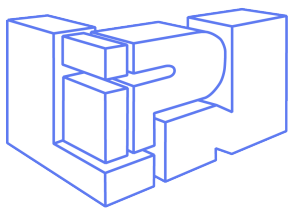
# Logique linéaire et classes de complexité sous-polynomiales

## Soutenance de thèse

Clément Aubert



26 novembre 2013

## Objectif

Capturer la difficulté d'un problème.

## Outils

- Thèse de Church
- Modèles de calcul abstraits
- Réduction, problèmes complets, etc.
- Classes de complexité

| | | |
|---|---|---|
| Circuits booléens | **AC** | « Efficace en parallèle » |
| | ⊂ | |
| Temps polynomial | **P** | « Raisonnable » |
| | ⊂ | |
| Espace logarithmique | (**N**)**L** | « Économe en espace » |

## Objectif

Caractériser les classes de complexité indépendamment des modèles de calcul.

## Techniques

- Schéma de récursion borné
- Issues de la théorie des modèles
- Quasi-interprétation
- Raffiner la Logique linéaire

## Objectif

Formaliser le raisonnement

## Techniques

- Étude des règles
- Recherche de preuves
- Théorie de la démonstration

## Correspondance entre preuves et programmes

L'élimination des coupures corresponds à l'exécution d'un programme typé : interprétation calculatoire.

## Objectif

Gérer les ressources : décomposition de $A \Rightarrow B$ en $!A \multimap B$

## Techniques

- Décomposition en fragments
- Contrôle des ressources
- Réseaux de preuves
- Géométrie de l'interaction

## Logique linéaire et complexité

- Logique linéaire allégée (ou par niveaux)
- Interprétation de cette dernière avec des moyens algébriques

## Objectif

Logique linéaire et classes de complexité sous-polynomiales

## ⊸ Avancées

Caractérisation. . .

du parallélisme avec des moyens syntaxiques

de l'espace logarithmique (non-)déterministe avec des moyens sémantiques

## Points communs

— Outils et buts

— Ponts avec complexité

— Représentation d'entrées de taille infinie

# Première partie

## Circuits de preuve

## Sources

- TERUI « Proof Nets and Boolean Circuits » (2004)
- MOGBIL et RAHLI « Uniform Circuits, & Boolean Proof Nets » et « Non-deterministic Boolean Proof Nets » (2009)

## Objectif

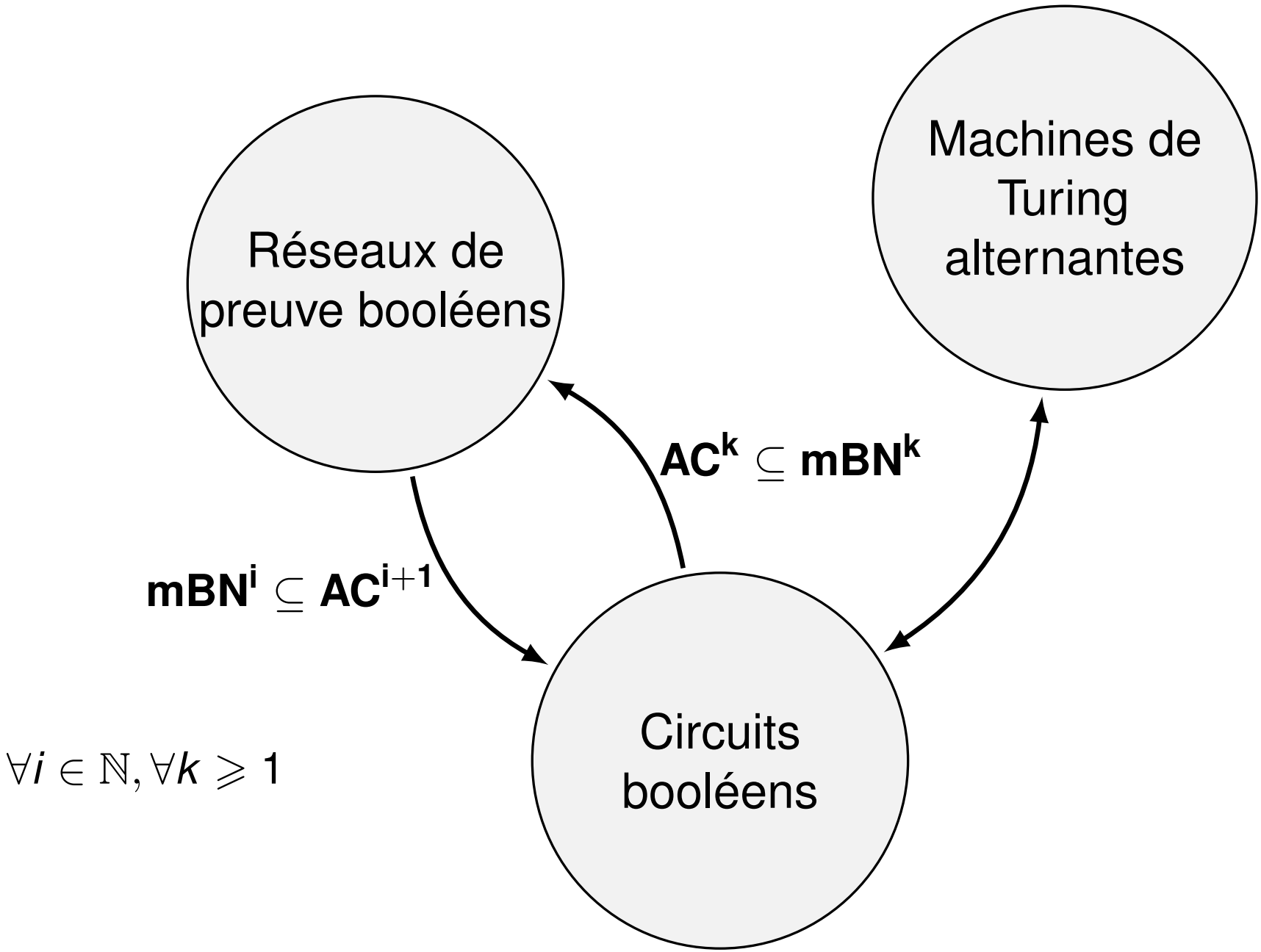
Faire calculer les réseaux de preuve à la manière des circuits booléens.

## Avancées

### Étendre la correspondance

- aux plus petites classes de complexité
- aux machines de Turing alternantes
- en la rendant plus modulaire

# Circuits de preuve : Situation initiale

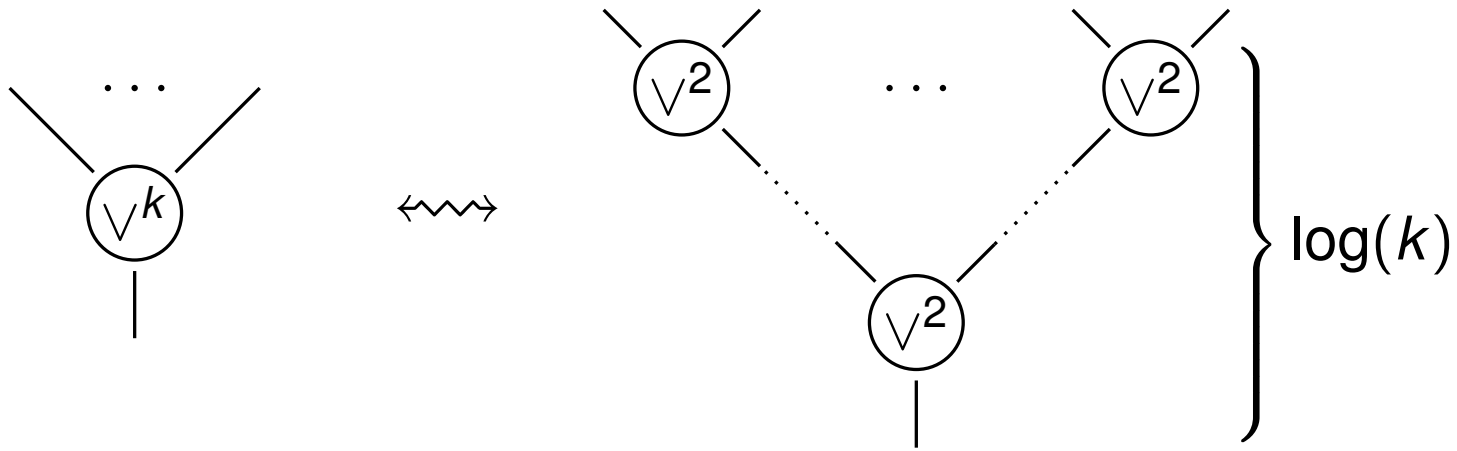


## Définition ($\mathbf{AC^i}$)

Pour tout $i \in \mathbb{N}$, l'ensemble des fonctions booléennes calculables par une famille uniforme de circuits booléens $C = (C_n)$ où pour tout $C_n$

- sa profondeur est $\mathcal{O}(\log^i(n))$,
- sa taille est $n^{\mathcal{O}(1)}$,
- ses portes sont étiquetées avec $\neg$, $\bigvee^k$ et $\bigwedge^k$ pour $k \in \mathbb{N}$.

Le cas borné $\bigvee^2$ et $\bigwedge^2$ définit $\mathbf{NC^i}$, $\mathbf{AC^i} \subseteq \mathbf{NC^{i+1}} \subseteq \mathbf{AC^{i+1}}$.



### Définition (Circuit de preuve)

Un circuit de preuve est un réseau de preuve booléen, composé inductivement à partir des pièces $b_0, b_1, \mathrm{Neg}, \mathrm{Dupl}^k, \mathrm{Disj}^k, \mathrm{Conj}^i$ pour $k \geqslant 2$.

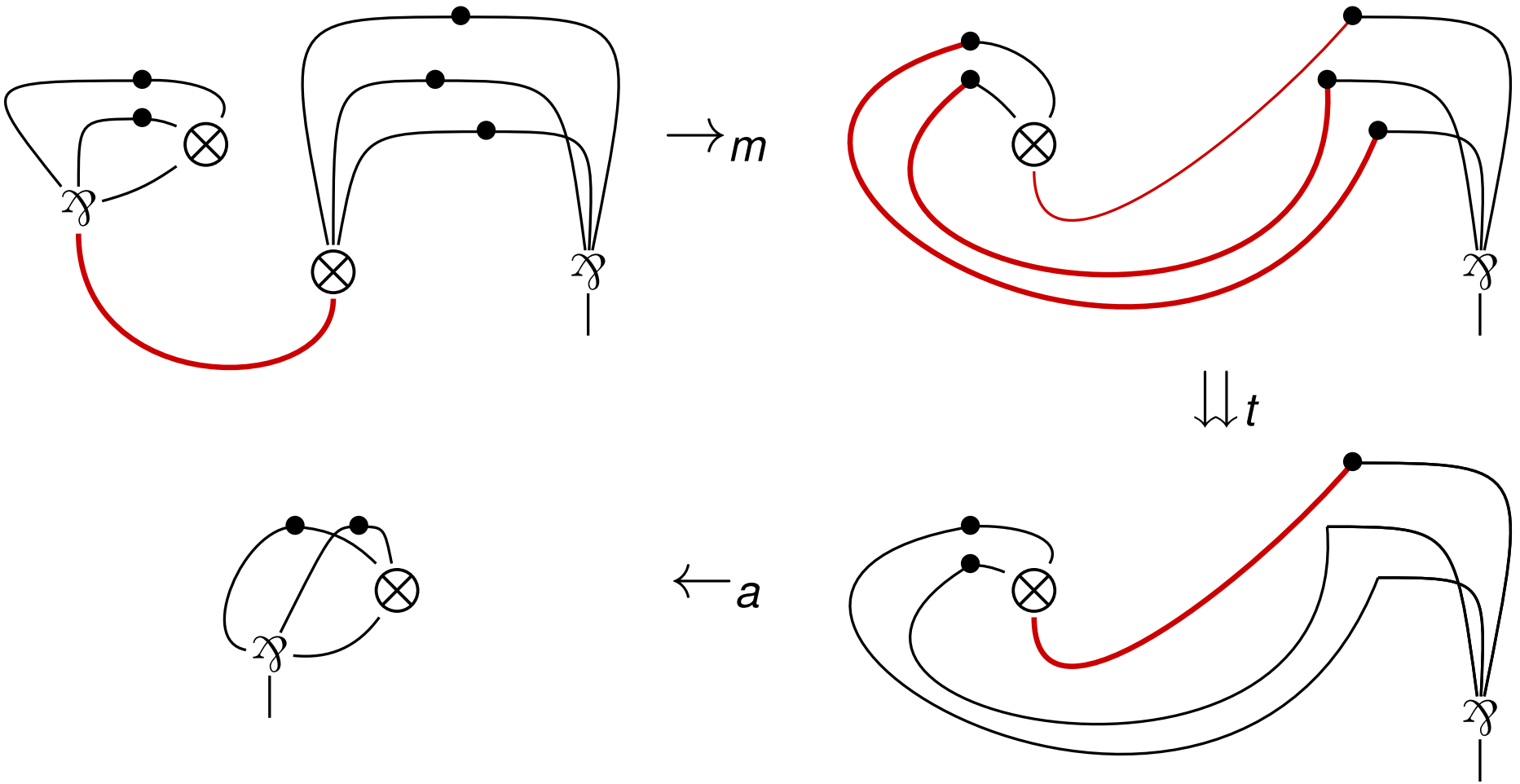
### Définition ($\mathbf{PCC^i}$)

Pour tout $i \in \mathbb{N}$, l'ensemble des fonctions booléennes calculables par une famille uniforme de circuits de preuves $P = (P_n)$ où pour tout $P_n$

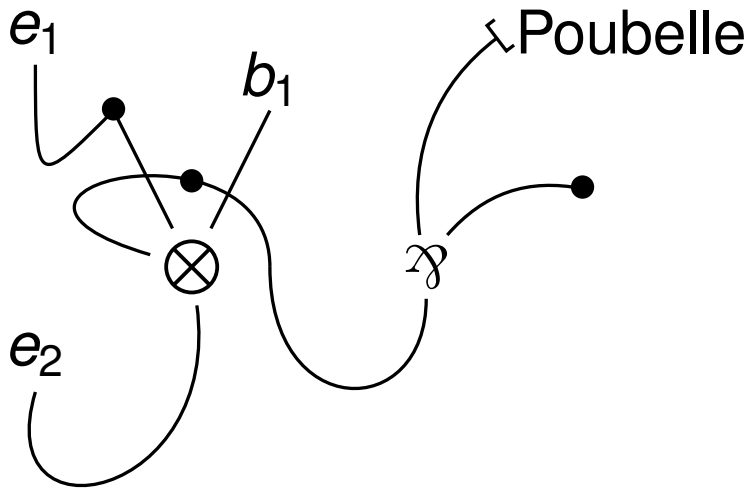
- sa profondeur est $\mathcal{O}(\log^i(n))$,
- sa taille est $n^{\mathcal{O}(1)}$.

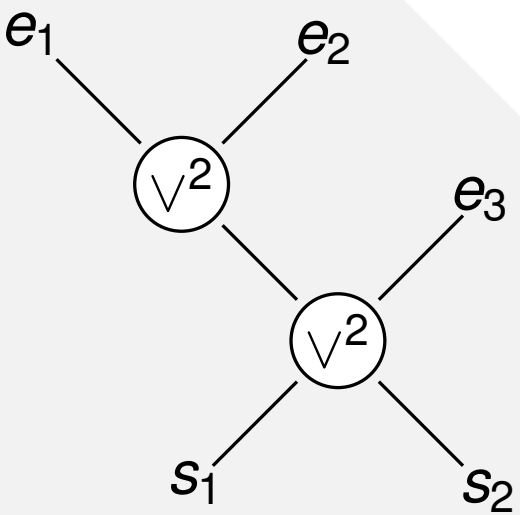
# Circuits de preuve : Comment calculer ?

Type booléen : $\parr^3(\alpha^\perp, \alpha^\perp, \otimes^2(\alpha, \alpha))$

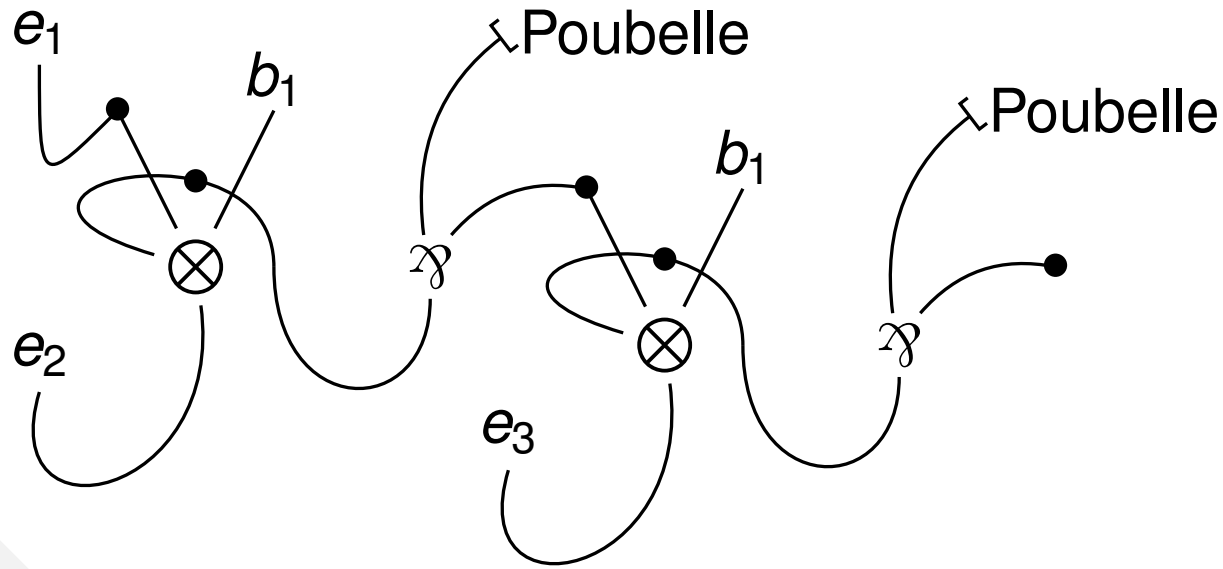


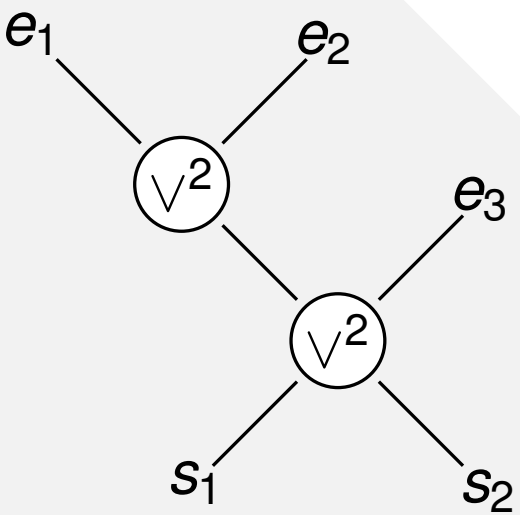
# Circuits de preuve : Disjonction et duplication



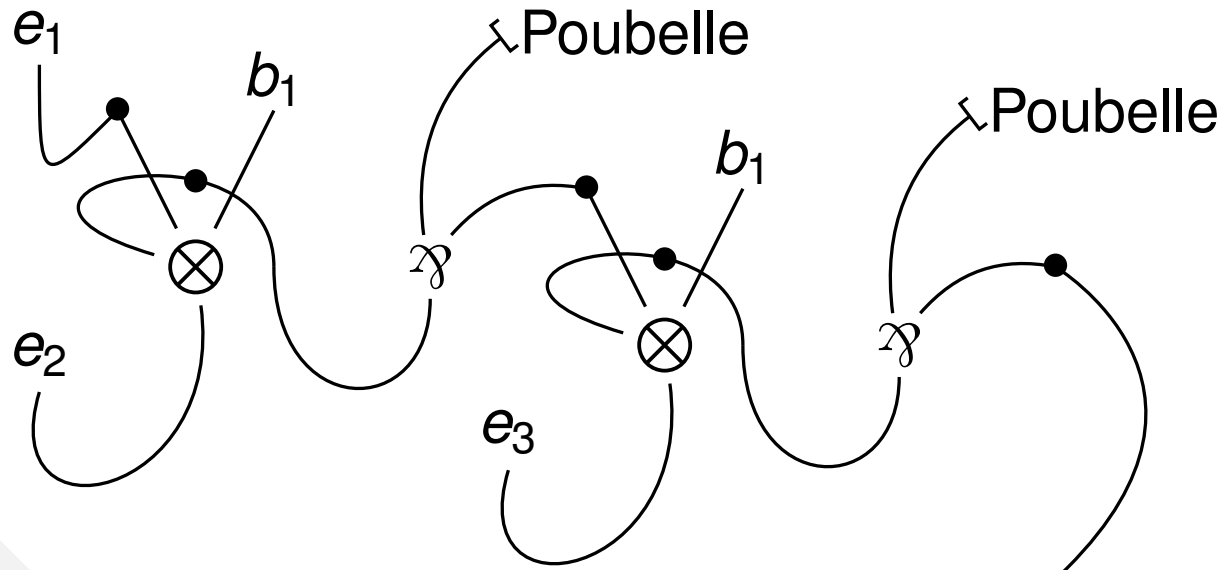



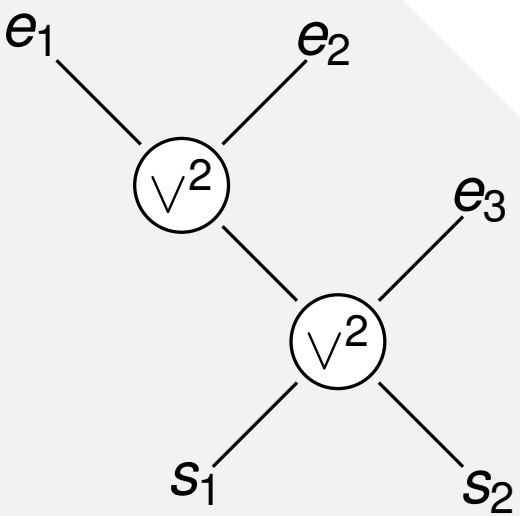
# Circuits de preuve : Disjonction et duplication

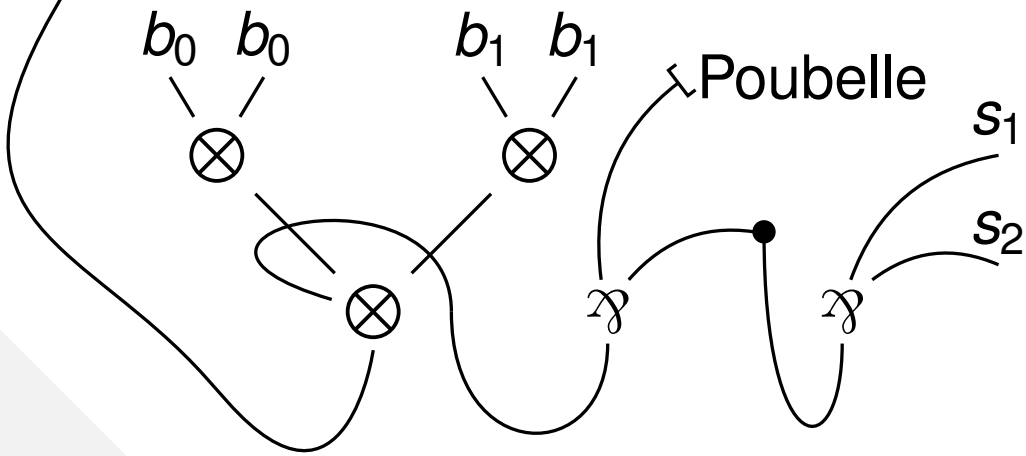


# Circuits de preuve : Disjonction et duplication



## ⚠ Difficultés

- Langage de description
- Uniformité
- Évaluation parallèle

## Avancées

Une présentation

- plus modulaire
- qui réduit la taille
- qui simplifie la traduction des circuits booléens en réseaux de preuves

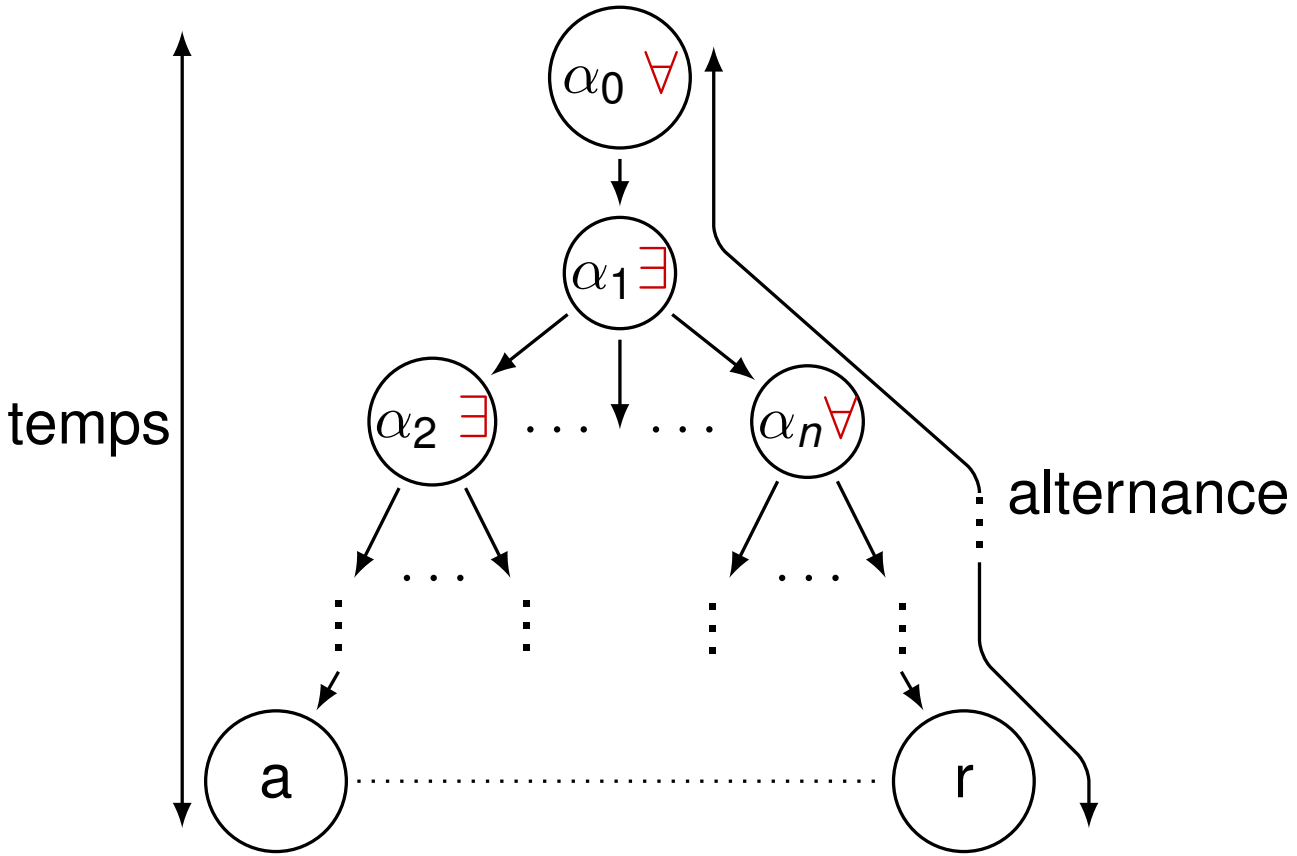
## Résultats

Pour tout $i \in \mathbb{N}$, $\mathbf{AC}^i \subseteq \mathbf{PCC}^i \subseteq \mathbf{mBN}^i$

## Définition

Une machine de Turing alternante à $k$ bandes est

- $Q$ un ensemble fini d'états, $Q = Q_\forall \cup Q_\exists$, $s \in Q$ l'état initial,
- $\Delta$ une relation entre $Q \times \Sigma^{k+1}$ et $Q \times \Sigma^k \times \{\leftarrow, \rightarrow, -\}^{k+1}$



# Circuits de preuve : Machines de Turing alternantes

## ⚠ Difficultés

- Familles de machines de Turing alternantes
- Tirer profit du parallélisme
- Complexité des réductions

## Avancées

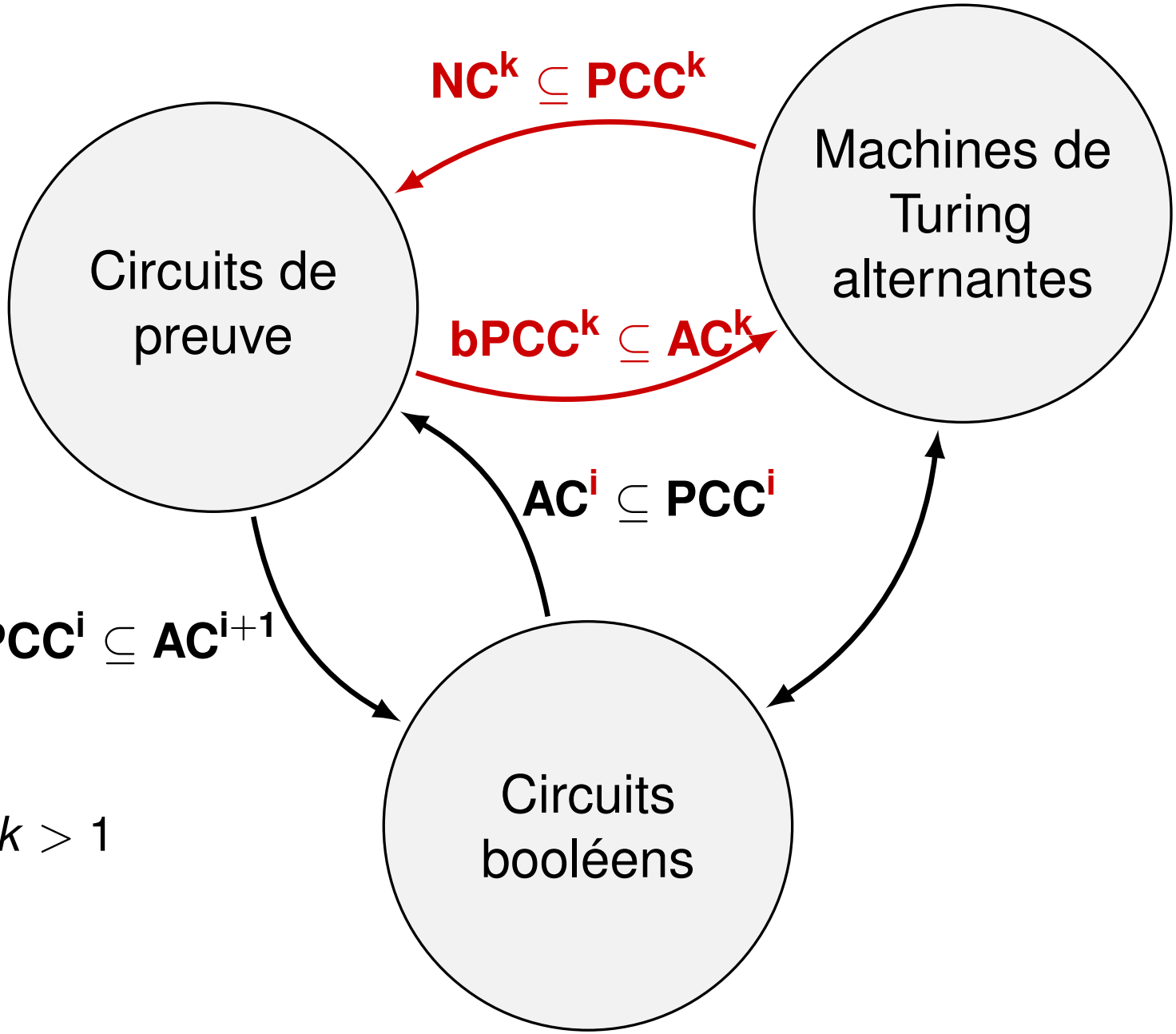
De nouveaux ponts

- grâce à la définition de $\mathbf{bPCC^i}$
- grâce à un algorithme de normalisation en espace log
- qui resserrent les liens entre modèles

## Résultats

Pour tout $k > 1$, $\mathbf{NC^k} \subseteq \mathbf{PCC^k}$ et $\mathbf{bPCC^k} \subseteq \mathbf{AC^k}$, en employant les machines de Turing alternantes.

# Circuits de preuve : Résultats




$\forall i \in \mathbb{N}, \forall k > 1$

# Deuxième partie

## Calcul et facteur hyperfini

## Sources

- GIRARD « Normativity in Logic » (2012)
- AUBERT et SEILLER « Characterizing co-NL by a group action » (2012) et « Logarithmic Space and Permutations » (2013)

## Objectif

Calculer avec l'interprétation de preuves dans une algèbre de von Neumann.

## Avancées

### Formaliser les intuitions

- pour améliorer la compréhension des objets manipulés
- qui permettent de relâcher certaines contraintes, d'en inventer d'autres

## Objectif

Interpréter dynamiquement les preuves

## Correspondance

| | | |
|---|---|---|
| Preuves | $\leftrightarrow$ | Opérateurs |
| Élimination des coupures | $\leftrightarrow$ | Équation de rétroaction |
| Normalisable | $\leftrightarrow$ | Nilpotent |

## Outils

- Chemins dans un réseau de preuve
- Algèbre d'opérateurs
- Quantiques

# Calcul et facteur hyperfini : Représenter les entiers

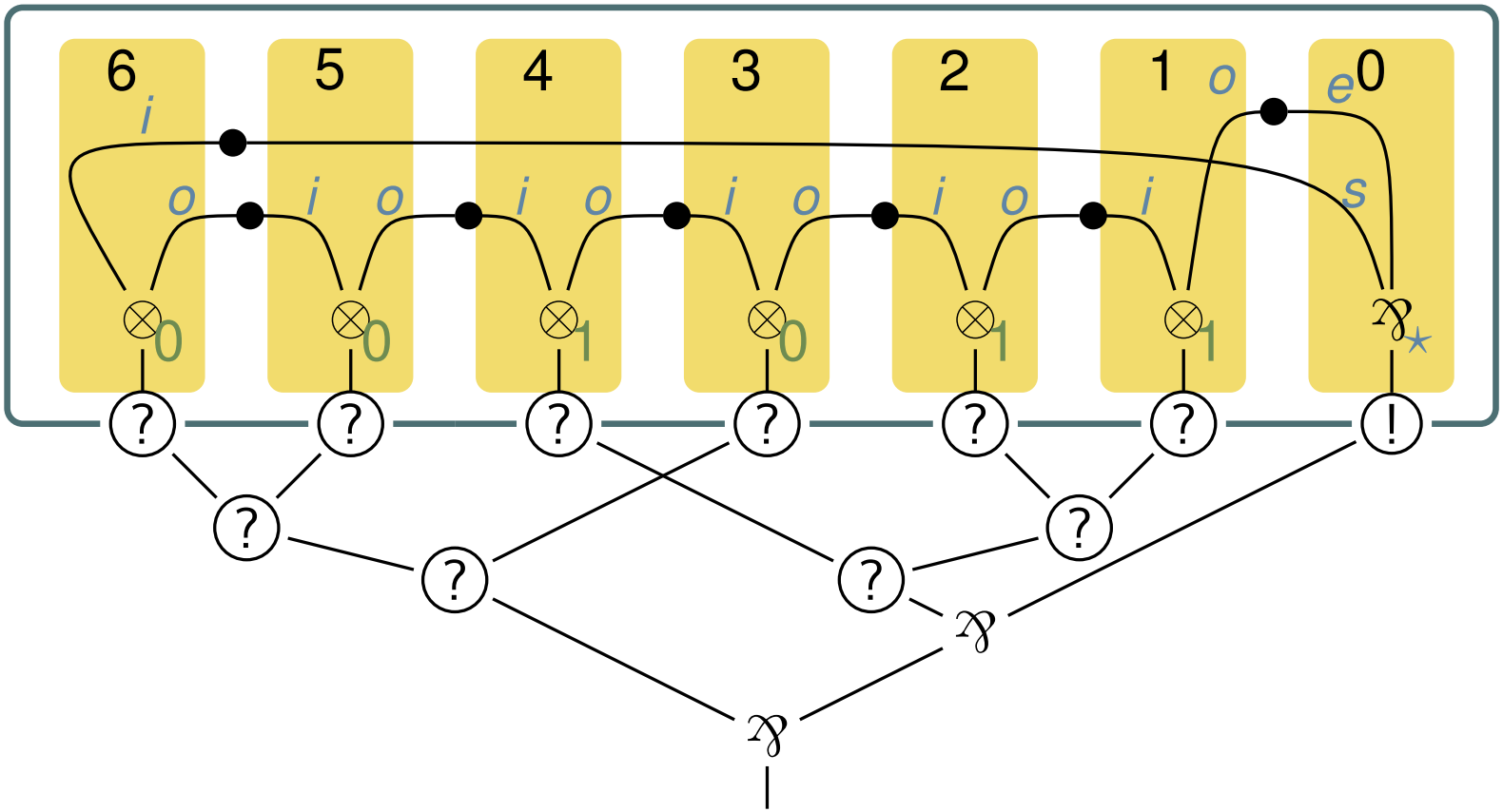
Entiers $\to 0, 1, 2, 3, \ldots$

$\hookrightarrow$ Listes binaires $\to 001, 010, 011, 100, \ldots$

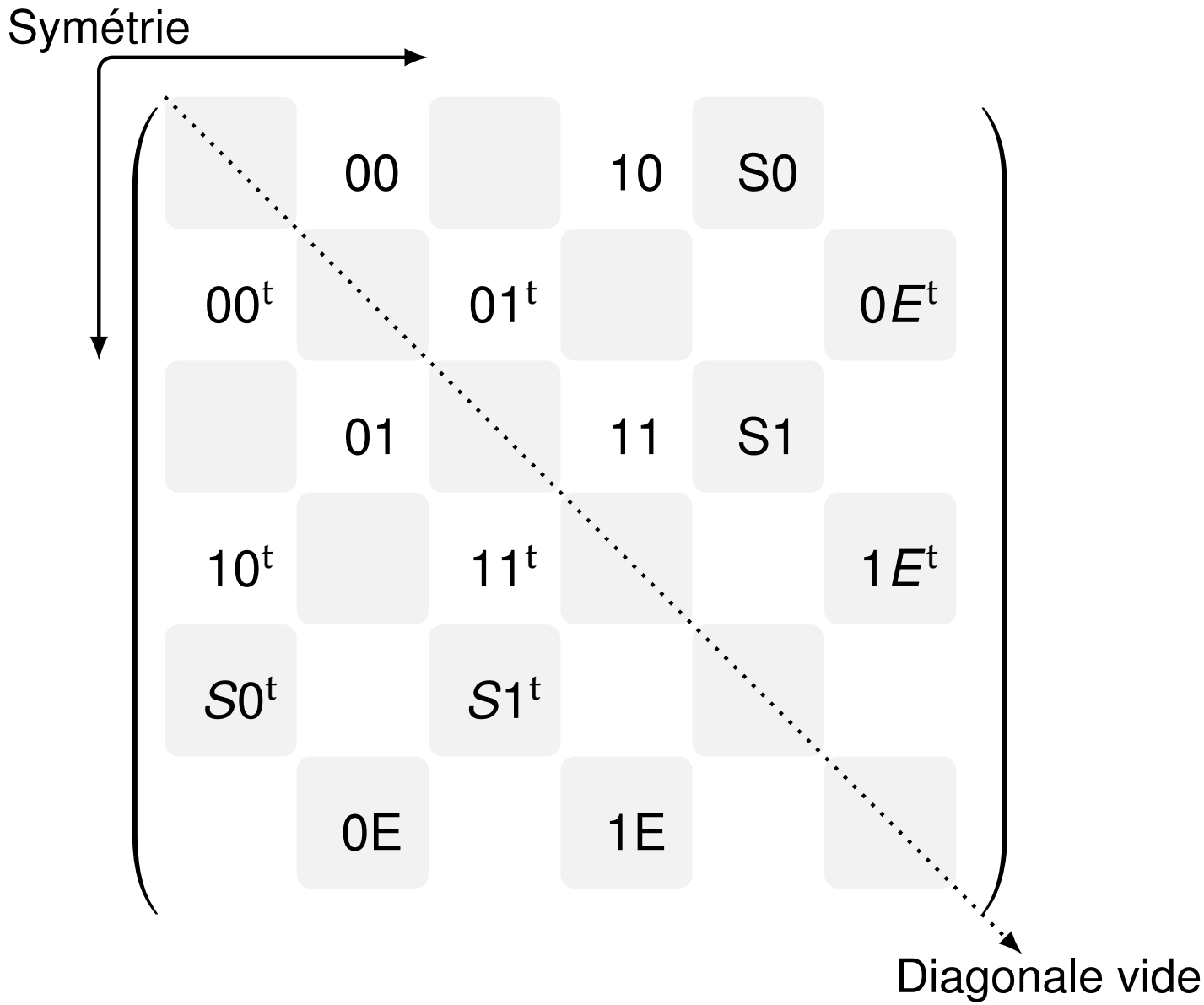
$\hookrightarrow$ $\lambda$-termes $\to \lambda f_0 \lambda f_1 \lambda_x \cdot f_0(f_1(f_1(\ldots(f_0 x)\ldots)$

$\hookrightarrow$ Preuves $\to \forall X!(X \multimap X) \multimap (!(X \multimap X) \multimap !(X \multimap X))$

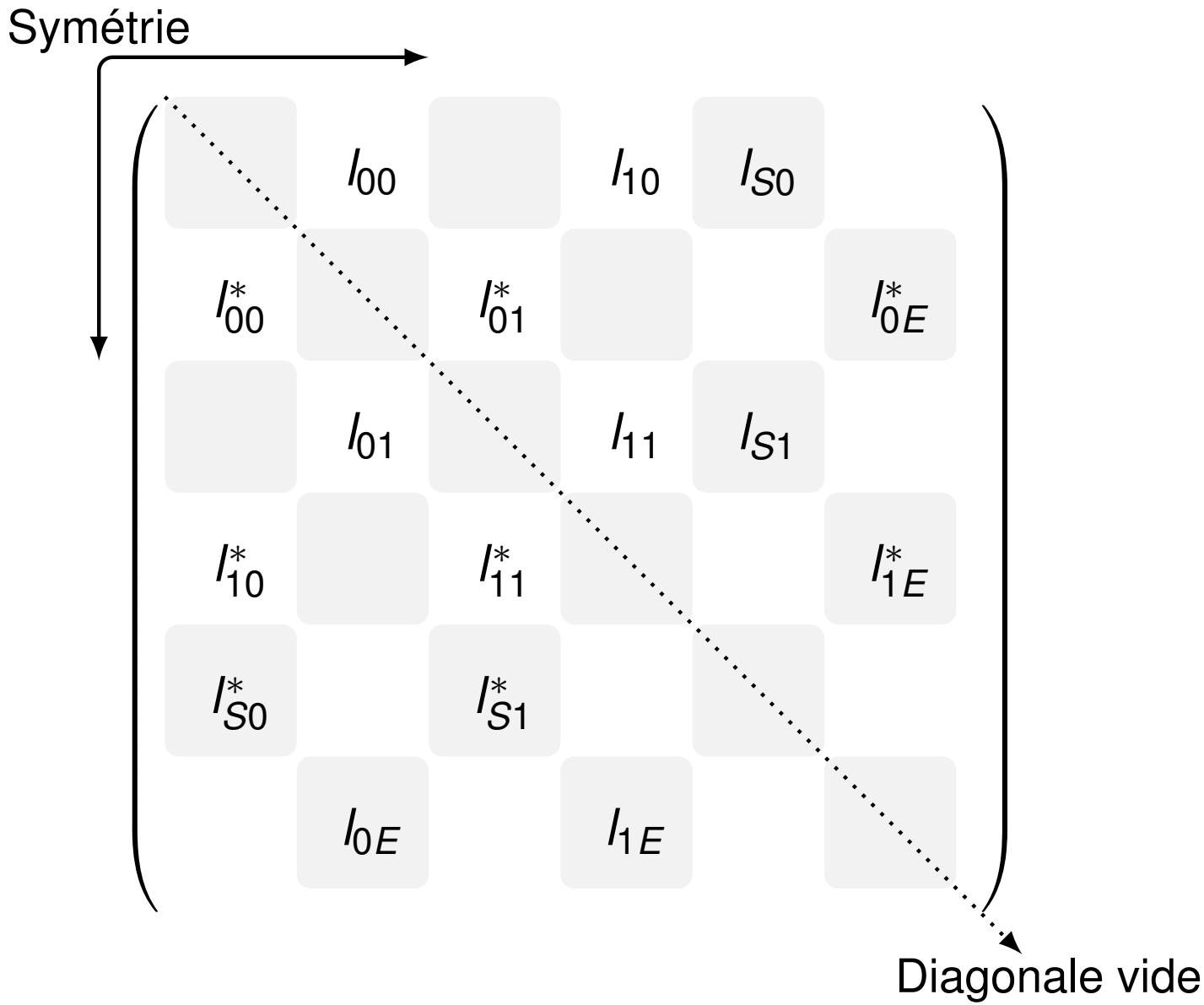
$\hookrightarrow$ Réseaux de preuves $\neg$



# Calcul et facteur hyperfini : Matrices et propriétés



# Calcul et facteur hyperfini : Opérateurs et propriétés



**Définition (Les entiers dans le facteur $\mathrm{II}_1$ hyperfini)**

On peut représenter $n \in \mathbb{N}$ comme un opérateur $N_n$ de $\mathrm{M}_6(\mathfrak{N})$.

**Définition (Observations)**

Une *observation* est un opérateur de $\phi \in \mathrm{M}_6(\mathfrak{S})$.

**Définition (Calculer, accepter)**

$\phi$ accepte $N_n$ si $\phi(N_n)$ est *nilpotent*, c.-à-d. si $\exists k \in \mathbb{N}$ tel que

$$(\phi(N_n))^k = 0$$

### Définition ($P_+$ et $P_{+,1}$)

On fixe une paire normative $(\mathfrak{N}, \mathfrak{G})$, pour $\phi \in \mathrm{M}_6(\mathfrak{G})$ une observation et $N_n \in \mathrm{M}_6(\mathfrak{N})$, $[\phi] = \{n \in \mathbb{N} \mid \phi(N_n) \text{ est nilpotent}\}$

$$\begin{aligned} P_+ &= \{\phi \mid \phi \text{ est une observation booléenne}\} \\ P_{+,1} &= \{\phi \mid \phi \in P_+ \text{ et } \|\phi\|_1 \leqslant 1\} \end{aligned}$$

$$\{P\} = \{[\phi] \mid \phi \in P\}$$

### Résultats

$$\{P_+\} \subseteq \textbf{co-NL}$$

$$\{P_{+,1}\} \subseteq \textbf{L}$$

### ⚠ Difficultés

Revenir au cadre fini nécessite un lemme puissant.

# Troisième partie

## Machines à pointeurs

## Sources

- HOFMANN et SCHÖPP, « Pure Pointer Programs with Iteration » (2010) et PURPLE.
- Immense littérature sur les pointeurs.

## Objectif

Établir un modèle qui rende compte du calcul des observations

## Avancées

### Représenter le calcul

- en développant un modèle *ad-hoc*
- de « bas niveau »
- en questionnant la notion de pointeur

## « Pointeurs = log-space » ?

Intuition à cadrer

- Pointeurs comme outil ? JAG, PURPLE,...
  Pointeur comme ruban ? SMM, KUM
- Algorithme de REINGOLD ne se résume pas à une manipulation de pointeurs.

## Fait classique

Automates finis bidirectionnels capturent naturellement **L**.

## Objectif

Adapter ce modèle aux opérateurs.

# Machines à pointeurs : Adapter le modèle

**La multiplication comme calcul** Une « nouvelle copie » de l'entrée et du programme est fournie à chaque étape

↪ Pas de possibilité de modification.

**Observation booléennes** « Branches » du calcul indépendantes.

↪ Calcul non-déterministe

**Nilpotence** *Toutes les « branches »* doivent s'annuler

↪ « Universellement non-déterministe »

**Produit croisé avec le groupe des permutations** Échanger de place plusieurs copies de l'entrée

↪ Plusieurs registres ou pointeurs.

**Entrée circulaire, initialisation particulière, etc.**

# Machines à pointeurs encodées comme opérateurs

## ⚠ Difficultés

- Trouver une représentation du rejet
- Préserver la contrainte sur la norme
- Décomposition des opérations élémentaires

## Théorème

*Pour toute machine à pointeur non-déterministe (resp. déterministe) $M$ il existe une observation $\phi_M \in P_+$ (resp. $\phi_M \in P_{+,1}$) tel que pour tout $n$, $M(n)$ accepte ssi $\phi_M N_n$ est nilpotent.*

## ⚘ Résultats

$$\textbf{co-NL} \subseteq \textbf{NPM} \subseteq \{P_+\}$$

$$\textbf{L} \subseteq \textbf{DPM} \subseteq \{P_{+,1}\}$$

$$\mathbf{NC^i} \subseteq \boxed{\mathbf{bPCC^i}} \subseteq \mathbf{AC^i} \subseteq \boxed{\mathbf{PCC^i}} \subseteq \mathbf{mBN^i}$$

$$\boxed{\begin{array}{c}\{P_{\leqslant 0}\} \\ \\ \{P_{+,1}\}\end{array}} = \begin{array}{c}\mathbf{co\text{-}NL} \\ \\ \mathbf{co\text{-}L}\end{array} = \begin{array}{c}\mathbf{NL} \\ \\ \mathbf{L}\end{array} = \begin{array}{c}\mathbf{NPM} \\ \\ \mathbf{DPM}\end{array}$$

## Circuits de preuve

- Mieux comprendre la modularité
- Inclusions strictes ?

## Pointeurs et opérateurs

- Algèbre d'unification
- Conserver les matrices finies de bout en bout
- Jouer avec la paire normative, les groupes
- Transférer les résultats des classes d'automates

# Bibliographie sélective I

AUBERT, Clément et Thomas SEILLER (2012). « Characterizing co-NL by a group action ». Dans : *Arxiv preprint* abs/1209.3422. arXiv :1209.3422 [cs.LO].

— (2013). « Logarithmic Space and Permutations ». Dans : *Arxiv preprint* abs/1301.3189. arXiv :1301.3189 [cs.LO].

GIRARD, Jean-Yves (2012). « Normativity in Logic ». Dans : *Epistemology versus Ontology*. Sous la dir. de Peter DYBJER, Sten LINDSTRÖM, Erik PALMGREN et Göran SUNDHOLM. T. 27. Logic, Epistemology, and the Unity of Science. Springer, p. 243–263.

HOFMANN, Martin et Ulrich SCHÖPP (2010). « Pure Pointer Programs with Iteration ». Dans : *ACM Trans. Comput. Log.* 11.4.

MOGBIL, Virgile (2009). « Non-deterministic Boolean Proof Nets ». Dans : *FOPARA*. Sous la dir. de Marko C. J. D. van EEKELEN et Olha SHKARAVSKA. T. 6324. Lecture Notes in Computer Science. Springer, p. 131–145.

MOGBIL, Virgile et Vincent RAHLI (2007). « Uniform Circuits, & Boolean Proof Nets ». Dans : *Proceedings of LFCS'07*. Sous la dir. de Sergei N. ARTËMOV et Anil NERODE. T. 4514. Lecture Notes in Computer Science. Springer, p. 401–421.

REINGOLD, Omer (2008). « Undirected connectivity in log-space ». Dans : *Journal of the ACM* 55.4, p. 17.

TERUI, Kazushige (2004). « Proof Nets and Boolean Circuits ». Dans : *LICS*. IEEE Computer Society, p. 182–191.