

Maths 4.2

Autres modèles de calcul

Florent Madelaine

BUT 2 Informatique



Plan

- ➊ Automate à pile
- ➋ Machine de Turing
- ➌ Langage et compilation

Introduction

Nous avons vu brièvement le modèle des automates finis et la correspondance avec les expressions régulières, avec des méthodes pour passer de l'un à l'autre, en particulier avec des ateliers disponibles dans JFLAP.

Nous avons également vu que le **lemme de la pompe** permet de montrer qu'un langage n'est pas régulier dans certains cas.

Nous allons maintenant évoquer des modèles plus puissants de calcul.

Automate à Pile (Pushdown Automata)

Idée

Automate + Pile = automate à pile.

- La mémoire n'est plus bornée
- Nous avons une mémoire « de contrôle » finie (les états : en gros les lignes du programme)
- Mais nous disposons de la **pile** pour stocker de l'information aussi haut que l'on souhaite.
- Par contre, ce n'est pas la princesse aux petits pois. On n'accède qu'à la cellule du haut de la pile.

Fonctionnement

Une instruction du programme

état actuel, lettre en haut de la pile, lettre courante du mot
~~~~~  
nouvel état, pile (pop ou push ou rien).

- Pour accepter un mot, il y a plusieurs définitions qui ne change pas la puissance du modèle.
  - Parfois on demande que la pile soit vide.
  - Parfois on demande que l'automate à pile soit dans un état acceptant.
- En pratique avec JFLAP, il y a un symbole de bas de pile Z et on indique l'opération en décrivant le haut de la pile avant et après la transition.

# Exemples d'opérations sur la pile

- (Push) Pousser un a sur pile vide :  $Z \rightsquigarrow aZ$
- (Push) Pousser un b sur pile qui contient un a :  $a \rightsquigarrow ba$
- (Read) Laisser la pile non vide qui contient un a :  $a \rightsquigarrow a$
- (Pop) Enlever le symbole en haut de la pile qui est un a :  
 $a \rightsquigarrow \lambda$

# Exemple pour les parenthèses

## Idée

- On empile chaque parenthèse ouvrante qu'on va lire
- On dépile chaque parenthèse ouvrante de la pile si on lit la parenthèse fermante correspondante (sinon on rejette)
- On accepte si la pile est vide (sinon on a oublié de fermer des parenthèses).

# La Machine de Turing

## Idée

Automate + Ruban sur lequel on peut écrire = ordinateur

- La mémoire n'est plus bornée
- Nous avons une mémoire « de contrôle » finie (les états : en gros les lignes du programme)
- Mais nous disposons du **ruban** pour lire l'entrée et stocker de l'information autant qu'on le souhaite.
- Une tête de lecture indique la position de la cellule qu'on peut lire.
- La tête peut écrire.

# Fonctionnement

## Une instruction du programme

état actuel, lettre lue



nouvel état, lettre écrite, mouvement de la tête

- variantes possibles : avec plusieurs rubans et des têtes qui se déplacent indépendamment sur chaque ruban.

## Exemple pour $a^n b^n$

### Idée

- On efface un a qu'on va lire
- On va à la fin du mot chercher un b à effacer
- On accepte si on a mangé le mot en entier

# Langage et Compilation

En pratique il y a souvent 2 étapes.

- lexing (automate classique)
- parsing (automate à pile ou un peu plus)

En pratique, on utilise plutôt des **grammaires formelles** pour décrire les langages.

# Hiérarchie de Chomsky

[https://en.wikipedia.org/wiki/Chomsky\\_hierarchy](https://en.wikipedia.org/wiki/Chomsky_hierarchy)

# Outils pour la compilation

Il y a de nombreux outils permettant de générer des parseurs.

[https://en.wikipedia.org/wiki/Comparison\\_of\\_parser\\_generators](https://en.wikipedia.org/wiki/Comparison_of_parser_generators)

Il peut être utile parfois de mettre en oeuvre un mini-langage et d'écrire un interpréteur ou un compilateur vers un langage classique.

[https://en.wikipedia.org/wiki/Domain-specific\\_language](https://en.wikipedia.org/wiki/Domain-specific_language)

Par exemple, pour décrire le calcul des impôts.

<https://blog.merigoux.ovh/fr/2019/12/20/impots-formels.html>