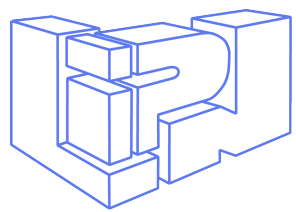
Proof circuits and other models of parallel computation

Logic and interactions 2012

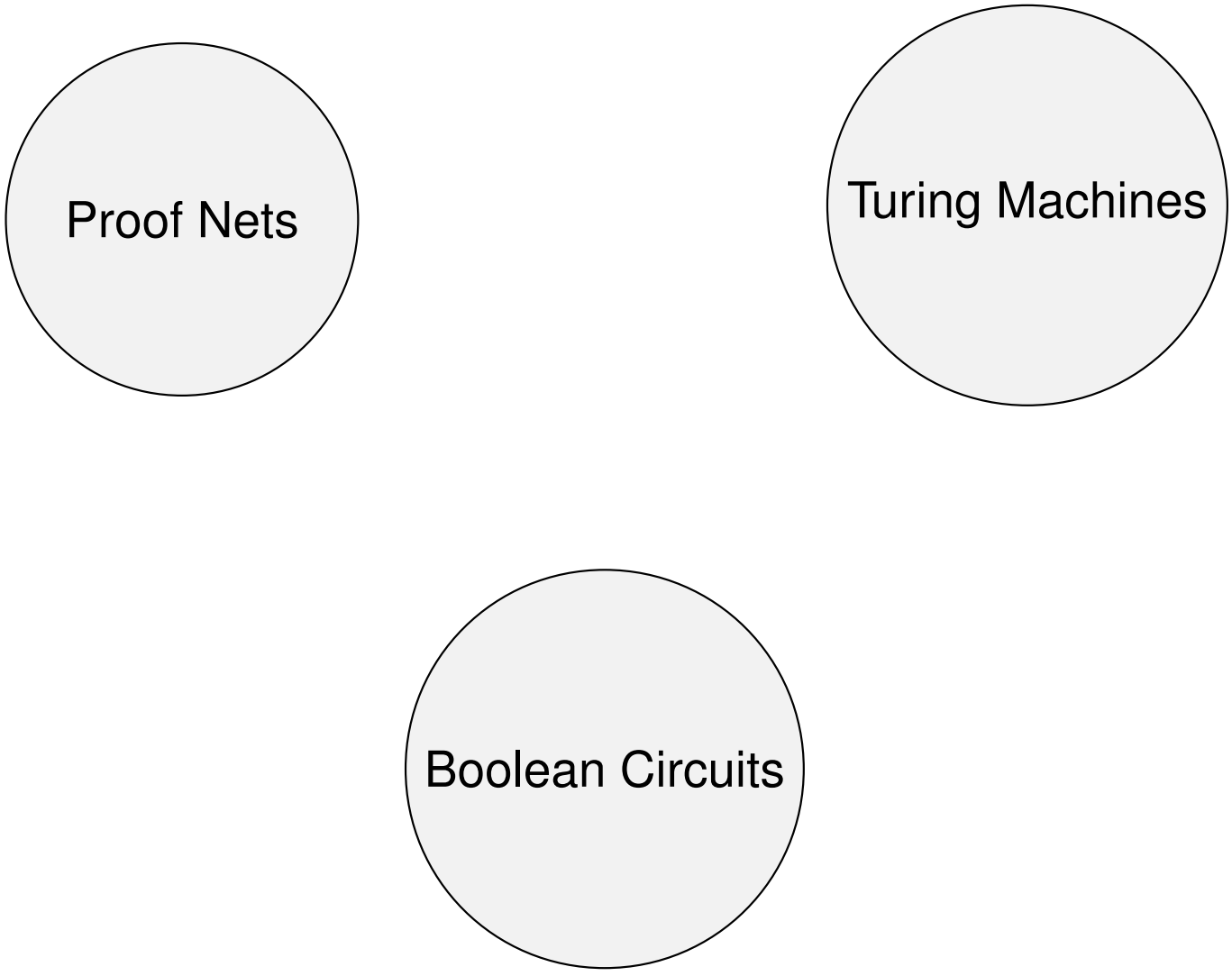
Clément Aubert
aubert@lipn.univ-paris13.fr



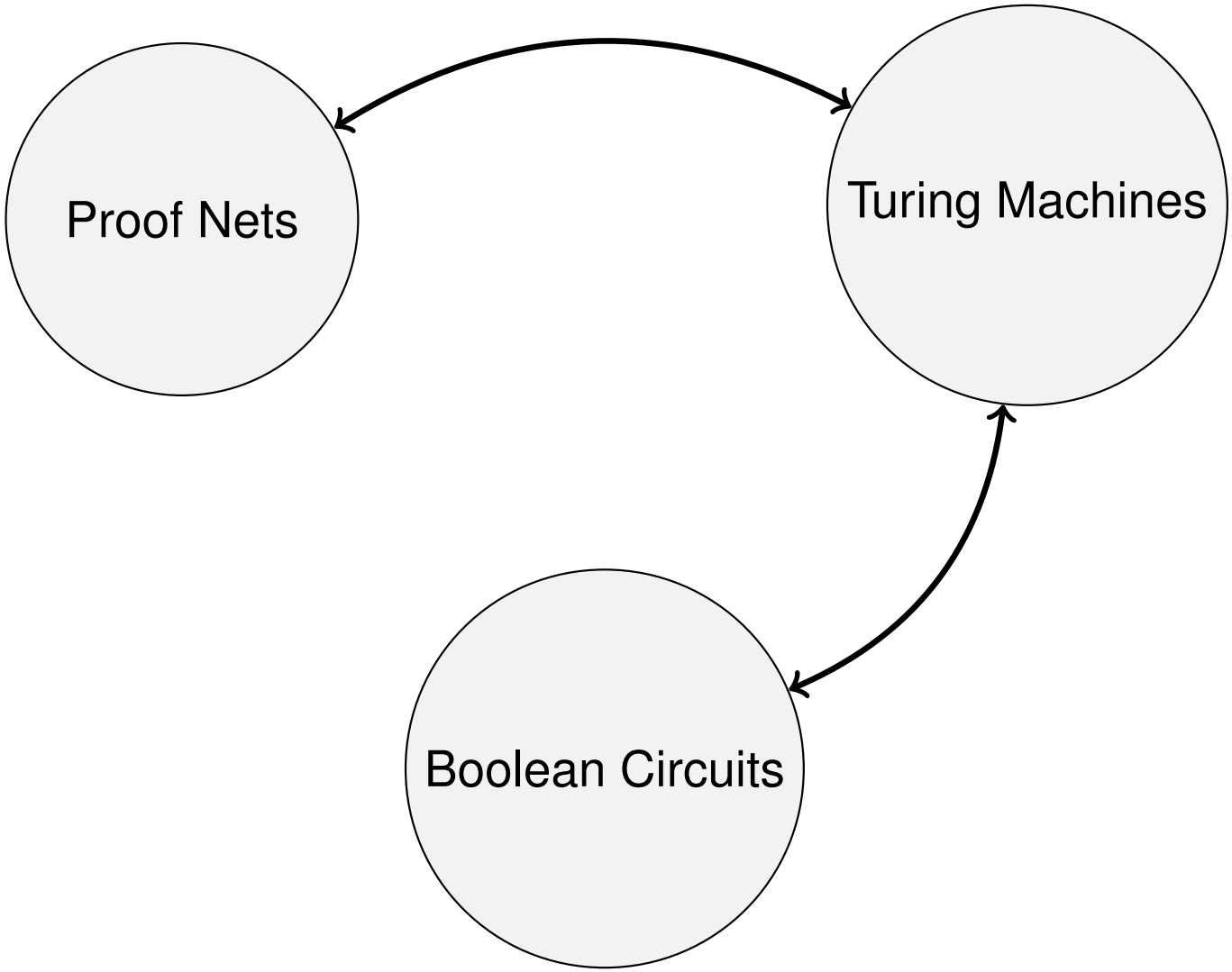
Institut Galilée - Université Paris-Nord
99, avenue Jean-Baptiste Clément
93430 Villetaneuse

Thursday 2 February 2012

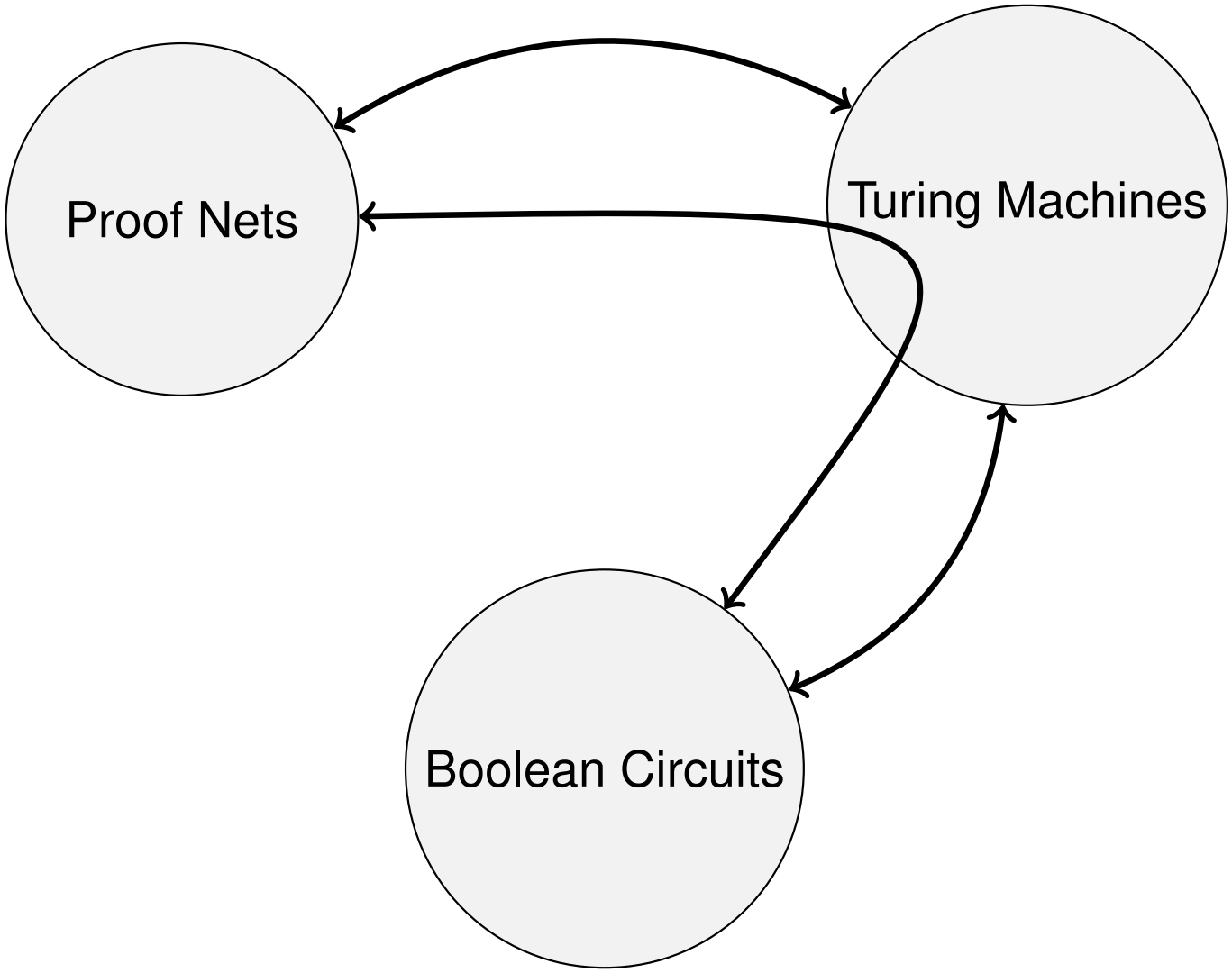
The big picture



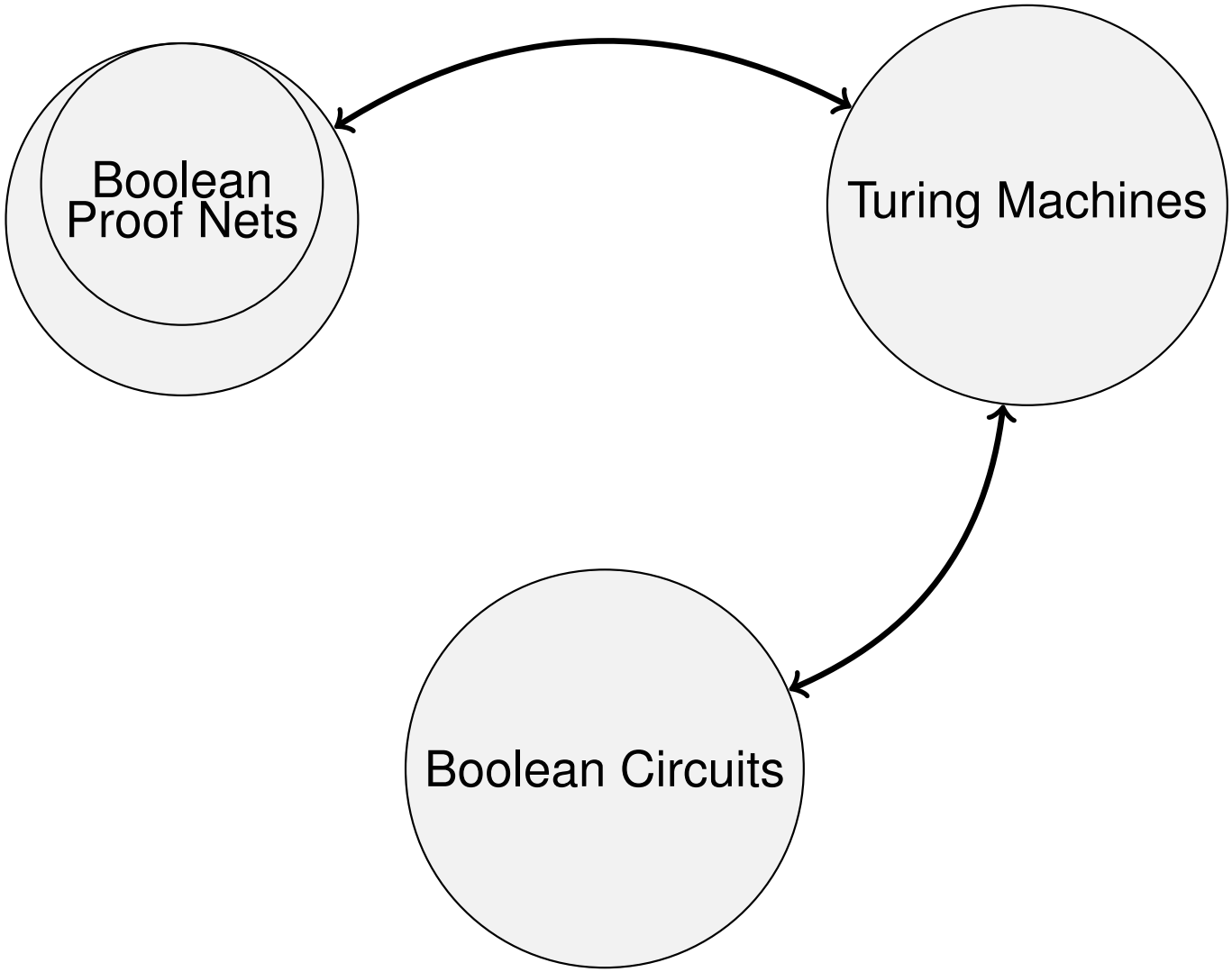
The big picture



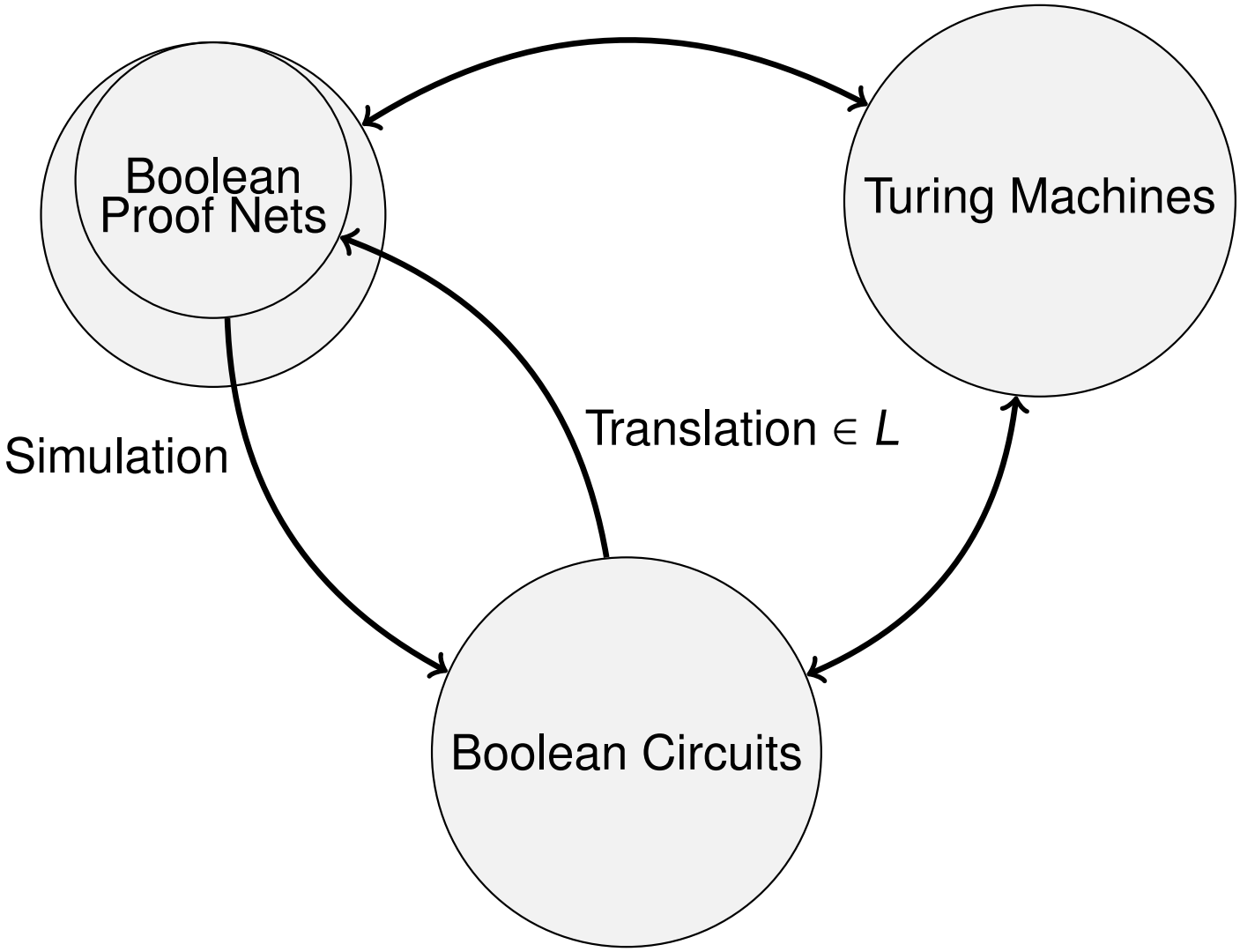
The big picture



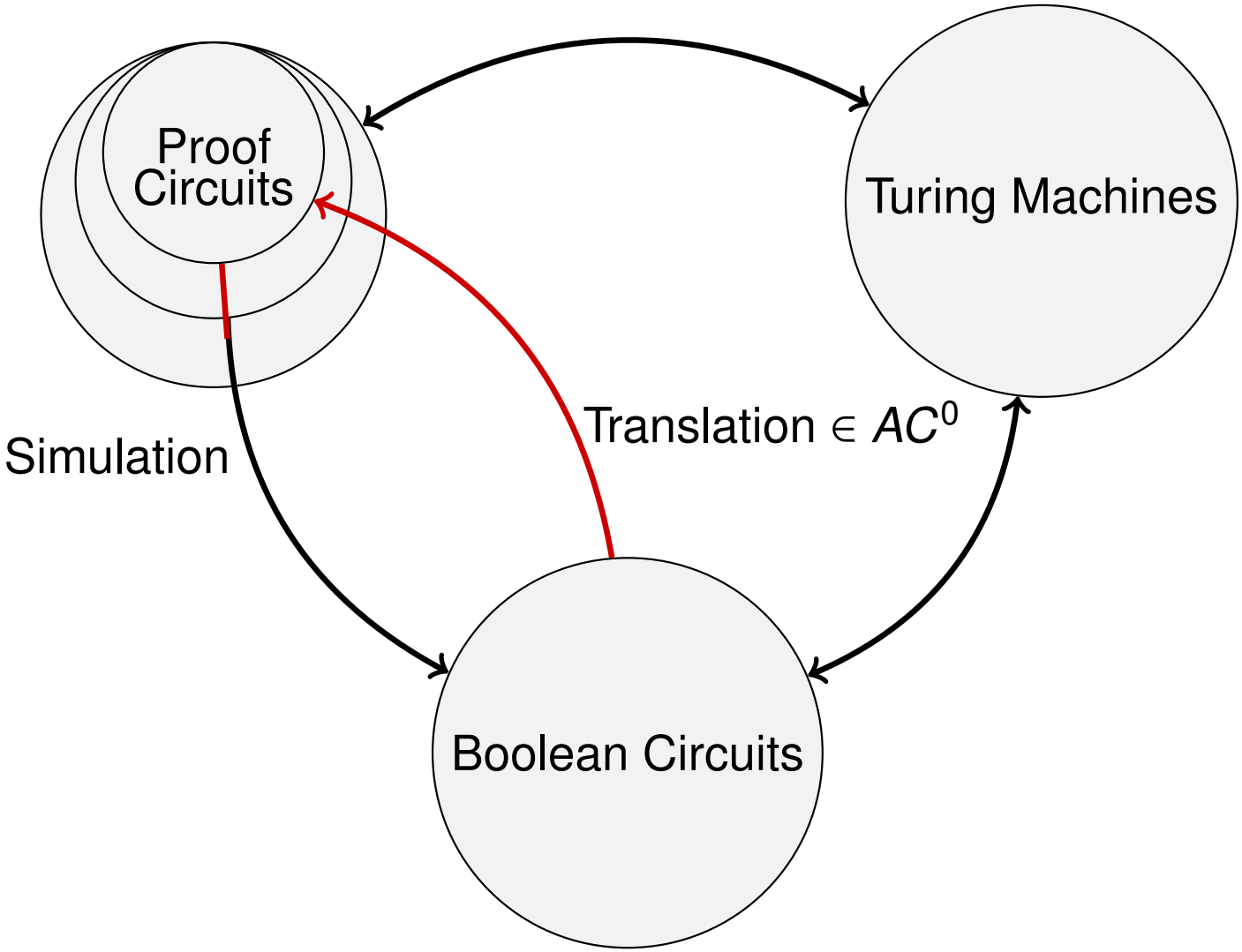
The big picture



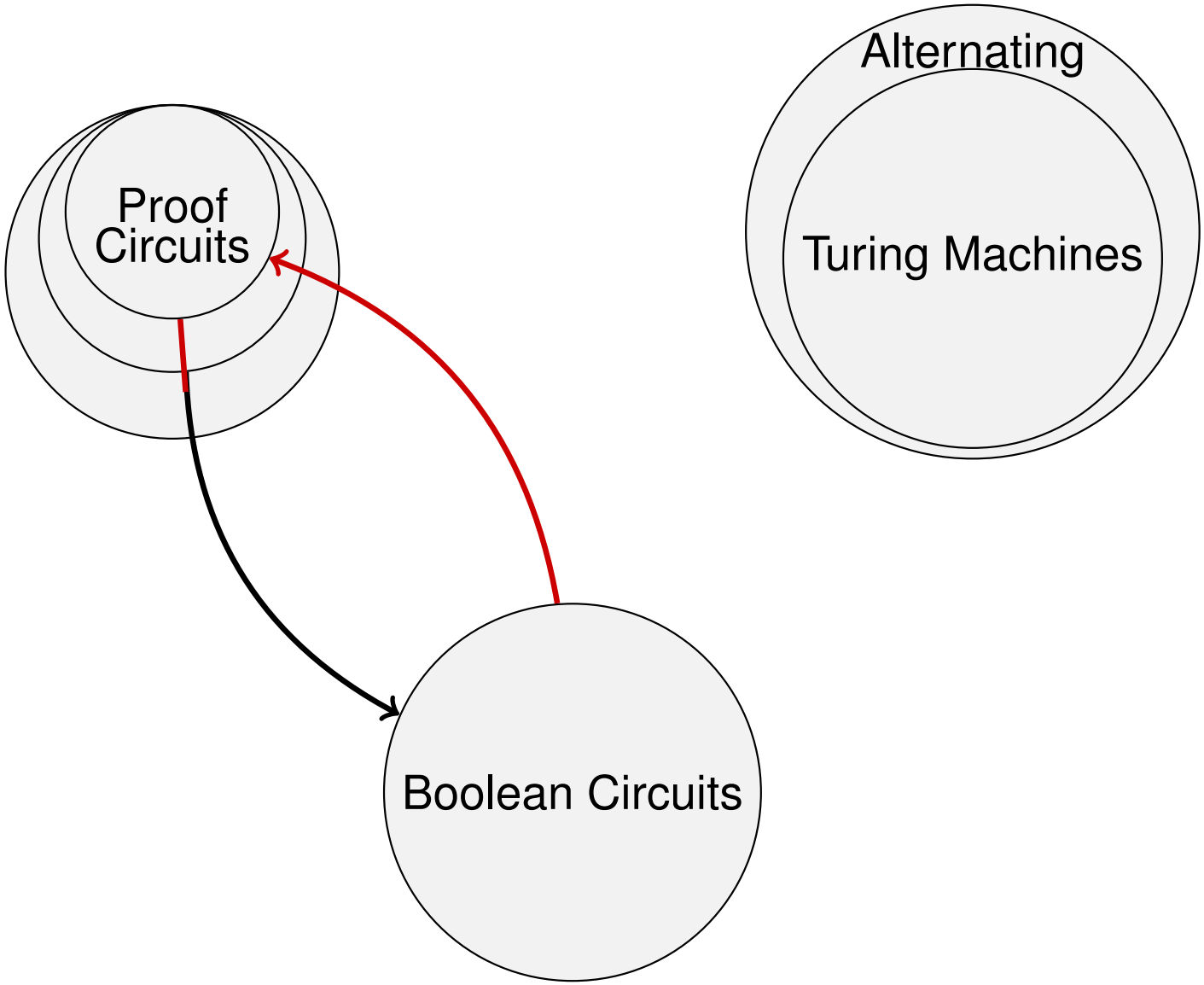
The big picture



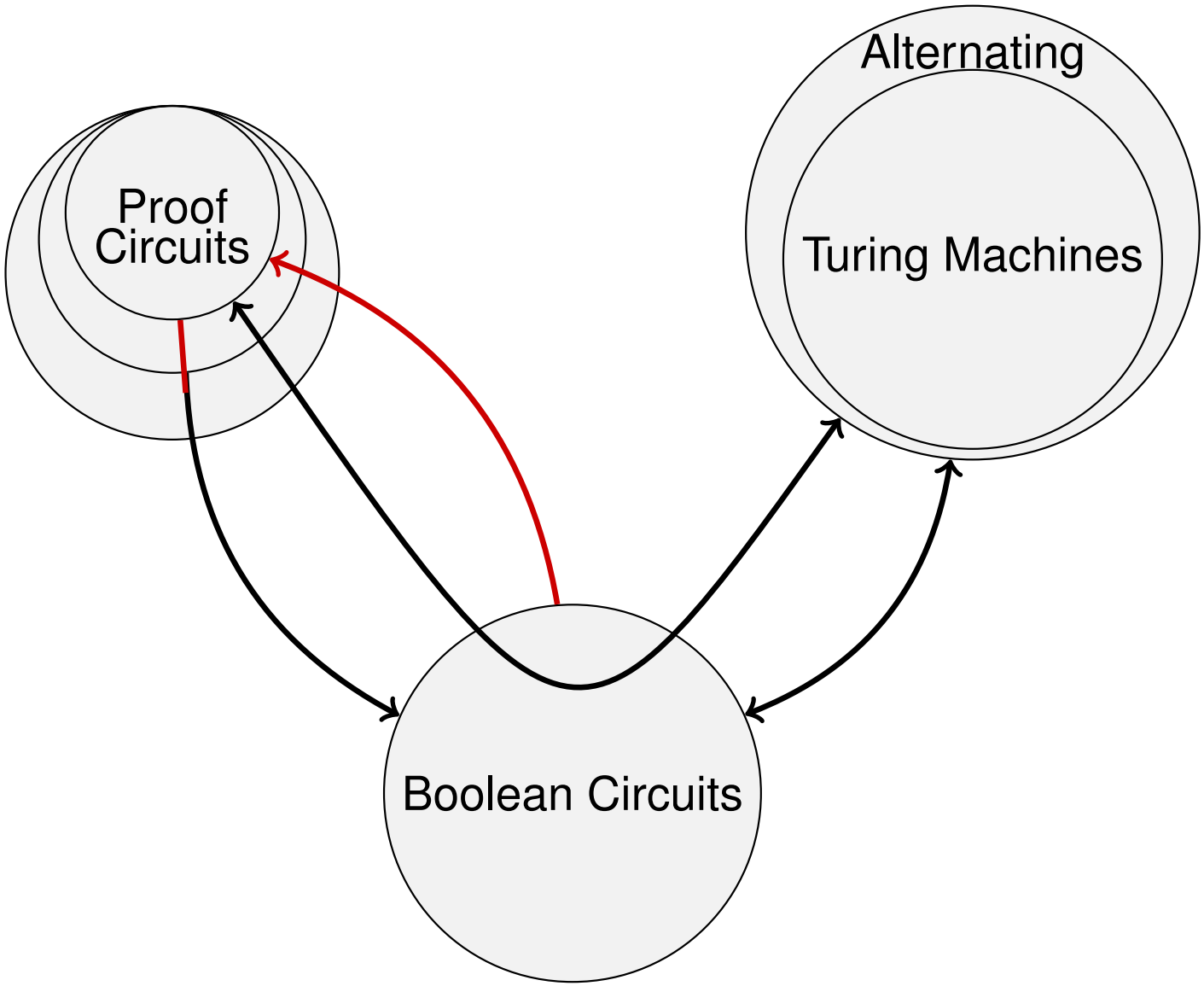
The big picture



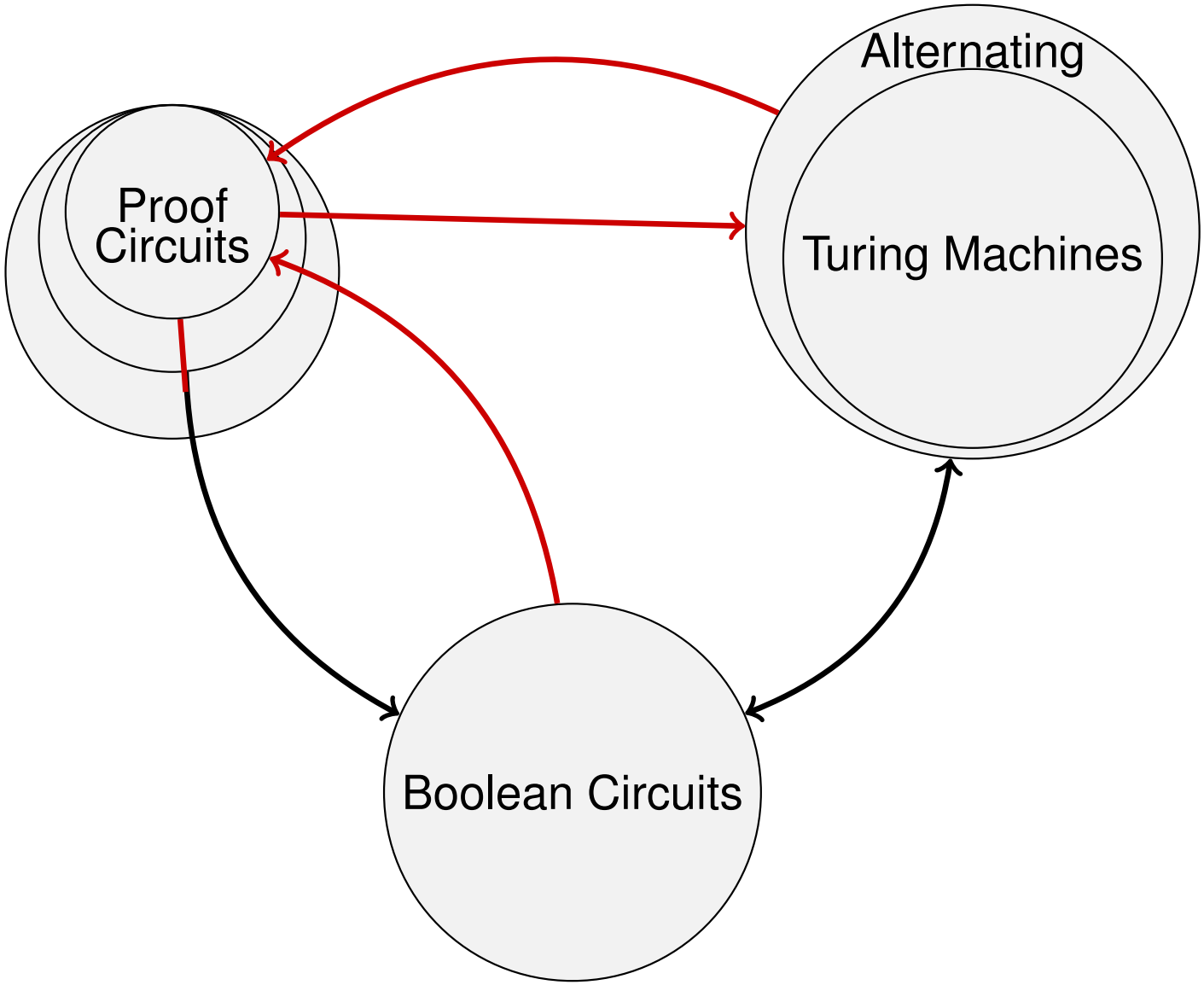
The big picture



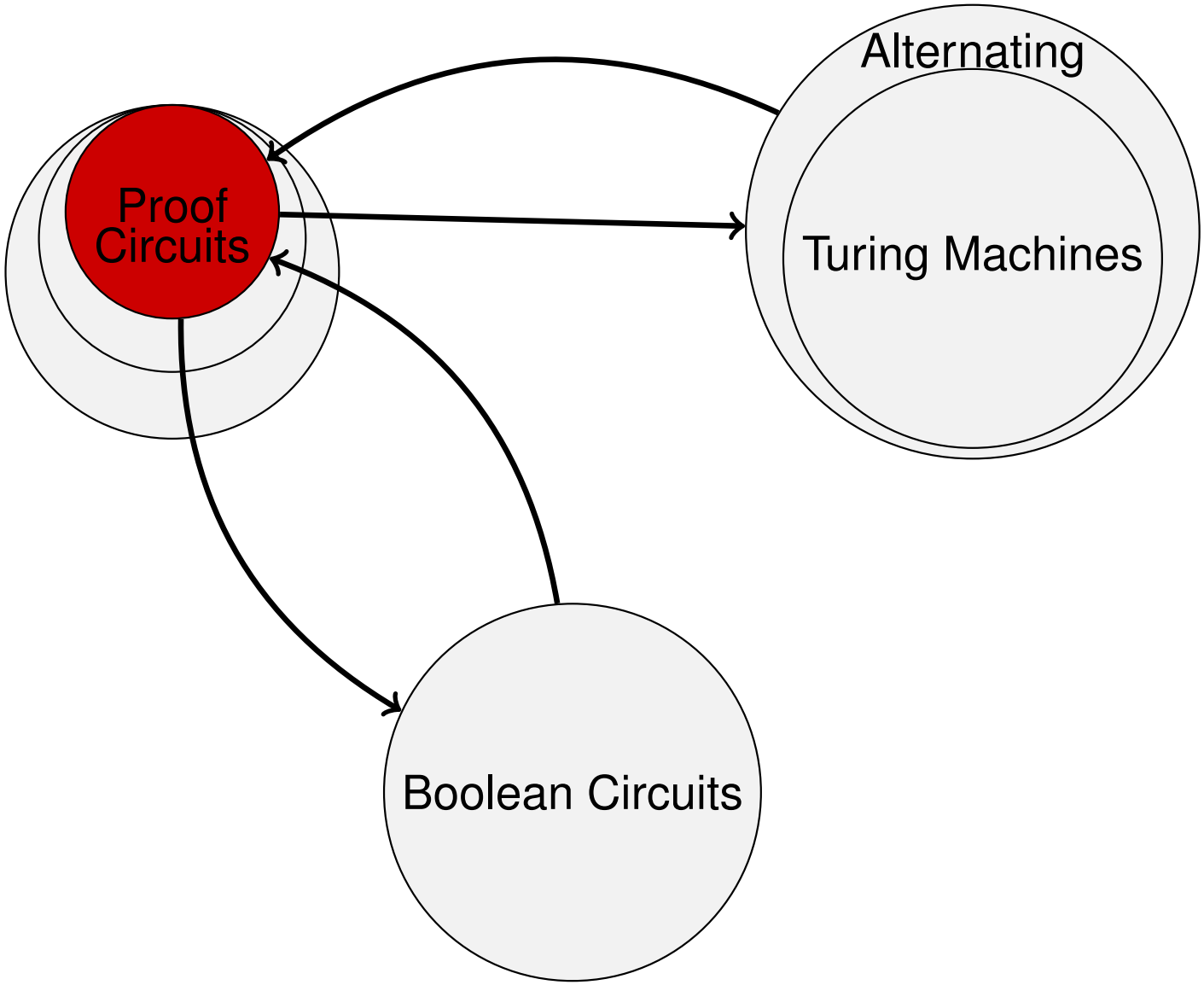
The big picture



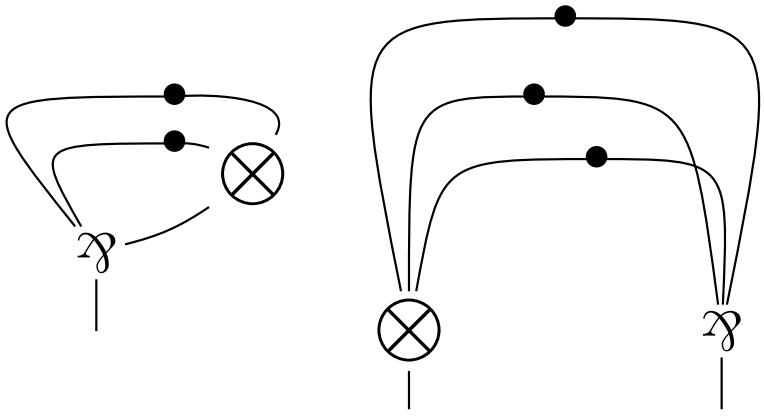
The big picture



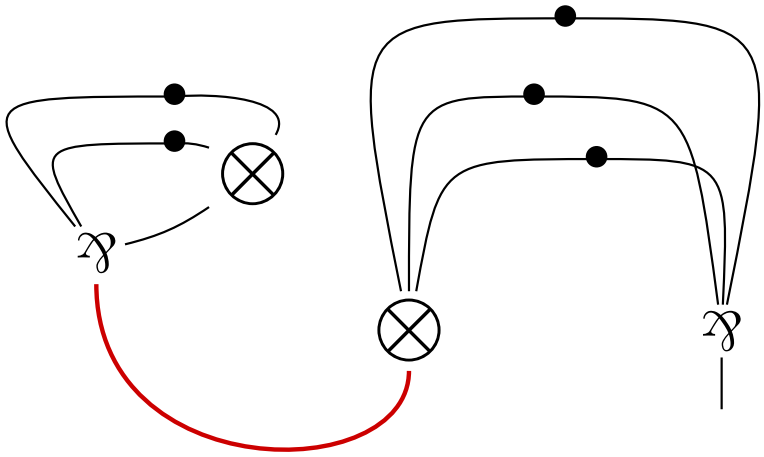
Map (1~5): Proof Circuits



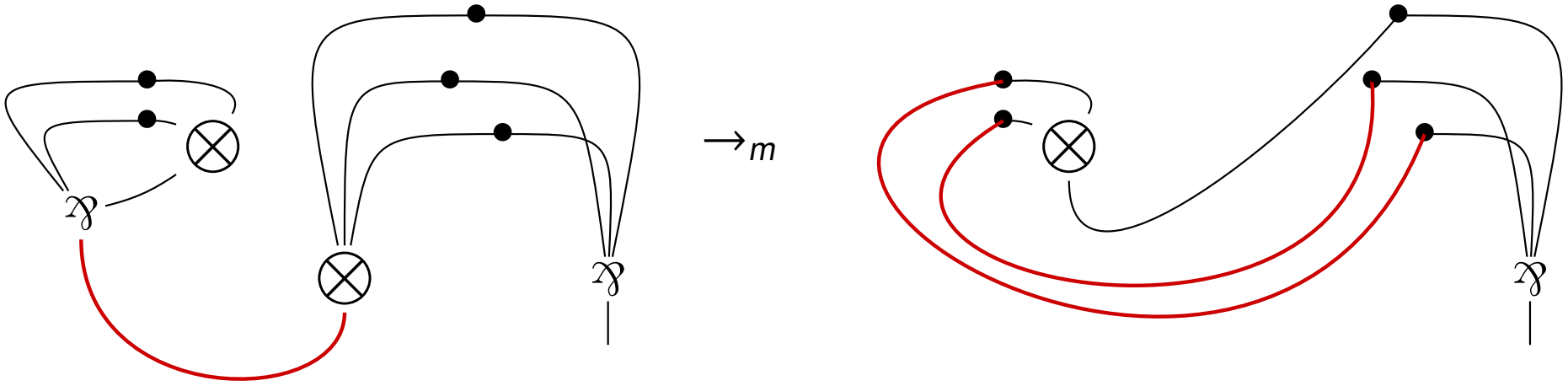
How to compute with Proof nets?

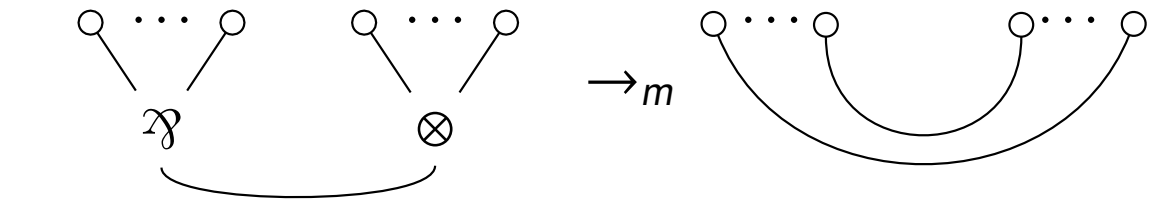


How to compute with Proof nets?

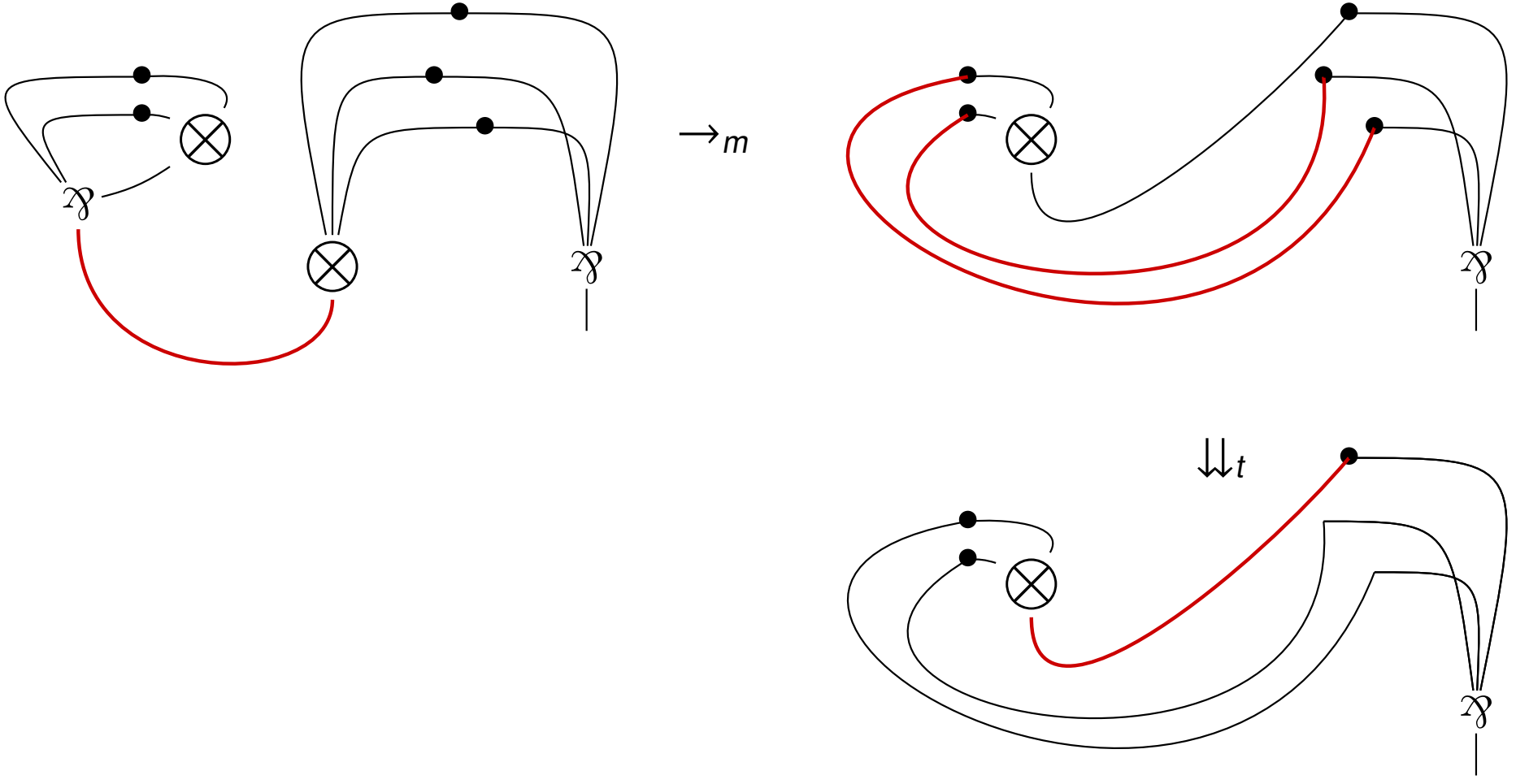


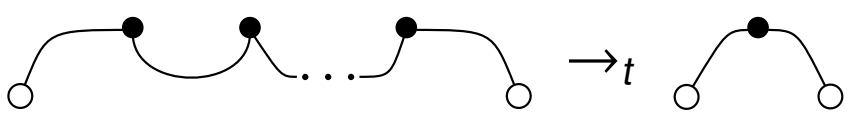
How to compute with Proof nets?



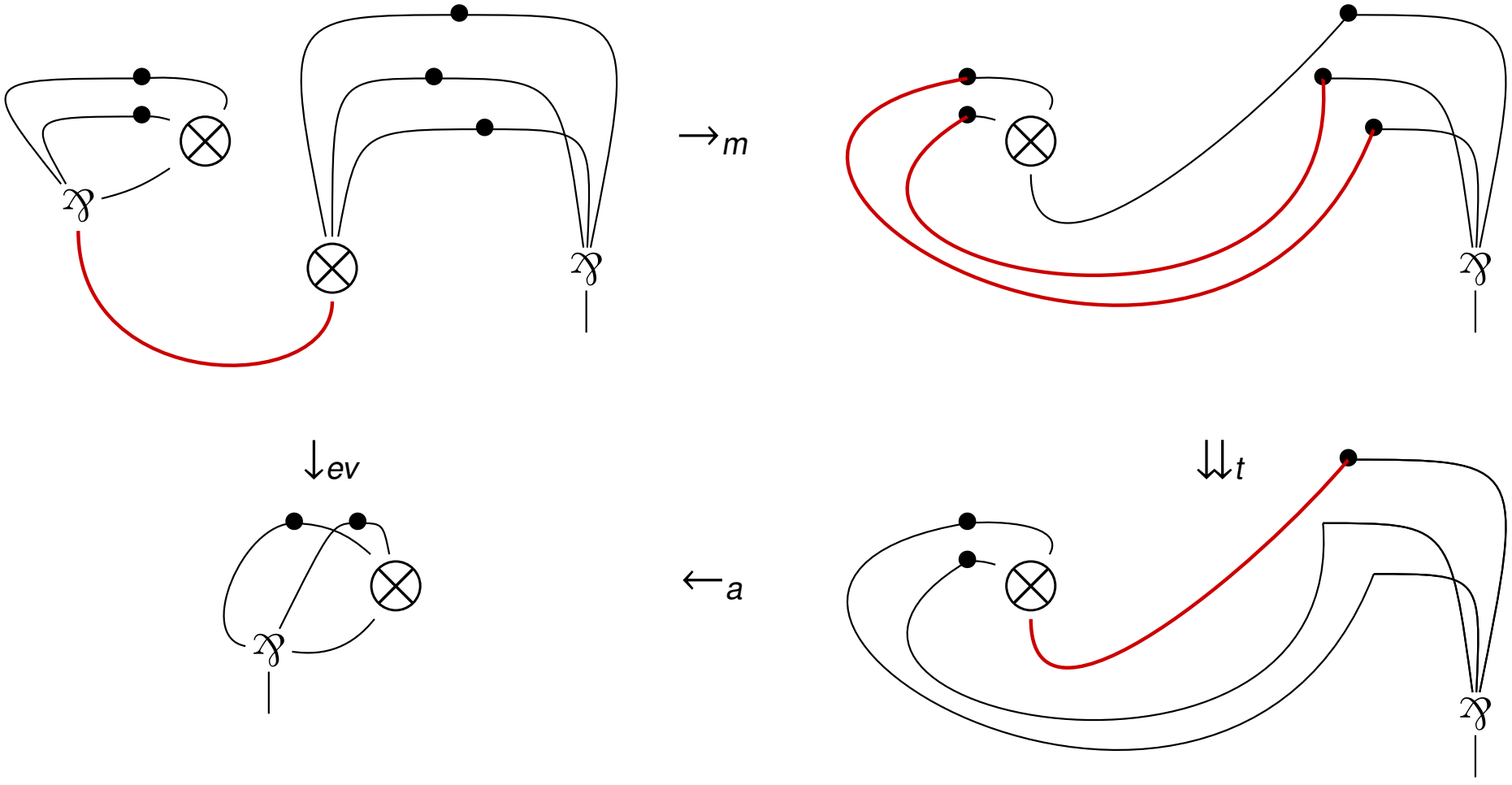


How to compute with Proof nets?

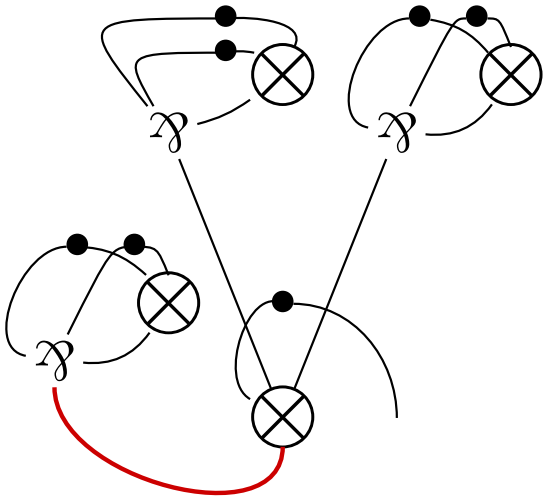




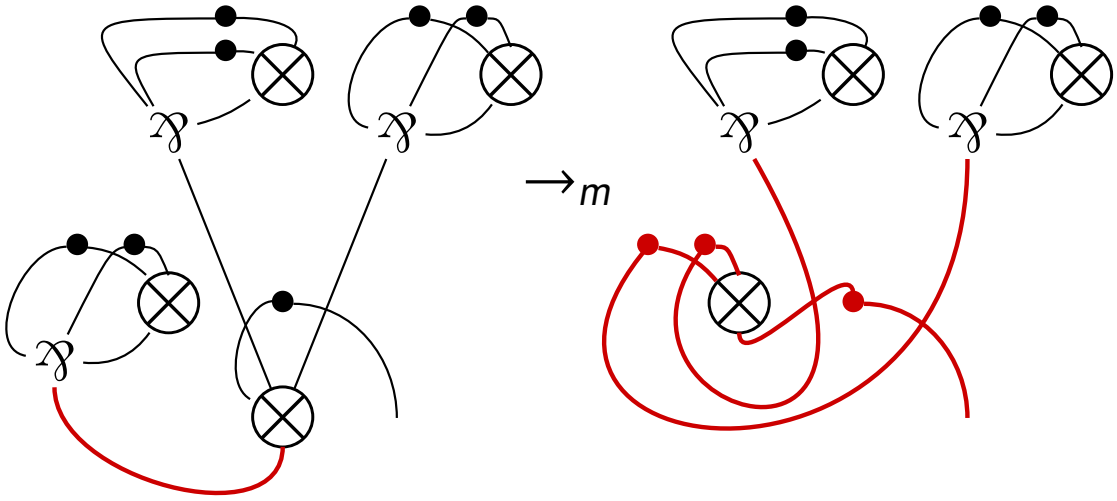
How to compute with Proof nets?



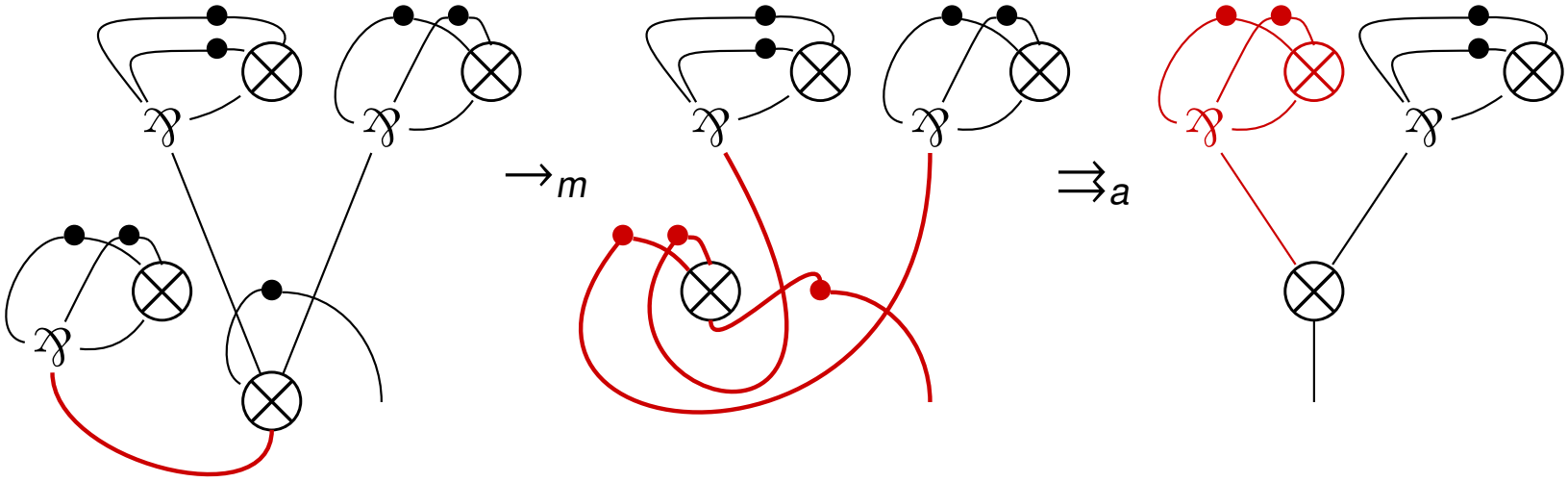
An example: the conditional

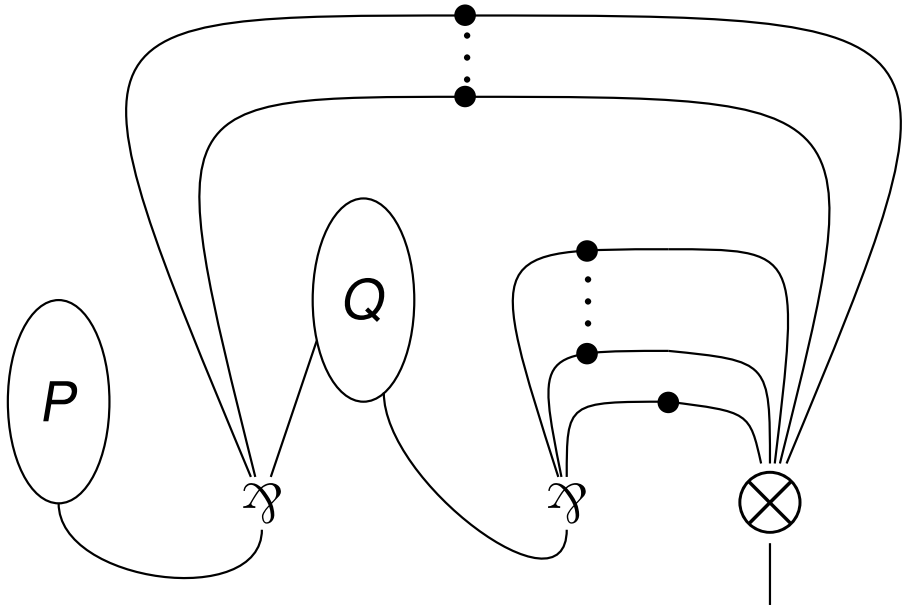


An example: the conditional

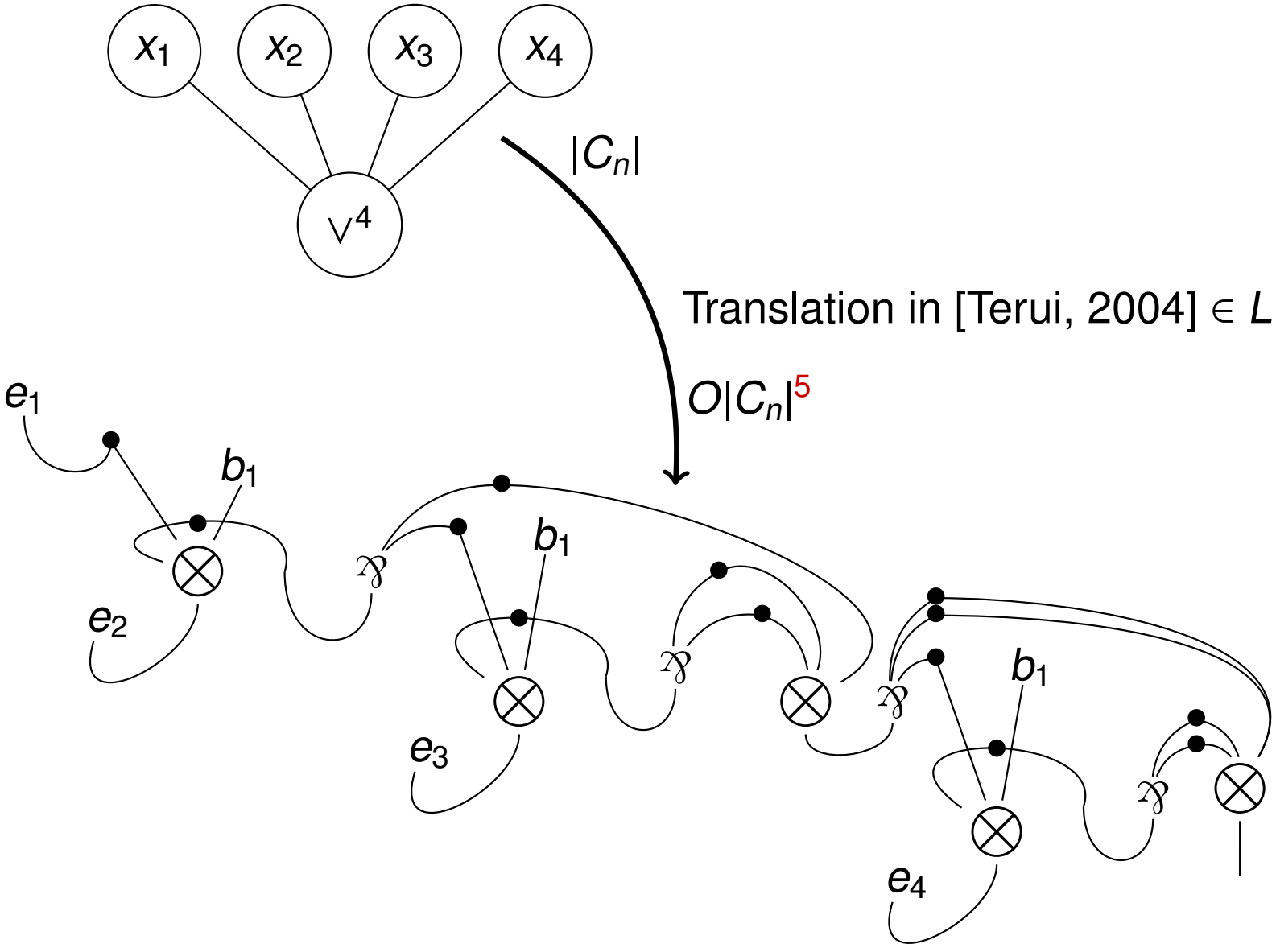


An example: the conditional

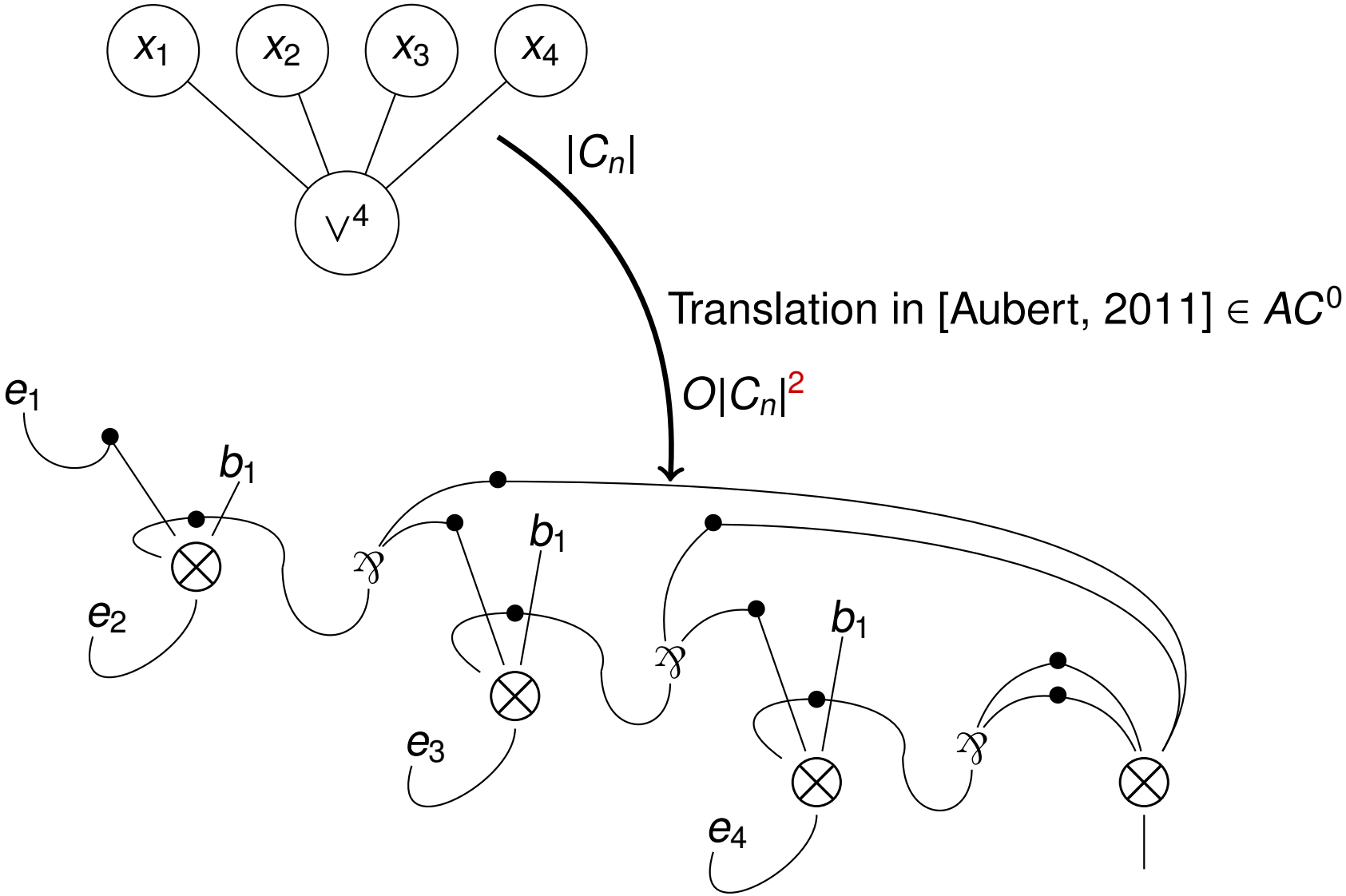




$\vee^4$ translated

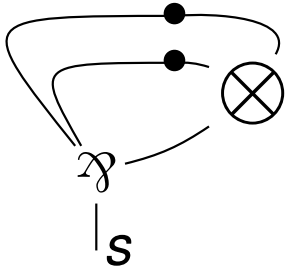


$\vee^4$ translated

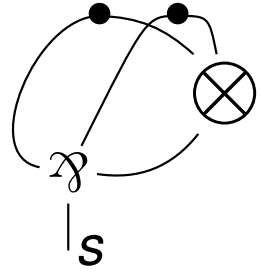


Pieces (1/4)

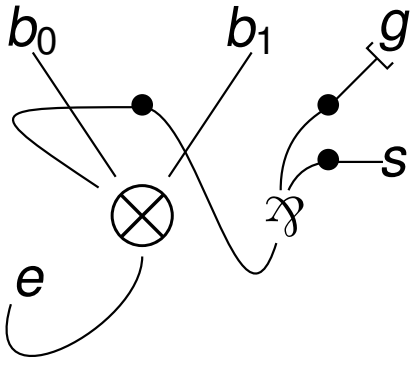
$b_0 \equiv$



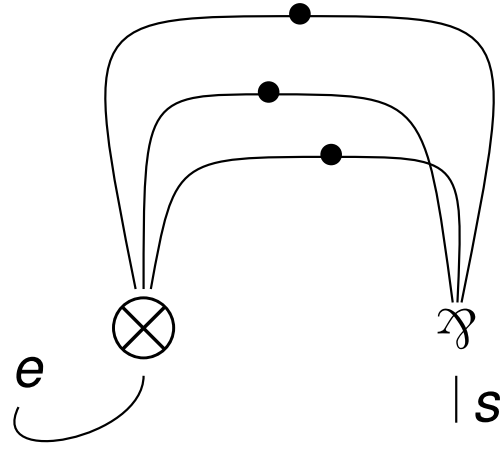
$b_1 \equiv$



COND $\equiv$

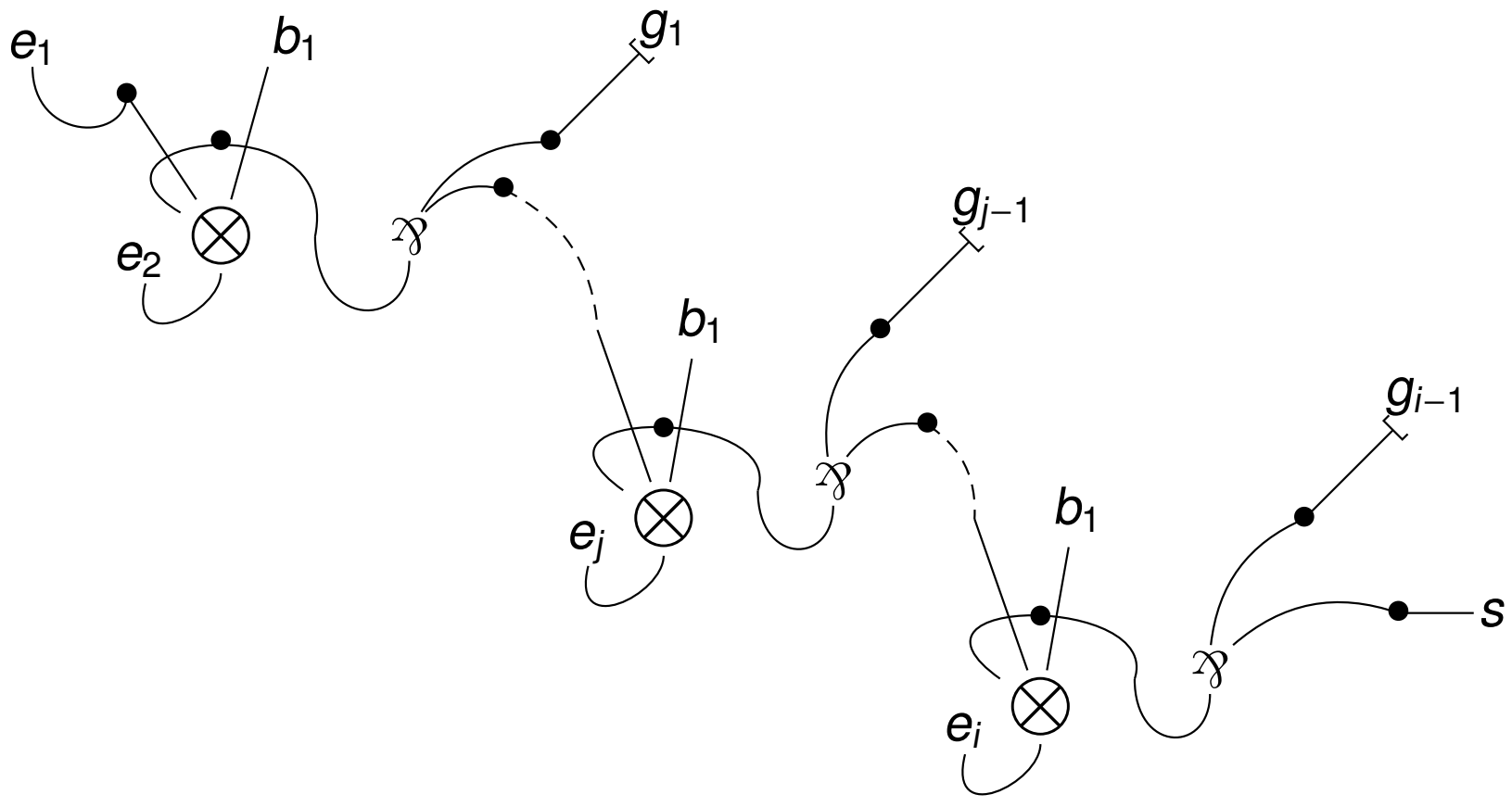


NEG $\equiv$



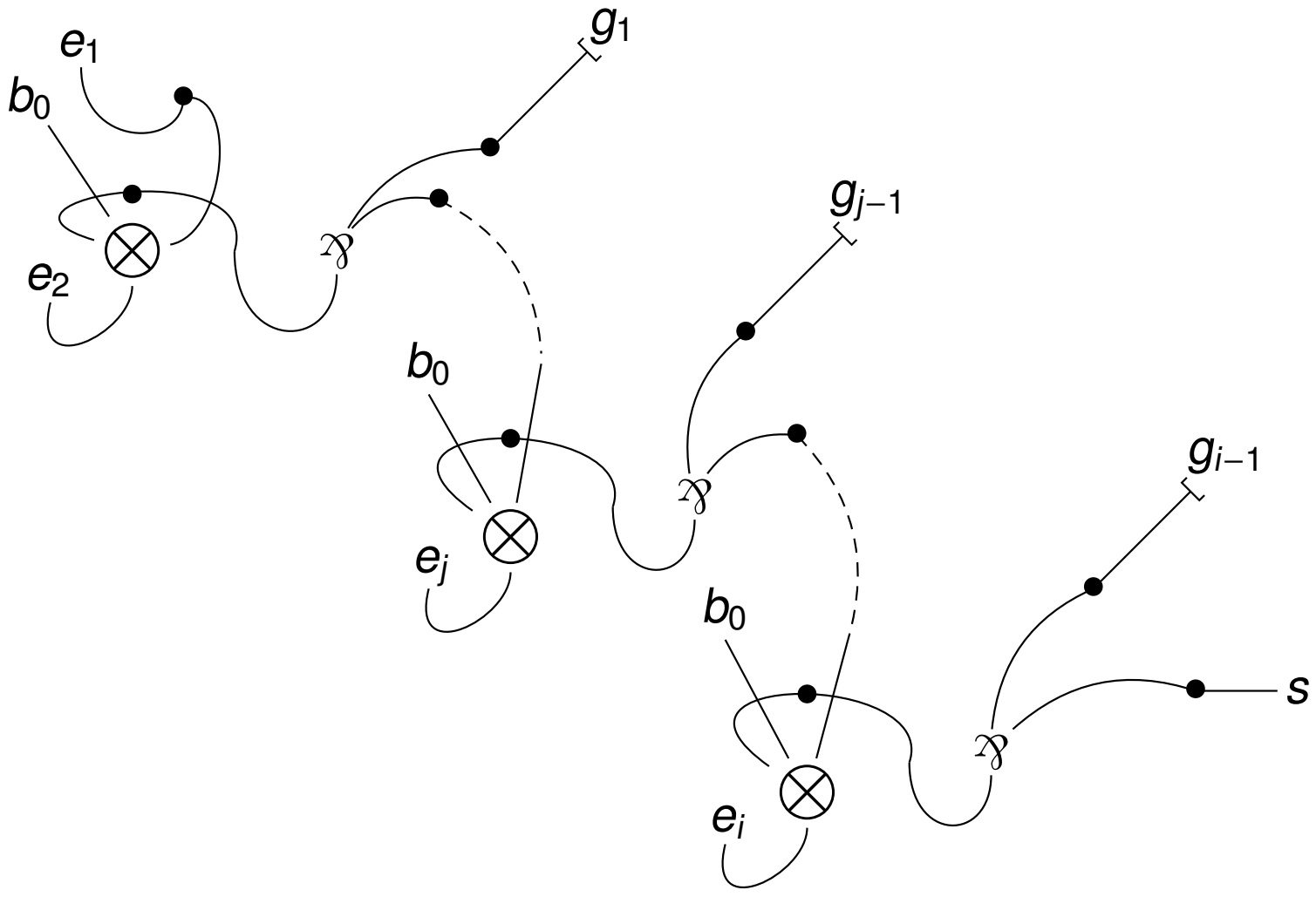
Pieces (2/4)

$DISJ^i \equiv$



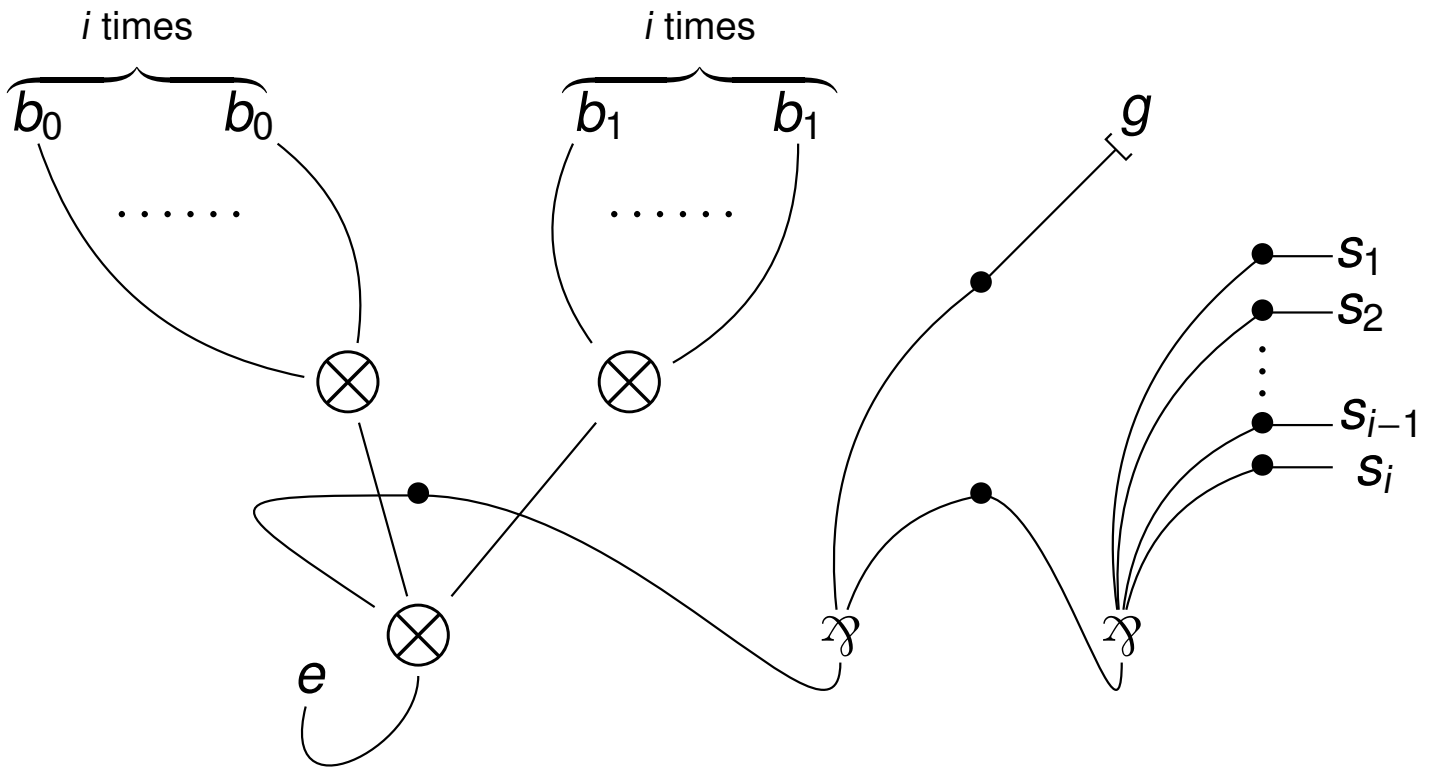
Pieces (3/4)

$CONJ^i \equiv$

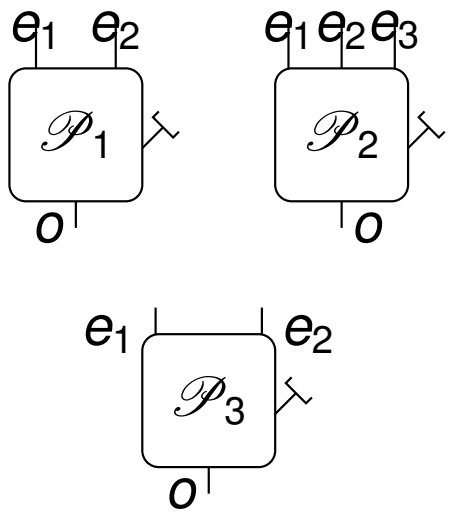


Pieces (4/4)

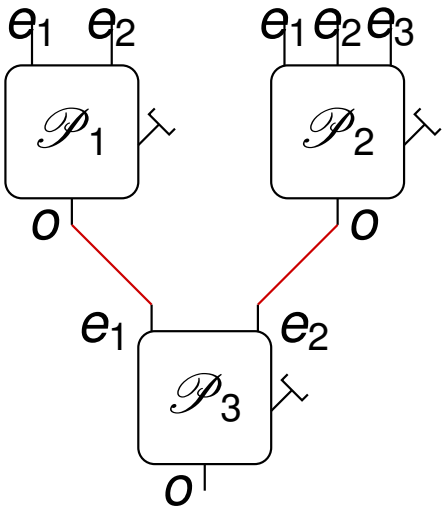
$DUPL^{i} \equiv$



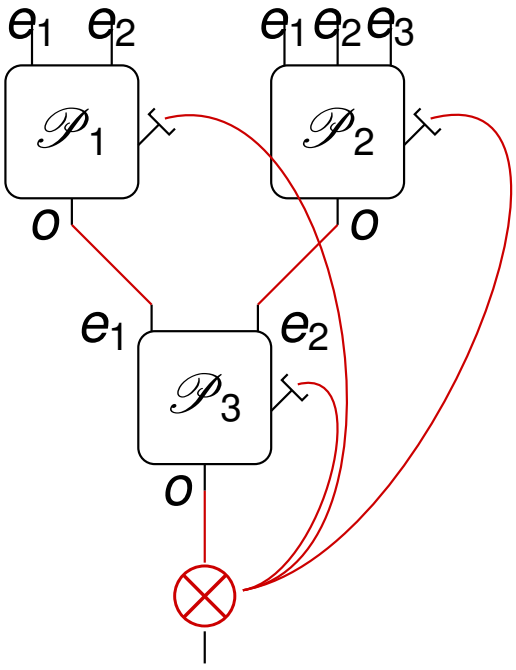
Proof Circuits



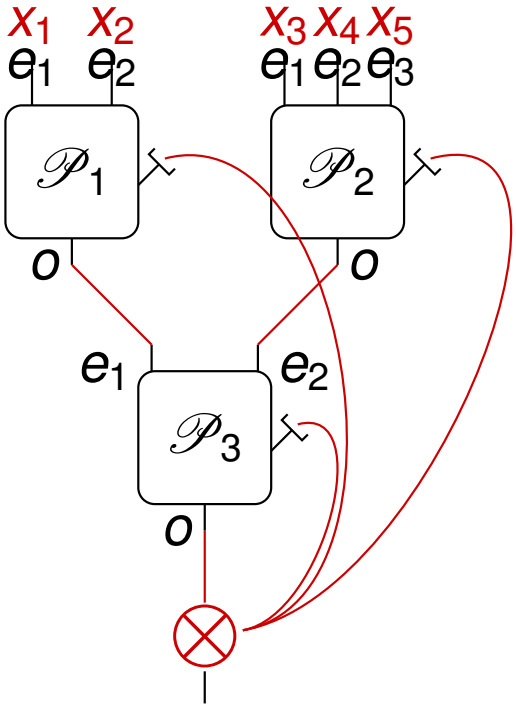
Proof Circuits



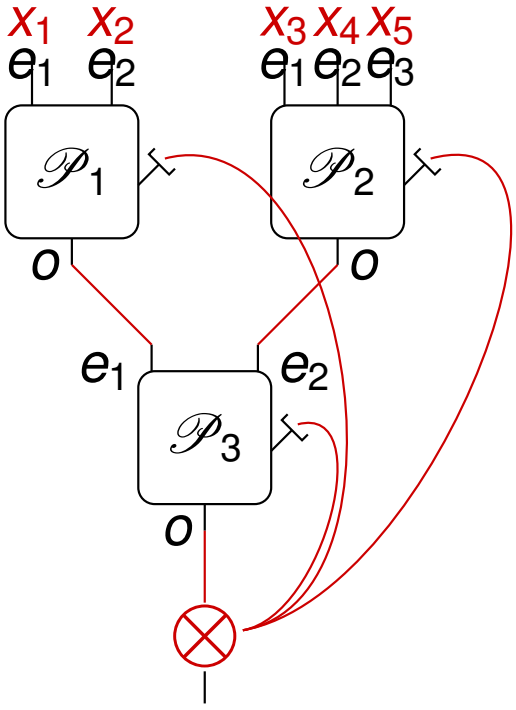
Proof Circuits

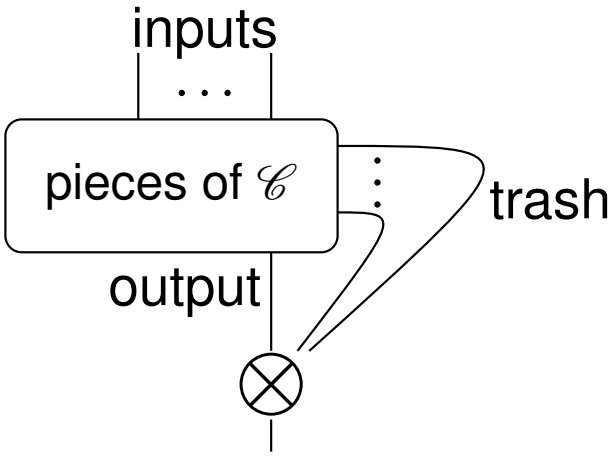


Proof Circuits

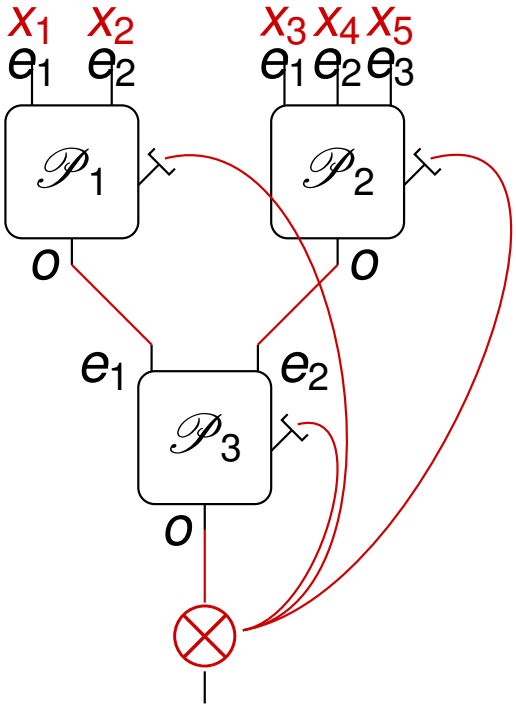


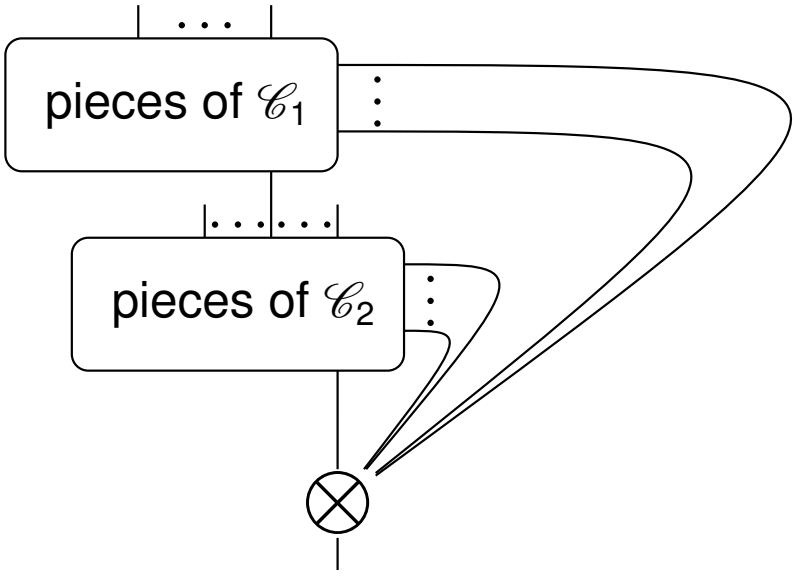
Proof Circuits



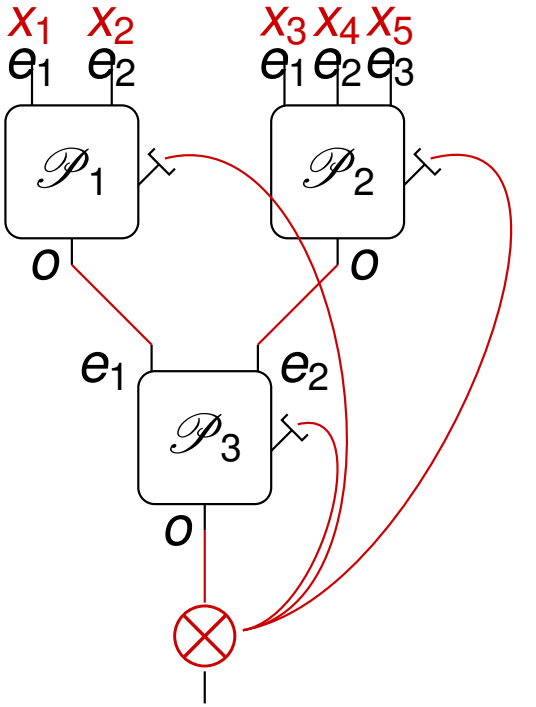


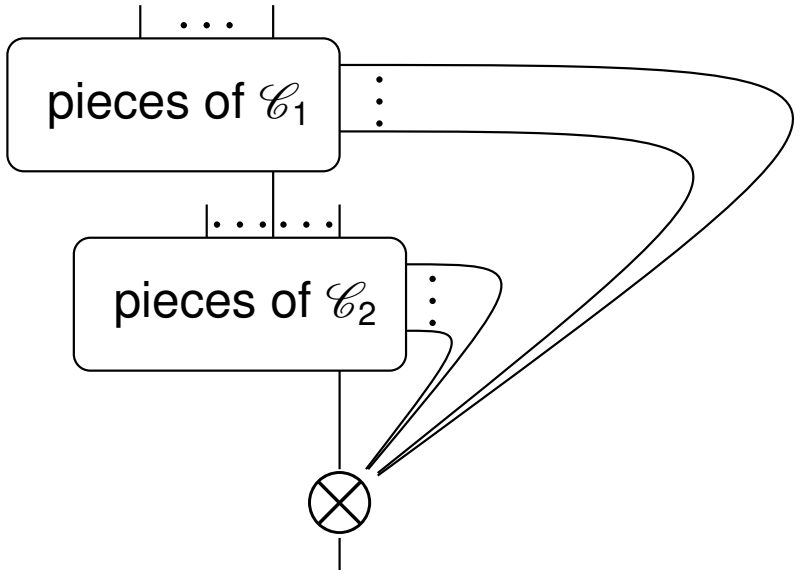
Proof Circuits

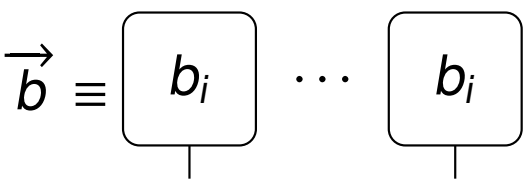


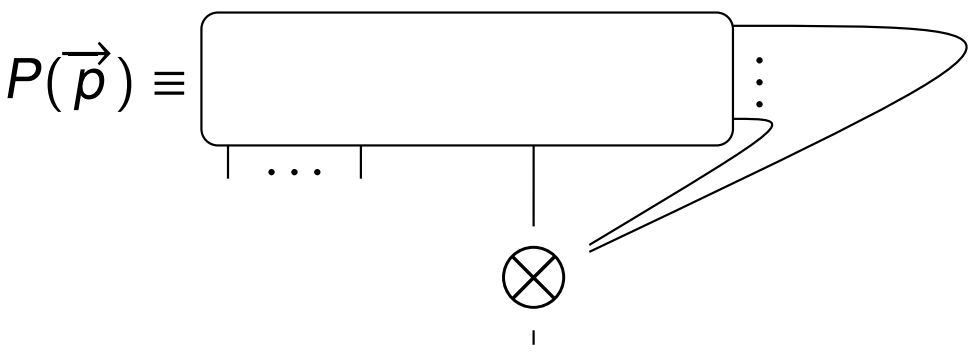


Proof Circuits

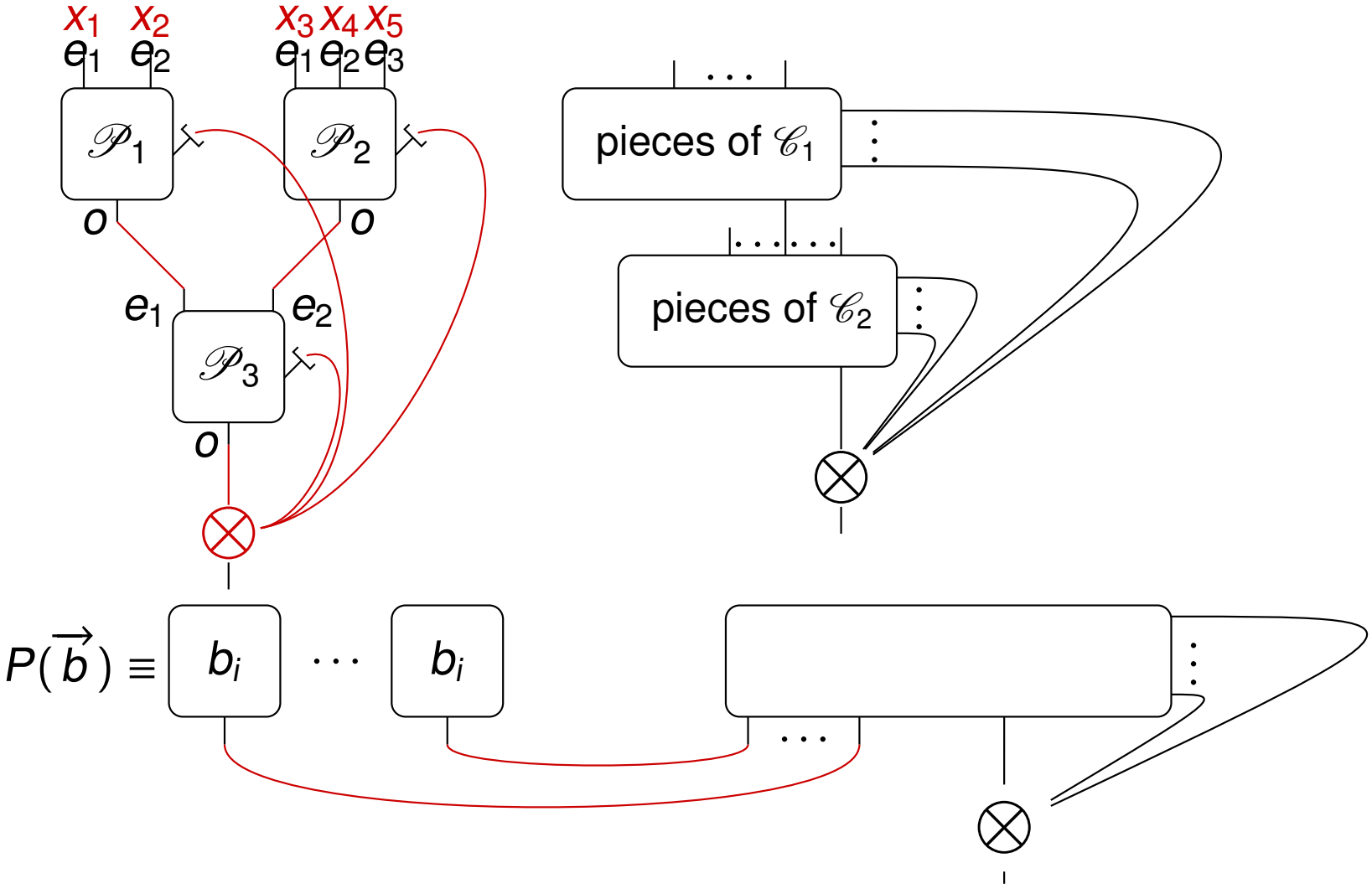




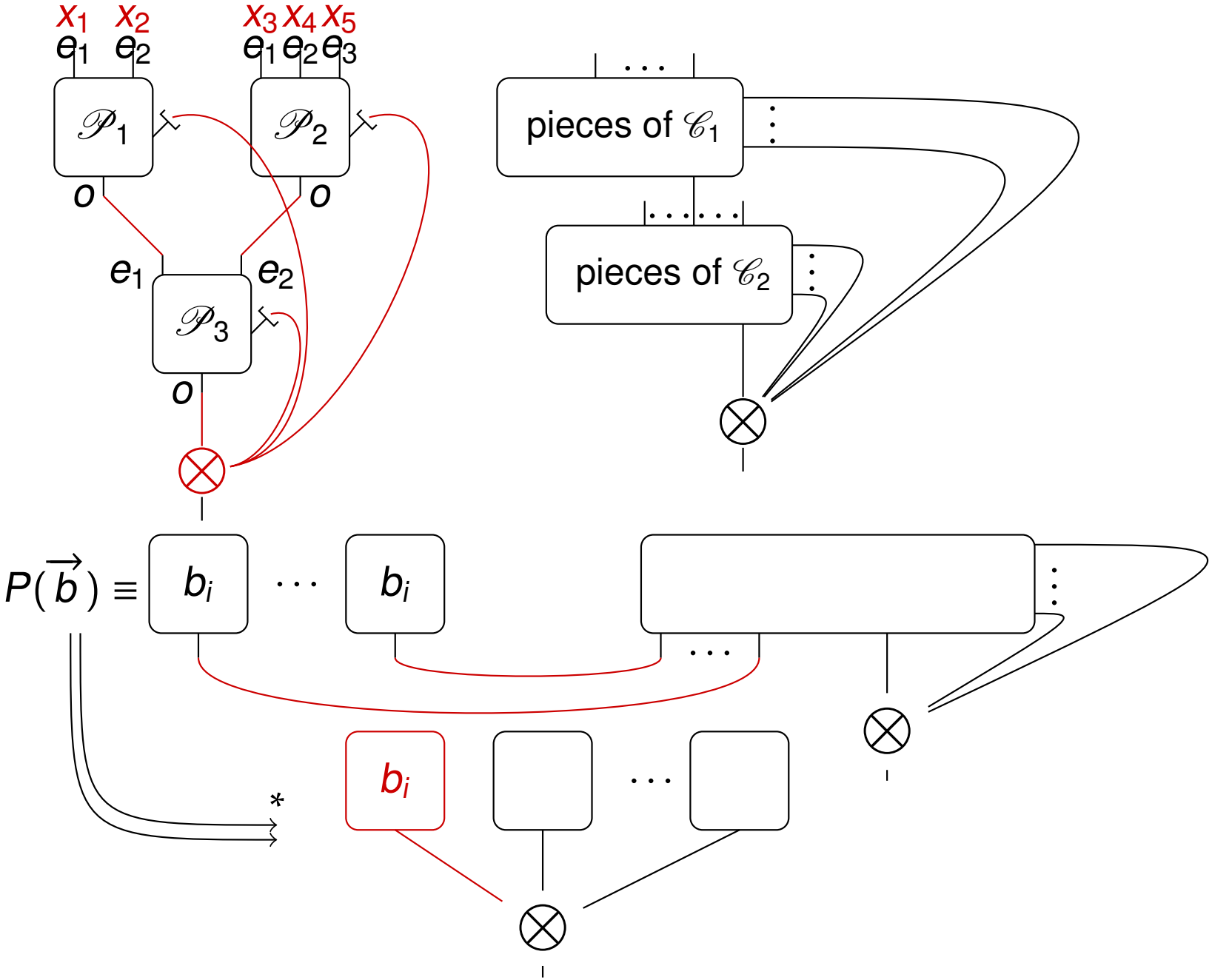




Proof Circuits



Proof Circuits



Definition (AC^i)

For $i \in \mathbb{N}$, set of boolean functions computable by a uniform family of Boolean Circuits $C = (C_n)$ where for all C_n

- *its depth is $O(\log^i n)$,*
- *its size is $n^{O(1)}$,*
- *its gate are labeled with functions of $\mathfrak{B}_1 = \{\neg, (\vee^j)_{j\in\mathbb{N}}, (\wedge^j)_{j\in\mathbb{N}}\}$.*

$$\bigcup_{i\in\mathbb{N}} AC^i = AC$$

Formally speaking...

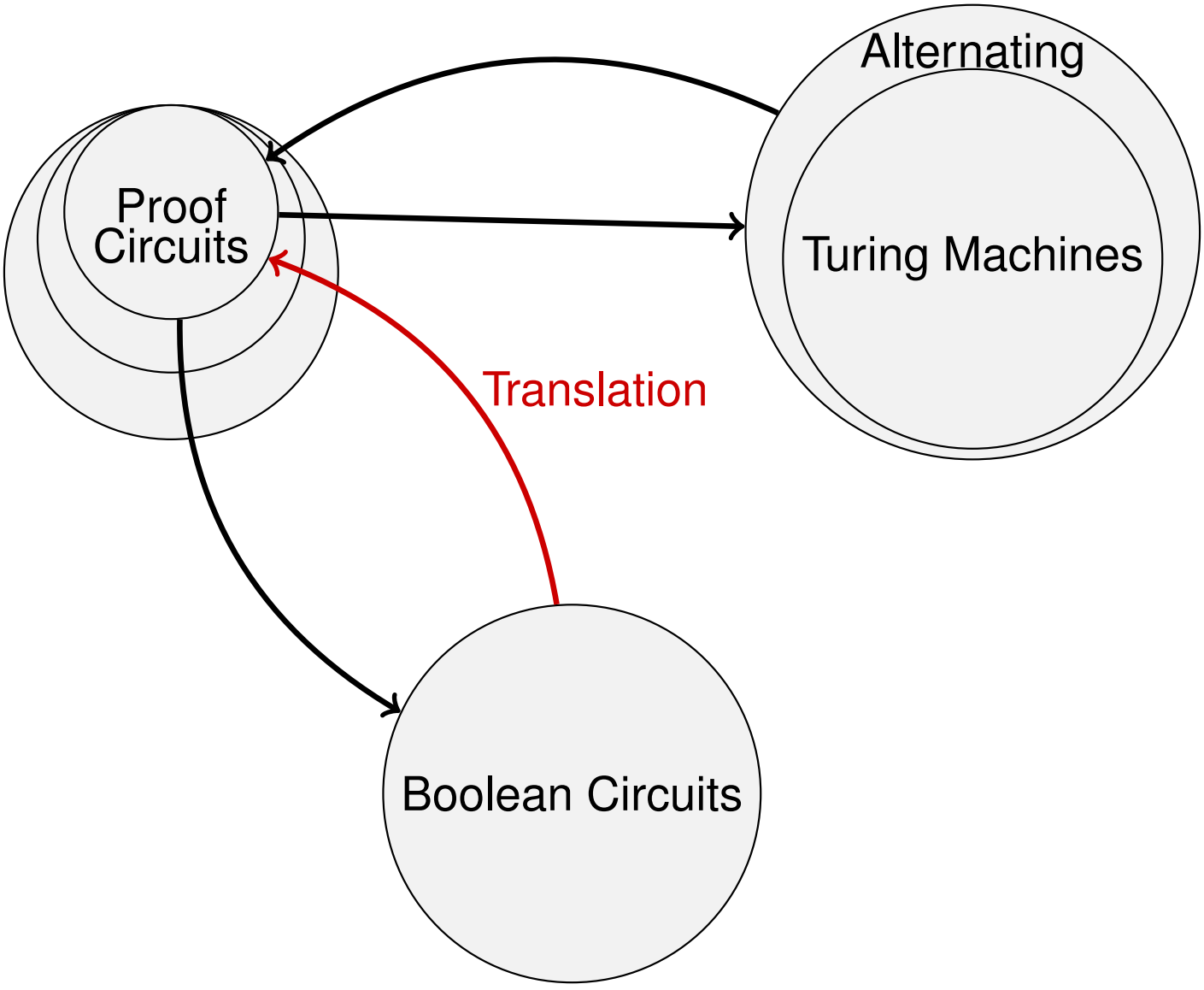
Definition (PCC^i)

For $i \in \mathbb{N}$, set of boolean functions computable by a uniform family of Proof Circuits $P = (P_n)$ where for all P_n

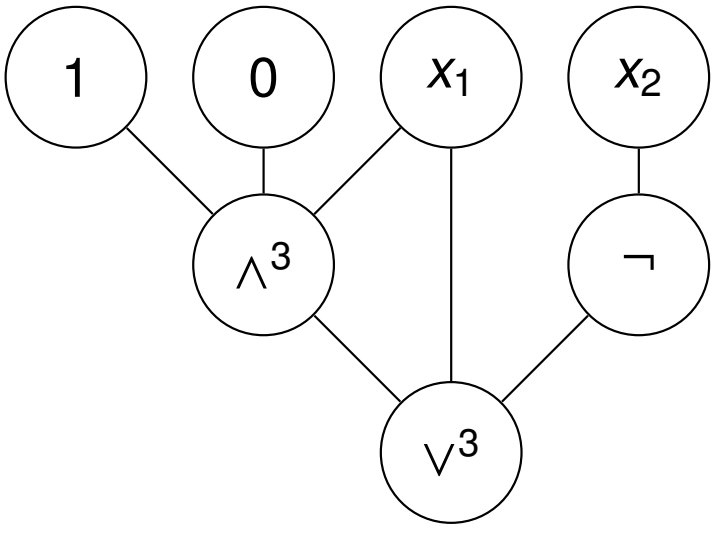
- *its depth is $O(\log^i n)$,*
- *its size is $n^{O(1)}$,*

$$\bigcup_{i \in \mathbb{N}} PCC^i = PCC$$

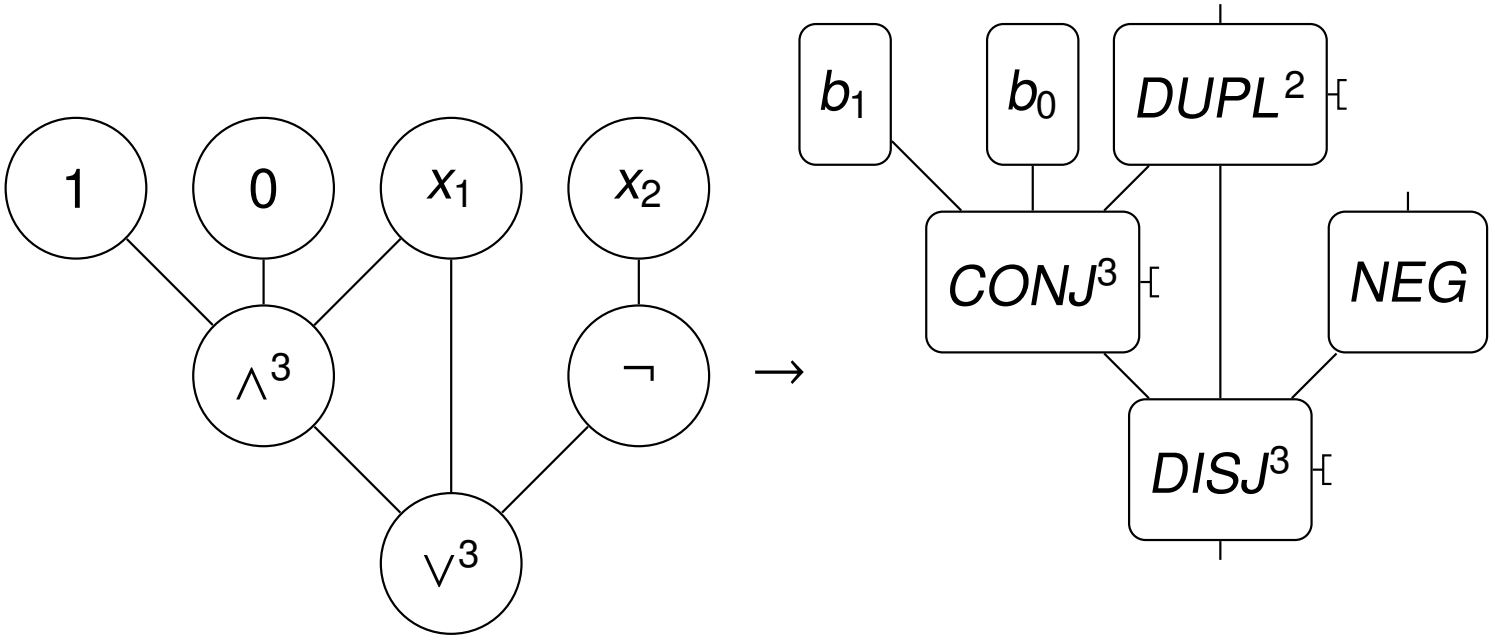
Map (2~5): From Boolean Circuits to Proof Circuits



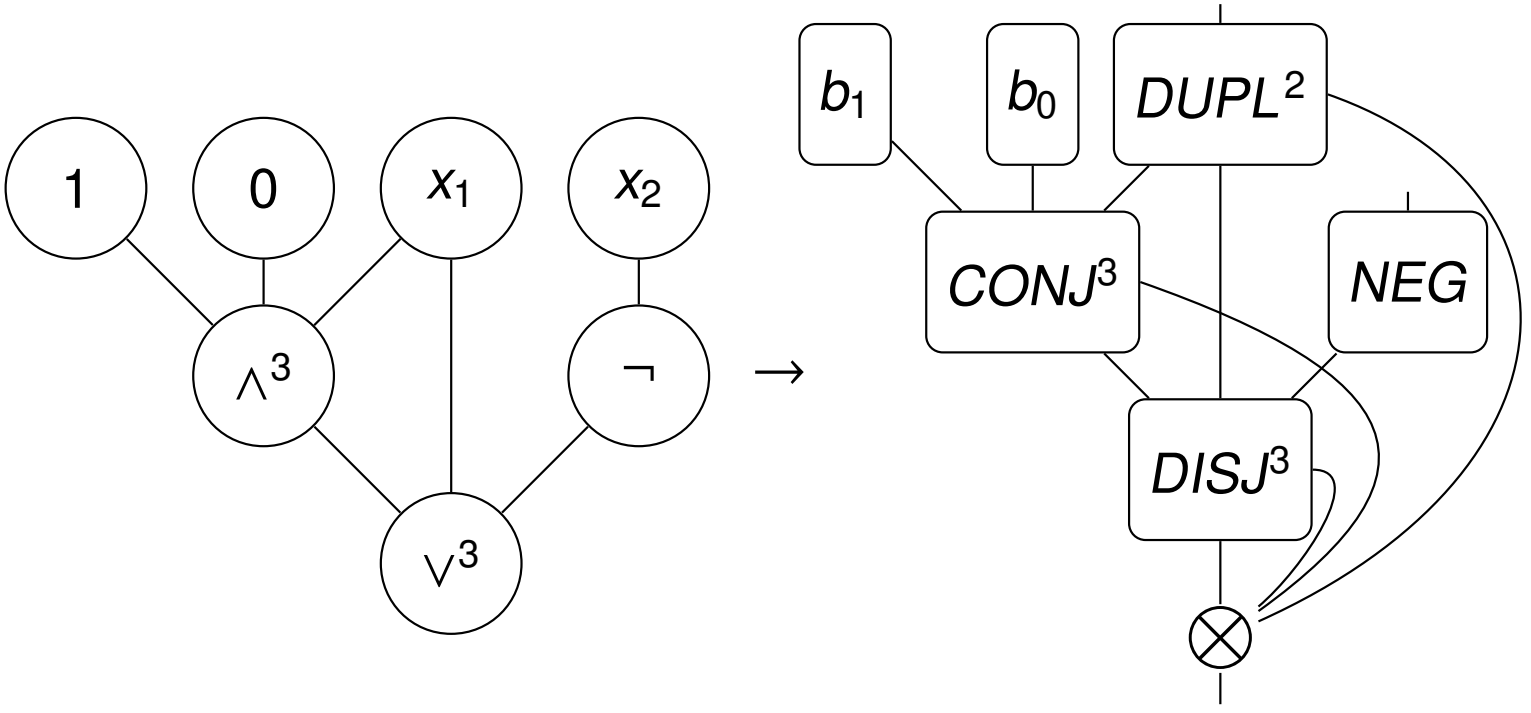
Complexity of the translation



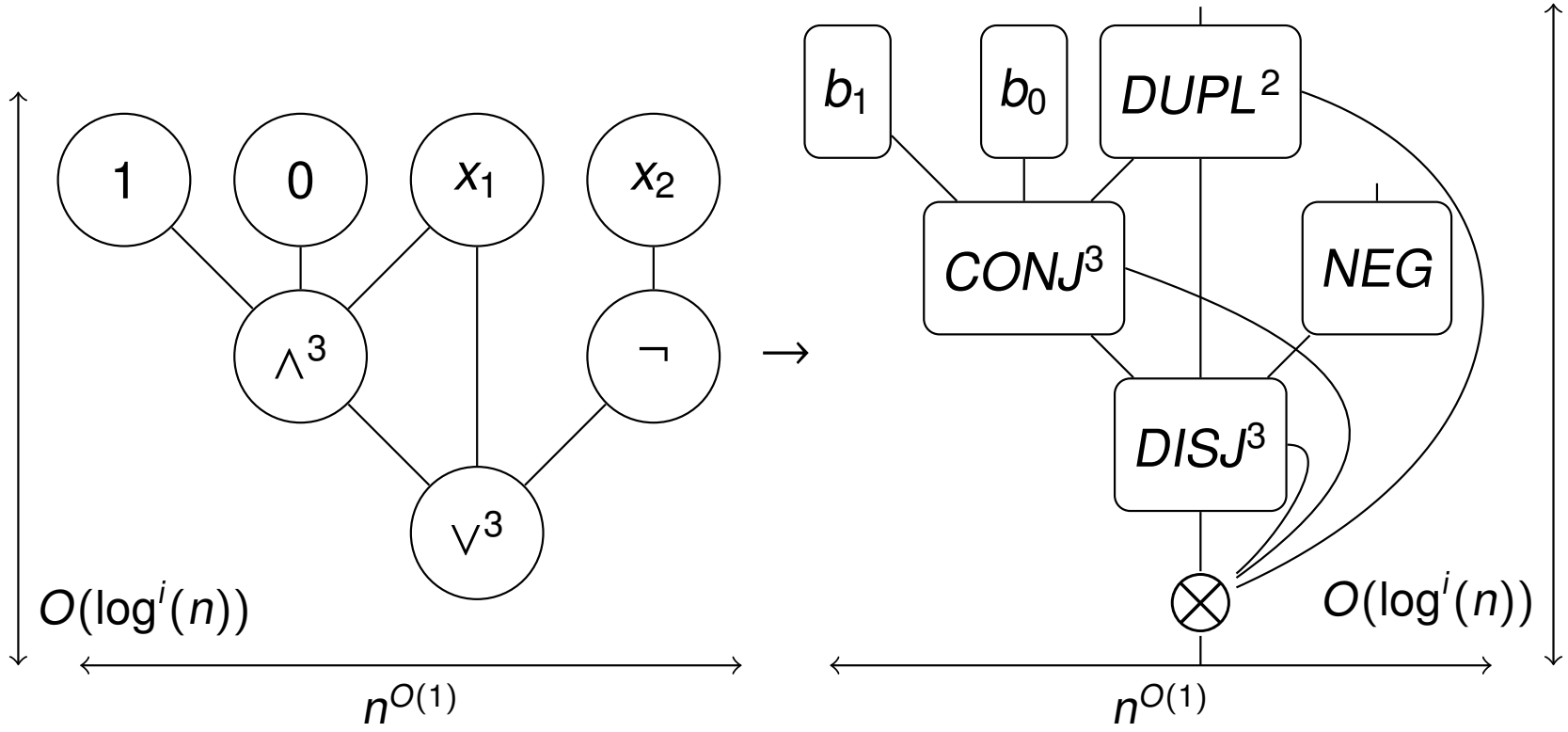
Complexity of the translation



Complexity of the translation



Complexity of the translation



Definition (Translation from AC^i to PCC^i)

Input: *Description of a uniform familly of Boolean Circuits* $C = (C_n)$ *in* AC^i.

Output: *Description of a familly of Proof Circuits* $P = (P_n)$ *in* PCC^i *so that for all* k, *for all* $\overrightarrow{b}$, $P_k(\overrightarrow{b}) \rightarrow_{ev} b_j$ *iff* $C_k(\overrightarrow{b})$ *evaluates to* j.

Theorem

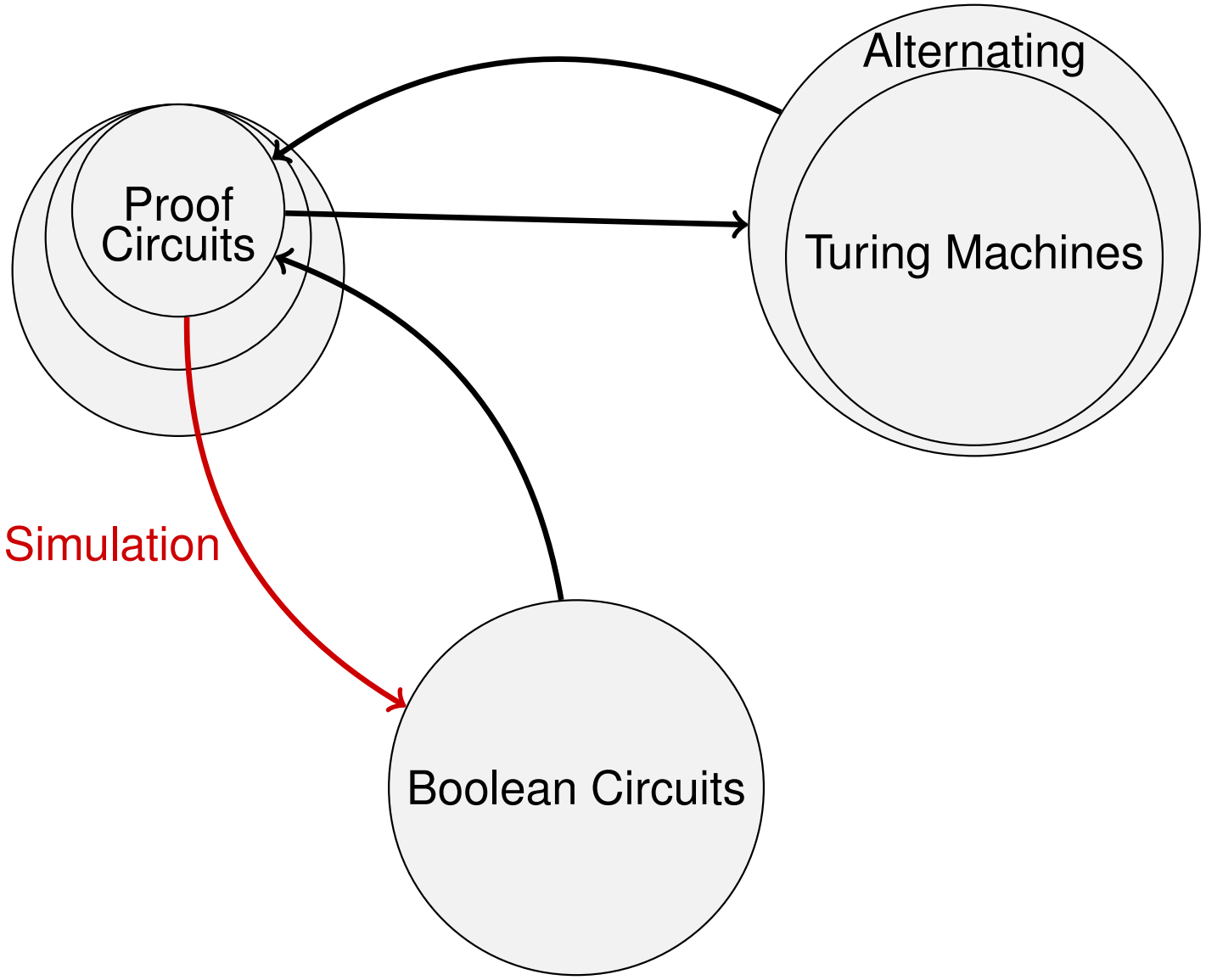
For all $i \in \mathbb{N}$, *Translation from* AC^i *to* PCC^i *belongs to* AC^0.

Theorem (Translation)

For all $i \in \mathbb{N}$,

$$AC^i \subseteq PCC^i$$

Map (3~5): From Proof Circuits to Boolean Circuits



Simulation of $\rightrightarrows$ by Boolean Circuits

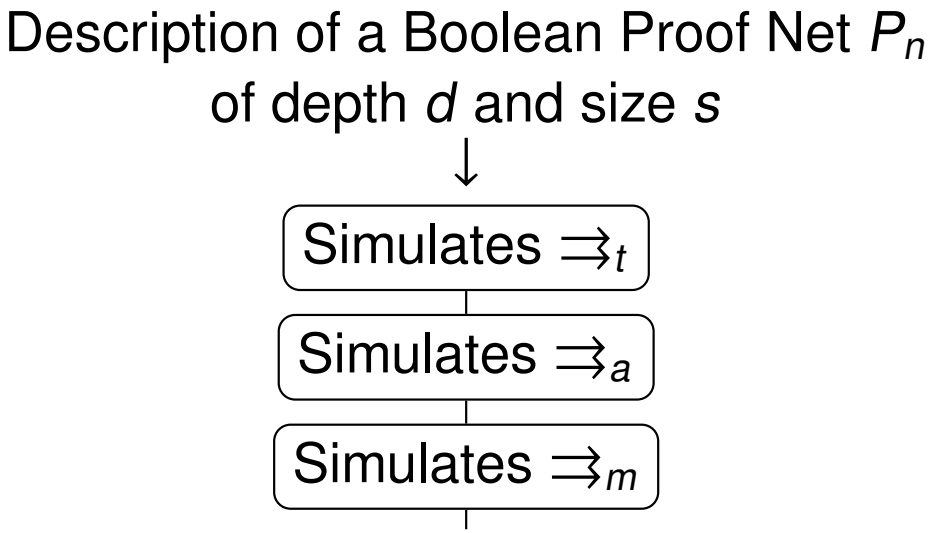
Theorem ([Terui, 2004])

Every Proof-Net P normalizes in at most $3\times$ its depth steps of parallel reduction ($\rightrightarrows_t$, $\rightrightarrows_a$, $\rightrightarrows_m$).

Simulation of $\rightrightarrows$ by Boolean Circuits

Theorem ([Terui, 2004])

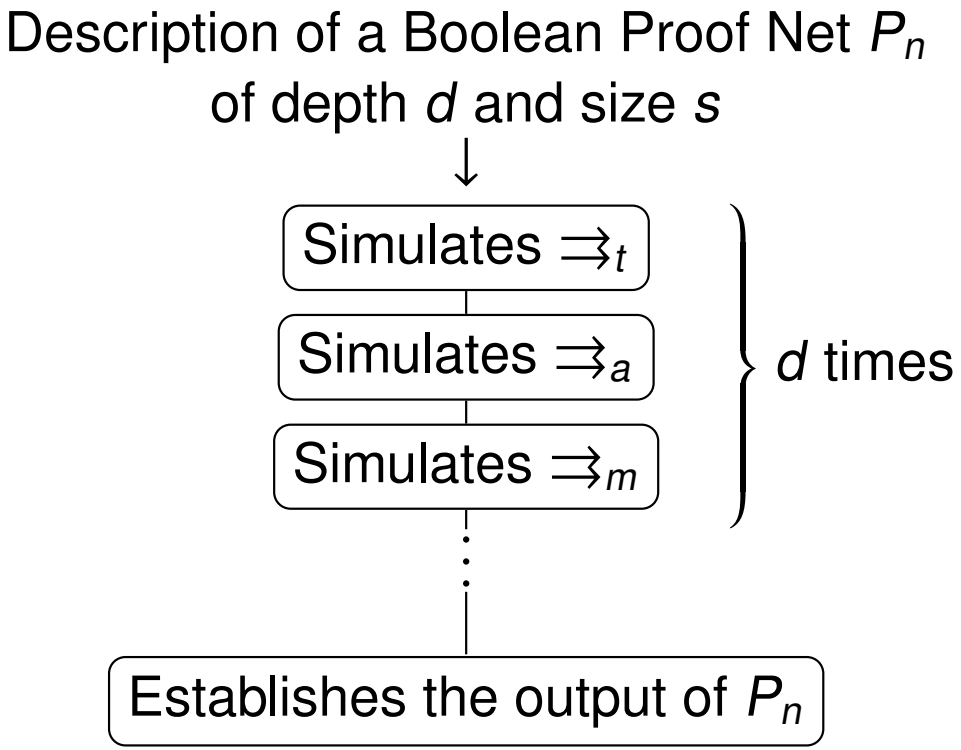
Every Proof-Net P normalizes in at most $3\times$ its depth steps of parallel reduction ($\rightrightarrows_t$, $\rightrightarrows_a$, $\rightrightarrows_m$).



Simulation of $\rightrightarrows$ by Boolean Circuits

Theorem ([Terui, 2004])

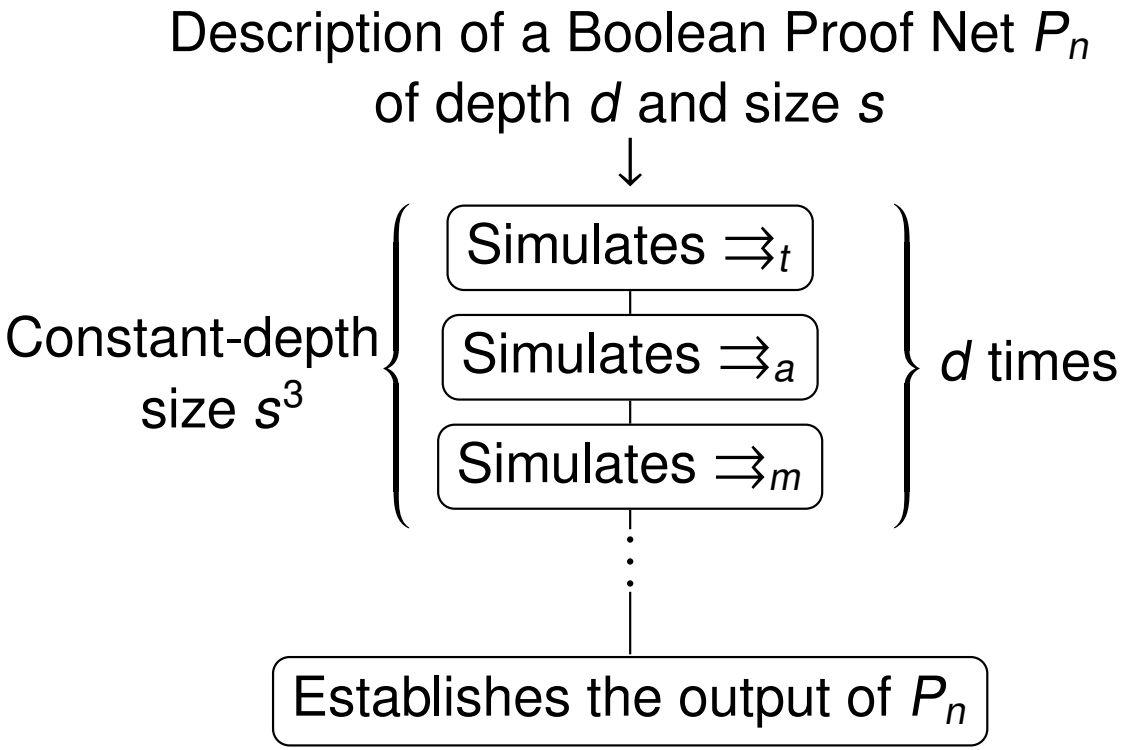
Every Proof-Net P normalizes in at most $3\times$ its depth steps of parallel reduction ($\rightrightarrows_t$, $\rightrightarrows_a$, $\rightrightarrows_m$).



Simulation of $\rightrightarrows$ by Boolean Circuits

Theorem ([Terui, 2004])

Every Proof-Net P normalizes in at most $3\times$ its depth steps of parallel reduction ($\rightrightarrows_t$, $\rightrightarrows_a$, $\rightrightarrows_m$).

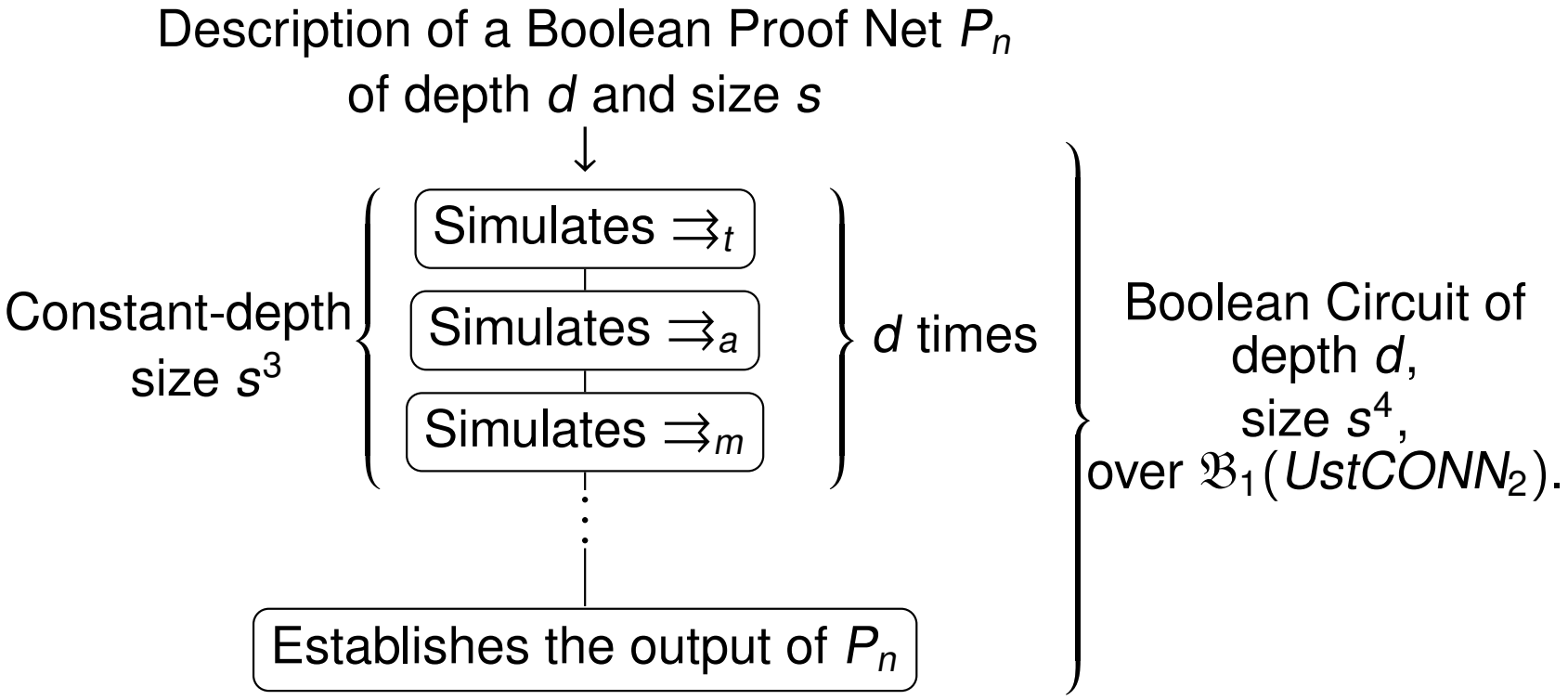


Simulation of $\rightrightarrows$ by Boolean Circuits

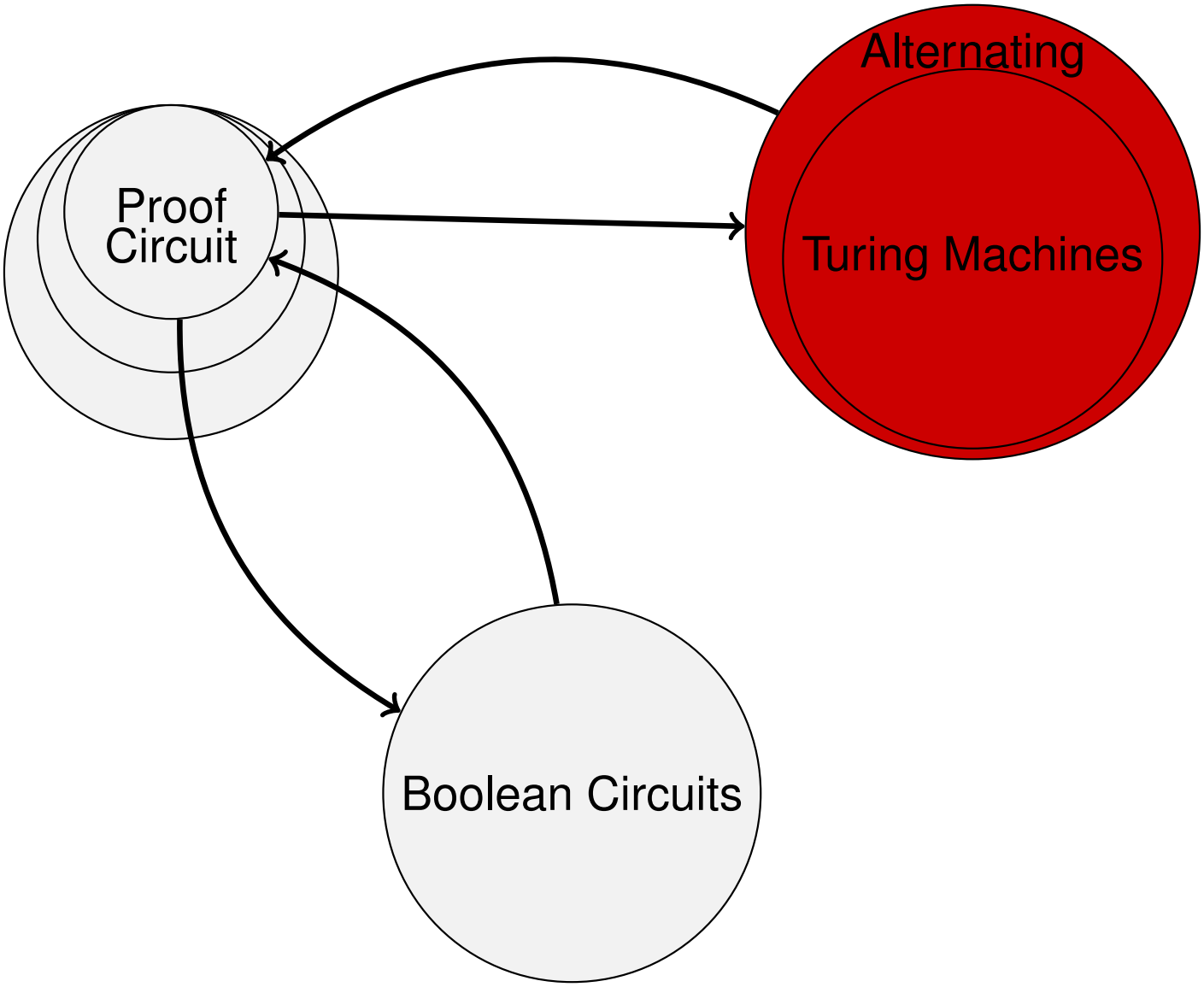
Theorem (Simulation)

For all $i \in \mathbb{N}$,

$$PCC^i \subseteq AC^i(UstCONN_2)$$



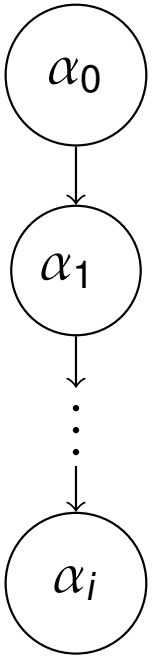
Map (4~5): Alternating Turing Machines



Quick presentation of the ATM

A k-tapes Turing Machine is a tuple $\{K, \Sigma, \delta, s\}$ with

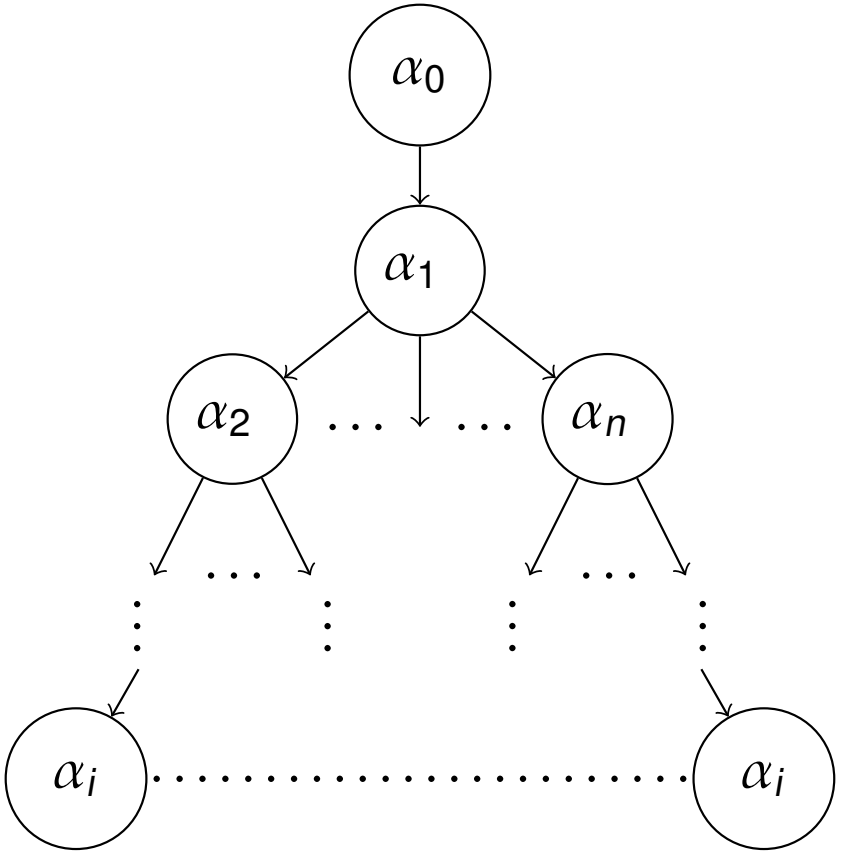
- K a finite set of *states*
- $\Sigma = \{\flat, 0, 1\}$
- δ a function from $K \times \Sigma^K$ to $(K \cup \{a, r\}) \times (\Sigma \times \{\leftarrow, \rightarrow, -\})^K$
- $s \in K$ the initial state



Quick presentation of the ATM

A k-tapes Nondeterminisic Turing Machine is a tuple $\{K, \Sigma, \Delta, s\}$ with

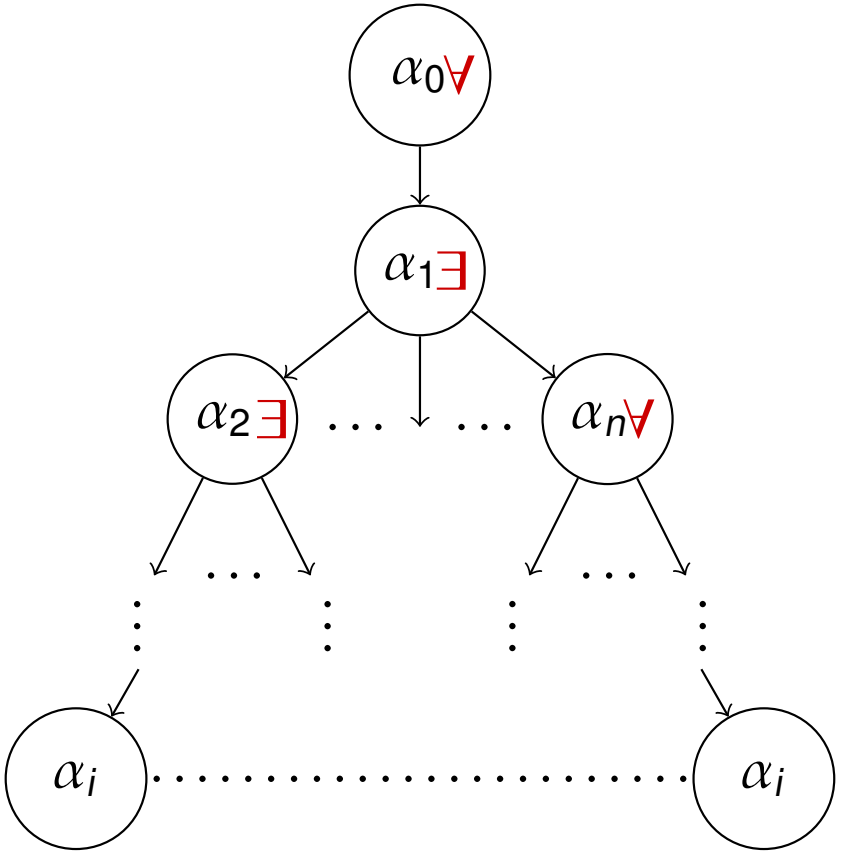
- K a finite set of *states*
- $\Sigma = \{\flat, 0, 1\}$
- Δ a relation $\subset (K \times \Sigma) \times [(K \cup \{a, r\}) \times \Sigma \times \{\leftarrow, \rightarrow, -\}]$
- $s \in K$ the initial state



Quick presentation of the ATM

A k-tapes Alternating Turing Machine is a tuple $\{K, \Sigma, \Delta, s\}$ with

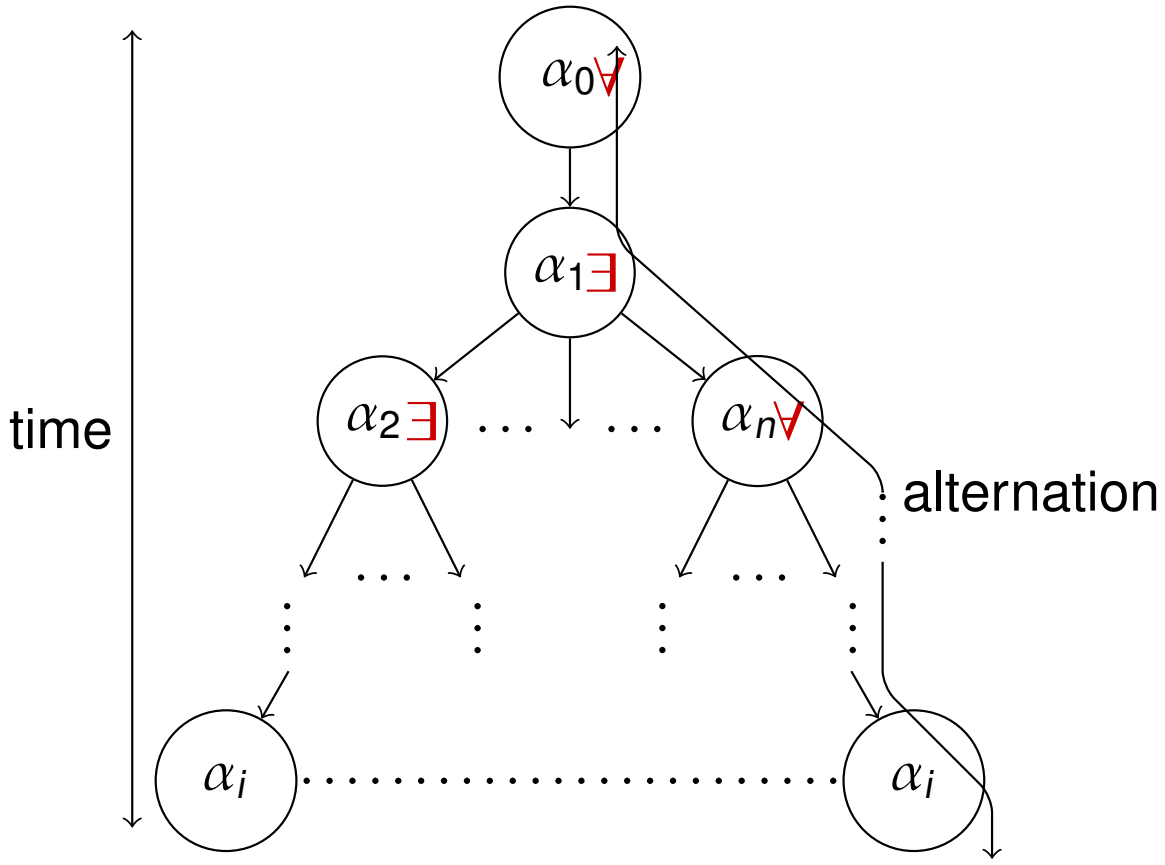
- K a finite set of *states*, partitionned into 2 sets, $K_\forall$ and $K_\exists$
- $\Sigma = \{\flat, 0, 1\}$
- Δ a relation $\subset (K \times \Sigma) \times [(K \cup \{a, r\}) \times \Sigma \times \{\leftarrow, \rightarrow, -\}]$
- $s \in K$ the initial state



Quick presentation of the ATM

A k-tapes Alternating Turing Machine is a tuple $\{K, \Sigma, \Delta, s\}$ with

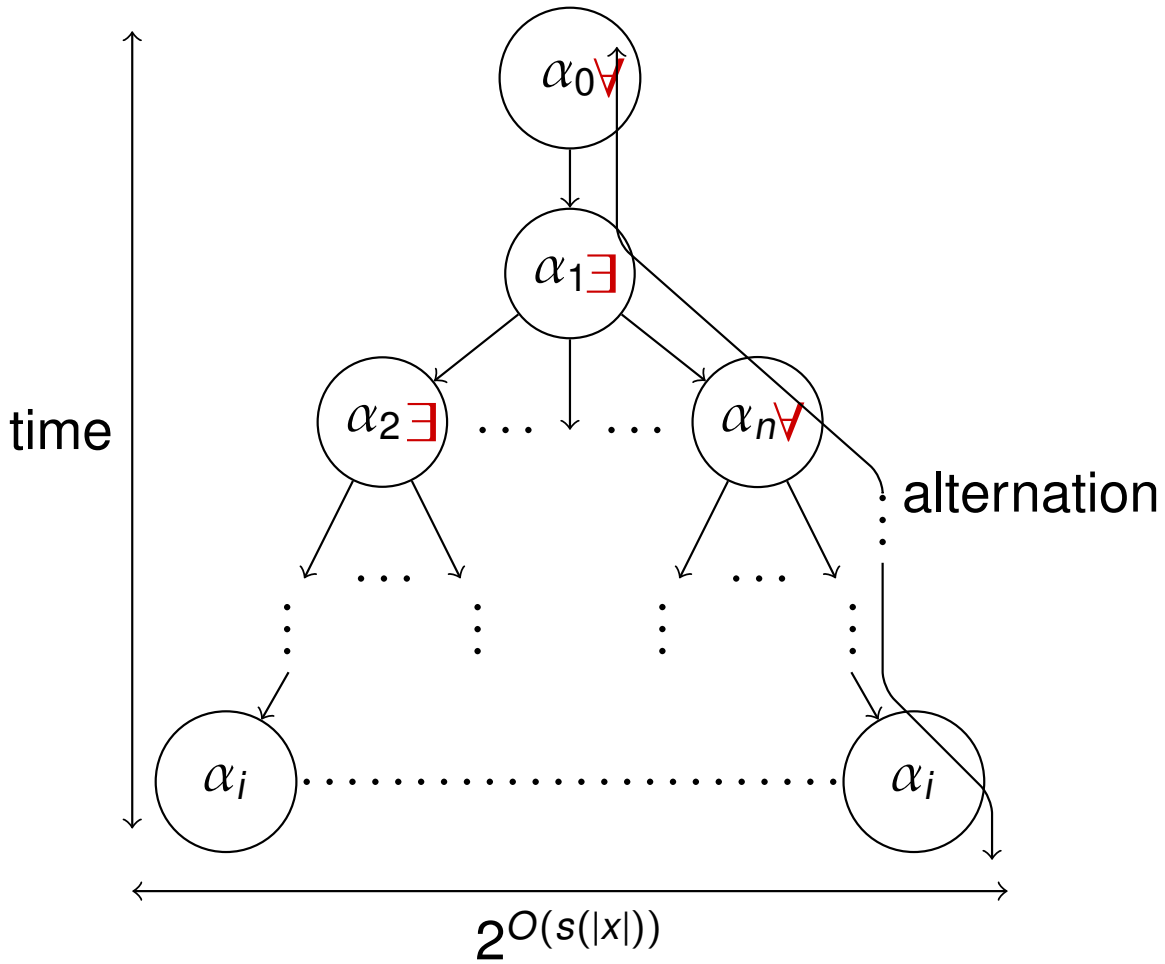
- K a finite set of *states*, partitionned into 2 sets, $K_\forall$ and $K_\exists$
- $\Sigma = \{\flat, 0, 1\}$
- Δ a relation $\subset (K \times \Sigma) \times [(K \cup \{a, r\}) \times \Sigma \times \{\leftarrow, \rightarrow, -\}]$
- $s \in K$ the initial state



Quick presentation of the ATM

A k-tapes Alternating Turing Machine is a tuple $\{K, \Sigma, \Delta, s\}$ with

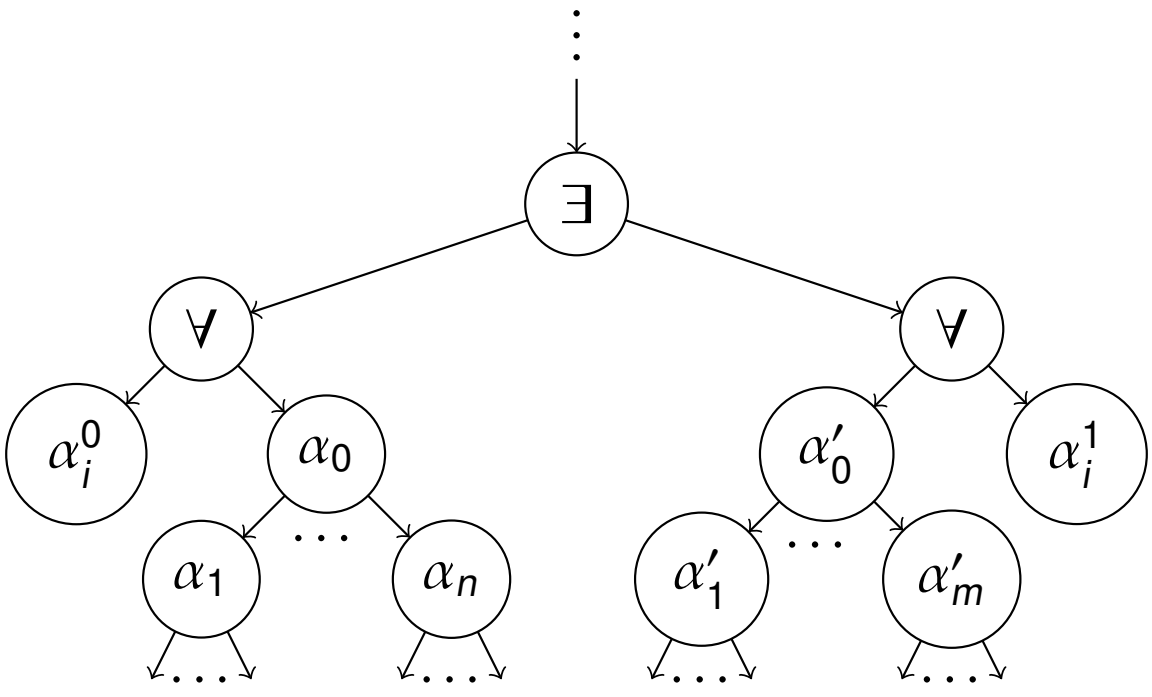
- K a finite set of *states*, partitionned into 2 sets, $K_\forall$ and $K_\exists$
- $\Sigma = \{\flat, 0, 1\}$
- Δ a relation $\subset (K \times \Sigma) \times [(K \cup \{a, r\}) \times \Sigma \times \{\leftarrow, \rightarrow, -\}]$
- $s \in K$ the initial state



Transformations on graphs of ATM

The aim is to obtain a finite tree that represents the behaviour of an ATM at given size of input.

- Input Normal Form

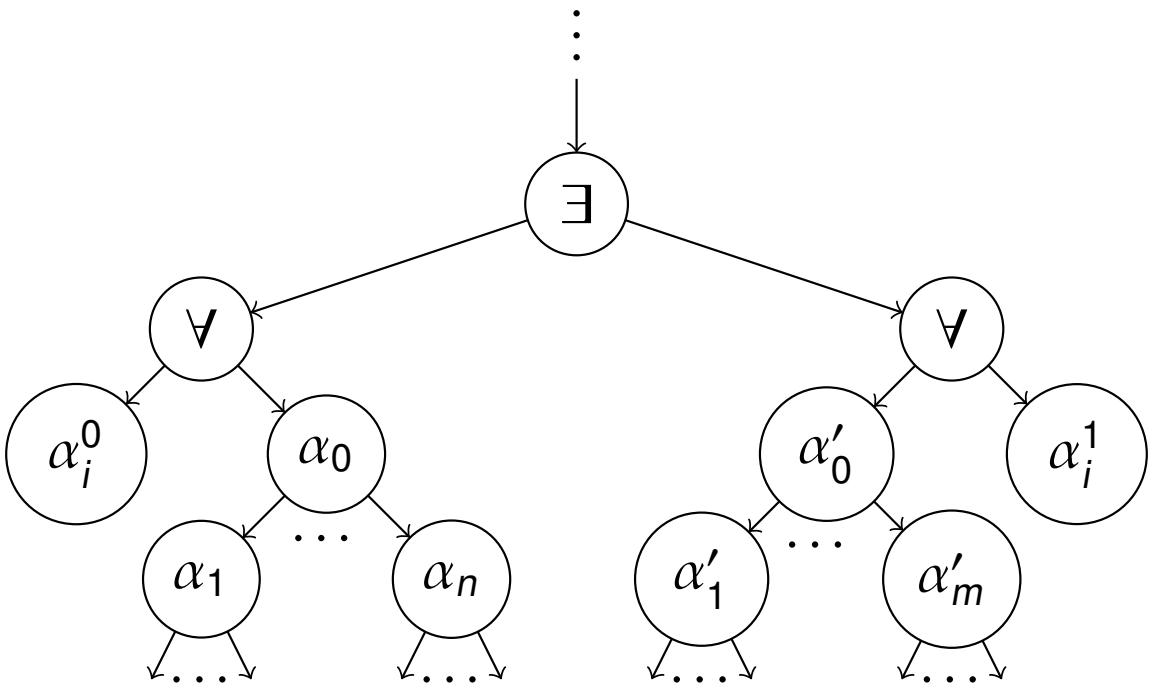


α_i^j is an accepting state iff $x_i = j$.

Transformations on graphs of ATM

The aim is to obtain a finite tree that represents the behaviour of an ATM at given size of input.

- Input Normal Form
- Clock: Remove cycles and infinite branchs

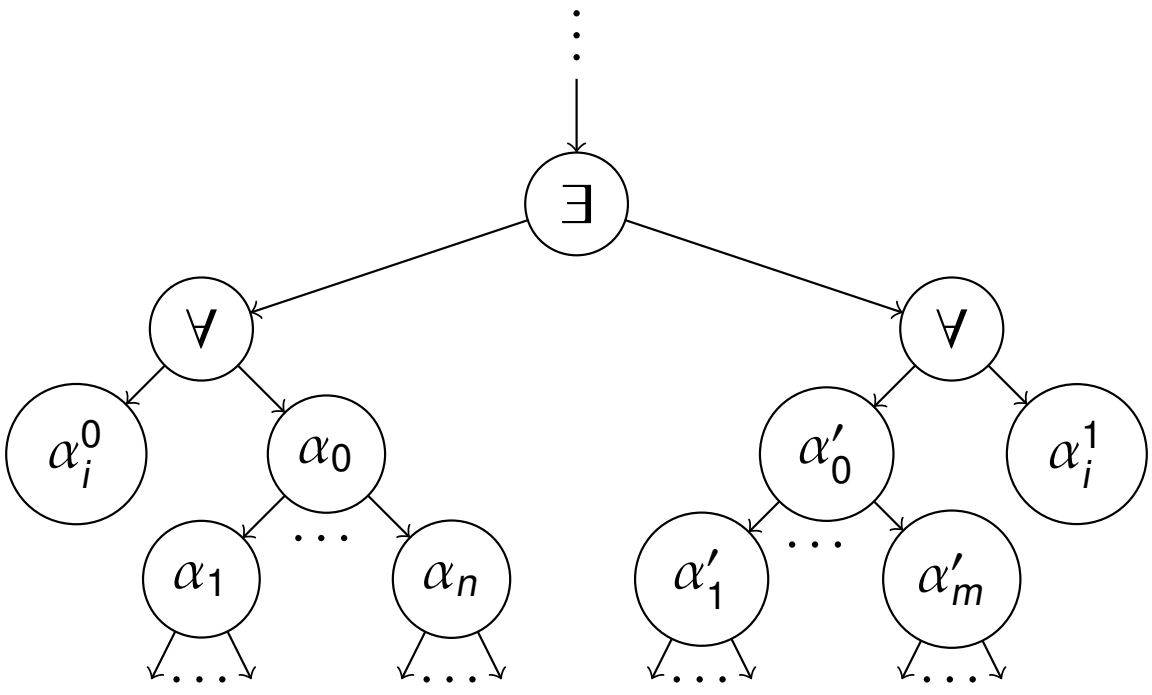


α_i^j is an accepting state iff $x_i = j$.

Transformations on graphs of ATM

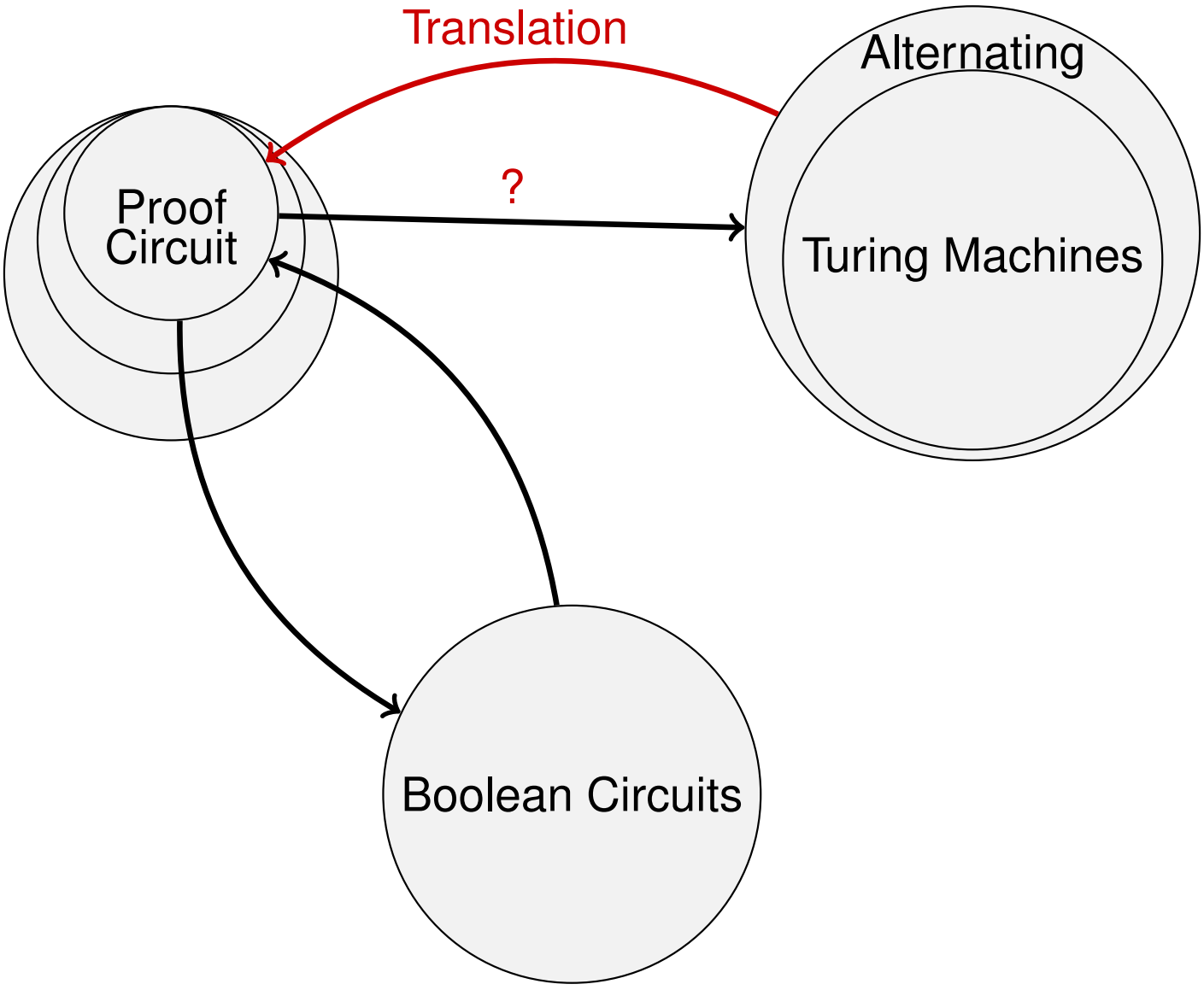
The aim is to obtain a finite tree that represents the behaviour of an ATM at given size of input.

- Input Normal Form
- Clock: Remove cycles and infinite branchs
- Every configuration has at most one predecessor

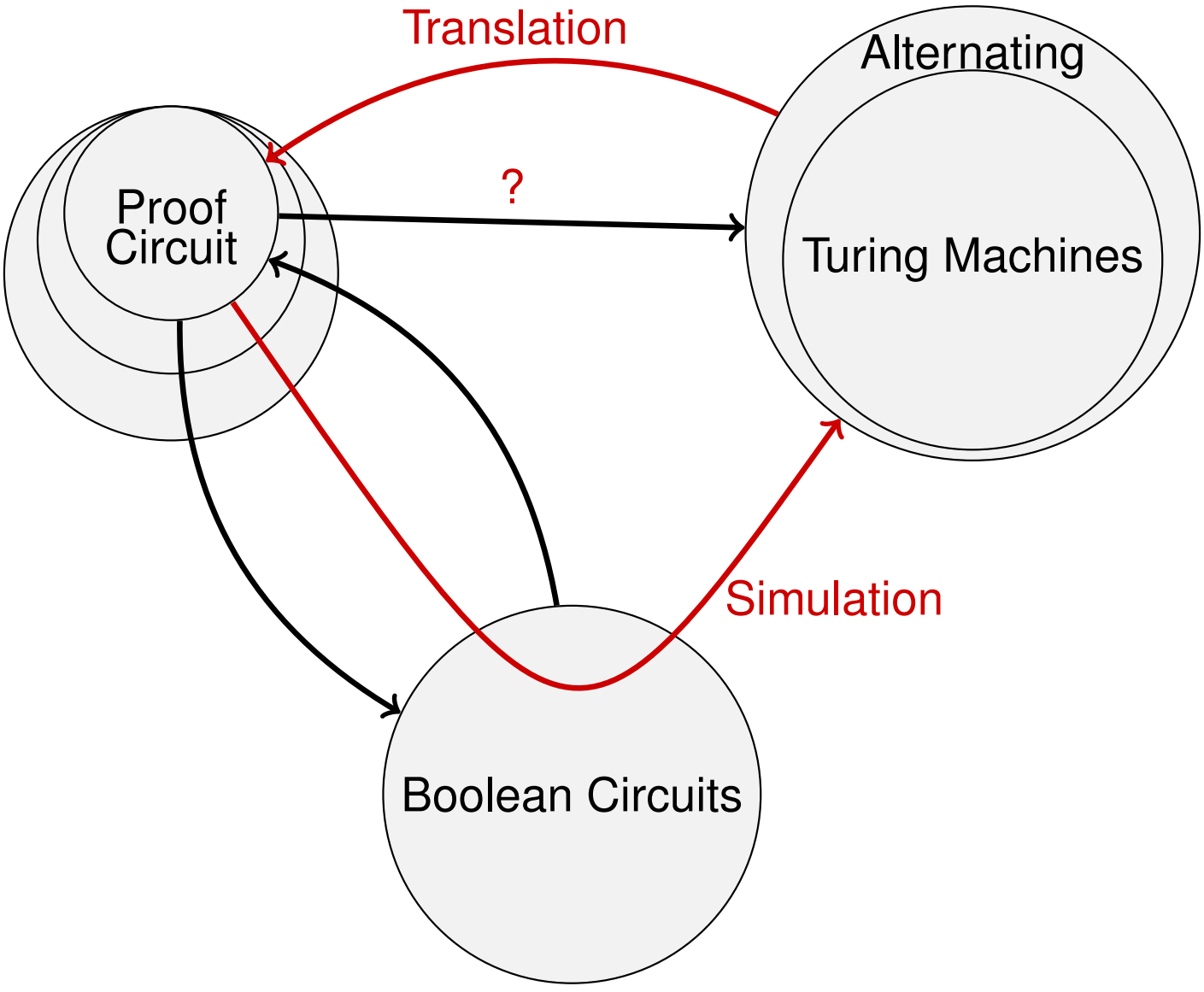


α_i^j is an accepting state iff $x_i = j$.

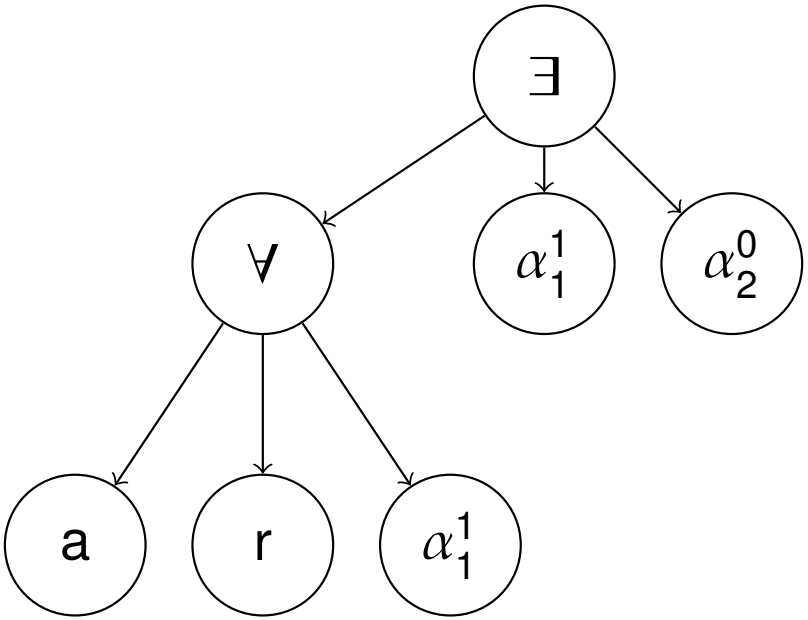
Map (5~5): A.T.M. and Proof Circuits



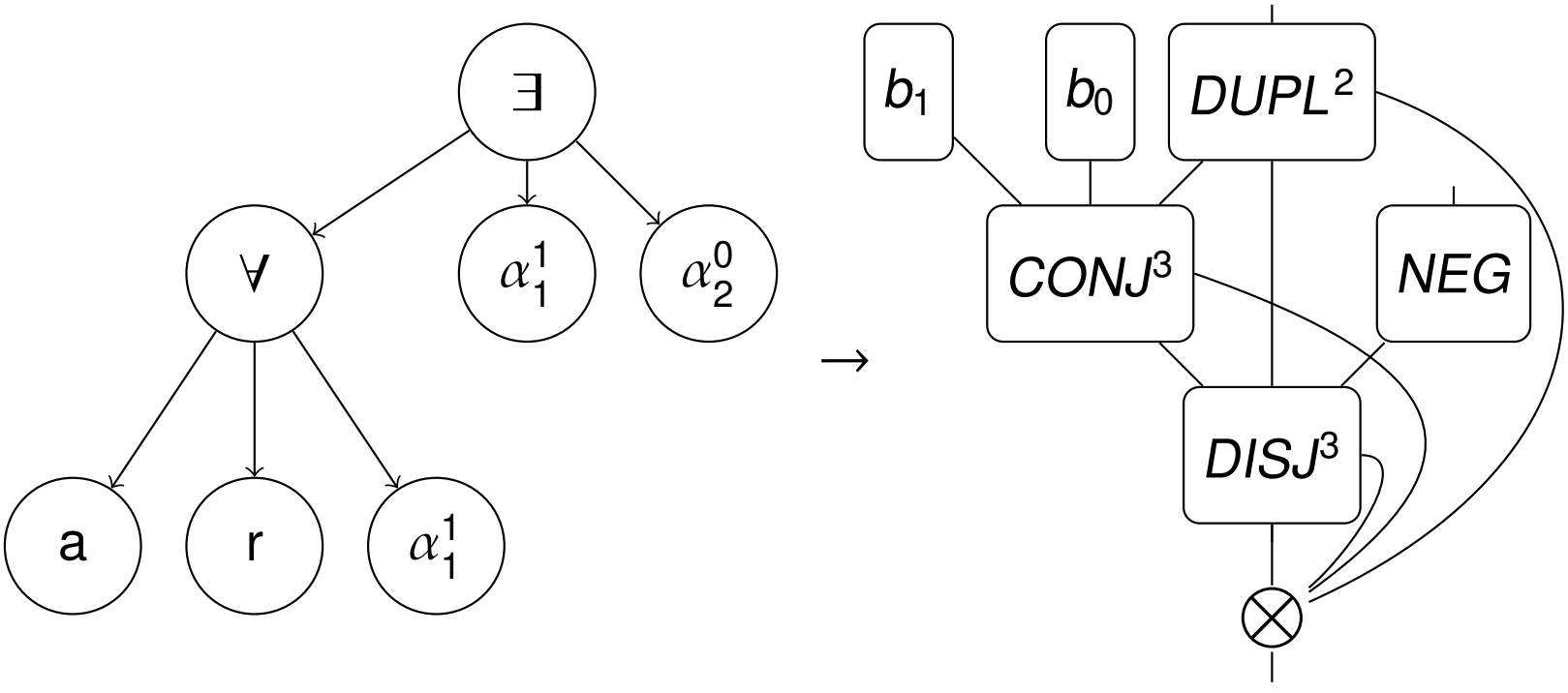
Map (5~5): A.T.M. and Proof Circuits



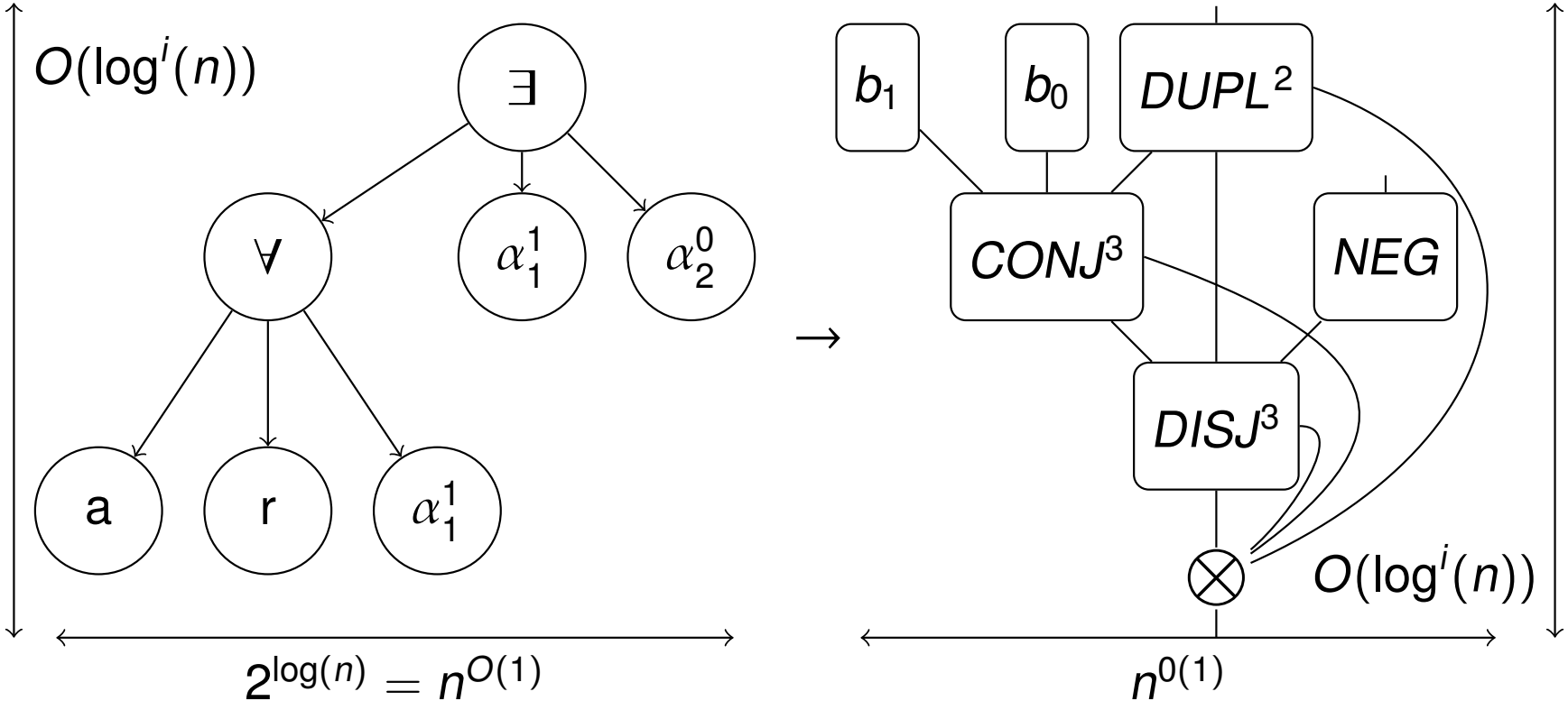
Translation from A.T.M to P.C.



Translation from A.T.M to P.C.



Translation from A.T.M to P.C.



Definition (STA)

$STA(s, t, a)$ for s, t and a proper complexity functions is the set of boolean functions computable by an *ATM* using space s, time t and alternation a.

Theorem (Translation)

For $i \geqslant 2$,

$$STA(\log(n), \log^i(n), *) \subseteq PCC^i$$

Definition (STA)

$STA(s, t, a)$ for s, t and a proper complexity functions is the set of boolean functions computable by an *ATM* using space s, time t and alternation a.

Theorem (Translation)

For $i \geqslant 2$,

$$STA(\log(n), \log^i(n), *) \subseteq PCC^i$$

Theorem (Simulation through Boolean Circuits, inspired by [Vollmer, 1999])

For $i \geqslant 2$,

$$PCC^i \subseteq STA(\log(n), \log^{i+2}, *)$$

That's all folks!


Aubert, C. (2011).
Sublogarithmic uniform boolean proof nets.
In *Proceedings of DICE'11*.


Terui, K. (2004).
Proof Nets and Boolean Circuits.
In *Proceedings of LICS'04*, pages 182–191.


Vollmer, H. (1999).
Introduction to Circuit Complexity: A Uniform Approach.
Springer Verlag.

aubert@lipn.univ-paris13.fr
lipn.fr/∼aubert