Fiche 1: Préliminaires et Machines de Turing

Objectifs de l'UE

1. Comprendre ce qu'est la **complexité intrinsèque d'un algorithme** au sens informatique, c'est-à-dire d'une *procédure de calcul* réalisable au moyen d'un **programme** qui s'arrête sur toute entrée, et cela *indépendamment* :
   — du langage de programmation utilisé pour "coder" (écrire des programmes),
   — du programme particulier qui *calcule* ou *réalise* cet algorithme (il y en a toujours une infinité),
   — et de l'ordinateur sur lequel on exécute ce programme.
   ⇒ Cela passe par une modélisation (abstraite) robuste de ce qu'est un algorithme et son exécution, et par la notion d'**ordre de grandeur** de son coût (en temps et éventuellement en place).
2. Comprendre ce qu'est la **complexité intrinsèque** d'un *problème* ou d'une *fonction* respectivement *soluble* ou *calculable* par l'informatique (c'est-à-dire à l'aide d'un algorithme).
   ⇒ Cela n'a de sens que pour des problèmes **décidables** ou des fonctions **calculables**, c'est-à-dire pour lesquels on sait qu'il existe au moins un algorithme pour les résoudre ou les calculer.

   L'objectif essentiel de cette UE est de fournir les concepts et les techniques pour classer les problèmes **décidables** en **classes de complexité** :
   — La classe P des problèmes pour lesquels il existe des algorithmes capables de les résoudre avec une complexité (temps) polynomiale.
   — La classe NP des problèmes pour lesquels on ne connait pas d'algorithmes de complexité polynomiale capables de les résoudre, et la sous-classe des problèmes *NP-complets* qui en constitue le noyau dur.
   ⇒ La clef pour cette classification de problèmes en classes de complexité est la notion de **réduction** d'un problème à un autre.

Préliminaires

Les problèmes que nous considérerons surtout sont des **problèmes de décision**. Un problème de décision B se définit sur un ensemble X d'entrées (les instances du problème) par une question booléenne "$B(x)$?" posable sur chaque instance $x \in X$. On dira que x est une *instance positive* de B si $B(x) = VRAI$, et que x est une *instance négative* de B si $B(x) = FAUX$).
Un problème de décision B est dit **décidable** s'il existe un algorithme qui, prenant en entrée une instance quelconque x de B, retourne comme résultat la valeur de $B(x)$.

Nous mentionnerons aussi des *problèmes de construction*, par exemple celui du voyageur de commerce (produire un chemin de coût minimal faisant passer exactement une fois par chaque nœud d'un graphe orienté pondéré).

Question : *Donner des exemples de problèmes de décision portant sur les entiers, puis sur les graphes.*

Les fonctions que nous considérerons dans un premier temps sont des fonctions (pouvant être partielles) de $\mathbb{N}^k$ vers $\mathbb{N}$, c'est-à-dire des fonctions qui prennent en entrée des k-uplets d'entiers $(n_1, n_2, \ldots, n_k)$ et qui renvoient comme résultat un entier (quand la fonction est définie sur l'entrée $(n_1, n_2, \ldots, n_k)$). Nous considérerons dans un second temps des fonctions sur des chaînes de caractères, par exemple la fonction miroir.

Exemples et notations :
— La fonction f de $\mathbb{N} \times \mathbb{N}$ dans $\mathbb{N}$ de la division *exacte* par 3 de la somme de deux arguments est partielle car elle n'est définie que pour les couples d'entiers (x, y) dont la somme est divisible par 3.
On utilisera les deux notations suivantes : $f(x,y) = \begin{cases} p & \texttt{si } x+y = 3p \\ \uparrow & \texttt{sinon} \end{cases}$
ou bien : $f = \lambda xy.\frac{x+y}{3}$.
— La fonction *Diff* de différence entière est une fonction totale de de $\mathbb{N} \times \mathbb{N}$ dans $\mathbb{N}$ qui étant donné un couple d'entiers (x, y) renvoie comme résultat $x - y$ si $x \geq y$ et 0 sinon.
$Diff(x,y) = \begin{cases} x-y & \texttt{si } x-y \geq 0 \\ 0 & \texttt{sinon} \end{cases}$
Autre notation : $Diff = \lambda xy[x \dot{-} y]$.

Question : *Donner des exemples de fonctions (au moins une fonction totale et une fonction partielle) de* $\mathbb{N}$ *dans* $\mathbb{N}$.

Machines de Turing

Le modèle des machines de Turing (MT) est le modèle abstrait de référence pour définir ce qu'est un algorithme et sa complexité. Il y a plusieurs variantes de ce modèle (à quadruplets, à quintuplets, à une bande, à plusieurs bandes, à demi-bandes, à deux piles...), toutes équivalentes en termes de calculabilité : elles permettent de calculer exactement les mêmes fonctions et de résoudre exactement les mêmes problèmes.

On s'intéresse d'abord au modèle le plus simple : les machines de Turing à quadruplets et à une bande.

1. Pour définir la classe des machines de Turing et permettre de les combiner, on définit deux ensembles infinis d'**états**, $Q = \{ q_0, q_1 \ldots, q_n \ldots \}$ et de **symboles**, $S = \{ S_0, S_1 \ldots, S_m \ldots \}$. Toutes les MT auront q_0 comme état initial, et S_0, noté B (le **blanc**) comme symbole par défaut. On note S_1 par | (la barre verticale).
2. **Définitions.** Une **MT simple** Z est définie par
   (1) un ensemble fini $E(Z) \subset Q$ d'**états**, tel que $q_0 \in E(Z)$,
   (2) un alphabet fini $A(Z) \subset S$ de **symboles**, tel que $S_0 = B \in A(Z)$,
   (3) une **bande** infinie des deux côtés, constituée de cellules ou "cases", chacune contenant exactement un symbole de $A(Z)$, et toutes sauf une partie finie contenant le blanc,
   (4) une **tête de lecture-écriture** pointant à chaque instant sur une case, dite "case courante", et enfin
   (5) un **programme** qui est un ensemble fini et consistant (ou encore déterministe) de quadruplets qui s'interprètent comme des instructions, ou des transitions dans le graphe fini dont les nœuds sont les états et les transitions les quadruplets.

   Les symboles servent à coder les entrées et les résultats de l'algorithme modélisé par l'ensemble des quadruplets. Un **quadruplet "simple"** est de forme $t = (q_i \quad S_j \quad [\ S_k \mid L \mid R\ ] \quad q_l)$. Deux quadruplets t et t' sont dits **consistants** s'ils ne commencent pas par le même couple : $(q_i, S_j) \neq (q'_i, S'_j)$. Les quadruplets (qui sont les "instructions" des MT) ne sont pas ordonnés, mais le déterminisme du modèle de calcul est garanti par l'exigence que l'ensemble des quadruplets soit consistant (ils sont consistants 2 à 2).
3. **Configuration d'une MT.** Une **configuration** de la MT Z est la donnée d'un état, d'une instance de sa bande (à B sauf sur une partie finie), et de la cellule (ou case) courante. Une configuration peut s'écrire par une **description instantanée** (d.i.) de forme $\gamma\ q_i\ S_j\ \delta \in A(Z)^*\ E(Z)\ A(Z)^+$: Z est dans l'état q_i, la cellule courante contient S_j, la partie de bande à sa gauche est $B^\infty\gamma$, et la partie de bande à sa droite est $\delta\ B^\infty$. Enfin, $(\forall m, n)$, $B^m\gamma\ q_i\ S_j\ \delta B^n$ est une d.i. de la même configuration.
4. **Pas de calcul.** Si la MT Z est dans l'état q_i et lit le symbole S_j, et si son programme contient un quadruplet de forme $(q_i\ S_j\ [\ X \in S_k|L|R\ ]\ q_l)$, elle passe dans l'état q_l et effectue l'une des 3 actions suivantes : écrire le symbole S_k (écrasant ainsi S_i), aller à droite d'une case (si $X = R$), ou aller à gauche d'une case (si $X = L$).
   Si Z ne contient pas de tel quadruplet, elle s'arrête. Il n'y a pas d'état final "en soi".
5. **Exemple :** la MT Z_1 avec 2 symboles ($A(Z_1) = \{ B, | \}$) et trois états ($E(Z_1) = \{q_0, q_1, q_2\}$).

$$Z_1 = \begin{bmatrix} q_0 & | & R & q_0 \\ q_0 & B & | & q_1 \\ q_1 & | & R & q_2 \\ q_2 & | & | & q_2 \end{bmatrix}$$

   Une configuration possible de Z_1 est $\alpha = BB|||q_1||B||B$.
   Questions :
   (a) *Simuler l'exécution de la MT* Z_1 *lorsqu'elle est appliquée sur le mot d'entrée* $|||$ *(qu'on peut noter* $|^3$*) inscrit sur la bande, dans l'état* q_0*, sa tête de lecture étant positionnée sur la case contenant le premier symbole* | *du mot d'entrée* ($c = q_0|||$).
   (b) *Pour un entier* $|^n$ *quelconque donné, simuler l'exécution de la MT* Z_1 *lorsqu'elle est appliquée sur le mot d'entrée* $|^n$ *inscrit sur la bande, dans l'état* q_0*, sa tête de lecture étant positionnée sur la case contenant le premier symbole* | *du mot d'entrée* ($c = q_0|^n B$).
   (c) *Donner un mot d'entrée (formé sur les symboles* | *et* B*) pour lequel* Z_1 *ne s'arrête pas.*

6. **Notation d'un pas de calcul et d'un calcul.** L'application d'une instruction (c'est-à-dire d'un quadruplet) qui fait passer d'une configuration de d.i. α à une configuration de d.i. β de la MT Z est notée : $\alpha \vdash_{\overline{Z}} \beta$.

   Par exemple, pour la MT Z_1 de l'exemple, $|^3q_0B \vdash_{\overline{Z_1}} |^3q_1|$ et aussi $|^3q_1| \vdash_{\overline{Z_1}} |^4q_2B$.
7. **Configuration terminale.** Une configuration α est **terminale** s'il n'existe pas de configuration β telle que $\alpha \vdash_{\overline{Z}} \beta$.
   Par exemple : $|^4q_2B$ est terminale pour la MT Z_1.
8. Un **calcul** d'une MT Z est une suite finie de configurations $\alpha_0 \dots \alpha_p$ telle que $(\forall i < p)\ [\alpha_i \vdash_{\overline{Z}} \alpha_{i+1}]$ et que α_p est une configuration terminale. On dit que α_p est le **résultat** du calcul de Z sur la configuration α_1, ce qui sera noté : $\boxed{\alpha_p = Res_Z(\alpha_0)}$.

   L'**interprétation d'un calcul** (sur des entiers, des graphes, ou d'autres structures) passe par le codage choisi pour représenter par des mots de S^* les entrées et les sorties du calcul.
9. **Complexité d'une MT Z :** la complexité (en temps) d'une MT Z est la fonction $\lambda n C(n)$ qui détermine (en ordre de grandeur) le nombre maximum de pas de calcul effectués par Z pour une entrée (un mot d'entrée) de taille n sur la bande dans la configuration initiale.
10. **Fonctions $\Psi_Z^{(n)}$ calculées par une MT Z :** on considère des MT Z dont l'ensemble de symboles contient B et | qui sont utilisés pour coder les entiers en unaire.

    Le codage unaire d'un entier x est le mot $\overline{x} = \overbrace{||\cdots|}^{x+1 \text{ fois}} B$,
    et le codage unaire d'un n-uplets d'entiers $(x_1 \dots x_n)$ est le mot $\overline{x_1} \dots \overline{x_n} = |^{x_1+1}B|^{x_2+1}B \dots |^{x_n+1}B$.
    Pour chaque arité n, la fonction (pouvant être partielle) $\Psi_Z^{(n)}$ est définie de la façon suivante :
    — si la MT Z s'arrête à partir de la configuration initiale $q_0\overline{x_1} \dots \overline{x_n}$, alors $\Psi_Z^{(n)}(x_1, \dots, x_n) = \langle Res_Z(q_0\overline{x_1} \dots \overline{x_n})\rangle$, où $\langle\alpha\rangle$ dénote le nombre de | sur la bande dans la configuration α.
    — sinon,$\Psi_Z^{(n)}(x_1, \dots, x_n)$ n'est pas défini.

    Question : *Quelles sont les fonctions définies par la MT Z_1 de l'exemple précédent ?*

11. **Fonctions Turing-calculables :** Une fonction f (pouvant être partielle) de $\mathbb{N}^k$ dans $\mathbb{N}$ est Turing-calculable s'il existe une MT Z telle que $f = \Psi_Z^{(k)}$

    Question : *Donner un exemple de fonction de $\mathbb{N}$ dans $\mathbb{N}$ qui est Turing-calculable.*

Variantes du modèle des Machines de Turing simples

Plusieurs variantes des MT simples existent, que l'on peut obtenir en modifiant le langage d'instructions pour permettre le déplacement de la tête de lecture et l'écriture d'un symbole en une seule instruction, ou en permettant que plusieurs bandes puissent être lues et écrites en une seule instruction, ou bien au contraire en restreignant la bande (potentiellement infinie à droite et à gauche) à une demi-bande.
Les notions de **configuration**, **calcul** et **complexité** s'adaptent de façon naturelle à chacune de ces variantes. En particulier, la complexité est une notion relative à chaque variante puisqu'elle correspond au nombre (maximum) d'instructions (du modèle considéré) permettant de passer d'une configuration initiale à une configuration terminale (dans le modèle considéré).
On peut montrer que toutes ces variantes sont équivalentes du point de vue de la calculabilité : elles ne modifient pas ce qui est calculable et ce qui ne l'est pas. Cela montre la robustesse du modèle des Machines de Turing en termes de calculabilité. Par contre, il peut y avoir un impact sur la complexité qu'il est important d'étudier pour chacune des variantes.

1. **Machines de Turing à quintuplets :** on permet d'écrire et de déplacer la tête de lecture-écriture en un seul pas de calcul, à l'aide d'instructions qui sont des quintuplets de la forme $(q_i \quad S_j \quad [\,L \mid R \mid N\,] \quad S_k \quad q_l)$.
   On peut simuler facilement chaque instruction (c'est-à-dire chaque quintuplet) d'une MT à quintuplets Z par au plus deux instructions (c'est-à-dire deux quadruplets) d'une MT à quadruplets Z' (définie sur le même alphabet de symboles $A(Z)$).
   Il en résulte le théorème suivant et son corollaire.
   Théorème. *Tout calcul d'une MT à quintuplets de k pas de calcul peut être simulé par un calcul d'au plus $2 \times k$ pas de calcul d'une MT à quadruplets.*

Corollaire. *L*es deux modèles de calcul que sont les MT à quadruplets et les MT à quintuplets sont équivalents en termes de calculabilité et en ordre de grandeur de complexité.
Attention : l'équivalence en ordre de grandeur de complexité n'est pas garantie pour toutes les variantes.

2. **Machines de Turing multi-bandes :** en une seule instruction, on peut gérer k bandes avec une tête de lecture-écriture par bande, à l'aide d'instructions de la forme $q \begin{pmatrix} s^1 \\ \vdots \\ s^k \end{pmatrix} \begin{pmatrix} a^1 \\ \vdots \\ a^k \end{pmatrix} q'$
où, pour tout $i \in [1..k]$, $a^i = L$, R ou s'^i.

Questions :
— *Montrer qu'on peut simuler une MT à 2 bandes par une MT simple.*
[Indication : on représente le contenu des 2 bandes ainsi que la position de leur tête de lecture-écriture à l'aide de cases d'une seule bande où chaque case contient un vecteur $\begin{pmatrix} v^1 \\ v^2 \\ v^3 \\ v^4 \end{pmatrix}$ de 4 symboles tels que (v^1, v^2) (respectivement (v^3, v^4)) représente le contenu d'une case de la première (respectivement seconde) bande et le fait qu'elle est ou non pointée par la tête de lecture-écriture de cette bande (on n'a besoin que d'un seul symbole supplémentaire pour cela).]
— Essayer d'évaluer le coût de cette simulation.
— Essayer de généraliser la simulation et sa complexité au cas des MT avec k bandes (où k est un entier quelconque).

3. **Machines de Turing à demi-bande.** On considère des MT avec une seule bande et la restriction supplémentaire qu'il n'y a pas de case à gauche de la case pointée par la tête de lecture-écriture dans la configuration initiale.

Question. Définir ce modèle et montrer qu'il est aussi puissant que celui des MT simples (à bande doublement infinie).

Pour en savoir plus

— "Introduction to Automata Theory, Languages and Computation". J. Hopcroft, R. Motwani, J. Ullman, 2nd edition, Addison-Wesley, 2001.
— "Introduction à la calculabilité". Pierre Wolper, 2ieme édition, Dunod, 2001.
— site Interstices (https ://interstices.info/jcms/jalios_5127/accueil).