13 Machines de Turing non déterministes (MTN)

Définition : idem MT standard (déterministes, MTD) sauf

$$\forall q \in Q, X \in \Gamma, \delta(q, X) \in \mathfrak{P}(Q \times \Gamma \times \mathcal{M})$$

Une config. $\alpha q X \beta$ a un ensemble *fini*, éventuellement vide, de cardinal borné par $|Q| \times |\Gamma| \times 3$, de configurations suivantes

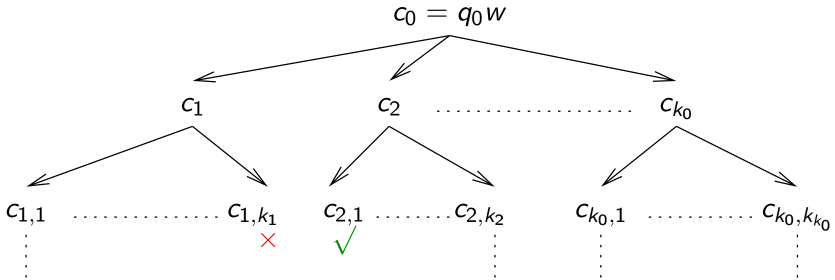
$\rightsquigarrow$ Nombre *éventuellement infini* d'exécutions possibles

$w \in \Sigma^*$ accepté par $m \Leftrightarrow$

$\exists\ q_0 w \vdash^* \alpha p X \beta \not\vdash$ et $p \in F$

$\exists\ q_0 w \vdash^* \alpha p X \beta$ et $p \in F$ (si $\forall (p, X) \in F \times \Gamma, \delta(p, X) = \emptyset$)

Arbre des exécutions : (nœuds : *configurations* atteignables)



Langage accepté par m :

$$L(m) = \{w \in \Sigma^* \ : \ w \text{ accepté par } m\}$$

L est *décidé* par m si $L = L(m)$ et, $\forall x \notin L$...

toutes les exécutions sont finies (et s'arrêtent donc avec $q \notin F$)

Exemples de MTN :

$m_1 = (Q_1, \Gamma_1, \Sigma, B, q_0^1, F_1, \delta_1)$, $m_2 = (Q_2, \Gamma_2, \Sigma, B, q_0^2, F_2, \delta_2)$ $\quad Q_1 \cap Q_2 = \emptyset$

$m = (Q_1 \cup Q_2 \cup \{q_0\}, \Gamma_1 \cup \Gamma_2, \Sigma, B, q_0, F_1 \cup F_2, \delta)$ $\quad q_0 \notin Q_1 \cup Q_2$

$\delta = \delta_1 \oplus \delta_2 \oplus [\delta(q_0, X) = \{(q_0^1, X, S), (q_0^2, X, S) : X \in \Sigma \cup \{B\}\}]$

alors $L(m) = L(m_1) \cup L(m_2)$

$\delta(q_0, 0) = \{(q_1, 0, D)\}$ $\quad \delta(q_1, 0) = \{(q_1, 0, D)\}$ $\quad \delta(q_1, 1) = \{(q_2, 1, D)\}$

$\delta(q_2, 0) = \{(q_3, 0, D)\}$ $\quad \delta(q_3, 0) = \{(q_3, 0, D)\}$ $\quad \delta(q_3, 1) = \{(q_4, 1, D)\}$

$\delta(q_4, 0) = \{(q_5, 0, D)\}$ $\quad \delta(q_5, 0) = \{(q_5, 0, D)\}$ $\quad \delta(q_5, 1) = \{(q_6, 1, D)\}$

$\delta(q_6, 0) = \{(q_7, 0, D)\}$ $\quad \delta(q_7, 0) = \{(q_7, 0, D)\}$ $\quad \delta(q_7, 1) = \{(q_8, 1, D)\}$

$\delta(q_8, 0) = \{(q_9, 0, D)\}$ $\quad \delta(q_9, 0) = \{(q_9, 0, D)\}$ $\quad \delta(q_9, 1) = \{(q_0, 1, D)\}$

$\delta(q_0, B) = \{(p_1, B, D)\}$ $\quad \to$ config : wBp_1, w code d'une MTD m'...

$\delta(p_1, B) = \{(p_1, 0, D), (p_1, 1, D), (p_2, B, G)\}$ $\quad$ «poser» 0 ou 1, ou passer en p_2

$\delta(p_2, X) = \{(p_2, X, G)\}$ $\quad X \in \{0, 1\}$ $\quad$ «reculer» sur les 0 et 1 posés

$\delta(p_2, B) = \{(p_3, 1, S)\}$ $\quad \to$ config : $wp_3 1 w'$, $w' \in \{0, 1\}^*$ quelconque...

$\delta(p_3, X) = \{(p_3, X, G)\}$ $\quad X \in \{0, 1\}$ $\quad$ «reculer» le long de w

$\delta(p_3, B) = \{(s, B, D)\}$ $\quad \to$ config : $sw1w'$

En s, faire comme MTU... Alors $L(m) = \{<m'> \mid L(m') \neq \emptyset\}$

Théorème : $\forall m_N$ MTN $\exists m_D$ MTD : $L(m_D) = L(m_N)$

Idée : «explorer» l'arbre des exécutions

surtout pas en profondeur (pb. si une branche est infinie !!!)

mais en largeur : $c_0 c_1 c_2 \ldots c_{k_0} c_{1,1} \ldots c_{1,k_1} c_{2,1} \ldots c_{2,k_2} \ldots c_{k_0,1} \ldots c_{k_0,k_{k_0}} \ldots$

$\rightarrow$ un ruban avec cette séq., construite peu à peu ; init : $q_0 w \sharp$

$\rightarrow$ une 2e *piste* pour «marquer» la conf. $c_i = \alpha q X \beta$ à traiter

1. Si $q \in F$, s'arrêter et accepter
2. $\forall t \in \delta(q, X)$ ajouter $\mathcal{S}(c_i, t)\sharp$ au bout du ruban
   où $\mathcal{S}(c_i, t)$ est la configuration suivante de c_i selon t...
3. Avancer la marque, retourner en 1. ou refuser si plus de conf.

m_D accepte $\Leftrightarrow$ elle traite un c_i avec $q \in F$ (1.) $\Rightarrow$ m_N accepte

m_N accepte $\Leftrightarrow$ une de ses exécutions accepte (disons en p pas)

Si k est le max des cardinaux des $\delta(q, X)$ (dans m_N) alors

m_D traite au plus $1 + k + k^2 \ldots + k^p = \theta(k^p)$ confs. avant d'accepter

$\Rightarrow$ m_D peut prendre un temps (un nombre de pas)

exponentiellement plus grand que m_N pour accepter

Question ouverte : peut-on faire mieux ?

Note : toutes les simulations antérieures (r rubans $\rightarrow$ 1 ruban...)

ne modifiaient le *coût* que *polynomialement* ($p \rightarrow p^c$)

14 *Éléments* de complexité théorique

On ne s'intéresse *ici* qu'aux *problèmes de décision*... *décidables*

On ne s'intéresse *ici* qu'à la *complexité temporelle*

On cherche à distinguer ce qui est *raisonnable* de ce qui ne l'est pas

Frontière : temps *polynomial* / *non polynomial* (quoique : n^{1000}...)

Les MT nous facilitent la formalisation :

- n : (*«taille des données»*) = lg de l'encodage de l'instance
- $T_m(n) = \mathrm{Max}$(nb de pas d'exéc. de la MT m sur $\{w : |w| = n\}$)

Arguments en faveur de la frontière :

Modèles de calculs (MTD, ordis actuels...) : *polynomialement* $\equiv$

1. $T(n) = \theta(n^c)$, quand les ordis iront $2\times$ plus vite, on pourra traiter dans le même temps des données de taille $\sqrt[c]{2} \cdot n$
2. $T(n) = \theta(c^n)$, quand les ordis iront $c\times$ plus vite, on pourra traiter dans le même temps des données de taille $n + 1$...

Exo : 1. et 2. : quand pourra-t-on traiter des données de taille $2n$?

Note : polynômes fermés par $+$, $\times$, composition...

Note : Tout ce qui est non polynomial (pas $\mathcal{O}(n^c)$) est qualifié d'«exponentiel», mais il existe des intermédiaires (ex : $n^{\log_2 n}$)

14.1 Les classes $\mathcal{P}$ et $\mathcal{NP}$

Un problème L est dans la classe $\mathcal{P}$ («deterministic $\mathcal{P}$olynomial»)
s'*il existe* une MTD m qui décide L et un polynôme π tels que
$T_m(n) = \mathcal{O}(\pi(n))$

En français : il existe une MTD qui *décide* L en temps polynomial

L est dans la classe $\mathcal{NP}$ («$\mathcal{N}$on-deterministic $\mathcal{P}$olynomial»)
s'*il existe* une MTN m qui décide L et un polynôme π tels que
$T_m(n) = \mathcal{O}(\pi(n))$
($\Leftrightarrow \forall w \in L$, m accepte w en temps $\leq \pi(n)$)

En français : il existe une MTN qui *accepte* L en temps polynomial

Question fondamentale : $\mathcal{P} = \mathcal{NP}$? (note : $\mathcal{P} \subseteq \mathcal{NP}$)
... on espère que non... sinon on pourrait e.g. «cracker» RSA :
de n GRAND entier (clé publique), sachant $\exists p, q : n = p \cdot q$
(p, q premiers), trouver p et q (clé privée)
MTN : «deviner» p et q (cf. 13., ex 2), tester leur produit...

Toute la suite est basée sur la notion de «*réduction polynomiale*»
$L_1 \preceq L_2 \Leftrightarrow$ on peut réduire L_1 à L_2 *en temps polynomial*

Alors $L_2 \in \mathcal{P} \Rightarrow L_1 \in \mathcal{P}$ ... $L_2 \in \mathcal{NP} \Rightarrow L_1 \in \mathcal{NP}$

14.2 Problèmes $\mathcal{NP}$-complets (classe $\mathcal{NPC}$)

... les plus «difficiles» de $\mathcal{NP}$

Déf : L est $\mathcal{NP}$-complet $\Leftrightarrow L \in \mathcal{NP}$ et $\forall L' \in \mathcal{NP} : L' \preceq L$

Intérêt : si on trouve un $L \in \mathcal{NPC}$ tel que $L \in \mathcal{P}$ alors $\mathcal{P} = \mathcal{NP}$

Prouver $L \in \mathcal{NPC}$: deux solutions après avoir montré 1. $L \in \mathcal{NP}$:

2. $\forall L' \in \mathcal{NP} : L' \preceq L$ (déf directement)

2'. $\exists L' \in \mathcal{NPC} : L' \preceq L$ (transitivité de $\preceq$)

Tant qu'on n'a pas trouvé un $L \in \mathcal{NPC}$ (par 2.)

on ne peut pas utiliser 2'.

Histoire : le premier problème $\mathcal{NP}$-complet (SAT)

Exps booléennes : variables $x_1, x_2 ...$, opérateurs $\wedge, \vee, \neg$, parenthèses

Exemple : $E = (x_1 \vee \neg x_2) \wedge \neg(x_3 \vee (x_1 \wedge x_2))$

Problème : $\exists ? f : \{x_i\} \to \{T, F\} \mid E[f(x_i)/x_i] = T$

En français : «E est-elle *satisfaisable* ?» *Une réponse sur l'exemple ?*

Théorème de Cook (1961) : SAT est $\mathcal{NP}$-complet...

Preuve : 1. SAT $\in \mathcal{NP}$: «deviner» un f, vérifier...

2. transformer *en temps polynomial* une MTN m décidant w *en temps polynomial* ... en E : m accepte $w \Leftrightarrow E$ est satisfaisable

... Preuve laissée en exercice...

Le plus dur est fait : une fois qu'on a un premier problème (SAT) $\mathcal{NP}$-complet on peut montrer que L est $\mathcal{NP}$-complet

en réduisant polynomialement SAT (ou tout autre $L' \in \mathcal{NPC}$) à L...

On connaît un nombre considérable de problèmes $\mathcal{NP}$-complets

et pour aucun on n'a trouvé d'algo déterministe polynomial...

$\Rightarrow$ FORTE présomption que $\mathcal{P} \neq \mathcal{NP}$

Compléments

$\star$ co-$\mathcal{NP} = \{L : \overline{L} \in \mathcal{NP}\}$

Ex : USAT : «E est-elle insatisfaisable ?»

$\simeq$ TAUT : «E est-elle une tautologie ?»(penser à $\neg E$)

$\mathcal{P} \subseteq$ co-$\mathcal{NP}$ (car $\mathcal{P}$ fermé par complémentation... exo...)

Conjecture : $\forall L \in \mathcal{NPC} : \overline{L} \notin \mathcal{NP} (\Rightarrow \forall L \in \mathcal{NPC} : L \notin$ co-$\mathcal{NP})$

Théorème : $\mathcal{NP} =$ co-$\mathcal{NP} \Leftrightarrow \exists L \in \mathcal{NPC} : \overline{L} \in \mathcal{NP}$

$\star$ $\mathcal{PS}$ / $\mathcal{NPS}$: idem $\mathcal{P}$ / $\mathcal{NP}$ mais en termes d'*espace* polynomial

On peut facilement montrer : $\mathcal{P} \subseteq \mathcal{NP} \subseteq \underline{\mathcal{PS} = \mathcal{NPS}}$

et aussi co-$\mathcal{NP} \subseteq \mathcal{PS}$... *Note* : $\mathcal{PS}$ souvent noté PSPACE

On peut s'intéresser à $\mathcal{PSC}$... ... ...