Langages Formels, Calculabilité, Complexité

Mickaël Thomazo
Lucas Larroque

23 novembre 2023

Table des matières

Première partie

Cours 1 28/09

1 Langages, Automates, RegExp, Monoïdes finis

Définition 1.0.1. *On appelle* alphabet *un ensemble fini* Σ *de lettres.*
On appelle mot *une suite finie de lettres.*
On appelle langage *un ensemble de mots*

Définition 1.0.2. *On appelle* automate sur l'alphabet Σ *un graphe orienté dont les arêtes sont étiquetées par les lettres de l'alphabet* Σ
Formellement, c'est un quadruplet $\mathcal{A} = (Q, \Sigma, I, F, \delta)$ *ou :*

- Q *est un ensemble fini d'états*
- Σ *est un alphabet*
- $I \subseteq Q$
- $F \subseteq Q$
- $\delta : Q \times \Sigma \to 2^Q$

Un calcul de $\mathcal{A}$ *sur* $w = a_0 \dots a_n$ *est une séquence* $q_0 \dots q_n$ *telle que* $q_0 \in I$, $\forall i \geq 1,\ q_i \in \delta(q_{i-1}, a_i)$
On appelle Langage reconnu par $\mathcal{A}$ *l'ensemble* $\mathcal{L}(\mathcal{A}) = \{w \in \Sigma^* \mid \exists q_0 \dots q_n$ *calcul de* $\mathcal{A}$ *sur* w *où* $q_n \in F\}$
On dit que $\mathcal{A}$ *est déterministe si :*

- $\forall q, a, |\delta(q, a)| \leq 1$
- $|I| = 1$

Définition 1.0.3. *Une expression régulière est de la forme :*

- $a \in \Sigma$
- $\varnothing$
- $r + r$ *(+ désigne l'union :* $L_1 + L_2 = \{w \in L_1 \cup L_2\}$*)*
- $r \cdot r$ *($\cdot$ désigne la concaténation :* $L_1 \cdot L_2 = \{w_1 w_2 \mid w_1 \in L_1,\ w_2 \in L_2\}$*)*
- r^* *($*$ désigne l'étoile de Kleene,* $L^* = \left\{ \bigodot_{w \in s} w \ \mid\ s \in \bigcup_{n \in \mathbb{N}} L^n \right\}$*)*

Définition 1.0.4 (Automate des Parties)**.** *On pose, si* $\mathcal{A} = (Q, \Sigma, I, F, \delta)$ *est un automate :*

- $\hat{Q} = 2^Q = \{q_S \mid S \subset Q\}$
- $\hat{I} = \{q_I\}$
- $\hat{F} = \{q_S \mid S \cap F \neq \varnothing\}$
- $\hat{\delta}(q_S, a) = \{q_{S'}\}$ *avec* $S' = \bigcup_{q \in S} \delta(q, a)$

Alors, $\hat{\mathcal{A}} = (\hat{Q}, \Sigma, \hat{I}, \hat{F}, \hat{\delta})$ *est un automate déterministe reconnaissant* $\mathcal{L}(\mathcal{A})$

Démonstration. On procède par double inclusion :

- ($\subset$) On introduit un calcul de $w \in \mathcal{L}(\mathcal{A})$ sur $\hat{\mathcal{A}}$ et on vérifie par récurrence que son dernier état est final.
- On procède de même pour la réciproque.

$\blacksquare$

Définition 1.0.5. *Un monoïde est un magma associatif unifère.*
Un morphisme de monoïde est une application $\varphi : (N, \cdot_N) \to (M, \cdot_M)$ *telle que :*

- $\varphi(1_N) = 1_M$
- $\varphi(n_1 n_2) = \varphi(n_1)\varphi(n_2)$

Un langage L est reconnu par $(M, \times)$ ssi il existe $P \subset M$ tel que $L = \varphi^{-1}(P)$ où φ est un morphisme de Σ^* dans M

Proposition 1.0.1. *$L \subseteq \Sigma^*$ est reconnu par un automate ssi L est reconnu par un monoïde fini.*

Démonstration.
- Soit L reconnu par un monoïde fini $(M, \times)$. Soit φ un morphisme tel que $L = \varphi^{-1}(P),\ P \subset M$. On pose $\mathcal{A} = (M, \Sigma, \{1\}, P, \delta)$ où $\delta(q, a) = q \times \varphi(a)$. Alors, $\mathcal{A}$ reconnaît L.
- Soit $\mathcal{A}$, déterministe, complet, reconnaissant L. Pour $a \in \Sigma$, $a \to \varphi_a : q \in Q \mapsto \delta(q, a)$ induit par induction un morphisme de $(\Sigma^*, \cdot)$ dans $(Q^Q, \circ)$. Alors, avec $P = \left\{f \in Q^Q \mid f(i) \in F_{\mathcal{A}}\right\}$. On a défini le monoïde des transitions de $\mathcal{A}$.

■

Deuxième partie
Cours 2 - 5/10

2 Lemme de Pompage

Théorème 2.0.1 (Lemme de Pompage/Lemme de l'Etoile)**.** *Si L est un langage régulier, $\exists n \in \mathbb{N}$ $\forall w \in L, |w| \geq n \Rightarrow \exists x, y, z$ tels que :*
- $w = xyz$
- $|xy| \leq n$
- $y \neq \varepsilon$
- $\forall n \geq 0, xy^n z \in L$

Démonstration. Faire un calcul de $\mathcal{A}$ sur w tel que $|w| \geq n$. Celui-ci passe deux fois par le même état. ■

3 Langages Quotients

3.1 Quotients d'un Langage à Gauche

Définition 3.1.1 (Quotient à Gauche)**.** *Soit $L, K \subseteq \Sigma^*, u \in \Sigma^*$.*
Le quotient à gauche de L par u noté $u^{-1}L$ est : $\{v \in \Sigma^* \mid uv \in L\}$
Le quotient à gauche de L par K, $K^{-1}L$ est $\bigcup_{u \in K} u^{-1}L$

Proposition 3.1.1.
- $w^{-1}(K + L) = w^{-1}K + w^{-1}L$
- $(wa)^{-1}L = a^{-1}(w^{-1}L)$
- $w^{-1}(KL) = (w^{-1}K \cdot L) + \sum\limits_{u \in L, v \in \Sigma^* w = uv} v^{-1}L$

3.2 Quotient d'un Automate à Gauche

Définition 3.2.1. *On définit le quotient à gauche d'un automate par un mot u comme celui obtenu en remplaçant les états initiaux par les résultats d'un calcul de l'automate sur u.*

Proposition 3.2.1. *L est régulier si et seulement si il a un nombre fini de quotients à gauche.*

Démonstration.
- Un automate reconnaissant L a au plus un quotient par état.
- Posons $A_L = \left(\Sigma, \left\{u^{-1}L \mid u \in \Sigma^*\right\}, I = L = \varepsilon^{-1}L, F =, \delta(w^{-1}L, a) = a^{-1}(w^{-1}L)\right)$
  Par récurrence, le calcul de A_L sur w termine en $w^{-1}L$

■

3.3 Construction de l'Automate Minimal

Définition 3.3.1. *Deux états q_1, q_2 sont distingables si : $\exists w \in \Sigma^*, \delta(q_1, w) \in F, \delta(q_2 \notin F)$.*

Proposition 3.3.1. *q_1 et q_2 sont distingables s'ils n'ont pas même quotient à gauche. Si $\delta(q, a)$ est distingable $\delta(q^{'}, a)$, $q, q^{'}$ sont distingables.*
La relation q, q' sont distingables est une relation d'équivalence.

Troisième partie

Cours 3 - 12/10

4 Langages et Logique

4.1 Objectif

On associe à $w \in \Sigma^*$ une structure D_w et à $L \subseteq \Sigma^*$ une structure φ_L telles que : $w \in L \setminus \{\varepsilon\} \Leftrightarrow D_w \vdash \varphi_L$. On se place dans le cadre de la logique du premier ordre et de la monadique du second ordre.

Définition 4.1.1. *On définit $\mathsf{pos}(w) = \{0, \dots, |w| - 1\}$. On définit une signature i.e. un ensemble de relations :*

$$\forall a \in \Sigma,\ L_a \text{ d'arité } 1$$
$$\leq, \text{ l'ordre strict sur } \mathsf{pos}(w)$$

On définit alors la structure $D_w = \Big(\mathsf{pos}(w), \left\{L_a^{D_w}\right\}_{a\in\Sigma}, <_w\Big)$

Remarque 4.1.0.1. *On aurait pû remplacer $<_w$ par $succ_w$, mais on perd en expressivité.*

4.2 Logique du Premier Ordre et Monadique du Second Ordre

Définition 4.2.1 (Logique du Premier Ordre)**.** *On définit par induction la logique du premier ordre.*

- *Constantes*
- *Variables*
- *Si R est une relation d'arité n, t_i des termes : $R(t_1, \dots, t_n)$*
- $\neg\varphi$, $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$
- $\forall x, \varphi, \exists\varphi$

On cherche à associer à $\varphi : \exists x,\ (L_0(x) \wedge \forall y(y < x \to (L_1(y))))$, un langage $L_\varphi = \{w \mid D_w \vdash \varphi\}$.

Définition 4.2.2. *Monadique du Second Ordre Sont des formules :*

- *$\forall X,\ \varphi$ avec X, variable du second ordre qui a une arité associée.*
- *$\exists X,\ \varphi$ avec X, variable du second ordre qui a une arité associée.*
- *$(x_1, \dots, x_n) \in X$ avec X d'arité n. On trouve aussi une formule pour les graphes qui mettent en relation deux sommets s, t :*

On se restreint dans la suite à des variables d'arité 1

On considère un vocabulaire qui contient une relation E ("arêtes d'un graphe"). On représente un graphe par D_G l'ensemble de ses sommets et E^{D_G} l'ensemble des arêtes de ce graphe.

Exemple 4.2.1. *On trouve alors une formule pour représenter tous les graphes 3-coloriables :*

$$\begin{aligned}\exists X_1, \exists X_2, \exists X_3\, &(\forall x\,(X_1(x) \vee X_2(x) \vee X_3(x))) \\ &\wedge (\forall x \forall y\,(E(x, y) \to (\neg\,(X_1(x) \wedge X_1(y)) \wedge \neg\,(X_2(x) \wedge X_2(y)) \wedge \neg\,(X_3(x) \wedge X_3(y)))))\end{aligned} \tag{1}$$

Exemple 4.2.2. *On trouve aussi une formule pour les graphes qui mettent en relation deux sommets s, t. Il s'agit de trouver une relation close par successeur qui contient s :*

$$\forall R\left([s \in R \wedge \forall x, y, (R(x) \wedge E(x,y)) \rightarrow R(y)] \rightarrow t \in R\right)$$

Ainsi, on peut en déduire une méthode pour reconnaître le langage d'un automate $\mathcal{A} = (\{0, \ldots, k\}, \Sigma, 0, \Delta, F)$ avec une formule $\varphi_{\mathcal{A}}$ de la monadique du second ordre.

Théorème 4.2.1. *Un langage $L = L(\mathcal{A})$ est régulier, si et seulement si il existe une formule $\varphi = \varphi_{\mathcal{A}}$ telle que $\forall w \in L, D_w \vdash \varphi$.*

Démonstration.

- ($\Rightarrow$) : On peut obtenir le premier élément d'un mot par la formule $\mathsf{first}(x) = \forall y((x = y) \vee x < y)$. On peut faire de même pour savoir si un couple est composé d'une paire successeur-successeuse de l'automate, ou si x est la dernière lettre.
  On sépare les positions d'un mot selon l'état de l'automate depuis lequel on part. Il faut alors vérifier que le premier élément est bien dans un état initial, que toute transition est bien valide, qu'on est dans au moins un état avant chaque lettre, et que la dernière position est bien écrite depuis une transition vers un état acceptant.
  On obtient alors, en notant k le nombre d'états, et 0 l'état initial :

$$\begin{aligned}\varphi_{\mathcal{A}} : \exists X_0, \ldots, \exists X_k & \left(\bigwedge_{i \neq j} \forall x,\ \neg\left(X_i(x) \wedge X_j(x)\right)\right) \\ & \forall x\ \ (\mathsf{first}(x) \rightarrow X_0(x)) \\ & \forall x, \forall y\ \left(\mathsf{succ}(x,y) \rightarrow \bigvee_{(i,a,j) \in \Delta} (X_i(x) \wedge L_a(x) \wedge X_j(y))\right) \\ & \forall x\ \left(\mathsf{last}(x) \rightarrow \bigvee_{\exists j \in F \mid (i,a,j) \in \Delta} (X_i(x) \wedge L_a(x))\right)\end{aligned} \tag{2}$$

- ($\Leftarrow$) : On procède par induction.
  - Initialisation : On peut facilement exhiber des automates qui reconnaissent les formules atomiques : $\mathsf{Sing}(X), X \subseteq Y, X \subseteq L_a$.
  - Hérédité : On raisonne sur les connecteurs, et on vérifie aisément, par les propriétés de cloture des langages réguliers le résultat. Pour ce qui est de la quantification existentielle, si le langage L défini par $\psi(X_1, \ldots, X_n)$ sur $\Sigma \times \{0,1\}^n$ est reconnu par $\mathcal{A}$. On exhibe un automate reconnu par $\varphi(X_1, \ldots, X_{n-1}) = \exists X_n \psi(X_1, \ldots, X_n)$, il n'a alors plus qu'a trouver une suite de $0-1$ qui définit la n-ième composante additionnelle et fonctionne sur $\Sigma \times \{0,1\}^n$ comme $\mathcal{A}$. Pour le $\forall$, il suffit de prendre la négation du $\exists$

■

Quatrième partie

Cours 4 : 26/10

On a déjà vu le type 3 de la hiérarchie de Chomsky : les langages réguliers. On passe aux langages algébriques, ou hors-contexte, définis par des grammaires hors-contextes.

5 Limites des Langages Réguliers

Le langage $\{a^n b^n \mid n \in \mathbb{N}\}$ n'est pas régulier. On va donc définir une classe de langage plus grande.

5.1 Grammaires

Définition 5.1.1 (Grammaire hors-contexte)**.** *Une grammaire hors-contexte est un quadruplet* (Σ, V, S, R) *où :*

- Σ *est un alphabet fini*
- V *est un alphabet fini disjoint de* Σ
- $S \in V$ *est un axiome*
- R *une sous partie de* $V \times (\Sigma \cup V)^\star$

Définition 5.1.2. *On dit que u produit (ou dérive) v en une étape si il existe* α, β, X, γ *tel que :*

- $u = \alpha X \beta$
- $v = \alpha \gamma \beta$
- $X \mapsto \gamma$ *est dans* R*.*

On note ceci $u \to v$*. On note* $u \xrightarrow{k} v$ *si u produit v en k étapes et* $u \xrightarrow{\star} v$ *si il existe un k.*

Définition 5.1.3. *Si G est une grammaire :* $\hat{\mathcal{L}}_G(x) = \left\{ w \in (\Sigma \cup V)^\star \mid x \xrightarrow{\star} w \right\}$ *On définit aussi* $\mathcal{L}_G(x) = \hat{\mathcal{L}}_G(x) \cap \Sigma^\star$*.*

Par exemple, pour la grammaire $S \to aSb + \varepsilon$, on a $\hat{\mathcal{L}}_{G_{a^n b^n}} = \{a^n S b^n \mid n \in \mathbb{N}\} \cup \{a^n b^n \mid n \in \mathbb{N}\}$.
Pour les langages de Dyck, on peut les reconnaître par :

$$\begin{aligned} S \to & \quad \varepsilon \\ S \to & \quad TS \\ T \to & ({}_1 S)_1 \mid ({}_2 S)_2 \mid \dots \mid ({}_n S)_n \end{aligned}$$

Remarque 5.1.0.1. *Le langages reconnu par T est le langage de Dyck primitif.*

5.2 Langages Algébriques et Clôture

Proposition 5.2.1. *On appelle algébrique un langage reconnu par une grammaire algébrique.*

Langages Réguliers	*Langages Algébriques*	
Clos par Union	*Clos par Union*	*On prend l'union des règles de grammaires :* $S \to S_1\ S_2$*, en faisant attention à disjoindre les symboles non-terminaux*
Clos par Concaténation	*Clos par Concaténation*	*On prend la concaténation des règles de grammaire :* $S \to S_1 \cdot S_2$*, en faisant attention à disjoindre les symboles non-terminaux*
Clos par Intersection	*NON Clos par Intersection*	
Clos par Complément	*...*	
Clos par Etoile de Kleene	*Clos par Etoile de Kleene*	$S \to SS_1 \mid \varepsilon$

Théorème 5.2.1 (Intersection Algébrique-Régulier)**.** *Si* L_1 *est algébrique et* L_2 *est régulier, alors* $L_1 \cap L_2$ *est algébrique.*

Définition 5.2.1 (Forme Normale de Chomsky)**.** *Une grammaire* $G = (\Sigma, V, S, R)$ *est sous forme normale de chomsky si toutes ses règles de grammaire sont de la forme :*

- $X \to a$
- $X \to X_1 X_2$
- $S \to \varepsilon$

avec $X \in V$ *et* $X_1, X_2 \in V \setminus \{S\}$*.*

Théorème 5.2.2. *Pour tout langage algébrique L, il existe G sous forme normale de Chomsky telle que $L_G(S) = L$.*

Définition 5.2.2. *On dit que x est accessible depuis S s'il existe $\alpha, \beta \in (\Sigma \cup V)^\star$ tels que $S \xrightarrow{\star} \alpha X \beta$.*
On dit que x est productif si il existe $w \in \Sigma^\star$ tel que $X \xrightarrow{\star} w$. On dit que x est utile s'il est accessible et productif.

Lemme 5.2.3. *Pour toute grammaire G, il existe $G^{'}$ telle que $L(G) = L(G^{'})$ et $G^{'}$ ne contient que des symboles accessibles.*

Démonstration. Soit $G^{'}$ obtenue en retirant tous les symboles non accessibles, i.e. on retire n'importe quelle règle qui contient un de ces symboles. Soit une dérivation sur G à partir de S. Par définition de l'accessibilité, c'est une dérivation sur $G^{'}$ à partir de S donc $L(G) \subseteq L(G^{'})$. Puisque toute production de $G^{'}$ est une production de G, on a bien le résultat. ■

Lemme 5.2.4. *Pour toute grammaire G, il existe $G^{'}$ telle que $L(G) = L(G^{'})$ ne contient que des symboles utiles.*

Démonstration. On marque les variables productives. Par récurrence, on trouve que X est productive si $X \to w$ où w est un mot sur Σ union l'ensemble des variables productives.
En prenant $V^{'}$ l'ensemble des variables productives de G, $R^{'} = R \cap V^{'} \times \left(\Sigma \cup V^{'}\right)^\star$. On a bien le résultat par les mêmes arguments que ci dessus. ■

Preuve du théorème sur la FNC 5.2.1. On utilise, dans cet ordre, 5.2.4 puis 5.2.3, pour se ramener à n'avoir que des états utiles.
Puis, on introduit de nouvelle règles :

- (TERM) : On introduit de nouveaux terminaux : Si $X \to aXb$, on écrit :
  — $X \to N_a X N_b$
  — $N_a \to a$
  — $N_b \to b$
- (INIT) : On introduit un nouvel axiome : $S^{'} \to S$
- (BIN) : On simplifie la règle : $X \to X_1\ X_2\ \ldots\ X_n$ pour $n \geq 3$. On introduit $X_2^{'}, \ldots, X_n^{'}$ de nouveaux non-terminaux et les règles :
  — $X \to X_1\ X_2^{'}$
  — $\forall i, X_i^{'} \to X_i\ X_{i+1}^{'}$
- (ε) : On simplifie $X_1 \to X\ X_2\ X\ X_3\ X$ en introduisant à la place :
  — $X_1 \to X\ X_2\ X\ X_3$
  — $X_1 \to X\ X_2\ X_3\ X$
  — Et ainsi de suite pour chaque règle où on peut choisir $X = \varepsilon$.
- (UNIT) : On va simplifier $X \to Y$, en remplaçant toute apparition de X dans une expression par Y.

En choisissant correctement l'ordre des règles, on a bien le résultat. ■

Preuve du théorème sur l'Intersection 5.2.1. On se donne une grammaire $G = (\Sigma, S, V, R)$ produisant L_1 et un automate $\mathcal{A} = (Q, \Sigma, I, F, \delta)$ reconnaissant L_2.
On passe notre grammaire G sous forme normale de Chomsky, i.e. les productions de G sont sous la forme :

- $S \to \varepsilon$
- $X \to X_1\ X_2$
- $X \to a$

On va construire une grammaire et des terminaux $(X_{p,q})_{\forall X \in V, p \in Q, q \in Q}$ tel que : $\mathcal{L}(X_{p,q}) = \mathcal{L}(X) \cap \mathcal{L}_2(p \to q)$. On introduit les règles :

- Si $X \to a \in R$, $X_{p,q} \to a$ si et seulement si $\delta(p, a) = q$.

- Si $X \to X_1 \ X_2 \in R$, $\forall p, q, q', X_{p,q} \to X_{p,q'} \ X_{q',q}$.
- On introduit un nouvel axiome S' par $S' \to S_{p,q}$ pour tous p, q.

On a bien défini une grammaire G'.

Soit $w \in L_1 \cap L_2$. Si $w = \varepsilon$, c'est bon. Sinon, on a alors un parmi :

- $S \to a$ et un état final F sur lequel un calcul sur w dans $\mathcal{A}$ termine, d'où $S' \to S_{I,F}$ dans G' par construction. D'où $S_{I,F} \to a$ car $\delta(I, a) = F$, car $a \in L_2$.
- $S \to X_1 \ X_2$, avec $X_1 \to w_1$ et $X_2 \to w_2$. $\mathcal{A}$ passe de i à q_1 en lisant w_1 puis que q_1 à F en lisant w_2 : $S \to X_{1_{i,q_1}} \ X_{2_{q_1,F}}$.

On obtient alors bien le résultat par induction. ■

Définition 5.2.3. *On appelle arbre de dérivation un arbre étiqueté par* $(\Sigma \cup V)$ *tel que :*

- *La racine est étiquetée par* S
- *Si un noeud étiqueté par* X *a* k *enfants étiquetés par* $a_1, \dots, a_k$ *éléments de* $(\Sigma \cup V)$*, alors* $X \to a_1 \ \cdots \ a_k \in R$*. Le mot associé est la concaténation des étiquettes des feuilles.*

Une grammaire est dite ambigue lorsqu'il existe un mot qui possède au moins deux arbres de dérivation syntaxique possible. Un langage est innéramment ambigu si toute grammaire qui le reconnaît est ambigu.

Cinquième partie

Cours 5 : 9/11

Sixième partie

Cours 6 : 16/11

6 Définitions

6.1 Décision

Définition 6.1.1 (Problèmes de Décision)**.** *Un problème de décision est un langage* $L \subseteq \Sigma^\star$ *sur l'alphabet* Σ*.*

Exemple 6.1.1. *On écrit un problème sous cette forme :*

- *Entrée : Un graphe non orienté* G
- *Sortie : Vrai ssi* G *est* 3*-coloriable*

Passer d'une forme à l'autre est un problème d'encodage.

6.2 Machine de Turing Déterministe à une Bande

Prenons $L = \left\{ w\#w \mid w \in \{0,1\}^\star \right\}$. Une machine de Turing contient une mémoire, qu'on représente par une bande, et une tête de lecture et d'écriture, qui peut accéder à une case, et la remplir ou d'une lettre, ou d'un blanc. Ici, par exemple, on peut lire le premier caractère à gauche, s'en souvenir, l'effacer, lire le premier caractère à droite, le comparer au caractère lu à gauche, s'arrêter si c'est différent, l'effacer et continuer sinon...

Définition 6.2.1 (Machine de Turing)**.** *Une machine de Turing est un uplet* $(Q, \Sigma, \Sigma \subseteq \Gamma, \delta, q_0, q_{accept}, q_{reject})$ *où :*

- Q *est un ensemble d'états*
- Σ *est un alphabet*
- Γ *est un alphabet (caractères sur la bande)*

- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{\leftarrow, \to\}$ *est une fonction de transition.*
- $\in \Gamma \setminus \Sigma$ *est un symbole blanc.*

Une configuration $w_1 q w_2$ où $w_1, w_2 \in \Gamma^\star$, $q \in Q$.

Définition 6.2.2. *La configuration initiale est $q_0 w$. Une configuration acceptante est $w_1 q_{accept} w_2$ et une configuration rejetante est $w_1 q_{reject} w_2$.*
Si on est dans une configuration $w_1 a q b w_2$, on utilise donc la transition $\delta(q, b) = q^{'}, b^{'} \leftarrow$, et on atterit dans la configuration $w_1 q^{'} a b^{'} w_2$. On note cela $w_1 a q b w_2 \vdash w_1 q^{'} q b^{'} w_2$.

Définition 6.2.3. $\mathcal{L}(T) = \{w \in \Sigma^* a \mid \exists C_{accept}, q_0 w \vdash^* C_{accept}\}$

Remarque 6.2.0.1. *Si à un moment, il n'y a plus de transition possible, cela revient à rejeter le résultat.*

Définition 6.2.4. *Soit M une machine de Turing, on dit que : M décide L si :*

- *Tout calcul est fini*
- $\mathcal{L}(M) = L$

M reconnaît L si $\mathcal{L}(M) = L$.
L est décidable (récursivement énumérable) s'il existe une TM qui décide (reconnaît) L.

6.3 Extensions

On peut avoir plusieurs bandes, mais cela n'apporte pas d'expressivité.

Définition 6.3.1 (Machines de Turing non-déterministes)**.** *C'est une machine de Turing pour laquelle $\delta : Q \times \Gamma \to 2^Q \times \Gamma \times \{\leftarrow, \to\}$*

Proposition 6.3.1. *Le non-déterminisme n'apporte pas d'expressivité.*

Démonstration. On fait un BFS sur l'arbre des calculs possibles, sur une TM avec 3 bandes. ∎

6.4 Machine de Turing Universelle

Définition 6.4.1 (Machine de Turing Universelle)**.** *Une TM universelle est une machine qui permet de simuler, étant donné un encodage, une machine de Turing.*

On peut par exemple encoder une TM $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ et un encodage $\langle w \rangle$ d'un input sur une bande par le code :

$$|Q|_{binaire} \, \# \, |\Sigma| \, \# \dots \# \, |\Gamma| \, \# a, q, a^{'}, q^{'}, \to, \#, \dots, \langle w \rangle$$

On se donne ensuite une seconde bande sur laquelle on va effectuer les calculs.

7 Limites - 1ère Partie

7.1 indécidabilité

En réalité, tous les langages ne sont pas décidables :

Proposition 7.1.1. *Le langage $A_{TM} = \{\langle M \rangle, \langle w \rangle \mid M$ accepte $w\}$ n'est pas décidable.*

Démonstration. S'il était décidable, soit D une TM qui décide A_{TM}. Soit $D^{'}(\langle M \rangle)$ telle que si $D(\langle M \rangle, \langle M \rangle)$ accepte, $D^{'}$ rejette et si $D(\langle M \rangle, \langle M \rangle)$ rejette, $D^{'}$ accepte. Alors, $D^{'}(\left\langle D^{'} \right\rangle)$ accepte si et seulement si il rejette. Contradiction. ∎

Proposition 7.1.2. *Le problème de l'arrêt $H_{TM} = \{\langle M \rangle \mid M$ s'arrête sur le mot vide$\}$ est indécidable.*

7.2 Propriétés de Langages

Définition 7.2.1. *Une propriété de langage est un ensemble $\mathcal{P}$ de codes de machine de Turing, tq $\forall M_1, M_2,\ \langle M_1\rangle \in \mathcal{P} \wedge \mathcal{L}(M_1) = \mathcal{L}(M_2) \Leftrightarrow \langle M_2\rangle \in \mathcal{P}$.*
Une propriété est non triviale si : $\exists M_1,\ \langle M_1\rangle \notin \mathcal{P}$ et $\exists M_2,\ \langle M_2\rangle$.

Théorème 7.2.1 (Rice)**.** *Toute propriété de langage non triviale est indécidable.*

Démonstration. Cf plus tard ■

Proposition 7.2.1 (Réduction de Turing)**.** *Supposons que A décide L. Pour décider $L^{'}$, on construit B utilisant A comme sous-outil. Alors :*

- *Si L est décidable, $L^{'}$ est décidable*
- *Si $L^{'}$ est indécidable, L est indécidable*

On a construit : $f : \Sigma_L \to \Sigma_{L'}$ telle que $w \in L \Leftrightarrow f(w) \in L^{'}$.

Septième partie
Cours 7 : 23/11

8 Problèmes Indécidables

8.1 Théorème de Rice

Théorème 8.1.1 (Rice)**.** *Toute propriété de langage non triviale est indécidable.*

Démonstration. Supposons que $M_\varnothing \notin P$ avec $L\left(M_\varnothing\right) = \varnothing$. Soit M_1 tel que $\langle M_1\rangle \in P$. On cherche $M_{M_1,M,x}$ de langage $L = \varnothing$ si M ne reconnait pas X, $L(M_1)$ sinon. On la définit ainsi, en notant Y l'entrée :

- On simule M sur X.
- Si M rejette X, on rejette Y.
- Sinon, on simule M_1 sur Y.

■

8.2 Problèmes des Correspondances de Postes

Définition 8.2.1. *On appelle une tuile un objet de la forme : $\begin{bmatrix} u_1 \\ v_1 \end{bmatrix}$ où $u_1, v_1 \in \Sigma^\star$*

Définition 8.2.2 (PCP)**.** *Etant données plusieurs tuiles, on cherche à les concaténer pour obtenir la même suite de lettres en haut de la tuile et en bas. On définit aussi le MPCP, un problème modifié où on fixe la première tuile.*

Définition 8.2.3 (Mot)**.** *Le problème du mot consiste à renvoyer vrai si et seulement si une machine accepte un mot.*

Proposition 8.2.1. *Le problème PCP est indécidable.*

Démonstration.

1. On réduit d'abord PCP à $MPCP$: On rajoute un symbole hors de l'alphabet considéré avant chacune des lettres des u_i, autour des lettres de v_1 et après les lettres de $v_i, i \geq 2$. On rajoute de plus une tuile de la forme $\begin{bmatrix} \$\$ \\ \cdot\$ \end{bmatrix}$.
2. On réduit maintenant $MPCP$ à PCP : On construit des tuiles ainsi
   - La première paire $u_1 = \$q_0w\$$ et $v_1 = \$$
   - Une paire $u_a = a$ et $v_a = a$ pour tout $a \in \Gamma$ pour recopier cette lettre

- Une paire $u = \$$, $v = \$$ pour passer à la configuration suivante
- Une paire $u = \$$ et $v = \#\$$ pour passer à la configuration suivante en ajoutant un $\#$ implicite à la fin de la configuration
- Une paire $u_\tau = pa$ et $v_\tau = bq$ pur toute transition $\tau = p, a \to (q, b, \to)$ de M
- Une paire $u_\tau = cpa$ et $v_\tau = qcb$ pur toute transition $\tau = p, a \to (q, b, \leftarrow)$ de M et toute lettre $c \in \Gamma$.
- Une paire $u = aq_+$, $v = q_+$ et une paire $u = q_{accept}a$, $v = q_{accept}$ pour toute lettre $a \in \Gamma$ afin de réduire la configuration une fois l'état q_{accept} atteint.
- Une paire $u = q_{accept}\$$ et $v = \varepsilon$ pour terminer.

Sous forme de dominos :

$$\begin{bmatrix} \$q_0w\$ \\ \$ \end{bmatrix}, \begin{bmatrix} a \\ a \end{bmatrix}, \begin{bmatrix} b \\ b \end{bmatrix}, \dots, \begin{bmatrix} \# \\ \# \end{bmatrix}, \begin{bmatrix} \$ \\ \#\$ \end{bmatrix}$$

et :

$$\begin{bmatrix} pa \\ bq \end{bmatrix}, \dots, \begin{bmatrix} cpa \\ qcb \end{bmatrix}, \dots, \begin{bmatrix} aq_{accept} \\ q_{accept} \end{bmatrix}, \dots, \begin{bmatrix} q_{accept}a, q_{accept} \end{bmatrix}, \dots, \begin{bmatrix} q_{accept}\$ \\ \varepsilon \end{bmatrix}$$

■