Chapitre 4

Calculabilité, Décidabilité, Complexité

Hugo Salou MPI*

Dernière mise à jour le 31 mars 2023

Table des matières

Au chapitre 1, on s'est intéressé aux langages réguliers, et aux automates qui est un modèle de calcul relativement simple. Dans ce chapitre, on s'intéresse à un modèle plus puissant : un *ordinateur*, on va notamment re-définir la notion de *problème*. Le choix classique de définition d'un ordinateur n'est pas celui qui a été choisi au programme : un programme OCaml.

0 Remarques mathématiques

Définition : Étant donné une relation $\mathscr{R}$ sur $\mathscr{E} \times \mathscr{F}$, on dit que $\mathscr{R}$ est

- *totale à gauche* dès lors que $\forall e \in \mathscr{E},\ \exists f \in \mathscr{F},\ (e, f) \in \mathscr{R}$;
- *déterministe* dès lors que $\forall e \in \mathscr{E},\ \forall (f, f') \in \mathscr{F}^2$, si $(e, f) \in \mathscr{R}$ et $(e, f') \in \mathscr{R}$, alors $f = f'$.

Définition : On appelle *fonction totale* de $\mathscr{E}$ dans $\mathscr{F}$ une relation sur $\mathscr{E}\times\mathscr{F}$ déterministe et totale à gauche.

Définition : On appelle *fonction partielle* de $\mathscr{E}$ dans $\mathscr{F}$ une relation sur $\mathscr{E} \times \mathscr{F}$ déterministe. On note alors

$$\operatorname{def}(f) = \{x \in \mathscr{E} \mid \exists y \in \mathscr{F},\ (x, y) \in f\}.$$

Remarque :
Soit f une fonction partielle de $\mathscr{E}$ dans $\mathscr{F}$ tel que $\square \notin \mathscr{F}$, alors on peut completer f en une fonction totale de $\mathscr{E}$ dans $\mathscr{F} \cup \{\square\}$ de la manière suivante :

$$\begin{aligned} f : \mathscr{E} &\longrightarrow \mathscr{F} \cup \{\square\} \\ x &\longmapsto \begin{cases} f(x) & \text{si } x \in \operatorname{def}(f) \\ \square & \text{sinon.} \end{cases} \end{aligned}$$

1 Problèmes

Définition : Étant donnés un ensemble d'entrée $\mathscr{E}$, un ensemble de sortie $\mathscr{S}$, on appelle *problème* sur $\mathscr{E} \times \mathscr{S}$ une relation $\mathscr{R}$ [1]sur $\mathscr{E} \times \mathscr{S}$ totale à gauche.

Remarque :
De la même manière qu'en mathématiques, on n'écrit pas $\{(0, 1), (1, 2), (2, 3), \dots\}$ mais $x \mapsto x + 1$, en informatique, on préfère la notation sous forme de problème.

1. C'est l'ensemble des liens entrées/sorties

Remarque :
On précisera, en cas d'ambigüité la représentation choisie pour les entrées.

Remarque :
Lorsque ce n'est pas précisé, dans la suite du cours, les entiers sont représentés en base 2.

Définition : On appelle *problème de décision* un problème à valeurs dans $\mathbb{B}$ déterministe.

Remarque (Notation) :
Lorsque Q est un problème, on note $\mathscr{E}_Q$ son espace d'entrée, et $\mathscr{S}_Q$ son espace de sortie. De plus, si Q est un problème de décision, on note $Q^+ = \{e \in \mathscr{E}_Q \mid (e, \boldsymbol{V}) \in Q\}$ et $Q^- = \{e \in \mathscr{E}_Q \mid (e, \boldsymbol{F}) \in Q\}$. $\{Q^+, Q^-\}$ est une partition de $\mathscr{E}_Q$.

2 Décidabilité

La définition d'un algorithme comme une suite finie d'instruction élémentaire, nous montre que l'ensemble d'algorithmes est dénombrable.[2] Mais, l'ensemble de problèmes est indénombrable. En effet, pour $x \in [0,1[$, on définit le problème

$$\textsc{Bit}_x : \begin{cases} \textbf{Entrée} & : n \in \mathbb{N} \\ \textbf{Sortie} & : \text{le } n\text{-ième bit de } x. \end{cases}$$

Et, comme $[0,1[$ n'est pas dénombrable, l'ensemble des problèmes ne l'est pas non plus.

2.1 Modèles de calcul

Définition : On appellera *modèle de calcul* la donnée d'un ensemble de machines
- qu'il est possible d'exécuter sur des entrées ;
- qui peuvent ou pas retourner une réponse.

Dans ce chapitre, notre modèle de calcul sera l'ensemble des fonction OCaml ayant pour type `string` $\to$ `string` qu'il est possible d'exécuter, qui peuvent donner un réponse ou non (boucle infinie, erreur).

Remarque :
On se place dans un monde d'exécution idéal : *mémoire infinie*.

2. en bijection avec $\mathbb{N}$

Remarque (Notation) :
Dans la suite, lorsque $\mathscr{M}$ est une machine, et $w \in$ `string`, on notera

- $w \xrightarrow[\mathscr{M}]{} w'$ si l'exécution de $\mathscr{M}$ sur w conduit à $w' \in$ `string`.
- $w \xrightarrow[\mathscr{M}]{} \circlearrowleft$ si l'exécution de $\mathscr{M}$ sur w conduit à une erreur.

Dans la suite de ce chapitre, on fixe $\Sigma =$ `char` et donc $\Sigma^\star =$ `string`.

Remarque :
On pourra, dans la suite, généraliser la signature de nos machines à un type $\mathscr{E} \to \mathscr{F}$ dès lors qu'on exhibe une fonction de sérialisation $\varphi : \mathscr{E} \to \Sigma^\star$ inversible (sur son espace image) injective : avec $\varphi_{\mathscr{E}} : \mathscr{E} \to \Sigma^\star$ et $\varphi_{\mathscr{F}} : \mathscr{F} \to \Sigma^\star$, on a

```
let ma_super_fonction (e: 𝓔) : 𝓕 = ...

let ma_fonction (s: string) : string =
  φ_𝓕(ma_super_fonction(φ_𝓔^-1(s)))
```

Code 1 – Généralisation des machines ayant pour entrée un ensemble $\mathscr{E}$ et sortie $\mathscr{F}$

2.2 Décidabilité

Définition : Une fonction partielle $f : \mathscr{E} \to \mathscr{S}$ est dite *calculée* par une machine $\mathscr{M}$ dès lors que

$$\forall e \in \operatorname{def}(f), \quad e \xrightarrow[\mathscr{M}]{} f(e).$$

On dit alors d'une telle fonction qu'elle est *calculable*.

Remarque :
Cette définition ne spécifie aucunement le comportement de $\mathscr{M}$ sur une entrée $e \notin \operatorname{def}(f)$.

Définition : Étant donné qu'un problème de décision Q est un cas particulier de fonction totale $\mathscr{E}_Q \to \mathbb{B}$, on dit que Q est *décidé* par une machine $\mathscr{M}$ dès lors que

$$\forall e \in \mathscr{E}_Q, \qquad \left(e \in Q^+ \iff e \xrightarrow[\mathscr{M}]{} \boldsymbol{V} \quad \text{et} \quad e \in Q^- \iff e \xrightarrow[\mathscr{M}]{} \boldsymbol{F} \right).$$

On dit alors que ce problème Q est *décidable*.

2.3 Langages et problèmes de décision

Les notions de langages et problèmes de décision d'entrée $\Sigma^\star$ coïncident. En effet, à un langage $L \subseteq \Sigma^\star$, on associe le problème de décision

$$\textsc{Appartient}_L : \begin{cases} \textbf{Entrée} & : w \in \Sigma^\star \\ \textbf{Sortie} & : w \in L\ ?. \end{cases}$$

On a alors $(\textsc{Appartient}_L)^+ = L$. Réciproquement, à un problème de décision Q d'entrées $\Sigma^\star$, on associe le langage Q^+.

Définition : Un langage L est dit *décidable* lorsque le problème $\textsc{Appartient}_L$ est décidable.

Définition : Étant donné une machine $\mathscr{M}$ de type `string` $\to$ `bool`, on appelle *langage* de $\mathscr{M}$, que l'on note $\mathscr{L}(\mathscr{M})$, l'ensemble

$$\{w \in \Sigma^\star \mid w \underset{\mathscr{M}}{\longrightarrow} \boldsymbol{V}\}.$$

Remarque :
$\mathscr{L}(\mathscr{M})$ n'est pas le complémentaire de $\{w \in \Sigma^\star \mid w \underset{\mathscr{M}}{\longrightarrow} \boldsymbol{F}\}$. En effet, il peut exister $w \in \Sigma^\star$ tel que $w \underset{\mathscr{M}}{\longrightarrow} \circlearrowleft$.

Propriété : Un langage L est décidable si, et seulement si L est le langage d'une machine $\mathscr{M}$ telle que $\forall w \in \Sigma^\star$, $w \underset{\mathscr{M}}{\longrightarrow} \boldsymbol{V}$ ou $w \underset{\mathscr{M}}{\longrightarrow} \boldsymbol{F}$.

Propriété : Tout langage régulier est décidable.

■

Propriété (stabilité des langages décidables) **:** Un langage décidable est stable par

1. union ;
2. intersection ;
3. complémentaire.

■

Remarque :
$\varnothing$ est décidable (par `fun s -> false`) ; $\Sigma^\star$ est décidable (par `fun s -> true`).

2.4 Sérialisation

Définition : Étant donné un type OCaml `t`, on appelle *sérialisation calculable* de ce type `t`, la donnée d'une fonction `f` OCaml de type `t` $\to$ `string` qui soit telle que

- pour tout e : `t`, (`f` e) est bien parenthésée ;
- `f` est injective ;
- la réciproque de `f` (définie sur $\mathrm{Im}($`f`$)$) est définissable en OCaml.

2. *i.e.* $\forall w \in \Sigma^\star$, $w \in L_1 \iff \texttt{decide}_1(w) = \texttt{true}$

Propriété : Soit $\mathtt{t_a}$ et $\mathtt{t_b}$ deux types OCaml sérialisables, alors le type $\mathtt{t_a{\star}t_b}$ est sérialisable.

Remarque :
Une programme OCaml est trivialement sérialisable : c'est déjà une chaîne de caractères.

2.5 Machine universelle

Soit l'ensemble $\mathbb{O}$ des chaînes de caractères qui sont des sérialisations de programme OCaml valide.

Définition : Soit la fonction `interprète` : $\mathbb{O} \times \Sigma^\star \to \Sigma^\star$ définie par

$$\mathtt{interpr\grave{e}te}(\mathscr{M}, w) = \begin{cases} w' \text{ tel que } w \underset{\mathscr{M}}{\longrightarrow} w' \\ \text{non défini sinon.} \end{cases}$$

Théorème : La fonction `interprète` est calculable. On appelle *machine universelle* un programme OCaml la calculant.

De même, considérons le problème suivant

$$\begin{cases} \textbf{Entrée} & : M \in \mathbb{O}, w \in \Sigma^\star, n \in \mathbb{N} \\ \textbf{Sortie} & : M \text{ se termine-t-elle sur } w \text{ en moins de } n \text{ étapes élémentaires ?} \end{cases}$$

Ce problème est décidable.

2.6 Théorème de l'Arrêt

Dans cette sous-section, on considère le problème

$$\textrm{Arrêt} : \begin{cases} \textbf{Entrée} & : M \in \mathbb{O}, w \in \Sigma^\star \\ \textbf{Sortie} & : M \text{ s'arrête-t-elle sur } w.^{3} \end{cases}$$

Théorème : Le problème Arrêt est indécidable.

Corollaire : Il existe des problèmes indécidables.

3. *i.e.* $\exists ? w' \in \Sigma^\star,\ w \underset{M}{\longrightarrow} w'$

2.7 Réduction

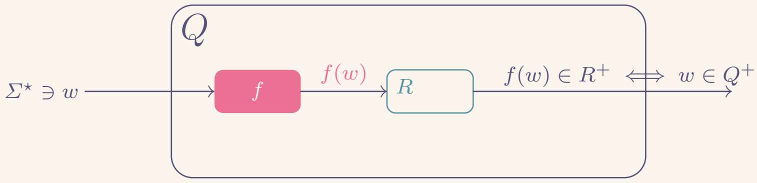

FIGURE 1 – Structure d'un sous-problème

Définition : Soit Q et R deux problèmes de décision. On dit que Q *se réduit au problème* R s'il existe $f : \mathscr{E}_Q \to \mathscr{E}_R$ totale et calculable, telle que

$$w \in Q^+ \iff f(w) \in R^+.$$

On note alors $Q \preccurlyeq R$.

Propriété : Si $Q \preccurlyeq R$, et que R est décidable, alors Q est décidable.

■

Corollaire : Si $Q \preccurlyeq R$, et Q non décidable, alors R non décidable.

Propriété : La relation $\preccurlyeq$ est un *pré-ordre* :
- $\preccurlyeq$ est réflective ;
- $\preccurlyeq$ est transitive.

■

3 Classe P et NP

Pour répondre à un problème, on peut le résoudre par des algorithmes plus ou moins rapides. Mais, l'objectif de cette section est de montrer que certains problèmes ne peuvent se résoudre que par des algorithmes lents, et que l'on ne peut pas faire mieux.

Définition : Le modèle de calcul impose une représentation des entrées par chaînes de caractères. Cela induit donc une notion de *taille d'entrée*, qui est la longueur de la chaîne de caractères.

3.1 Complexité d'une machine

Définition : Étant donné une machine $\mathcal{M}$ et une entrée $w \in \Sigma^\star$, on note $C^{\mathcal{M}}(w)$ le nombre d'opérations élémentaires effectuées lors de l'appel de $\mathcal{M}$ sur w. Lorsque $w \underset{\mathcal{M}}{\longrightarrow} \circlearrowleft$, on définit $C^{\mathcal{M}} = +\infty$.

Pour $n \in \mathbb{N}$, on définit alors

$$C_n^{\mathcal{M}} = \max\{C^{\mathcal{M}}(w) \mid w \in \Sigma^n\}.$$

Remarque :
On a, $\forall n \in \mathbb{N}$, $C_n^{\mathcal{M}} \in \bar{\mathbb{N}} = \mathbb{N} \cup \{+\infty\}$.

Définition : Soit $f : \mathbb{N} \to \mathbb{N}$ une fonction totale et calculable. On note $\textsc{Time}(f)$ l'ensemble des machines $\mathcal{M}$ telles que

- $\mathcal{M}$ s'arrête sur toute entrée ;
- $(C_n^{\mathcal{M}})_{n\in\mathbb{N}} = \mathcal{O}\big((f(n))_{n\in\mathbb{N}}\big)$.

3.2 Classe P

Définition : On dit d'une machine $\mathcal{M}$ qu'elle est de *complexité polynômiale* dès lors qu'il existe $k \in \mathbb{N}$ tel que $\mathcal{M} \in \textsc{Time}(n^k)$.

Définition : On dit d'une fonction (partielle ou non), qu'elle est *calculable en temps polynômial* dès lors qu'il existe une machine $\mathcal{M}$ de complexité polynômiale la calculant.

Propriété : La composée de deux fonctions totales calculables en temps polynômial est une fonction totale calculable en temps polynômial.

■

Remarque :
Dans la preuve précédente, l'ordre du polynôme change. En effet, la composée de deux programmes en $\mathcal{O}(n^2)$ est un $\mathcal{O}(n^4)$. L'espace des fonctions calculables en $\mathcal{O}(n^p)$, pour un p fixé, n'est pas stable par composition.

Définition (Classe P) : On dit qu'un problème est dans **P** dès lors qu'il est décidable en temps polynômial.

Propriété (Stabilité de la classe P) : La classe **P** est stable par
- union ;
- intersection ;
- complémentaire.

■

3.3 Classe NP

Définition (Classe **NP**) **:** On dit qu'un problème de décision Q est dans **NP** si et seulement si
- il existe un polynôme A ;
- un problème $\textsc{Verif} \in \mathbf{P}$;
- un ensemble $\mathscr{C}$ (certificats),

tels que
$$\forall w \in \Sigma^\star, \quad \left(w \in Q^+ \iff \exists u \in \mathscr{C},\ |u| \leqslant A(|w|) \text{ et } (w,u) \in \textsc{Verif}^+ \right).$$

La classe **NP** est la classe de problèmes tels qu'ils sont vérifiables en temps polynômial. Par exemple, si on a une formule logique, trouver un environnement propositionnel est coûteux en temps, MAIS vérifier la solution est très simple. Nous verrons ce résultat plus tard dans cette sous-section.

Le "**NP**" ne vient pas de **N**on **P**olynômial, mais vient de **N**on-déterministe **P**olynômial.

Propriété :
$$\mathbf{P} \subseteq \mathbf{NP}.$$

■

Propriété :
$$\textsc{Sat} \in \mathbf{NP}.$$

RAPPEL **:**
On rappelle la définition du problème SAT.
$$\textsc{Sat} : \begin{cases} \textbf{Entrée} & : \text{Une formule } G \\ \textbf{Sortie} & : \text{Existe-t-il } \rho \in \mathbb{B}^{\mathrm{vars}(G)} \text{ tel que } [\![G]\!]^\rho = \boldsymbol{V}\,? \end{cases}$$

■

3.4 NP-difficile

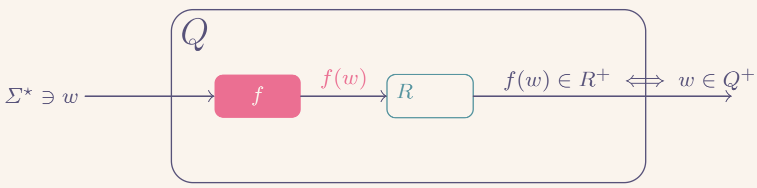

FIGURE 2 – Structure d'un sous-problème

Définition (Réduction polynômiale) **:** Soit Q et R deux problèmes de décision, on appelle *réduction polynômiale* de Q à R la donnée d'une fonction f totale, calculable en temps polynômial de $\mathscr{E}_Q$ dans $\mathscr{E}_R$ telle que
$$\forall w \in \mathscr{E}_Q, \quad w \in Q^+ \iff f(w) \in R^+.$$

On note alors $Q \preccurlyeq_{\mathrm{p}} R$.

Propriété : La relation $\preccurlyeq_{\mathrm{p}}$ est transitive et reflective : c'est un pré-ordre. ■

Propriété : Si $R \preccurlyeq_{\mathrm{p}} Q$, et $R \in \mathbf{P}$, alors $Q \in \mathbf{P}$. □

Définition (NP-difficile) : Un problème Q est **NP**-difficile si

$$\forall R \in \mathbf{NP},\ R \preccurlyeq_{\mathrm{p}} Q.$$

Propriété : Si Q est **NP**-difficile, et $Q \preccurlyeq_{\mathrm{p}} R$, alors R est **NP**-difficile. ■

On admet le théorème suivant.

Théorème (Cook-Levin) : Le problème Sat est **NP**-difficile. ■

Définition (n-fnc) : Soit $n \in \mathbb{N}$. Une formule G est sous forme n-fnc dès lors que G est sous forme fnc et chaque clause de G contient au plus n littéraux.

On parle aussi de forme cnf traduction anglaise de fnc. De même, on parle de forme n-cnf au lieu de n-fnc.

Définition : On définit le problème ci-dessous.

$$n\text{-cnf-Sat} : \begin{cases} \textbf{Entrée} & : G \text{ une } n\text{-cnf} \\ \textbf{Sortie} & : \text{Existe-t-il } \rho \text{ tel que } [\![G]\!]^\rho = \boldsymbol{V}\,? \end{cases}$$

Propriété : Soit 3sat = 3-cnf-Sat. Le problème 3sat est **NP**-difficile. ■

Remarque :
Pour tout $n \geqslant 3$, le problème n-cnf-sat est **NP**-difficile. Ceci peut être prouvé par réduction avec la fonction identité à 3sat.

Définition : On dit qu'un problème est **NP**-complet s'il est dans la classe **NP** et dans la classe **NP**-difficile :

$$\mathbf{NP}\text{-complet} = \mathbf{NP}\text{-difficile} \cap \mathbf{NP}.$$

REMARQUE :
Le problème SAT est **NP**-complet.