Thèse Church

- Notion informelle d’algorithme
- **Thèse de Church:** Tout algorithme peut se traduire en un programme de machine de Turing.

- Remarque: il n’est pas possible de prouver formellement ce résultat, car il fait appel à la notion informelle de ce qu’on appelle un algorithme.
- Par contre, on peut montrer que pour chaque notion d’algorithme que l’on se fixe formellement (exemple: algorithme = programme C, programme Java, ou Programme de machine RAM), tout algorithme se transforme un programme de machine de Turing.

Thèse Church: version effective

Thèse de l'invariance: La traduction en programme de machine de Turing introduit au plus un ralentissement quadratique, et un facteur constant sur l'espace.

- Exemple: on va le prouver pour les machines RAM.

Machine RAM

- **Les machines RAM** sont des systèmes de calcul beaucoup plus proches du fonctionnement réel des ordinateurs.

- **Définition: Machine RAM**. Une machine possède un certain nombre de registres $R_1, \ldots, R_n$. Chaque registre contient des mots sur un alphabet fini $\Sigma = \{a_0, \ldots, a_{m-1}\}$. La machine évolue selon un programme qui est une suite convenable d'instructions de l'un des types suivants:
  1. $X : add_j\ Y$: concaténer a_j à droite du mot contenu dans Y.
  2. $X : del\ Y$: effacer le premier caractère de Y.
  3. $X : clr\ Y$: effacer le contenu de Y.
  4. $X : Y \leftarrow Z$: substituer le contenu de Z à celui de Y.
  5. $X : jmp\ X'$: aller à la ligne X'.
  6. $X : Y\ jmp_j\ X'$: aller à la ligne X' si la première lettre de Y est a_j.
  7. $X : stop - accepte$: arrêter et accepter.
  8. $X : stop - rejette$: arrêter et rejeter.

- **Un langage $L \subset \Sigma^*$ est décidé par la machine** si pour tout $w \in \Sigma^*$, la machine partant avec le registre R_1 avec la valeur w et les autres registres blancs, finit ultimement par exécuter l'instruction $stop - accepte$ pour $w \in L$, et $stop - rejette$ pour $w \notin L$.

Équivalence RAM, Turing

- **Théorème:** Un langage $L \subset \Sigma^*$ est décidé par une machine RAM ssi il est décidé par machine de Turing.

Démonstration: passe par machines SRM = machine à un seul registre avec instructions

1. $X : add\ j$: ajouter a_j à droite du registre,
2. $X : del$: efface la lettre la plus à gauche si le registre n'est pas vide,
3. $X : jmp\ j, X'(a)$ (resp. $X : jmp\ j, X'(b)$) aller à la ligne X' en avant (resp. arrière) si la première lettre du registre est a_j.

- Étape 1: Suppose que a_0 est le symbole ″,″. On construit une machine $TOURNE$ qui transforme $x_1, \ldots, x_n$ en $x_2, \ldots, x_n, x_1$.

Définir le sous programme $JUMPX$ comme

1. $jmp\ 0, X$
2. $jmp\ 1, X$
3. **. . .**
4. $jmp\ k, X$

Définir le sous programme $ROTATE$ comme

1. $jmp\ 0, copy_0(b)$
2. **...**
3. $jmp\ k, copy_k(b)$
4. $copy_0\ add0$
5. $Jump\ suite(b)$
6. **...**
7. $copy_k : add\ k$
8. $Jump\ suite(b)$
9. $suite : del$**.**

Le programme de $TOURNE$ est donné par:

1. $add\ 0$
2. $loop : jmp\ 0, end(b)$
3. $Rotate$
4. $Jump\ loop(a)$
5. $end : del$.

- Étape 2: Montre que toute machine RAM est simulable par une machine SRM. Code état RAM $R_1, \ldots, R_n$ par le registre $*, R_1, \ldots, R_n$, où $*$ nouvelle lettre ajoutée à l'alphabet (ex: indice -1).

Définir le sous-programme $Init$ qui remet $*$ en début de mot défini par:

1. $loop : jmp\ -1, end(b)$
2. $TOURNE$
3. $Jump\ loop(a)$

Définir le sous-programme $Reg\ k$ qui met le registre k en début défini par:

$Init; TOURNE; \ldots; TOURNE;$**, où** $TOURNE$ **est répété** $k-1$ **fois:**

On remplace alors les instructions de la machine RAM comme suit:

1. $add\ j, R_p$ devient
   (a) $Reg\ p+1$
   (b) $add\ j$.
2. $dell\ R_p$ devient
   (a) $Reg\ p$
   (b) del.

3 $R_p\ jmp_j X$ devient

(a) $Reg\ p$

(b) $jmp\ j, doJump$

(c) $Jump\ noJump$

(d) $doJump :$

(e) $Init$

(f) $Jump\ X$

(g) $noJump : Init$.

Étape 3: On montre que calculable par SRM implique calculable par une machine de Turing.

Idée: simuler l'évolution machine SRM sur le registre R par une machine de Turing dont le ruban contient DRF où D et F sont des nouveaux symboles marquant le début et la fin.

On remplace chaque instruction de la machine SRM par un couple d'états de machine de Turing. Premier état sert à chercher début ou fin du mot, deuxième état cherchant à faire l'opération correspondante. Ajoute des états pour ajouter les marqueurs D, F en début et fin de mot. Connecte tout, et on a le résultat.

- **Théorème:** Si un langage $L \subset \Sigma^*$ est décidé en temps (resp. espace) $f(n)$ par une machine RAM, alors il est décidé par machine de Turing en temps $\leq K * f^2(n))$ (resp. en espace $\leq K * f(n)$) pour une certaine constante K.

Démonstration: les simulations précédentes introduisent un ralentissement au plus quadratique et augmente l'espace d'au plus un facteur constant.