hard system (1)

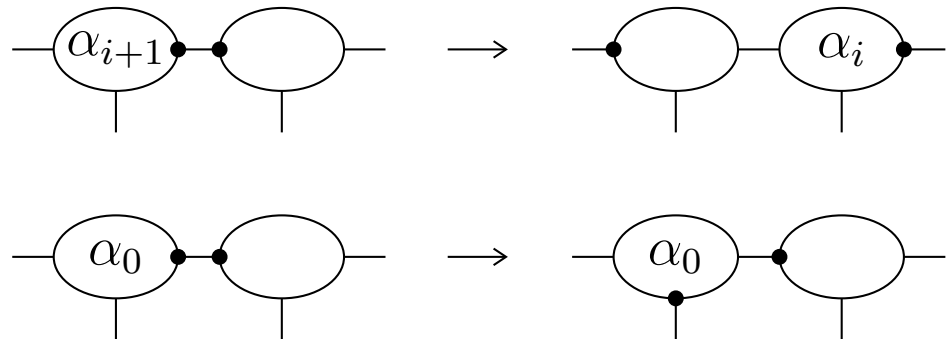
- For any symbol α of arity n we introduce $n + 1$ (binary) symbols $\alpha_0, \alpha_1, \dots, \alpha_n$
- We introduce a (binary) "void" symbol



- Binary Rules :



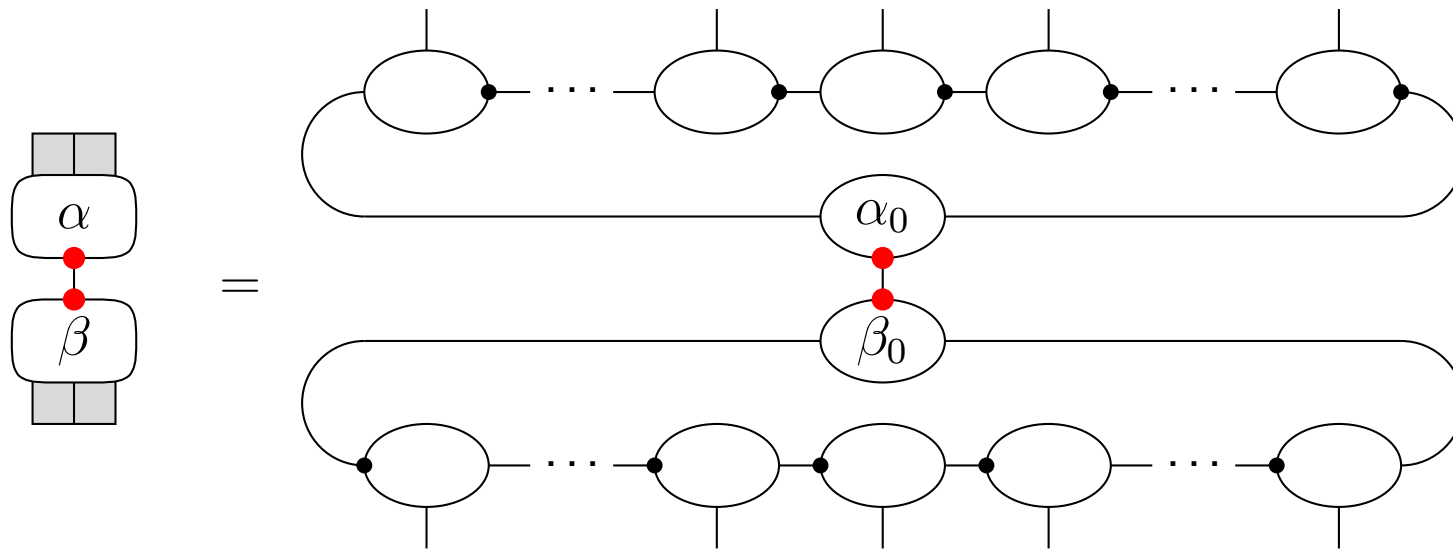
- For any symbol α_i ,



Translation into a binary hard system (2)

Proposition. The previous translation is compatible with reduction.

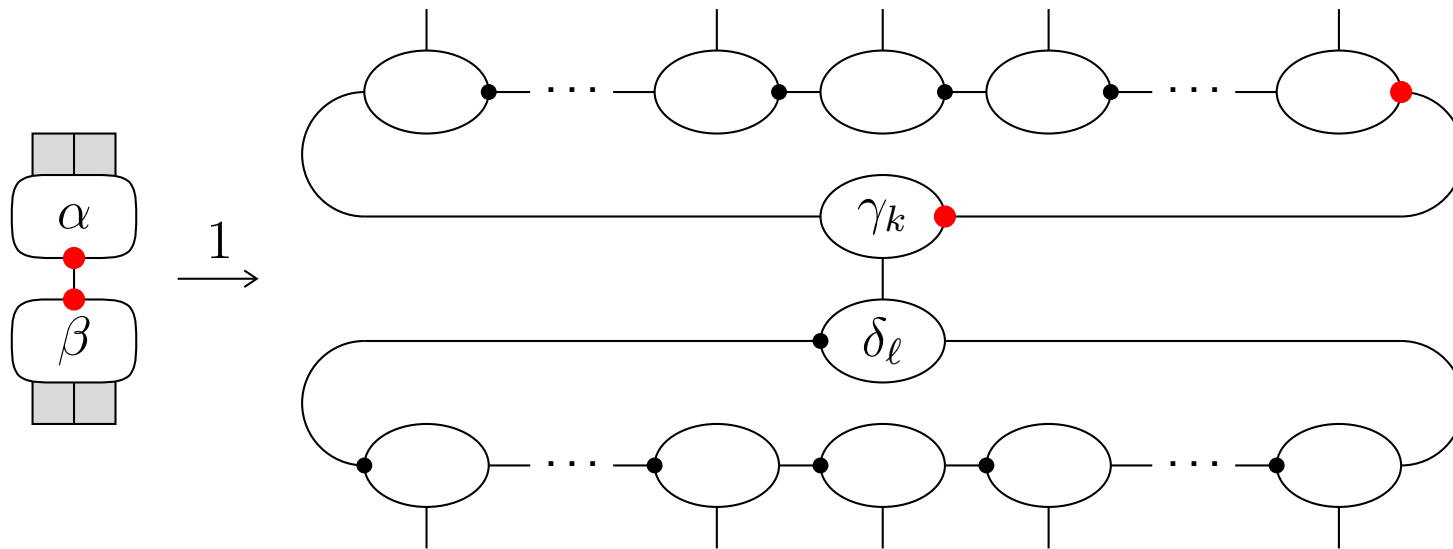
Proof.



Translation into a binary hard system (2)

Proposition. The previous translation is compatible with reduction.

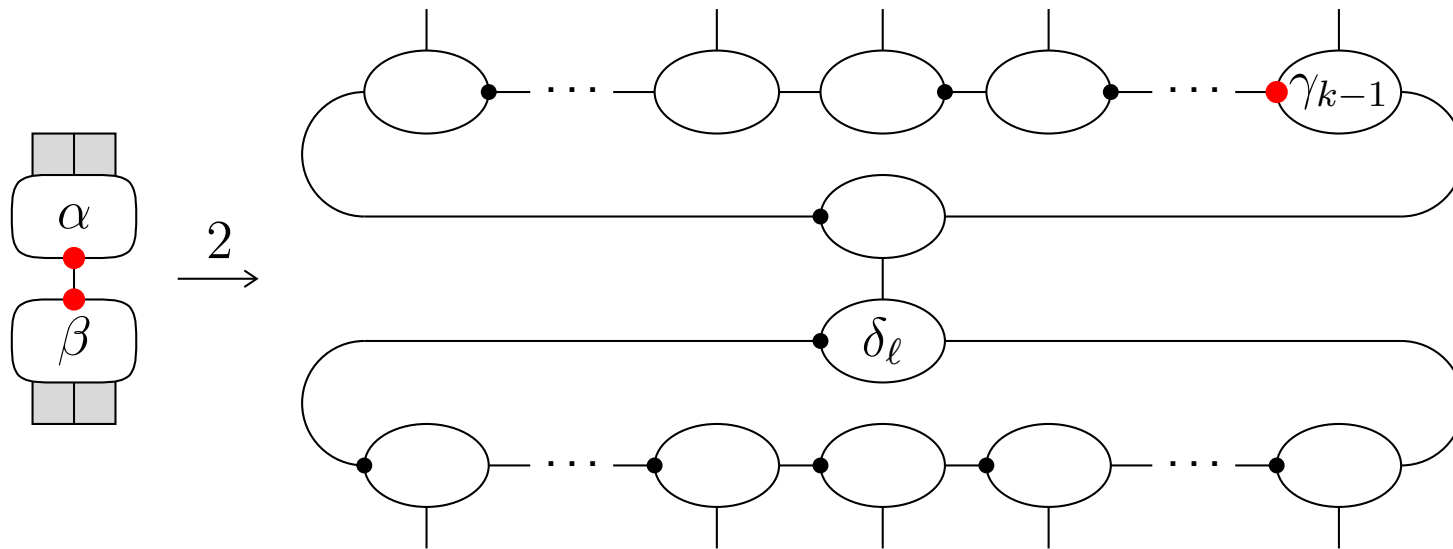
Proof.



Translation into a binary hard system (2)

Proposition. The previous translation is compatible with reduction.

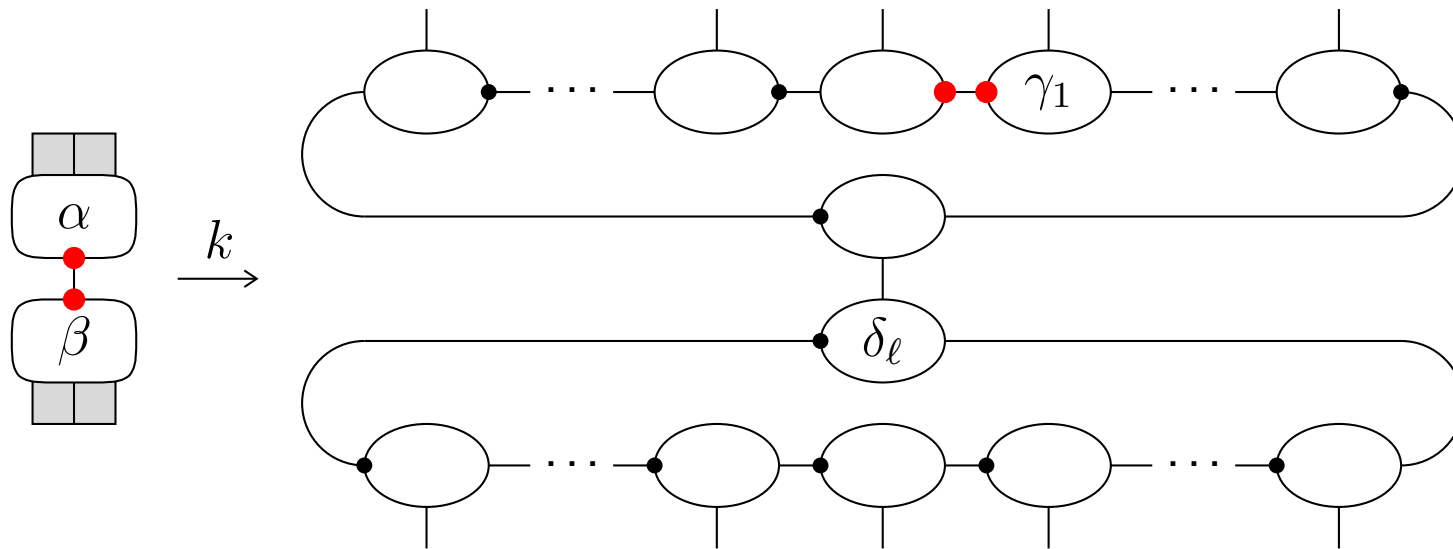
Proof.



Translation into a binary hard system (2)

Proposition. The previous translation is compatible with reduction.

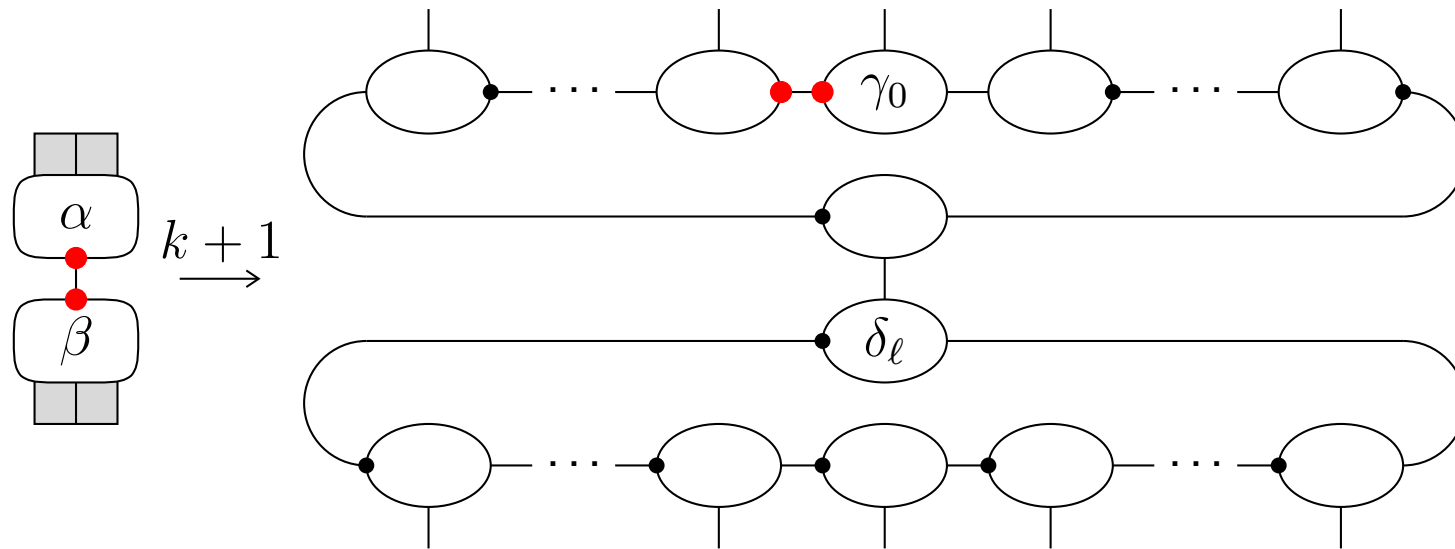
Proof.



Translation into a binary hard system (2)

Proposition. The previous translation is compatible with reduction.

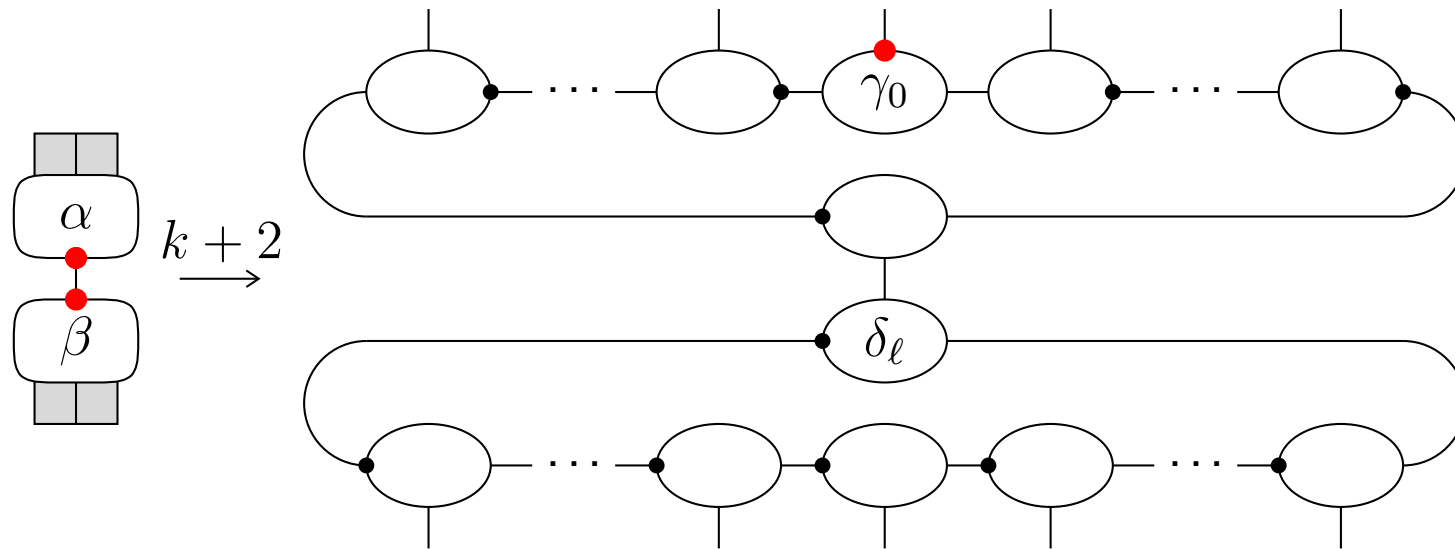
Proof.



Translation into a binary hard system (2)

Proposition. The previous translation is compatible with reduction.

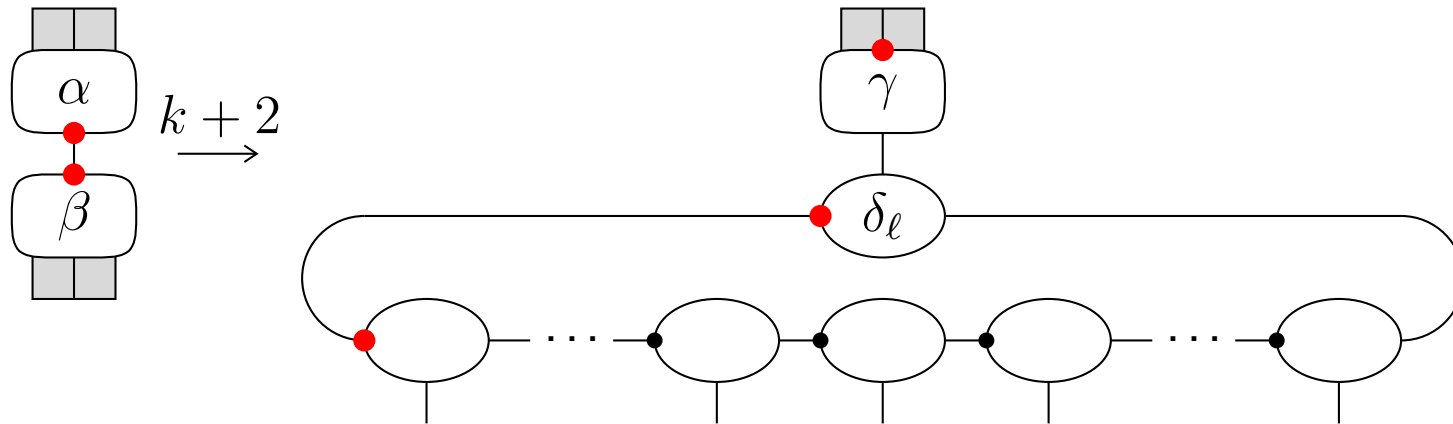
Proof.



Translation into a binary hard system (2)

Proposition. The previous translation is compatible with reduction.

Proof.



Translation into a binary hard system (2)

Proposition. The previous translation is compatible with reduction.

Proof.



Universal hard system

Question.

Find a hard system (*i.e* a set of *symbols* and *rules*) into which we can translate any hard system.

Universal hard system

Question.

Find a hard system (*i.e* a set of *symbols* and *rules*) into which we can translate any hard system.

Hints.

- Symbols are numbered and coded by their binary representation.

Universal hard system

Question.

Find a hard system (*i.e* a set of *symbols* and *rules*) into which we can translate any hard system.

Hints.

- Symbols are numbered and coded by their binary representation.
- Let us implement boolean functions

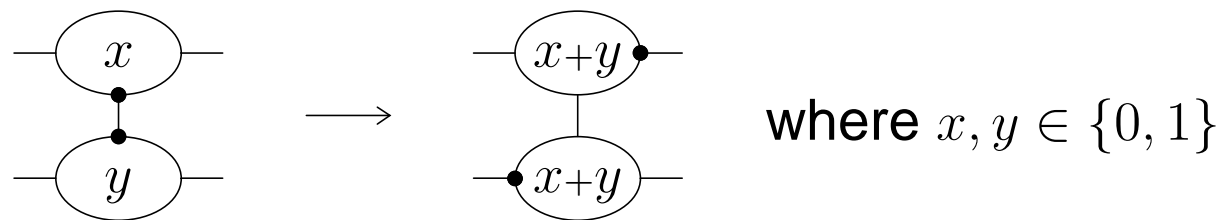
Universal hard system

Question.

Find a hard system (*i.e* a set of *symbols* and *rules*) into which we can translate any hard system.

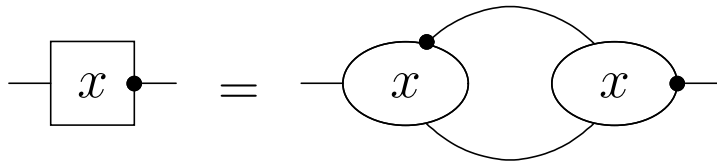
Hints.

- Symbols are numbered and coded by their binary representation.
- Let us implement boolean functions
- Let us introduce binary symbols 0 and 1 with the following rules,



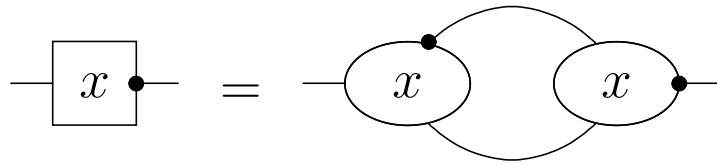
Pipes

Definition :

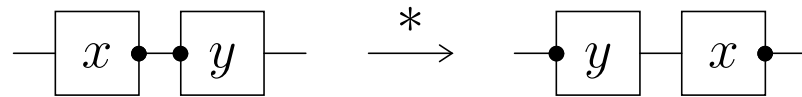


Pipes

Definition :

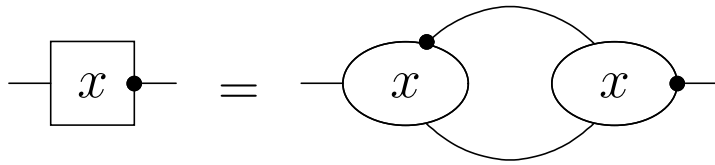


Lemma :

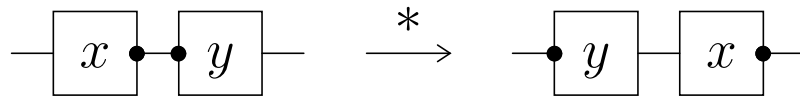


Pipes

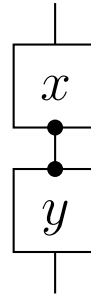
Definition :



Lemma :

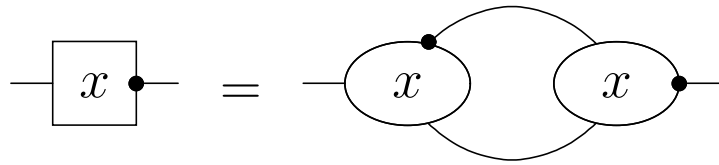


Proof : ($x + x = 0 \pmod{2}$)

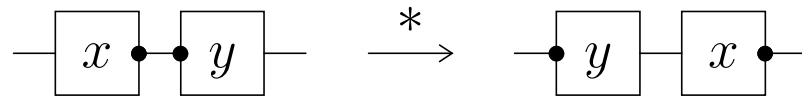


Pipes

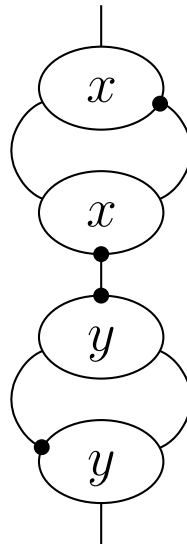
Definition :



Lemma :

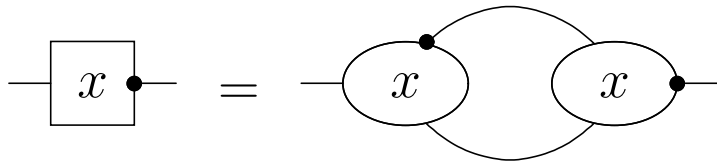


Proof : $(x + x = 0 \pmod{2})$

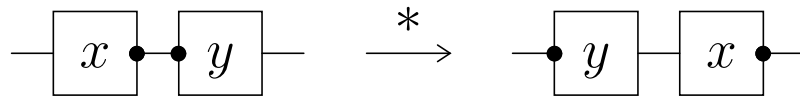


Pipes

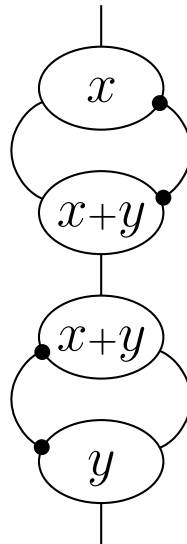
Definition :



Lemma :

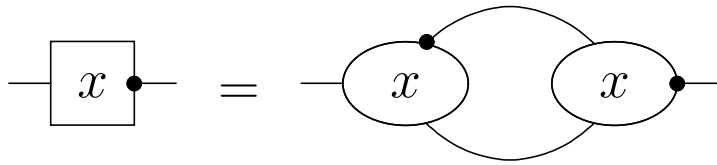


Proof : $(x + x = 0 \pmod{2})$

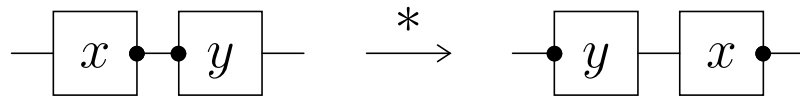


Pipes

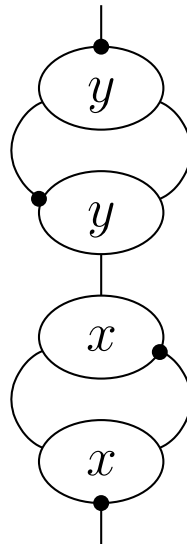
Definition :



Lemma :

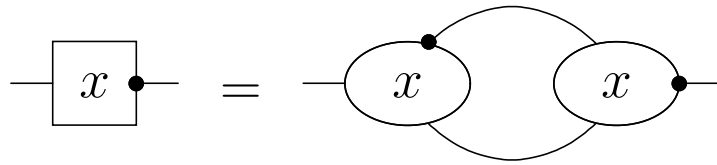


Proof : $(x + x = 0 \pmod{2})$

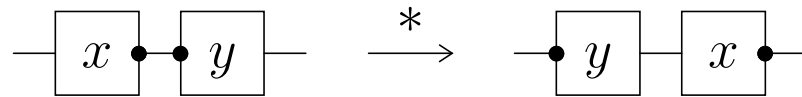


Pipes

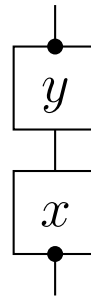
Definition :



Lemma :

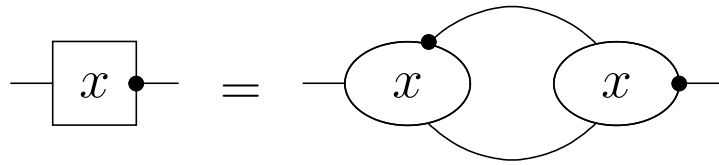


Proof : $(x + x = 0 \pmod{2})$

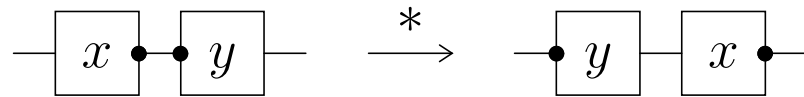


Pipes

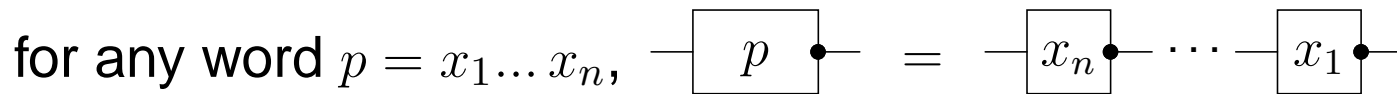
Definition :



Lemma :

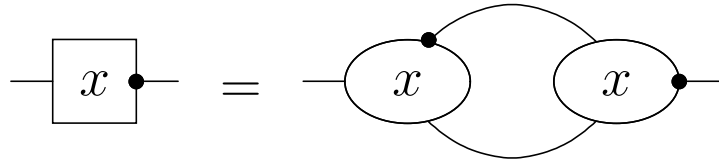


Definition :

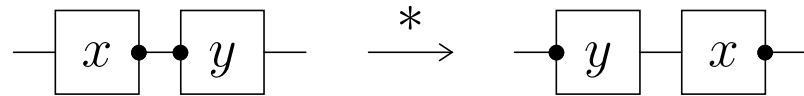


Pipes

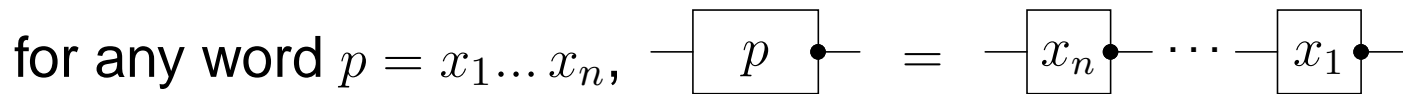
Definition :



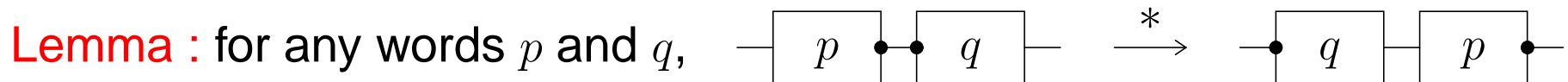
Lemma :



Definition :

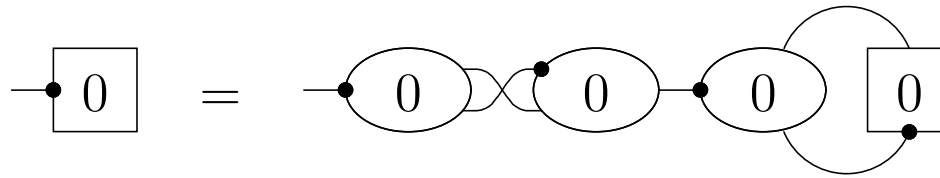


Lemma :

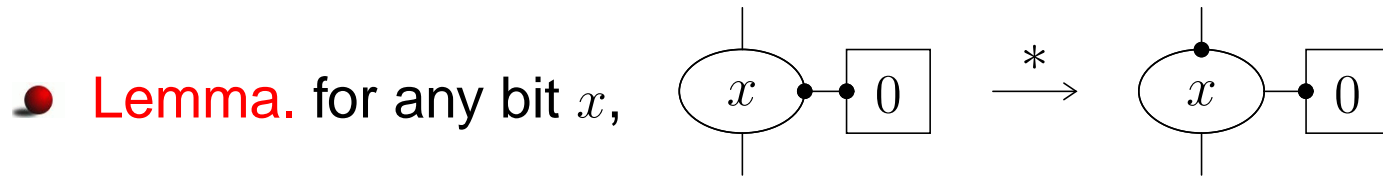
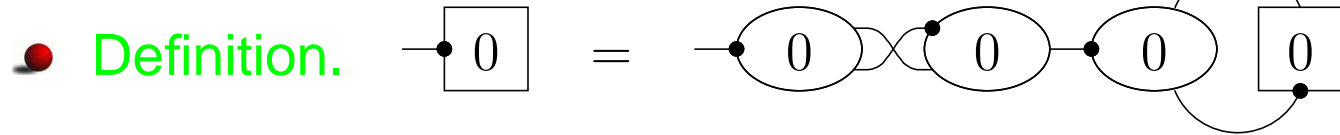


Constant Zero

• Definition.

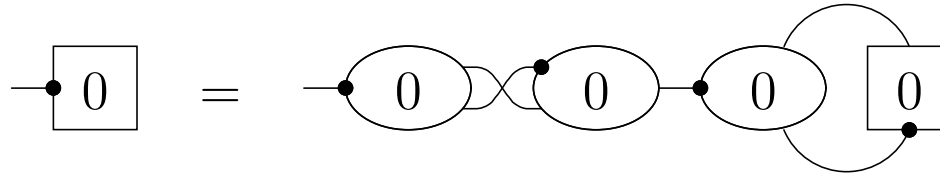


Constant Zero

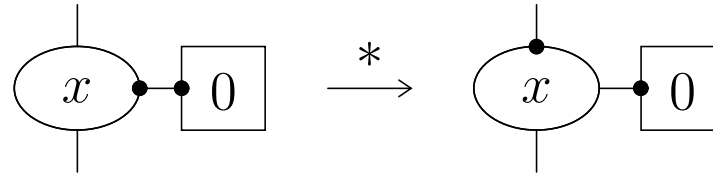


Constant Zero

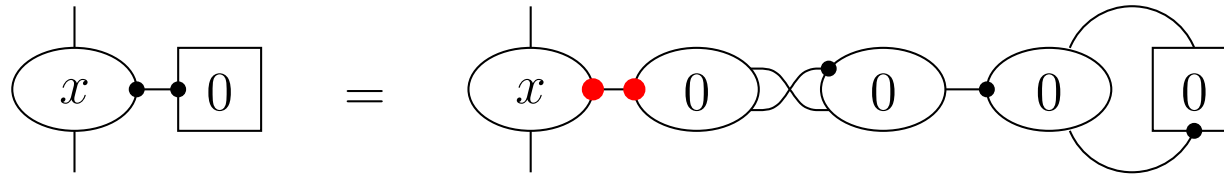
● Definition.



● Lemma. for any bit x ,

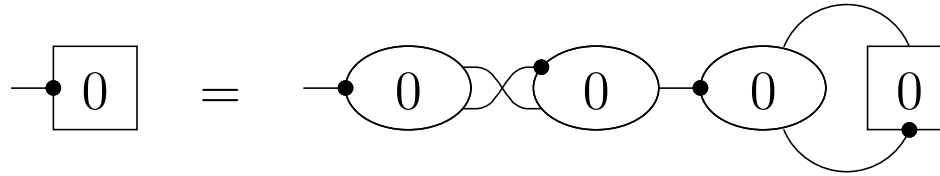


Proof.

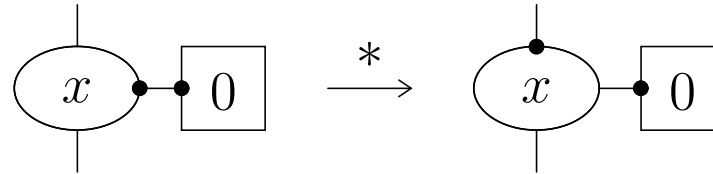


Constant Zero

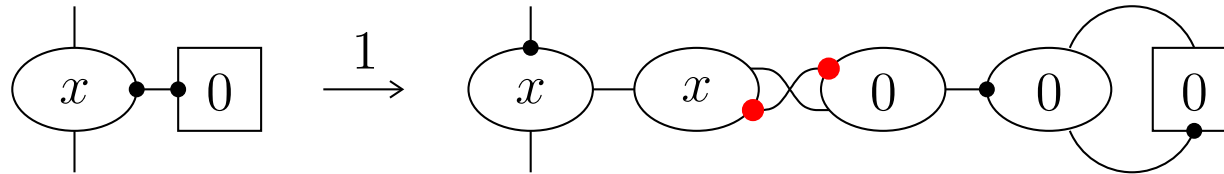
● Definition.



● Lemma. for any bit x ,

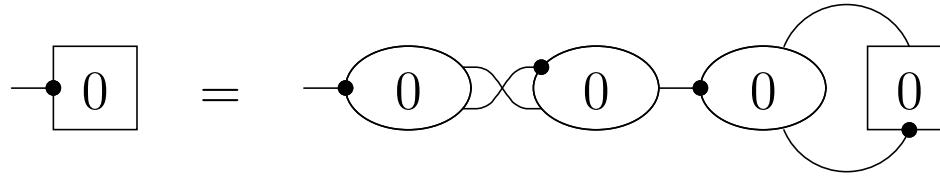


Proof.

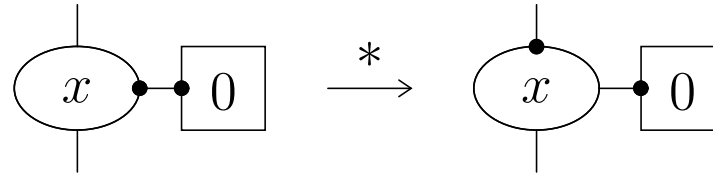


Constant Zero

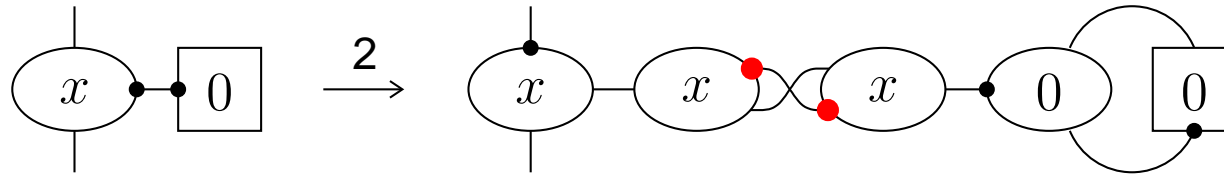
● Definition.



● Lemma. for any bit x ,

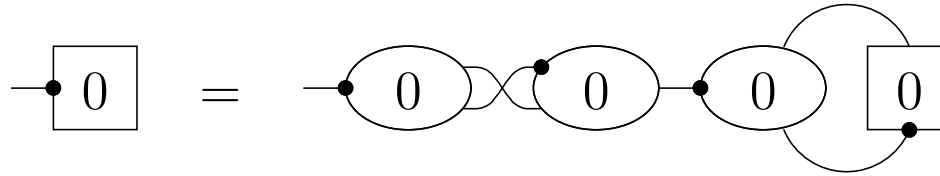


Proof.

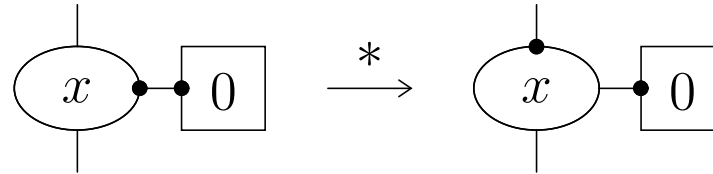


Constant Zero

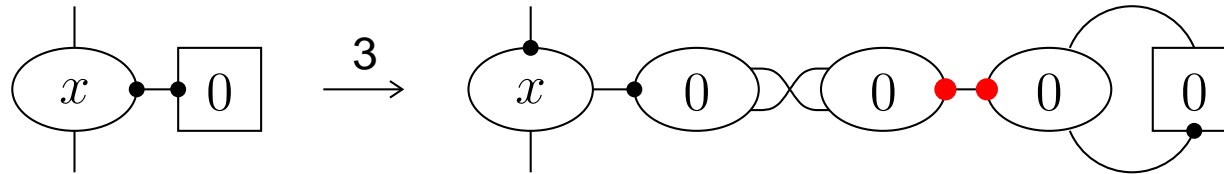
● Definition.



● Lemma. for any bit x ,

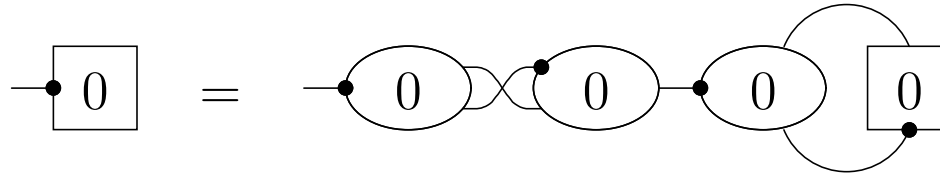


Proof.

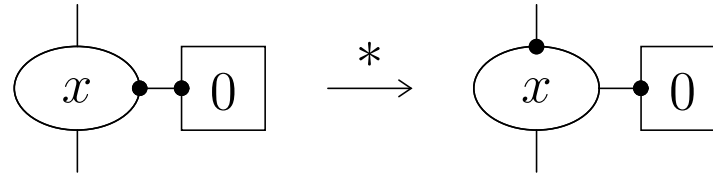


Constant Zero

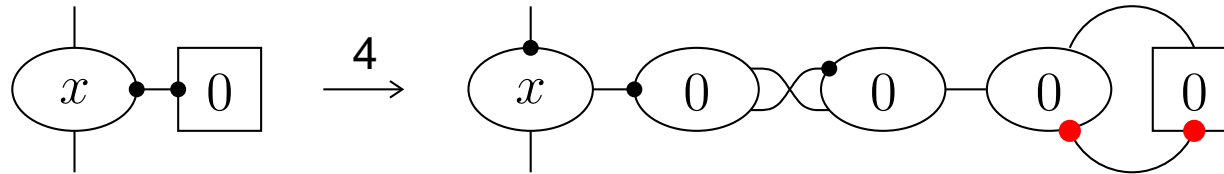
● Definition.



● Lemma. for any bit x ,

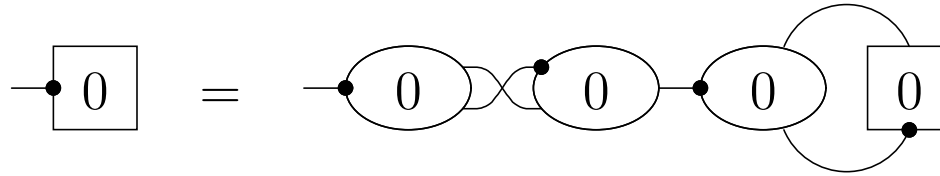


Proof.

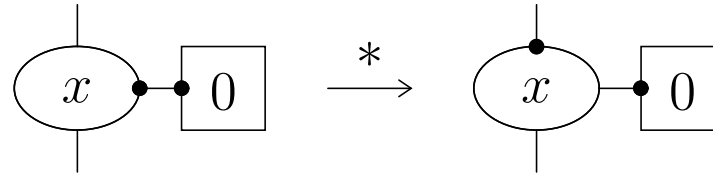


Constant Zero

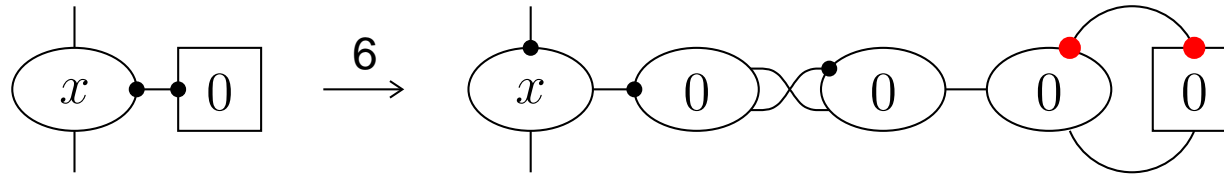
● Definition.



● Lemma. for any bit x ,

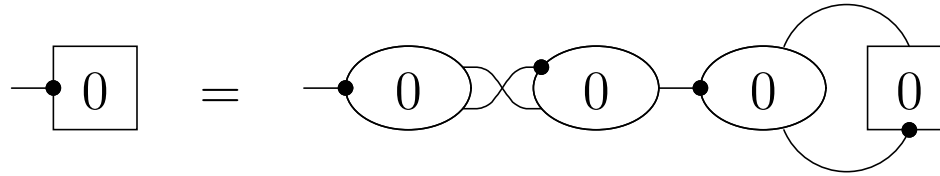


Proof.

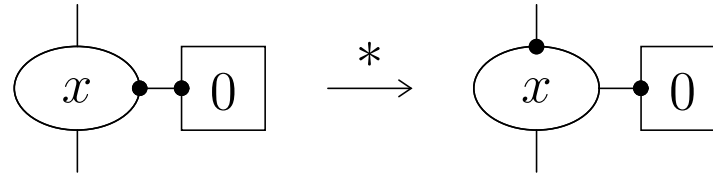


Constant Zero

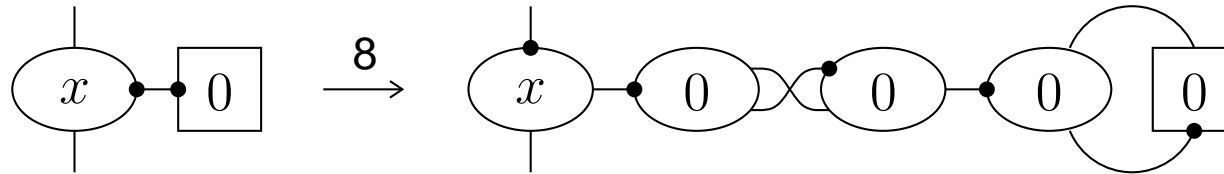
● Definition.



● Lemma. for any bit x ,

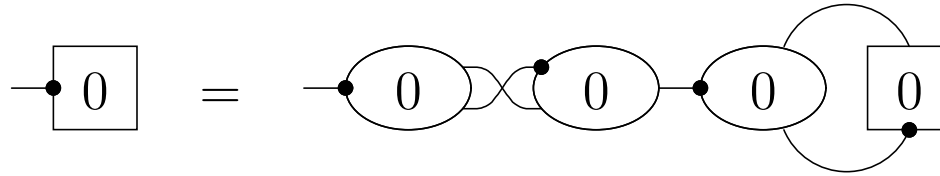


Proof.

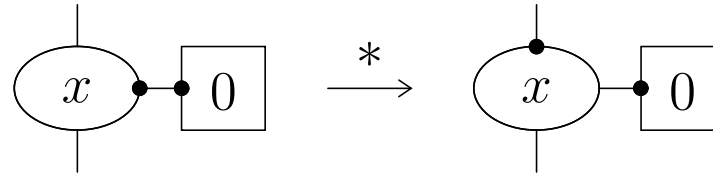


Constant Zero

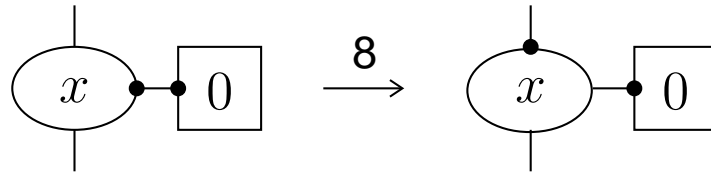
● Definition.



● Lemma. for any bit x ,

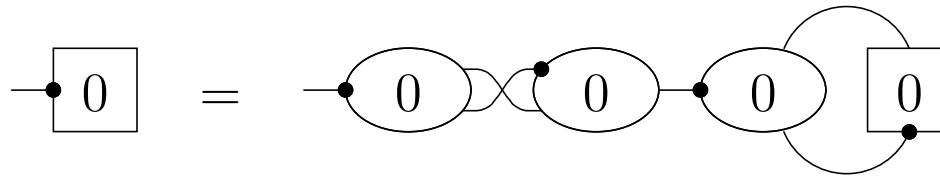


Proof.

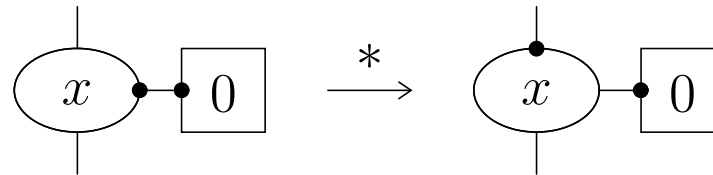


Constant Zero

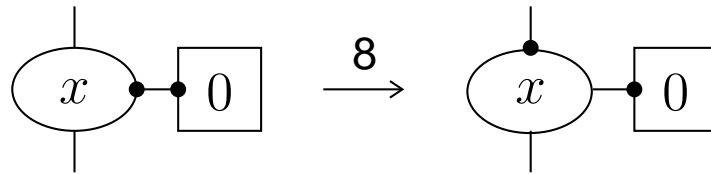
● Definition.



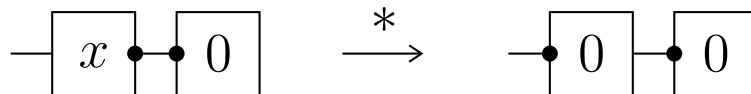
● Lemma. for any bit x ,



Proof.

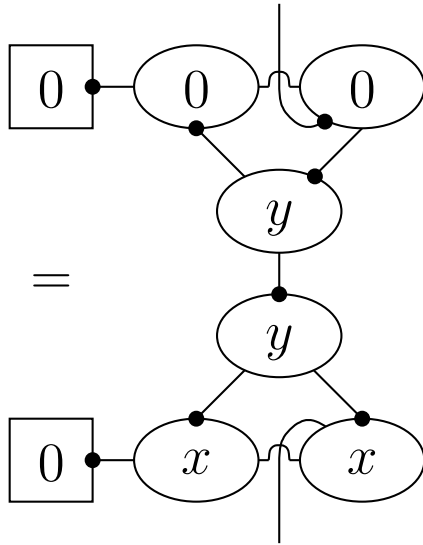


● Corollary.



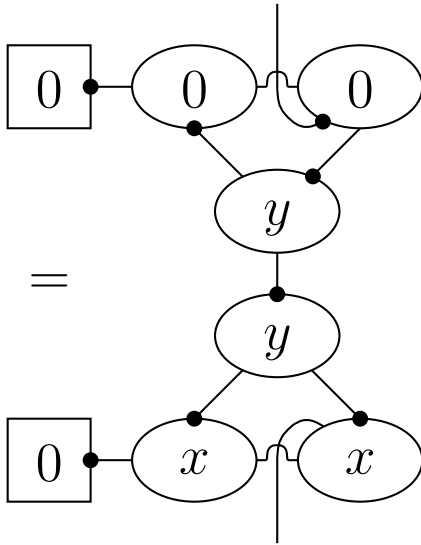
Diode

● Definition.

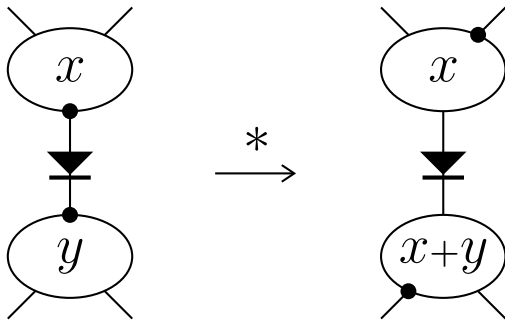


Diode

● Definition.

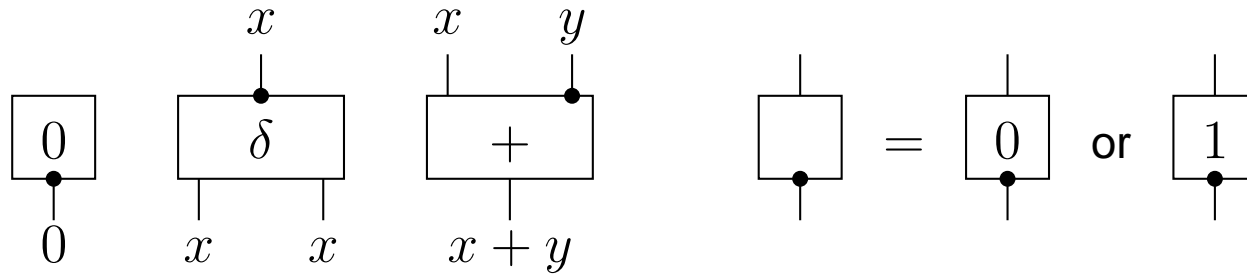


● Lemma.



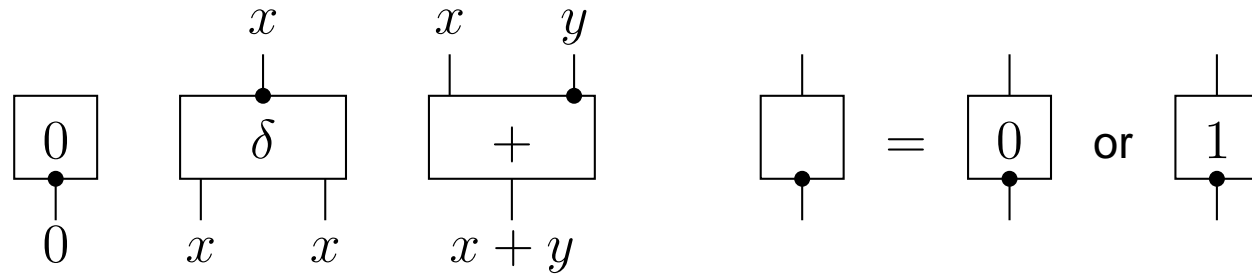
First boolean functions

• Notation.

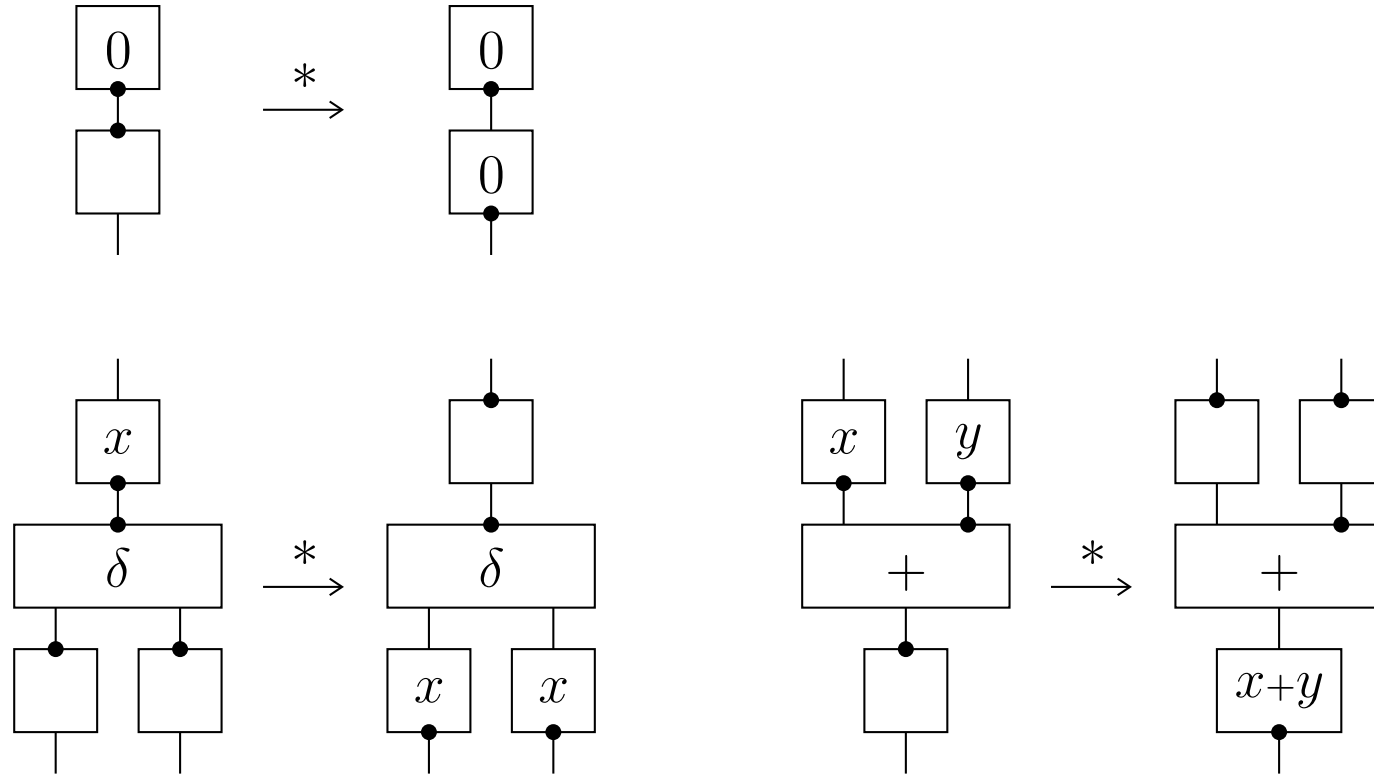


First boolean functions

● Notation.

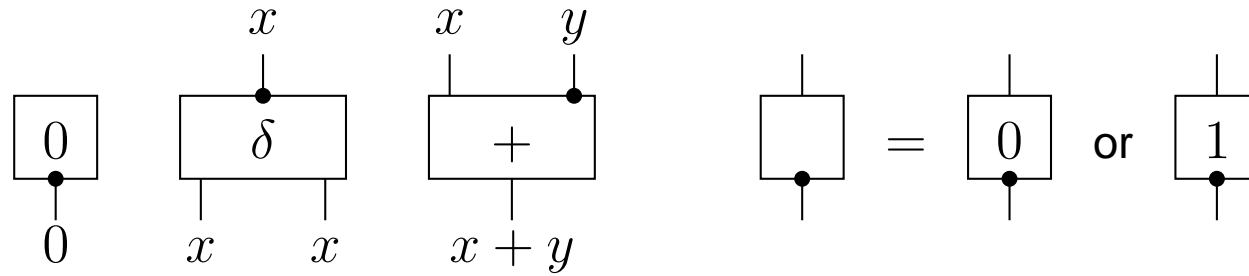


● Lemma.

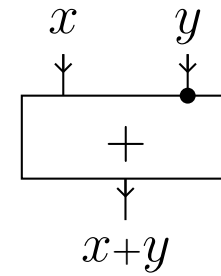
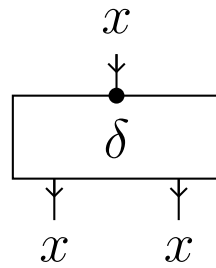
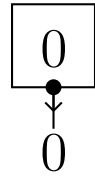


First boolean functions

● Notation.



● Lemma.



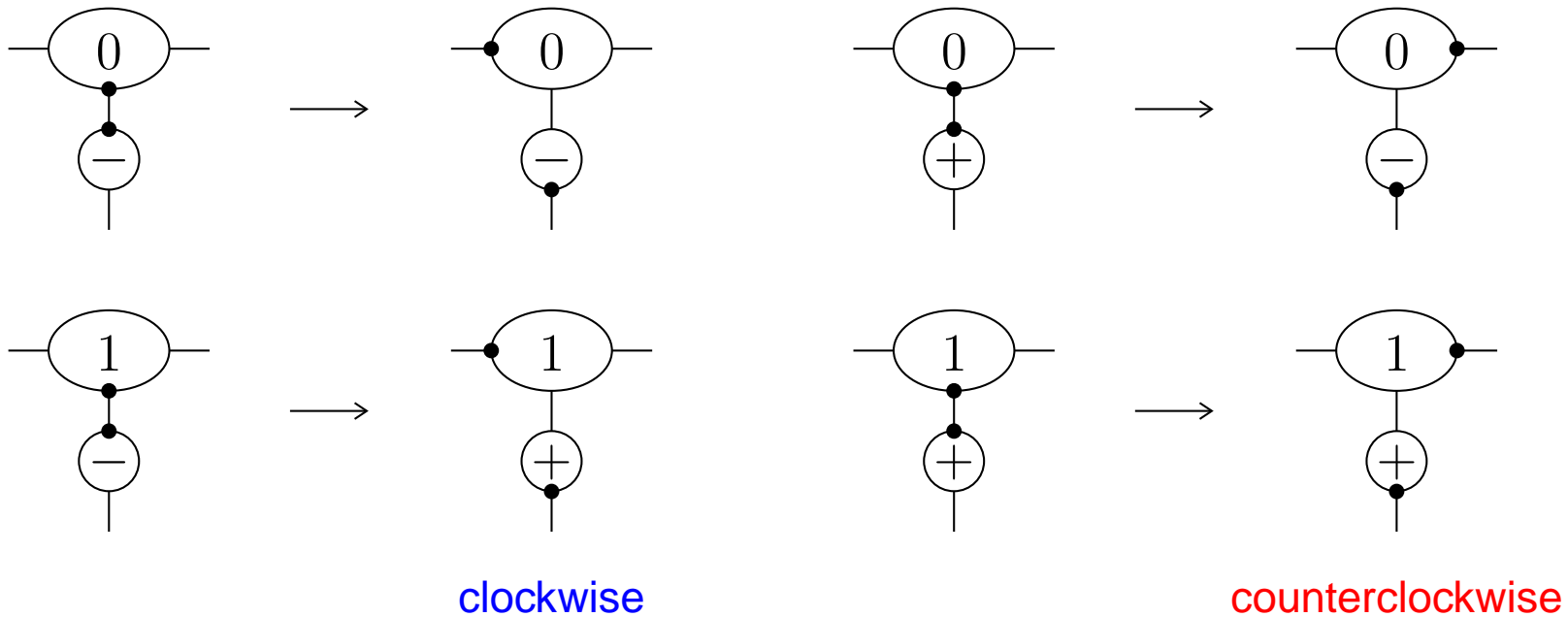
Non-uniform rules

Proposition. A uniform system is not universal.

Non-uniform rules

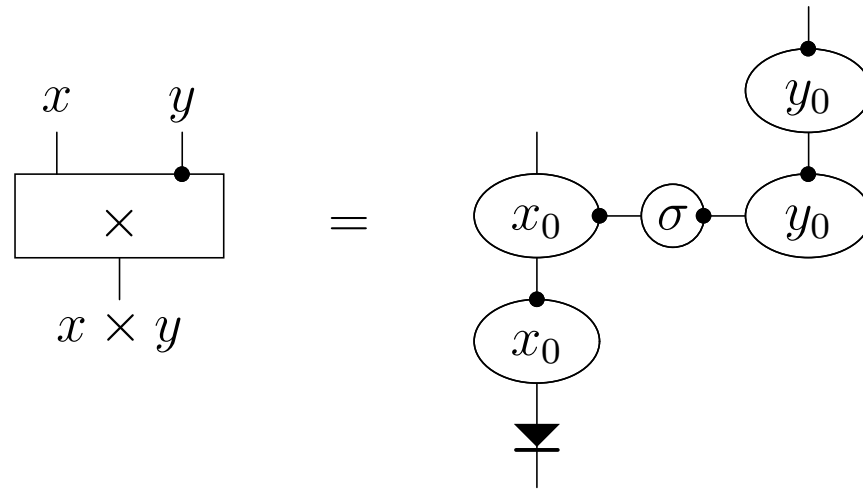
Proposition. A uniform system is not universal.

Consequently, let us introduce unary symbols $+$ and $-$ with the following rules,



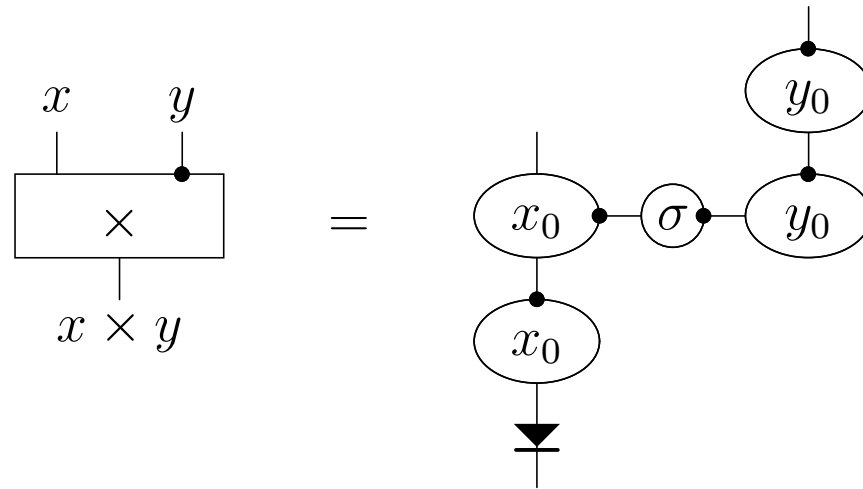
Non-uniform functions

• Sequential product.

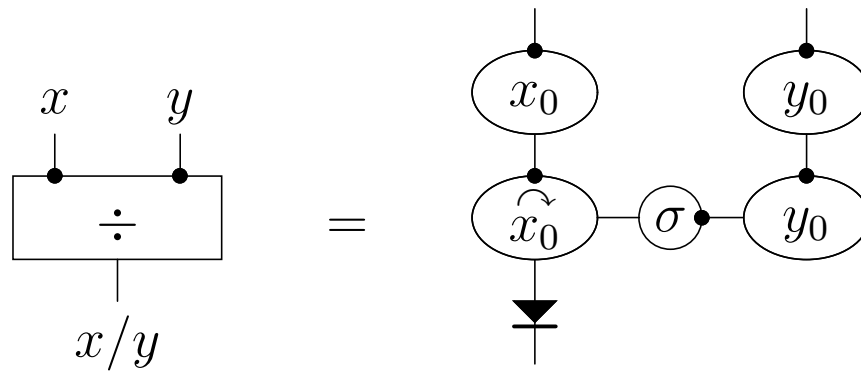


Non-uniform functions

• Sequential product.

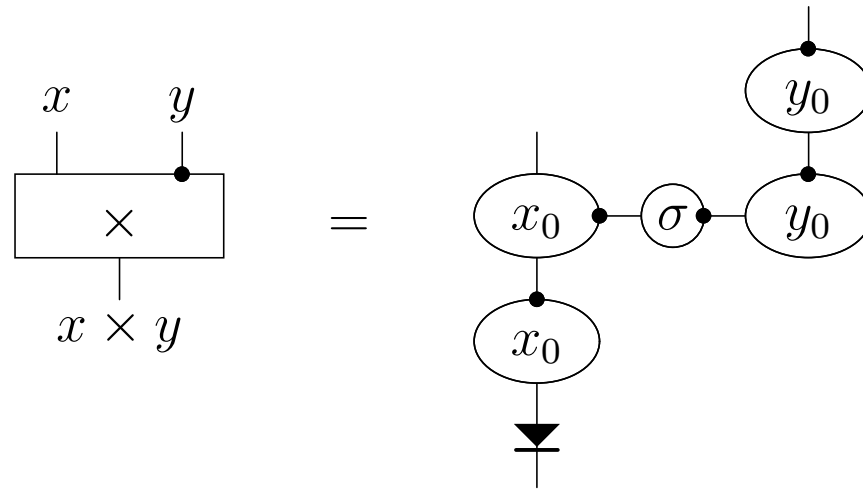


• Partial quotient.

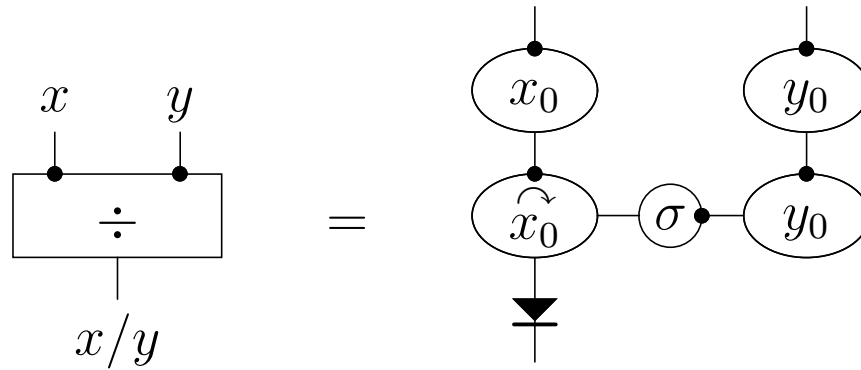


Non-uniform functions

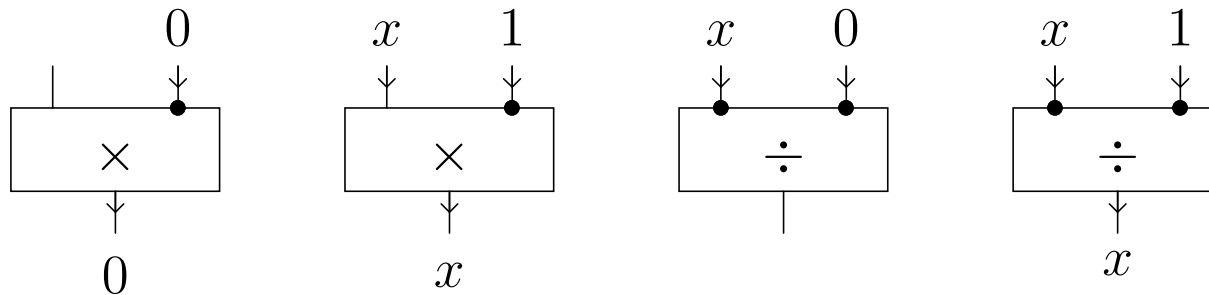
• Sequential product.



• Partial quotient.



Lemma.



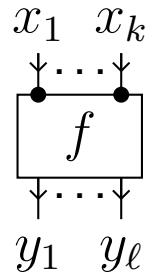
Boolean functions

Let $f : \{0, 1\}^k \rightarrow \{0, 1\}^\ell$
 $p = (x_1, \dots, x_k) \mapsto f(p) = (y_1, \dots, y_\ell)$

Boolean functions

$$\text{Let } f : \{0, 1\}^k \rightarrow \{0, 1\}^\ell$$
$$p = (x_1, \dots, x_k) \mapsto f(p) = (y_1, \dots, y_\ell)$$

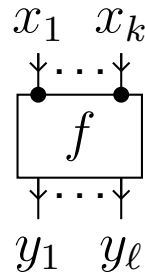
• Compose 0 , $+$, δ , \times , \div and build



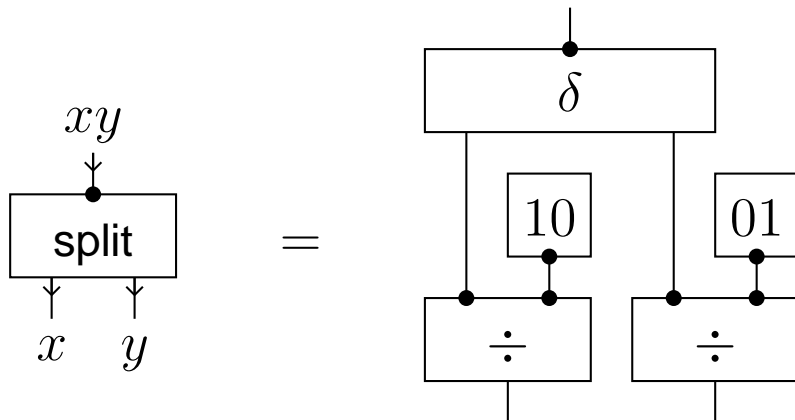
Boolean functions

Let $f : \{0, 1\}^k \rightarrow \{0, 1\}^\ell$
 $p = (x_1, \dots, x_k) \mapsto f(p) = (y_1, \dots, y_\ell)$

- Compose $0, +, \delta, \times, \div$ and build



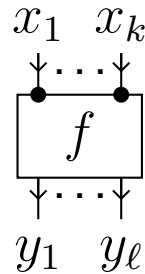
- Parallel / Sequential conversion



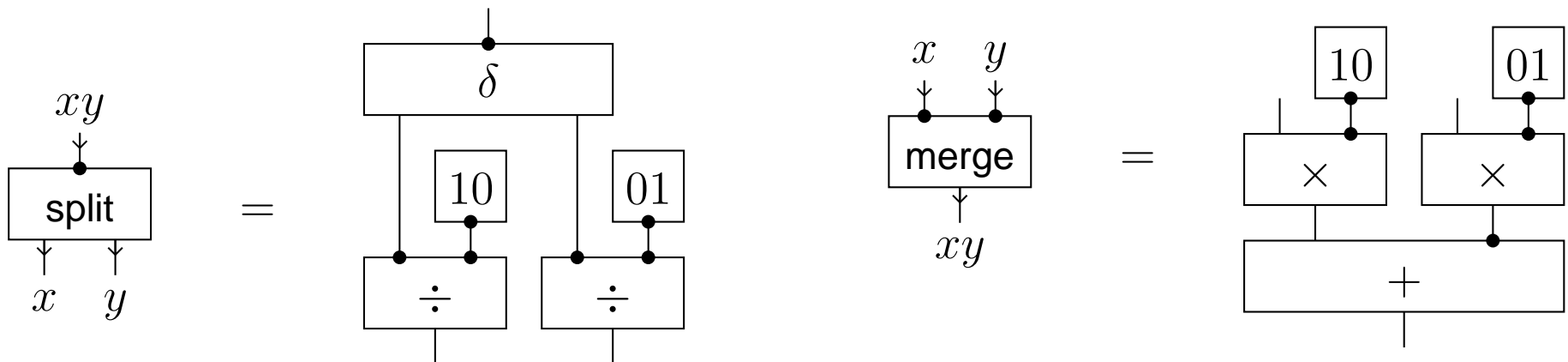
Boolean functions

Let $f : \{0, 1\}^k \rightarrow \{0, 1\}^\ell$
 $p = (x_1, \dots, x_k) \mapsto f(p) = (y_1, \dots, y_\ell)$

- Compose 0 , $+$, δ , \times , \div and build



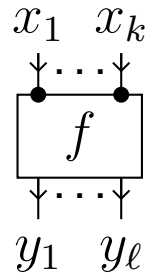
- Parallel / Sequential conversion



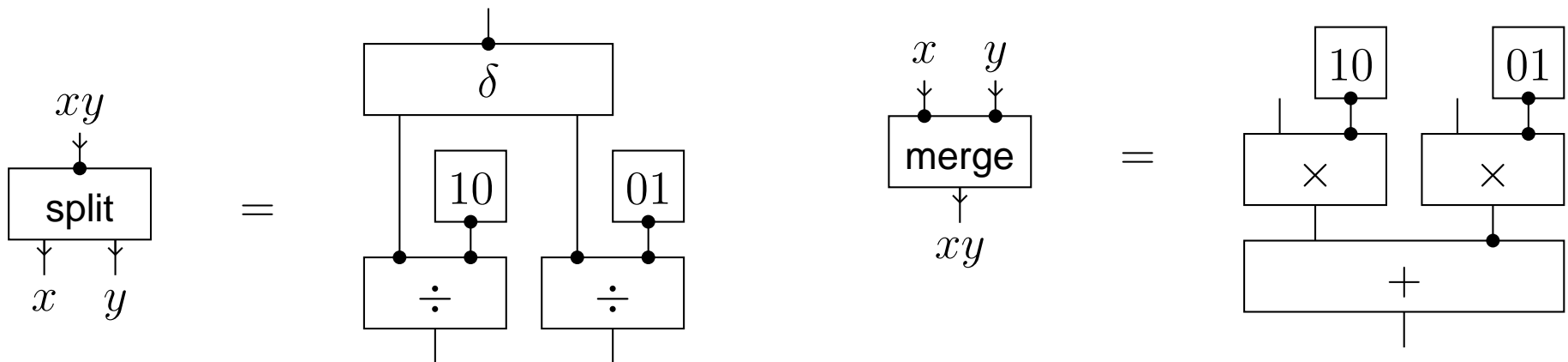
Boolean functions

Let $f : \{0, 1\}^k \rightarrow \{0, 1\}^\ell$
 $p = (x_1, \dots, x_k) \mapsto f(p) = (y_1, \dots, y_\ell)$

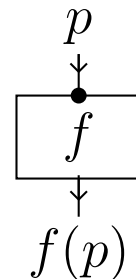
• Compose $0, +, \delta, \times, \div$ and build



• Parallel / Sequential conversion



• Compose f with split, merge and build



Translation

Let (Σ, R) a hard system and $R : \Sigma \times \Sigma \rightarrow \Sigma \times \mathbb{N}$
 $(\alpha, \beta) \mapsto (\gamma, k)$

Translation

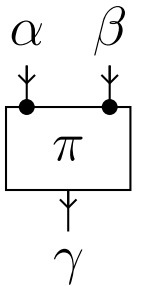
Let (Σ, R) a hard system and $R : \Sigma \times \Sigma \rightarrow \Sigma \times \mathbb{N}$
 $(\alpha, \beta) \mapsto (\gamma, k)$

- Symbols are encoded with binary numbers,

Translation

Let (Σ, R) a hard system and $R : \Sigma \times \Sigma \rightarrow \Sigma \times \mathbb{N}$
 $(\alpha, \beta) \mapsto (\gamma, k)$

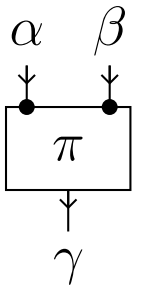
- Symbols are encoded with binary numbers,



Translation

Let (Σ, R) a hard system and $R : \Sigma \times \Sigma \rightarrow \Sigma \times \mathbb{N}$
 $(\alpha, \beta) \mapsto (\gamma, k)$

- Symbols are encoded with binary numbers,



- Let us introduce symbols $\alpha_0, \dots, \alpha_k$,

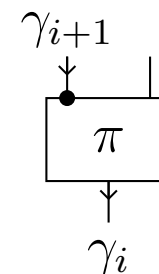
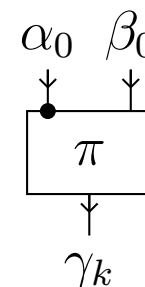
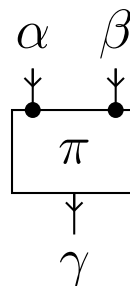
Translation

Let (Σ, R) a hard system and $R : \Sigma \times \Sigma \rightarrow \Sigma$

$$\begin{aligned}(\alpha_0, \beta_0) &\mapsto \gamma_k \\ (\gamma_{i+1}, -) &\mapsto \gamma_i\end{aligned}$$

- Symbols are encoded with binary numbers,

- Let us introduce symbols $\alpha_0, \dots, \alpha_k$,



Translation

Let (Σ, R) a hard system and

$$R : \Sigma \times \Sigma \rightarrow \Sigma$$

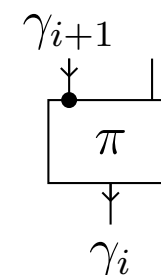
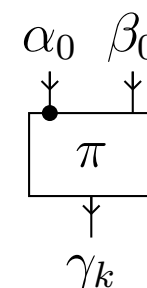
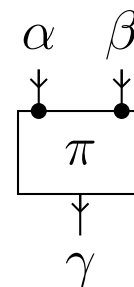
$$(\alpha_0, \beta_0) \mapsto \gamma_k$$

$$(\gamma_{i+1}, -) \mapsto \gamma_i$$

- Symbols are encoded with binary numbers,

- Let us introduce symbols $\alpha_0, \dots, \alpha_k$,

- α_0 is output for the other interacting cell,



Translation

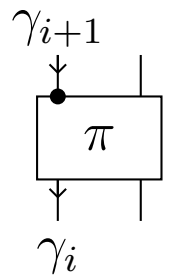
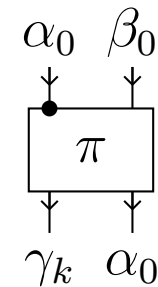
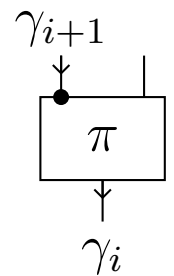
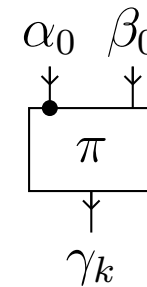
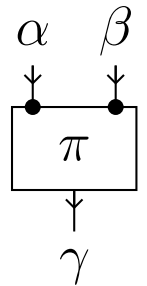
Let (Σ, R) a hard system and $R : \Sigma \times \Sigma \rightarrow \Sigma$

$$\begin{aligned} (\alpha_0, \beta_0) &\mapsto \gamma_k \\ (\gamma_{i+1}, -) &\mapsto \gamma_i \end{aligned}$$

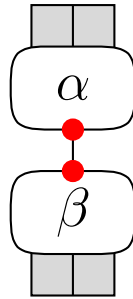
- Symbols are encoded with binary numbers,

- Let us introduce symbols $\alpha_0, \dots, \alpha_k$,

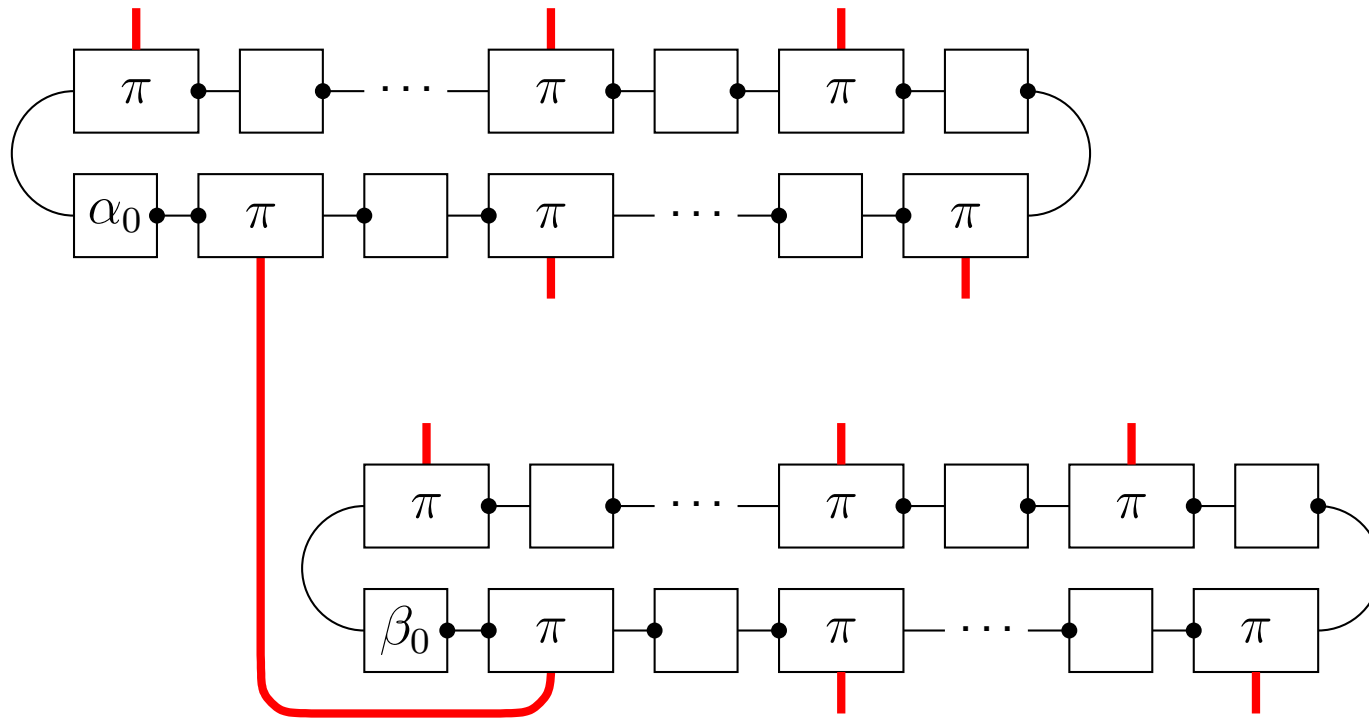
- α_0 is output for the other interacting cell,



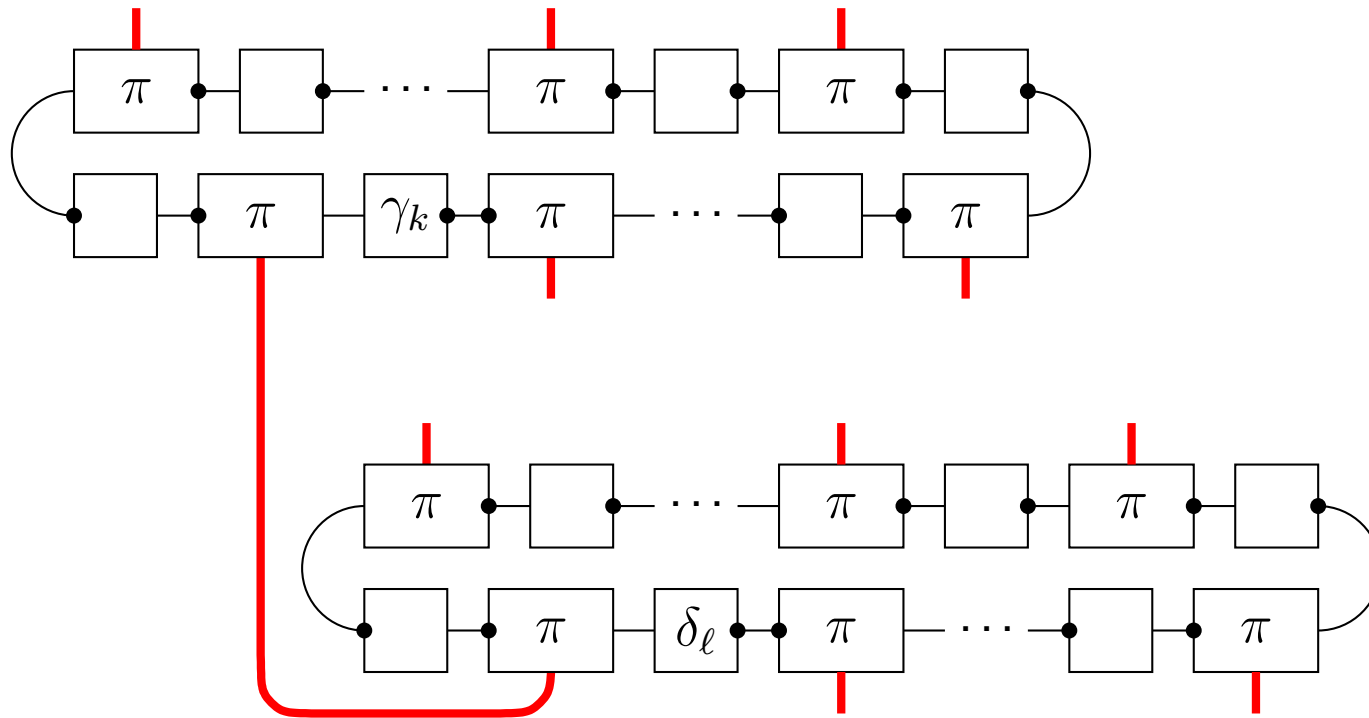
Translation is compatible with reduction



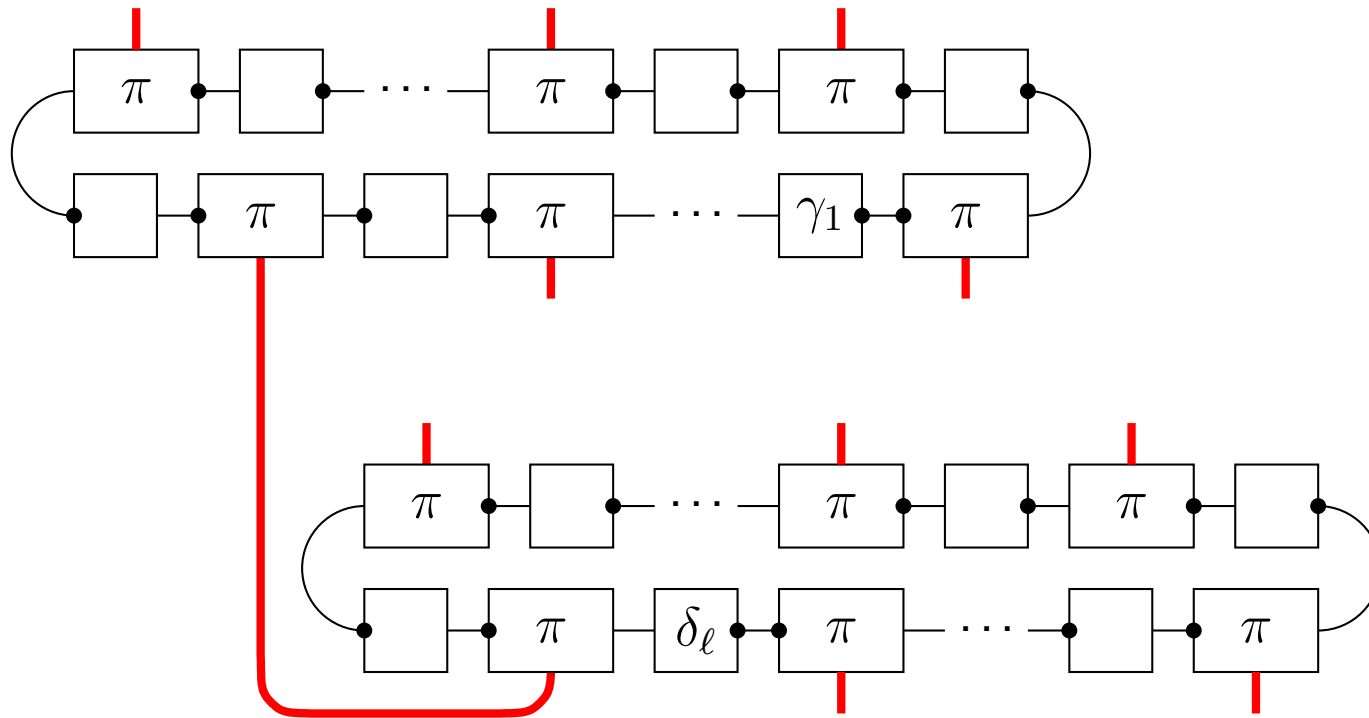
Translation is compatible with reduction



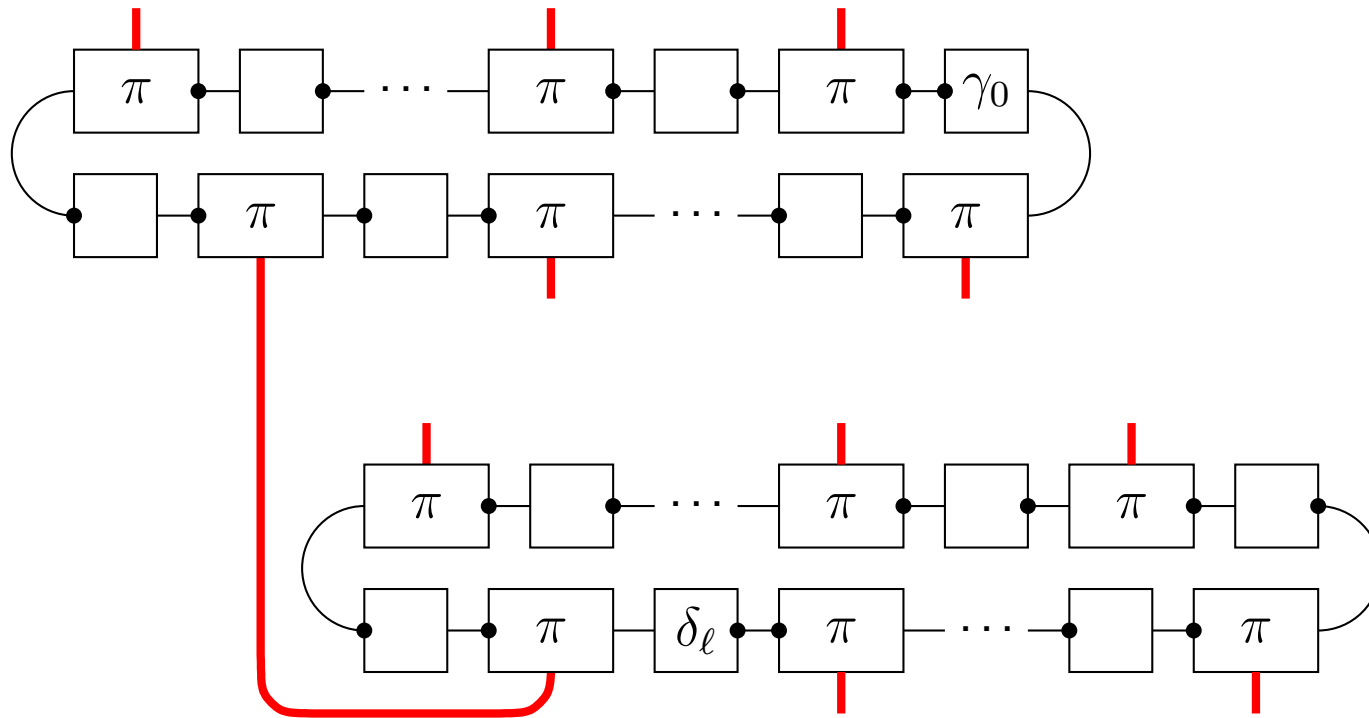
Translation is compatible with reduction



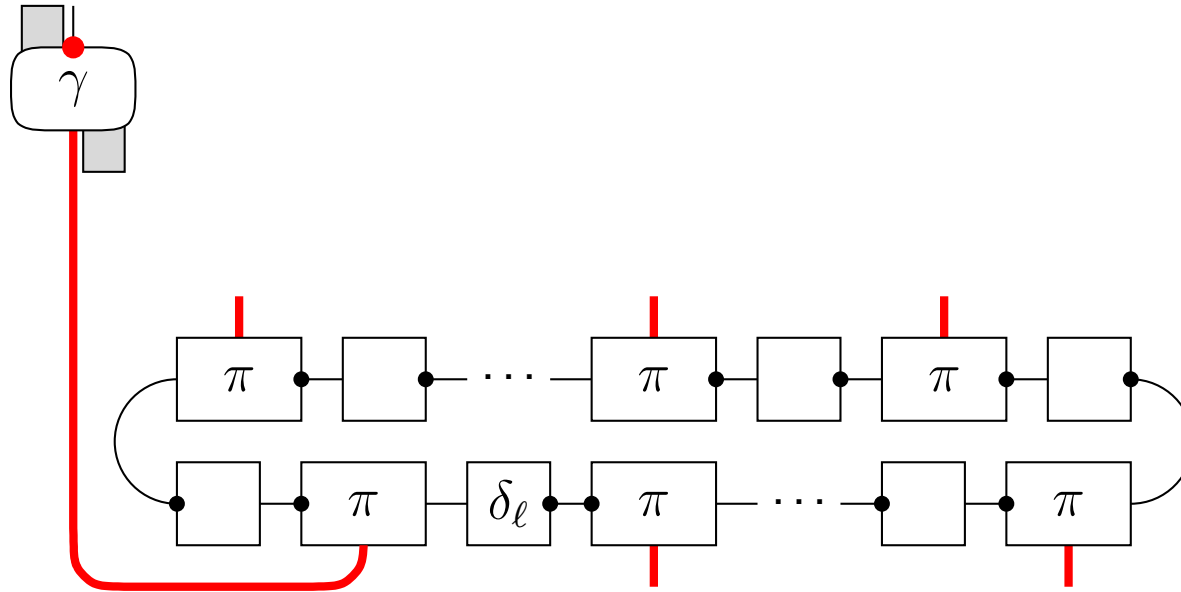
Translation is compatible with reduction



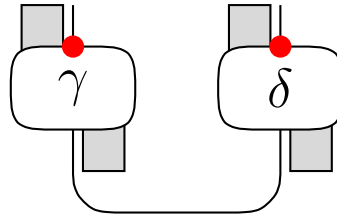
Translation is compatible with reduction



Translation is compatible with reduction



Translation is compatible with reduction



A first answer and few questions...

We have given a *universal* system with four symbols (0,1,+,-) and seven rules.

A first answer and few questions...

We have given a *universal* system with four symbols (0,1,+,-) and seven rules.

- Universal hard interaction
 - Minimality of this universal system
 - Components for asynchronous computer architecture

A first answer and few questions...

We have given a *universal* system with four symbols (0,1,+,-) and seven rules.

- Universal hard interaction
 - Minimality of this universal system
 - Components for asynchronous computer architecture
- Hard interaction
 - Detection of deadlocks
 - Comparison with other graph relabeling systems