

- Fonction d'Ackermann: fonction récursive non primitive (sans composition de fonctions)

$$\hookrightarrow A(m, n) = \begin{cases} n+1 & m=0 \\ A(m-1, 1) & m>0 \text{ et } n=0 \\ A(m-1, A(m, n-1)) & m>0 \text{ et } n>0 \end{cases}$$

utilisation pour tester l'implémentation d'un langage de programmation

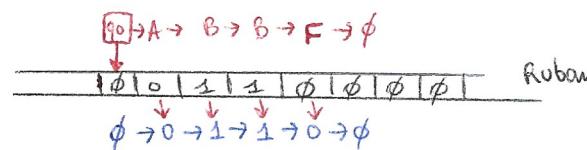
- Machine de Turing déterministe: formalisme de la notion d'algorithme.

Definit par quintuplet $(Q, \Gamma, q_0, \delta, F)$ où ; la fonction de transition est donnée par:

$$\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{\leftarrow, \rightarrow\}$$

ensemble fini d'état alphabet de travail déplacement de la tête de lecture

- multiplication par 2:



	0	1	ϕ
A	B,0 ↓	B,1 ↓	A, ϕ
B	B,0 ↓	B,1 ↓	F, ϕ

$$q_0 = A ; F = \text{état final}$$

→ On est passé de: 011 → 0110

- Universalité: simulation de n'importe quel machine de Turing sur un dénombrable

principe: Prog : $\{ M.Turing \} \rightarrow \{ mot sur Ruban \}$

$$T \mapsto \langle T \rangle$$

codage: $x \mapsto \langle x \rangle$, tel que $x \in (\langle T \rangle, \langle x \rangle) \in T.O.U.$

→ un problème est Turing-complet si son système peut être écrit par une machine de Turing

- Théorème de Rice: Toute propriété non triviale P sur les langages récursivement enumérables, le problème de savoir si le langage $L(M)$ d'une machine de Turing M vérifie P, est non décidable → généralisation du problème de l'arrêt:

- Input $\langle T \rangle$
- if A accepte $(\langle T \rangle, \langle T \rangle)$ then rejeter
- else accepter

- Lambda-calcul: système formel utilisé comme modèle de calcul, formalise la notion de fonction récursive. → simule toute machine de Turing

→ Construction des termes lambda et effectuer des opérations de réduction entre eux.

réductions • $(\lambda x.xz)(\lambda z.zx) \xrightarrow{\text{donne}} (\lambda y.y)(\lambda y.y)$

contraction • $(\lambda z.zy)a \xrightarrow{\text{donne}} (zy)[z := a] = ay$

- Hierarchie des Automates: (Logique combinatoire) \subset (Automate fini) \subset (Automate à pile) \subset (M.Turing)
 - emplir des données \uparrow (d'un ruban)
- Logique de Hoare: (Axione) L'affectation est l'instruction $x := E$, associant à la variable x , la valeur de l'expression E : $\{P[E/x]\}_{x:=E} = E\{P\}$
- Automate fini: modèle de calculs contenant des table de transitions et des symboles d'entrée
 - Def. quintuplet $A = (\Sigma, Q, R, I, F)$, où l'ensemble des transition R est défini par:
$$Q \times \Sigma \times I$$

ensemble d'état ↑ initiat

alphabet
- Langage rationnel: ensemble fini de lettres A (ou caractères), une chaîne est une suite finie de lettres
 - mot de langage R: $w = (a_1, a_2, \dots, a_k)$
 - operation:
 - concaténation: $X Y \Rightarrow \{ab, c\} \cdot \{ba, c\} = \{abba, abc, cba, cc\}$
 - union: $X \cup Y \Rightarrow "U" = \{ab, c, ba\}$
 - étoile de Kleene: $X^* \Rightarrow \{a, ab\}^* = \{\epsilon, a, aa, ab, aaa, \dots\}$
 - (Note). Théorème de Codd: algèbre relationnel \Leftrightarrow calcul relationnel \rightarrow base de données SQL
- Théorème de Kleene: Sur un alphabet fini A, il y a égalité entre langage rationnel et langage reconnaissable, tg: $\text{Rat } A^* = \text{Rec } A^*$

Complexité:

Etude de la quantité de ressources (temps, espace mémoire) d'un algorithme pour résoudre un problème. \rightarrow si décidable, échelle de temps raisonnable?

\rightarrow principe: Pour une boucle finie de taille n, la complexité O est $O(n)$, ainsi, une boucle dans une autre est: $O(n^2)$ si exécutée entièrement.

. Formellement: $C_T(n) = \max \left\{ m \mid \text{tpe de taille } n, D_T(x) = m \right\}$

. Mesure de la complexité: Si (u_n) et (v_n) sont deux suites numériques, on dit que (u_n) est d'ordre (v_n) et se note: $u_n = O(v_n)$

ssi: $\exists M \in \mathbb{R}^+, \exists n_0 \in \mathbb{N}, \forall n \geq n_0, |u_n| \leq M |v_n|$
pour complexité multiplicative $O(n^2)$

- Classe de complétude: Ensemble des problèmes qui peuvent être résolus par un modèle de calcul M, utilisant une quantité $O(f(n))$ de ressource de type R où "n" est la taille.
 - Inclusion: $NL \subseteq PSPACE \subseteq EXPSPACE \quad \left\{ \begin{array}{l} \text{si } f: \mathbb{N} \rightarrow \mathbb{N} \text{ vérifie } f(n) \geq c \cdot n \text{ avec } c > 1 \\ \text{alors } O(f(n)^2) \end{array} \right.$
 - $P \subseteq EXPTIME$
 - $NP \subseteq NEXPTIME$
 - Théorème hiérarchie temps déterministe: Soit f, g suite d'entiers naturels tels que $f(n)$ est non nul et g est constructible en temps et $f \cdot \log(f) = O(g)$, alors $\text{DTIME}(f) \subseteq \text{DTIME}(g)$
 - f est calculable par M-Turing
 - ↑ comparaison asymptotique (analyse)
 - Complexité polynomiale: Un algorithme est dit polynomial si $\forall n$, pour des données $< n$ octets, l'algorithme s'exécute en moins de $C \times n^k$ opérations élémentaires.

↳ cas NP: $D_T(x) = \min \{ m \mid \text{A execution de } T \text{ sur } x \text{ fini} = m \text{ etapes} \} \rightarrow$ meilleur cas
 $C_T(n) = \max \{ m \mid \exists \text{ mot}_x \text{ de taille } n \mid D_T(x) = m \}$ Temps de calcul

↳ Cryptographie: $e_K : \{0,1\}^n \rightarrow \{0,1\}^m$ } si def \rightarrow temps polynomial
 fct sens unique $\xrightarrow{\quad}$ (clair) (chiffré)

- Théorème de Cook: Le problème de satisfabilité est NP-complet $\xrightarrow{\text{(SAT)}}$ vérifie toutes les NPs.

Principe "clause": $a_1 \vee a_2 \vee \dots \vee a_p = (\underbrace{a_1 \vee a_2 \vee a_3}_{(a_1)} \wedge \dots \wedge \underbrace{a_{p-1} \vee a_p \vee a_{p-3}}_{(a_{p-3})})$

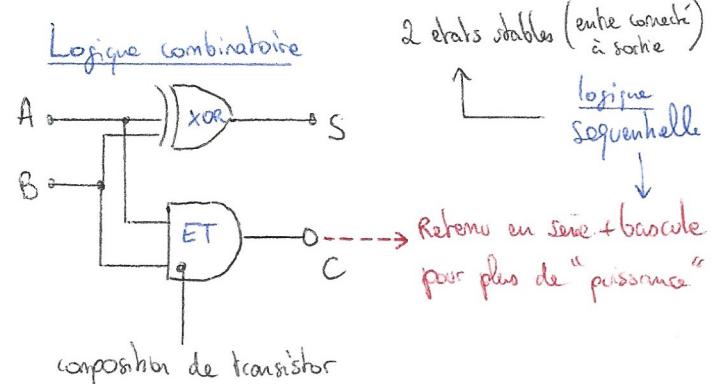
Passage problème différent (graph par ex.): $[uv \vee v \tau \tau]^{3\text{-SAT}}$ $[3\text{-SAT}]$

Nombre utilise Allume eteind ↓ ↓
• Les ordinateurs | le système numérique de base 2 (vrai/faux). Les chiffres de la numération
positionnel binaire | se nomme "bit". Un "bit" peut prendre 2 valeurs noté 0 et 1.

• Représentation nombre entiers: $x = \left(\begin{matrix} 1 \\ 30+5 \end{matrix} \right)_{B_{10}} = \left(\overbrace{}^1 + \overbrace{}^0 + \underbrace{1}_{1} \right)_{B_2}$ Passage d'une base à une autre par division euclidienne.

- Deni-additionneur:
(Logique combinatoire)

A	B	$A + B$	Retorno
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



Nombres (suite)INFORMATIQUE

- Nombre réel: Approximation par l'arithmétique à virgule flottante, représenté sous forme de triplet: (signe, mantisse, exposant) \rightarrow signe \times nombre $\times 10^k$, $k \in \mathbb{Z}$
- Décalage de bit pour calcul racine: $(a \ll n) = (101 \ll 2) = (a_m \dots a_0 \ 0_n \ 0_0) = (101\ 00)$
↳ si a
- Génération de nombre aléatoire: Algorithme générant une séquence de nombre pseudo-indépendant.
 ↳ méthode de congruence linéaire: $X_{n+1} = (a \cdot X_n + c) \bmod m$; $X_0 = \text{graine} \in [0, m-1]$
 ex ↳ choix du module "m":
 • choix du multiplicateur "a" et increment "c":
 → loi uniforme \rightarrow voir Box Muller pour passage à loi normal.
- Méthode Monte-Carlo: Approximation de la valeur numérique de l'intégrale d'une fonction quelconque par une suite d'événements aléatoires.

Théorème de transfert: $G = \mathbb{E}[g(x)] = \int g(x) f_x(x) dx$

Algorithme \longrightarrow variable aléatoire \uparrow jet dérivé uniforme

- Entropie de Shannon: Si X est une variable aléatoire discrète, l'entropie d'une source d'information (en bits) est défini par: $H(X) = - \sum_x P(X=x) \cdot \log(P(X=x))$

Opération sur les listes

Les listes sont des structures de données permettant de regrouper des données librement. Elles servent de base au donnée plus complexe comme les pile, files, arbre, etc.

- 2 types:
 - Tableau: (valeur, index)

45	100	58	25	32	25
0	1	2	3	4	5

 VALEUR INDEX
 - liste chaîne: (valeur, pointeur \rightarrow), (valeur, pointeur \rightarrow) \rightarrow on s'intéressera ici qu'au tableau.

- Tri: Algorithmes permettant d'organiser une collection d'objets selon une relation d'ordre.

- ↳ Tri rapide ($n \log n \rightarrow n^2$):
 - Pivot à la fin \rightarrow échange avec dernier élément du sous tableau.
 - Si élément < pivot \rightarrow début du sous tableau
 - pivot \rightarrow fin des éléments déplacés.

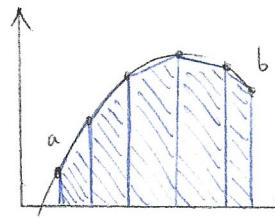
- ↳ principe du "diviser pour régner":
 - Diviser problème en sous-problèmes
 - Régner: résoudre sous-problème \rightarrow parallelisable
 - Combiner: calculer solution au problème initial.

- ↳ Algorithme glouton: résolution d'un problème constant à faire le choix localement optimal à chaque étapes → problème à 2 maximums dans la liste.
- ↳ Très utilisé dans les graphes: problème du plus court chemin (Djikstra) ↔ arbre courant
- Idee: noeud initial $Y_0 \rightarrow$ distance entre Y et $Y_0 \rightarrow$ Attribution distance + optimisation pas à pas
↳ nb lieu \Leftrightarrow Distance hop-dogue.

- Recherche de zéros: recherche d'un intervalle E dans lequel existe le zéro de la suite
↳ résoudre équation: $(u_n) = 0$. ↳ Théorème V.I. (voir cours)
- ↳ méthodes dichotomie (\Leftrightarrow diviser pour régner): $\underbrace{[a, c] \text{ ou } [c, b]}_{\text{changement de signe?}}$ avec $c = \frac{(a+b)}{2}$

- Intégration numérique: estimation de la surface défini sur le domaine particulier de la suite.

↳ méthodes du Trapèze:



$$I = h \left[\frac{f(a) + f(b)}{2} + \sum f(a + ih) \right]$$

Erreur: $E(f) = -\frac{(b-a)^3}{12} f''(n) \rightarrow O(h^2) \rightarrow \text{ordre 2} \rightarrow (\text{voir analyse})$

Note: La variation des éléments du tableau \Leftrightarrow dérivée

- Résolution équation différentiel: procédure pour résoudre des équations de fonctions/dérivées

[ex: $f = f'$] pour discréétisation de la dérivée.

• Ordre 1: (Euler)

$$\begin{array}{c} X_{n-1} \quad X_n \quad X_{n+1} \\ -1 \quad 0 \quad +1 \end{array}$$

$$X_{n+1} = X_n + h \cdot f(n \cdot h, X_n) \left. \begin{array}{l} \text{variation n+1} \\ \text{implicite} \end{array} \right\}$$

• ordre $\frac{3}{2}$: (point milieu)

$$\begin{array}{c} X_n \quad X_{n+1} \\ - \quad + \end{array}$$

$X_{n+\frac{1}{2}} \rightarrow$ estimation

$$K_2 = h \cdot f\left(Y_n + \frac{1}{2} K_1, t_n + \frac{1}{2} h\right)$$

$$Y_{n+1} = Y_n + K_2$$

- Théorème de Lax: stabilité nécessaire et suffisante pour convergence: $R = \frac{\Delta t}{\Delta x^2}$