# Logique

## Théorème de Fagin

Florent Madelaine et Mamadou Kanté

M2 Clermont

Logique et Graphes

Année 2013 - 2014

# Plan du cours

Rappels

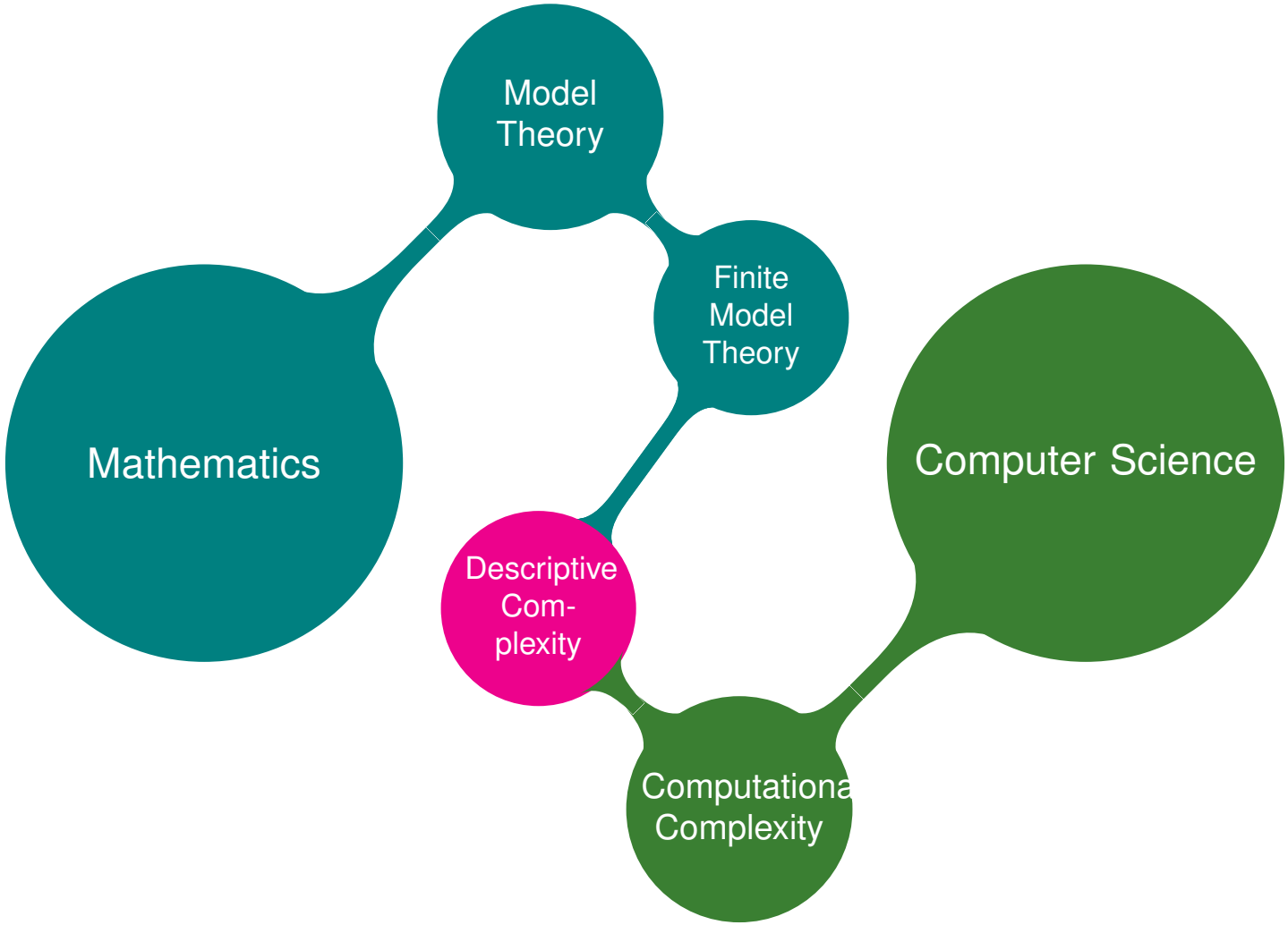
Preuve du théorème de Fagin

## Rappels

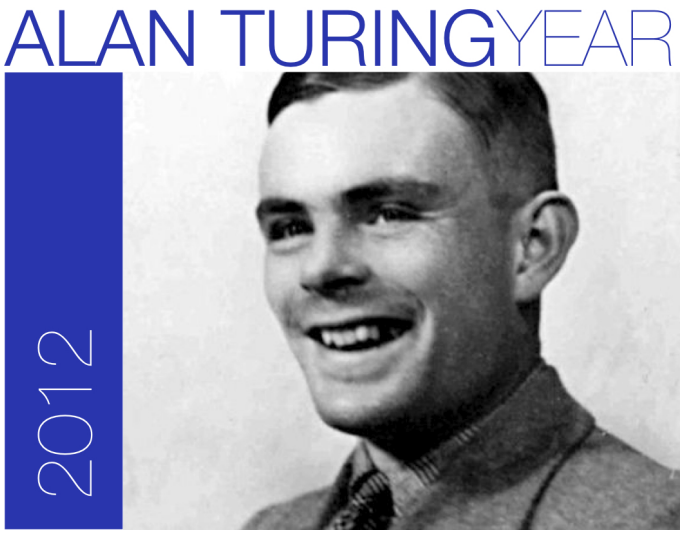
## Preuve du théorème de Fagin

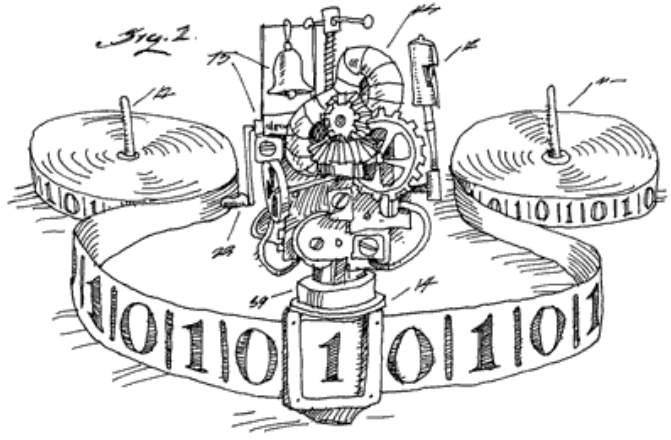
# Complexité Descriptive

La complexité descriptive fait le lien entre l'étude des logiques sur les structures finies (la théorie des modèles finis) et la complexité.



# Machine de Turing

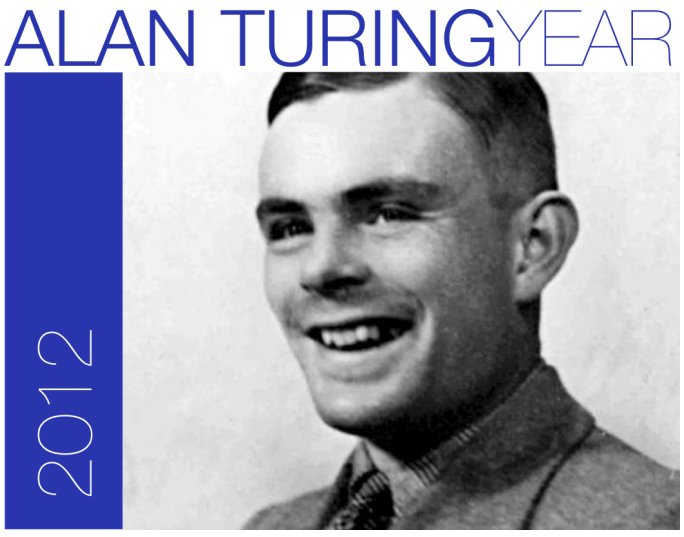


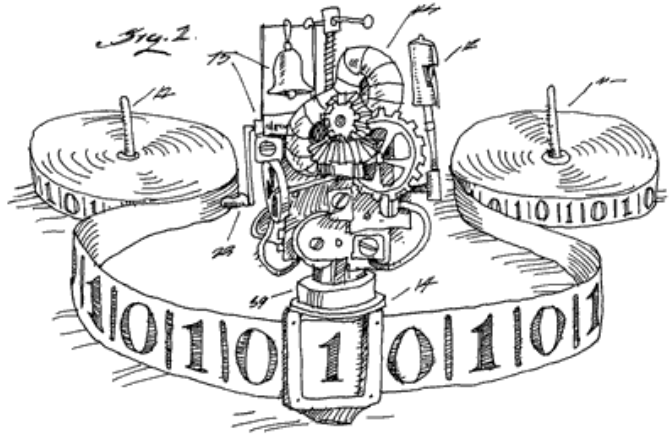



- unité de temps : 1 mouvement de la tête
- unité d'espace ?

  Pour mesurer finement la complexité, on ajoute un ruban de travail et on compte 1 unité d'espace par cellule utilisé de ce ruban de travail.

# Machine de Turing

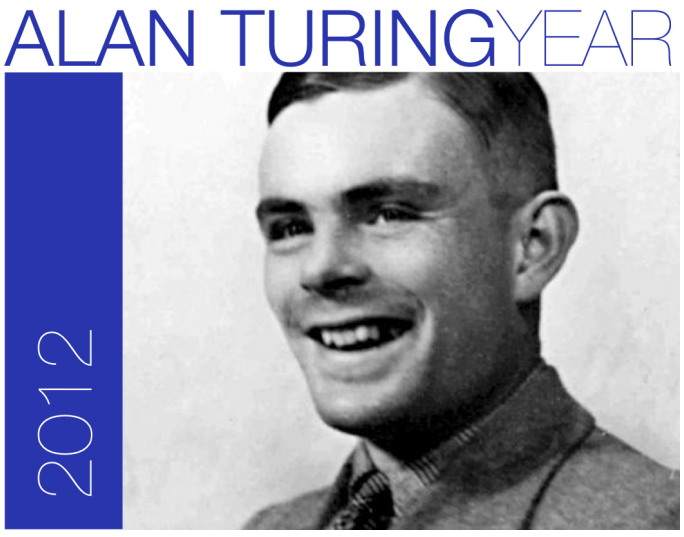


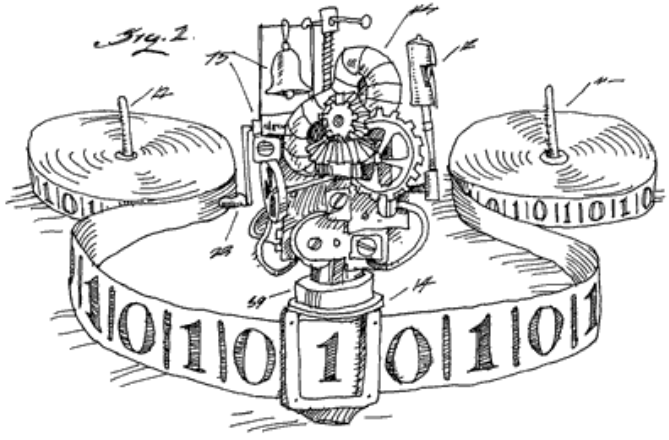




- unité de temps : 1 mouvement de la tête
- unité d'espace ?

  Pour mesurer finement la complexité, on ajoute un ruban de travail et on compte 1 unité d'espace par cellule utilisé de ce ruban de travail.

# Machine de Turing

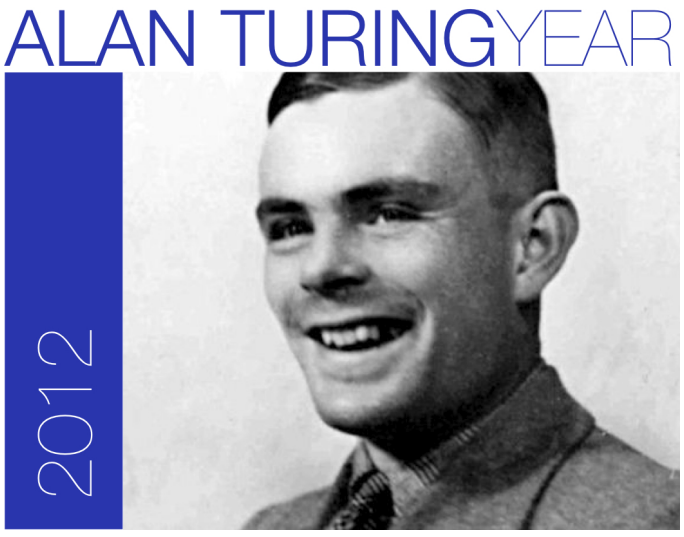


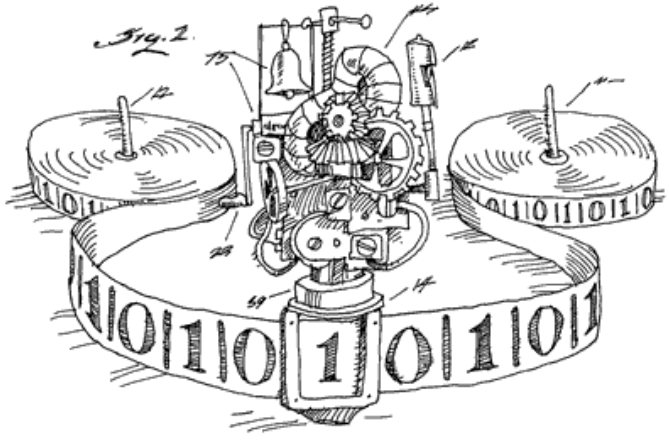




- unité de temps : 1 mouvement de la tête
- unité d'espace ?

  Pour mesurer finement la complexité, on ajoute un ruban de travail et on compte 1 unité d'espace par cellule utilisé de ce ruban de travail.

# Machine de Turing




- unité de temps : 1 mouvement de la tête
- unité d'espace ?

  Pour mesurer finement la complexité, on ajoute un ruban de travail et on compte 1 unité d'espace par cellule utilisé de ce ruban de travail.

# Machine de Turing non-déterministe

Non-déterminisme

- la fonction de transition devient une relation de transition
- On accepte si un calcul accepte
- On rejette si tous les calculs rejettent

Pour le temps polynomial, on peut opposer la machine déterministe qui calcule une réponse en temps polynomial à la machine non-déterministe qui vérifie en temps polynomial.

# Machine de Turing non-déterministe

Non-déterminisme

- la fonction de transition devient une relation de transition
- On accepte si un calcul accepte
- On rejette si tous les calculs rejettent

Pour le temps polynomial, on peut opposer la machine déterministe qui calcule une réponse en temps polynomial à la machine non-déterministe qui vérifie en temps polynomial.

# Machine de Turing non-déterministe

Non-déterminisme

- la fonction de transition devient une relation de transition
- On accepte si un calcul accepte
- On rejette si tous les calculs rejettent

Pour le temps polynomial, on peut opposer la machine déterministe qui calcule une réponse en temps polynomial à la machine non-déterministe qui vérifie en temps polynomial.

# Machine de Turing non-déterministe

Non-déterminisme

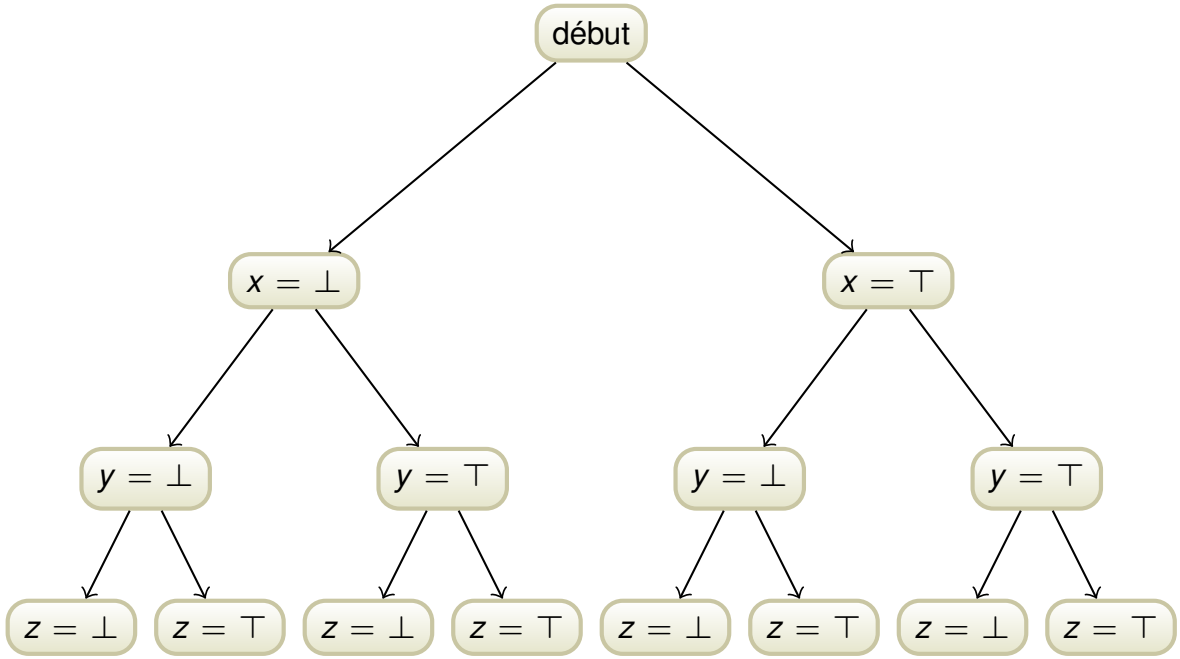
- la fonction de transition devient une relation de transition
- On accepte si un calcul accepte
- On rejette si tous les calculs rejettent

Pour le temps polynomial, on peut opposer la machine déterministe qui calcule une réponse en temps polynomial à la machine non-déterministe qui vérifie en temps polynomial.

# Qu'est-ce-qu'un temps raisonnable ?

P vs. NP

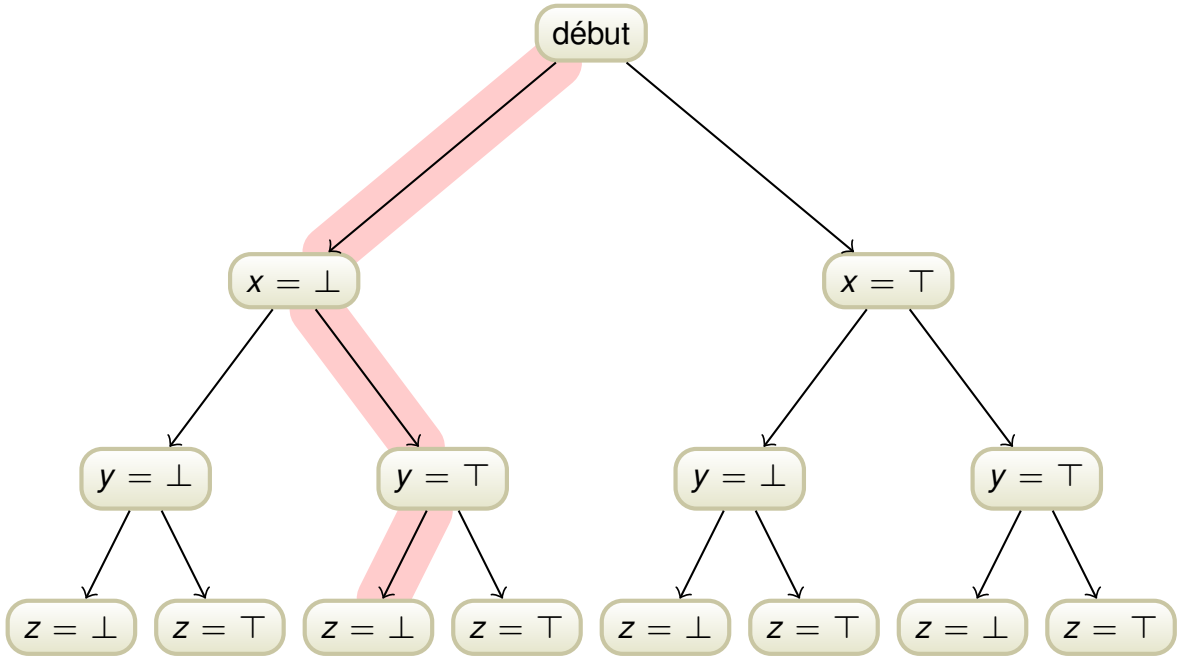
résolution polynomiale vs. vérification polynomiale



# Qu'est-ce-qu'un temps raisonnable ?

P vs. NP

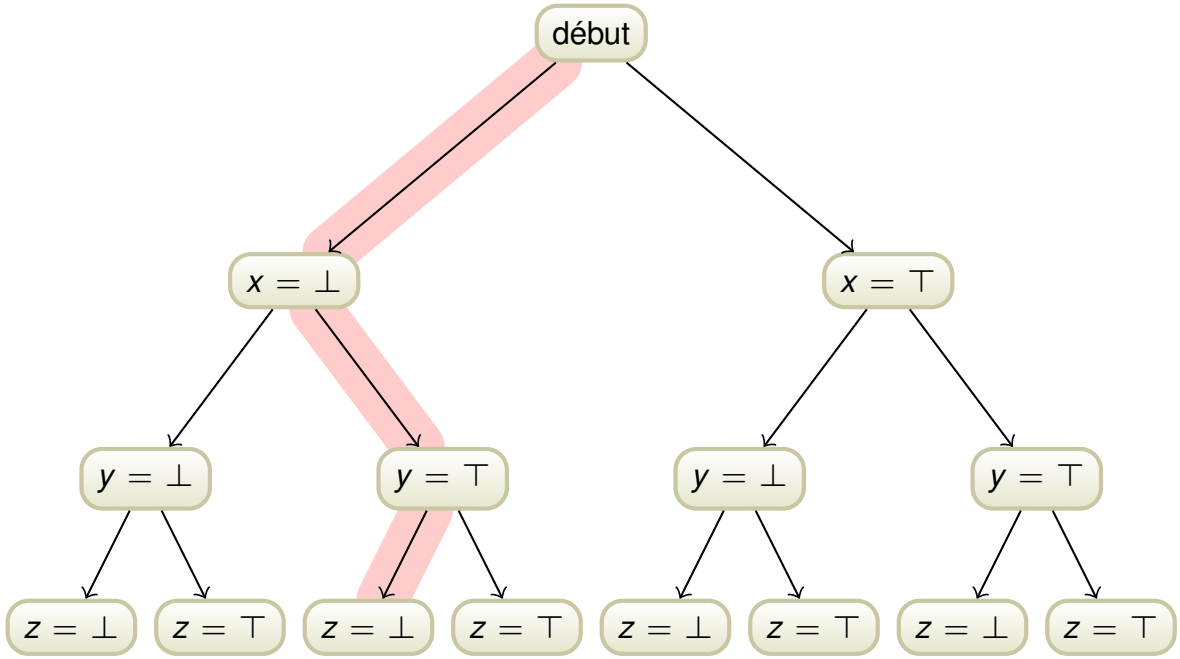
résolution polynomiale vs. vérification polynomiale



# Qu'est-ce-qu'un temps raisonnable ?

P vs. NP

résolution polynomiale vs. vérification polynomiale



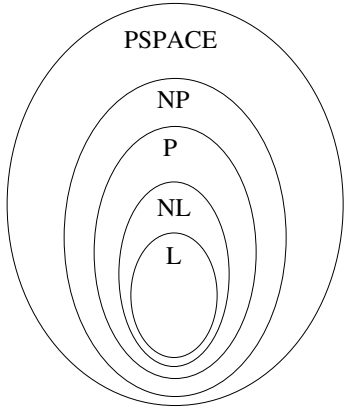

Pour SAT, $2^n$ possibilités pour $n$ variables.
Recherche brutale exhaustive d'une solution $\Rightarrow$ temps exponentiel.

# Classes de Complexité

Classe : ensemble de problèmes de décision (langages). Typiquement, pour lesquels il existe un algorithme (une machine de Turing) fonctionnant en limitant une ressource (temps, espace).

## Classes usuelles

| | | |
|---|---|---|
| Espace logarithmique | L | (Logspace) |
| (version non-dét.) | NL | (NLogspace) |
| Temps polynomial | NP | (NPtime) |
| (version non-dét.) | P | (Ptime) |
| Espace polynomial | Pspace | |



# Machine de Turing non-déterministe

Donnée par $(Q, \Sigma, \Delta, q_0, Q_a, Q_r)$ où

- $Q$ est un ensemble fini d'états
- $\Sigma$ est l'alphabet de l'entrée (sans perte de généralité, on supposera $\Sigma = \{0, 1\}$)
- $\Delta$ est l'alphabet du ruban de travail (sans perte de généralité on supposera $\Delta = \Sigma \cup \{\_\}$ où _ représente le symbole blanc)
- $\delta$ est la relation de transition, dans $(Q \times \Delta) \times (Q \times \Delta \times \{\leftarrow, \bullet, \rightarrow\})$
- $q_0$ est l'état initial
- $Q_a$ est l'ensemble des états acceptants
- $Q_r$ est l'ensemble des états rejetants

# Machine de Turing

exo

Donner une machine de Turing déterministe qui lit l'entrée de gauche à droite en intervertissant 0 et 1 et accepte l'entrée.

Pub

Vous pouvez utiliser le logiciel libre JFLAP pour simuler des machines de Turing (et d'autres machines comme des automates).

# Faiblesse de la logique du premier ordre

On peut démontrer qu'on ne peut pas exprimer des propriétés simples comme :

- le fait que l'univers a une taille paire
- l'accessibilité

De manière générale la logique du premier ordre ne peut exprimer que des propriétés locales (dans un sens très précis).

# La logique du second ordre

## Idée

On s'autorise à quantifier de nouveaux symboles de relations.

## Exemple

On peut « deviner »une relation binaire $L$ et dire qu'il s'agit d'un ordre linéaire.

$$\varphi_L := \exists L \forall x \neg (L(x,x) \wedge \forall x,y,z \big((x,y) \wedge L(y,z) \implies L(x,z)\big) \\ \wedge \forall x,y L(x,y) \vee L(y,x)$$

# Successeur

On peut déduire de l'ordre linéaire $L$, la relation de successeur sous-jacente.

$$\varphi_S := \varphi_L \wedge \exists S$$
$$\forall x, y S(x, y) \iff \big(L(x, y) \wedge \neg L(x, z) \wedge L(z, y)\big)$$

Une fois qu'on a un ordre, on peut facilement « marcher » le long du chemin défini par $B$ et vérifier qu'on a un nombre pair d'éléments.

# 3 colorabilité

$$\exists R \exists B \exists V \forall x \big(R(x) \vee B(x) \vee V(x)\big) \wedge$$
$$\forall x \Big(\neg\big(R(x) \wedge B(x)\big) \wedge \neg\big(R(x) \wedge V(x)\big) \wedge \neg\big(B(x) \wedge V(x)\big)\Big) \wedge$$
$$\forall x \forall y \neg\big(E(x,y) \wedge R(x) \wedge R(y)\big) \wedge$$
$$\neg\big(E(x,y) \wedge B(x) \wedge B(y)\big) \wedge \neg\big(E(x,y) \wedge V(x) \wedge V(y)\big)$$

- Dans l'exemple de la 3-colorabilité, les prédicats du second ordre sont tous des ensembles.
- Dans ce cas restreint on parle de logique monadique du second ordre (MSO)
- Notons qu'en général dans MSO le quantificateur universel est aussi autorisé
- La logique MSO joue un rôle central en théorie des langages

# Syntaxe et Sémantique

## Syntaxe

Comme pour la logique du premier ordre, mais on autorise les termes utilisant de nouveaux symboles [1] et on autorise leur quantification soit existentielle $\exists R$ ou universelle $\forall R$.

## Sémantique

Similaire à la logique du premier ordre mais on interprète maintenant un nouveau symbole $R$ d'arité $r$ comme une nouvelle relation $R^S$ d'arité $r$ sur la structure.

## Notation

On réserve en général les lettres majuscules aux nouveaux symboles et on utilise des lettres minuscules pour les variables du premier ordre.

1. On se donne une infinité de symboles pour chaque arité.

# Différents fragments de la logique du second ordre

| | | |
|---|---|---|
| SO | logique du second ordre | extension de la logique du premier ordre avec des « nouveaux symboles »représentant des relations |
| MSO | logique du second ordre monadique | restriction de la logique du second ordre où les nouveaux symboles sont forcément des ensembles de sommet (arité 1) |
| ESO | logique du second ordre existentielle | de la forme $\exists$ second ordre suivi d'une formule du premir ordre |
| FO | logique du premier ordre | pas de nouveaux symboles |

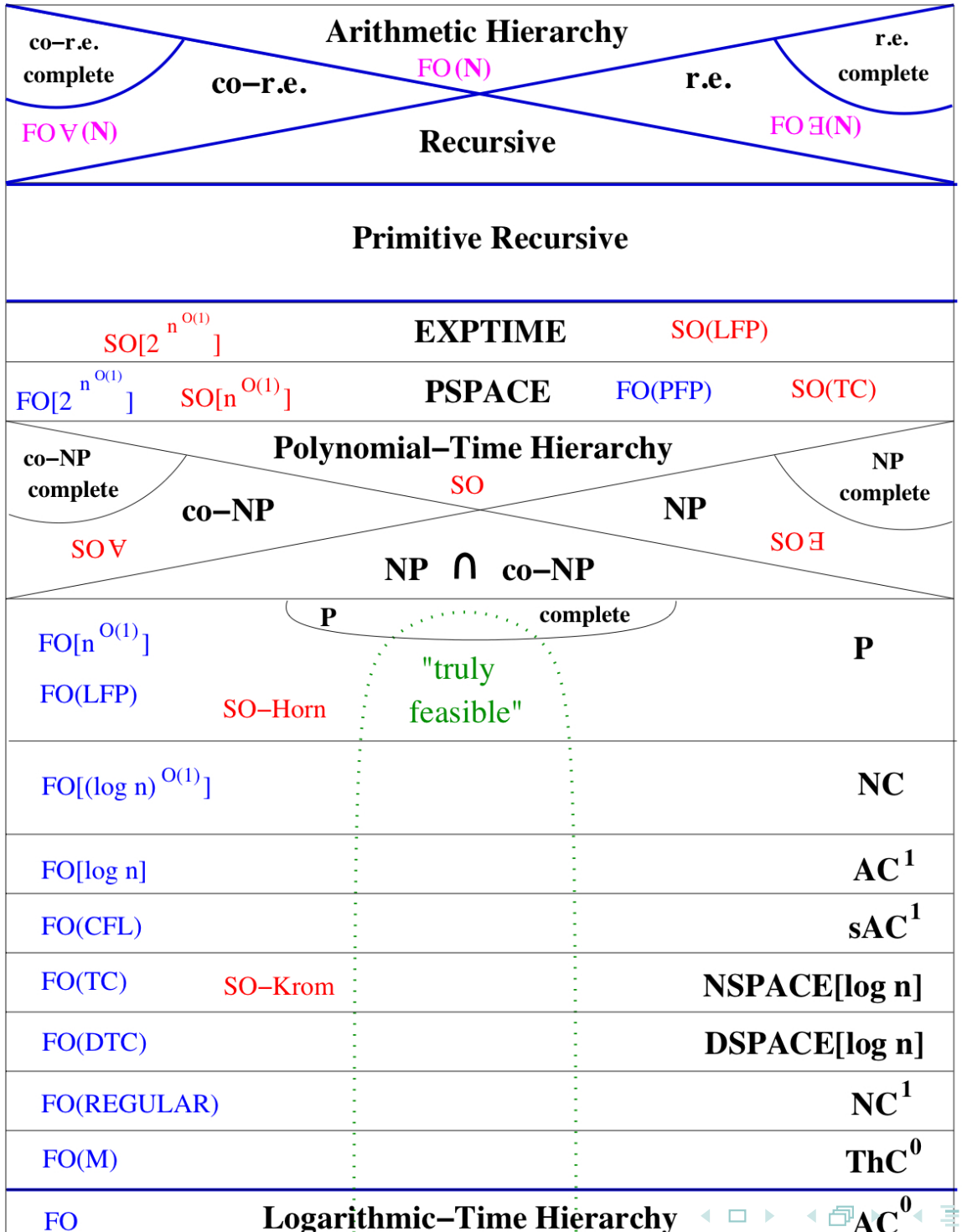
# Deux résultats de complexité descriptive

**Théorème (Büchi)**

*Pour les mots : Langage réguliers = MSO*

**Théorème (Fagin)**

*Pour les structures : NP = ESO*


Arithmetic Hierarchy
co−r.e. complete
co−r.e.
FO(N)
r.e.
r.e. complete
FO∀(N)
Recursive
FO∃(N)
Primitive Recursive
SO[2^n^O(1)]
EXPTIME
SO(LFP)
FO[2^n^O(1)]
SO[n^O(1)]
PSPACE
FO(PFP)
SO(TC)
Polynomial−Time Hierarchy
co−NP complete
co−NP
SO
NP
NP complete
SO∀
SO∃
NP ∩ co−NP
P
complete
FO[n^O(1)]
FO(LFP)
SO−Horn
"truly feasible"
P
FO[(log n)^O(1)]
NC
FO[log n]
AC^1
FO(CFL)
sAC^1
FO(TC)
SO−Krom
NSPACE[log n]
FO(DTC)
DSPACE[log n]
FO(REGULAR)
NC^1
FO(M)
ThC^0
FO
Logarithmic−Time Hierarchy
AC^0

Rappels

# Preuve du théorème de Fagin

# Théorème de Fagin

## Théorème

*Pour les structures : NP = ESO*

*C'est-à-dire que*

- *pour toute formule fixée $\Phi$ de ESO, le problème associé*

$$\{\textit{structures finies } S \textit{ telles que } S \models \Phi\}$$

  *est dans NP*

- *pour tout problème de décision sur les structures finies $\Omega$ de NP, il existe une formule $\Phi_\Omega$ de ESO telle que pour toute structure finie S,*

$$S \in \Omega \iff S \models \Phi_\Omega$$

# model checking de ESO est dans NP

## à montrer

Pour toute formule fixée $\Phi$ de ESO, le problème associé

$$\{\text{structures finies } S \text{ telles que } S \models \Phi\}$$

est dans NP.

## première étape

Montrer que le même problème pour FO est dans P

## deuxième étape

La formule $\Phi$ de ESO est de la forme $\exists R_1, \exists R_2, \ldots, \exists R_\ell \ \varphi$ où $\varphi$ est dans FO. Pour une structure finie $S$ donnée, la machine NP devine les valeurs de $R_1, R_2, \ldots, R_\ell$ puis utilise l'algo polynomial pour résoudre $S + R_1, R_2, \ldots, R_\ell \models \varphi$.

# Codage d'une structure

Il faut forcément qu'on détaille un peu comment on va coder une structure (pour nous un graphe pour simplifier).
Pour une structure $S$ de taille $n$,

- on aura $n$ zéros puis un 1 (code la taille du domaine).
- puis un mot binaire de longueur $n^k$ pour chaque prédicat d'arité $k$ de la signature.

# Exemple

le mot

$$000101010000010000000001$$

code une structure à trois sommets $\{0,1,2\}$ et deux prédicats binaires $R_1$ et $R_2$ où $R_1 = \{(0,1)\,(1,0)\,(2,2)\}$ et $R_2 = \{(2,2)\}$. Un 1 dans le mot en rouge correspond à un tuple de $R_1$. Les tuples sont classés dans l'ordre lexicographiques.

$$\begin{array}{ccccccccc} 0 & 0 & 0 & 1 & 1 & 1 & 2 & 2 & 2 \\ 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 \end{array}$$

$$\begin{array}{ccccccccc} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{array}$$

Notons que le code dépent de l'ordre des sommets qu'on a choisi.

# Exemple

le mot

$$000101010000010000000001$$

code une structure à trois sommets $\{0,1,2\}$ et deux prédicats binaires $R_1$ et $R_2$ où $R_1 = \{(0,1)\,(1,0)\,(2,2)\}$ et $R_2 = \{(2,2)\}$. Un 1 dans le mot en rouge correspond à un tuple de $R_1$. Les tuples sont classés dans l'ordre lexicographiques.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 1 | 2 | 2 | 2 |
| 0 | 1 | 2 | 0 | 1 | 2 | 0 | 1 | 2 |
| | | | | | | | | |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |

Notons que le code dépent de l'ordre des sommets qu'on a choisi.

# Importance de l'ordre

Notons que le code dépent de l'ordre des sommets qu'on a choisi.

Deux représentations pour notre exemple.

- Pour l'ordre $0, 1, 2$.

$$000101010000100000001$$

- Pour l'ordre $2, 0, 1$.

$$000110000101010000000$$

# Détails

Pour une formule $\varphi$ du première ordre dont les sous-formules ont au plus $k$ variables libres, on peut donner un algo en $O(n^k)$.
L'idée consiste à proposer un algorithme par induction sur la forme syntaxique de la formule.
Voir détails donnés en cours au tableau.

# NP vers ESO

Soit $\Omega$ un problème de NP sur des structures finies (on prendra la signature d'un graphe pour simplifier la preuve). Soit $M$ une machine de Turing non-déterministe qui décide $\Omega$ et fonctionne en temps polynomial $O(n^k)$.
On suppose sans perte de généralité que la machine lit l'entrée dans son intégralité. La machine visite au plus $n^k$ positions distinctes du ruban en temps au plus $n^k$.
On construit une formule $\Phi_\Omega$ de ESO telle que pour toute structure finie $S$,

$$S \in \Omega \iff S \models \Phi_\Omega$$

# NP vers ESO

## Les prédicats existentiels de la formule

Ils sont tous d'arité $2k$ qu'on voit comme des coordonnées position $\times$ instant

- $T_0, T_1, T_2$ "symbole lu $0, 1$ ou _"
- pour chaque état $q$ un prédicat $H_q$ "position/état tête de la machine"

à l'exception de $L$ qui est un prédicat binaire

- $L$ prédicat binaire

# Ce que la formule dit

- $L$ est un ordre linéaire (cf cours précédent)
- Dans chaque configuration, chaque case du ruban contient exactement un symbole
- La machine est dans un seul état à chaque instant
- Un jour la machine accepte (entre dans un état de $Q_a$)
- Les $T_i$ et les $H_q$ respecte la relation de transition $\delta$
- Une sous-formule détaillant la configuration initiale.

# Exemple

Dans chaque configuration, chaque case du ruban contient exactement un symbole.

$$\forall \bar{p} \forall \bar{t}$$

$$\begin{aligned}
&\big(T_0(\bar{p},\bar{t}) \vee T_1(\bar{p},\bar{t}) \vee T_2(\bar{p},\bar{t})\big) \\
&\quad \wedge \neg\big(T_0(\bar{p},\bar{t}) \wedge T_1(\bar{p},\bar{t})\big) \\
&\quad \wedge \neg\big(T_0(\bar{p},\bar{t}) \wedge T_2(\bar{p},\bar{t})\big) \\
&\quad \wedge \neg\big(T_1(\bar{p},\bar{t}) \wedge T_2(\bar{p},\bar{t})\big)
\end{aligned}$$

# Détails

Pour les autres cas, voir cours au tableau.
Pour le dernier cas : une sous-formule détaillant la configuration initiale.
On doit décrire logiquement que à l'instant initial (pour $\bar{t} = \overline{\text{min}} = (\text{min}, \text{min}, \ldots, \text{min})$ où min est la constante correspondant au premier élément de l'ordre linéaire $L$) on a sur le ruban le code du graphe correspondant à l'ordre $L$.
C'est-à-dire que les prédicats $T_i(\bar{p}, \overline{\text{min}})$ ont les bonnes valeurs par rapport à notre codage des structures modulo l'ordre $L$ sur les sommets deviné.
Ce n'est pas complètement trivial : reportez vous svp à la preuve du livre de Libkin.

# Corollaire

exo

Donnez une caractérisation logique de coNP

exo

Montrer le théorème de Cook (à savoir que SAT est NP-complet).

exo

Regardez la preuve de l'exo précédent dans le bouquin de Libkin. Appliquez sa construction à la formule pour la 3-colorabilité et le graphe à 3 sommets représentant un chemin de longueur 2.

# Prochains cours

4,6 MSO et décomposition arborescentes (théorème de Courcelle) (Mamadou Kanté)

5 problème de contraintes et décomposition (moi)