Planification Automatique & Techniques d'Intelligence Artificielle

Les fondements de l'informatique et de l'Intelligence Artificielle

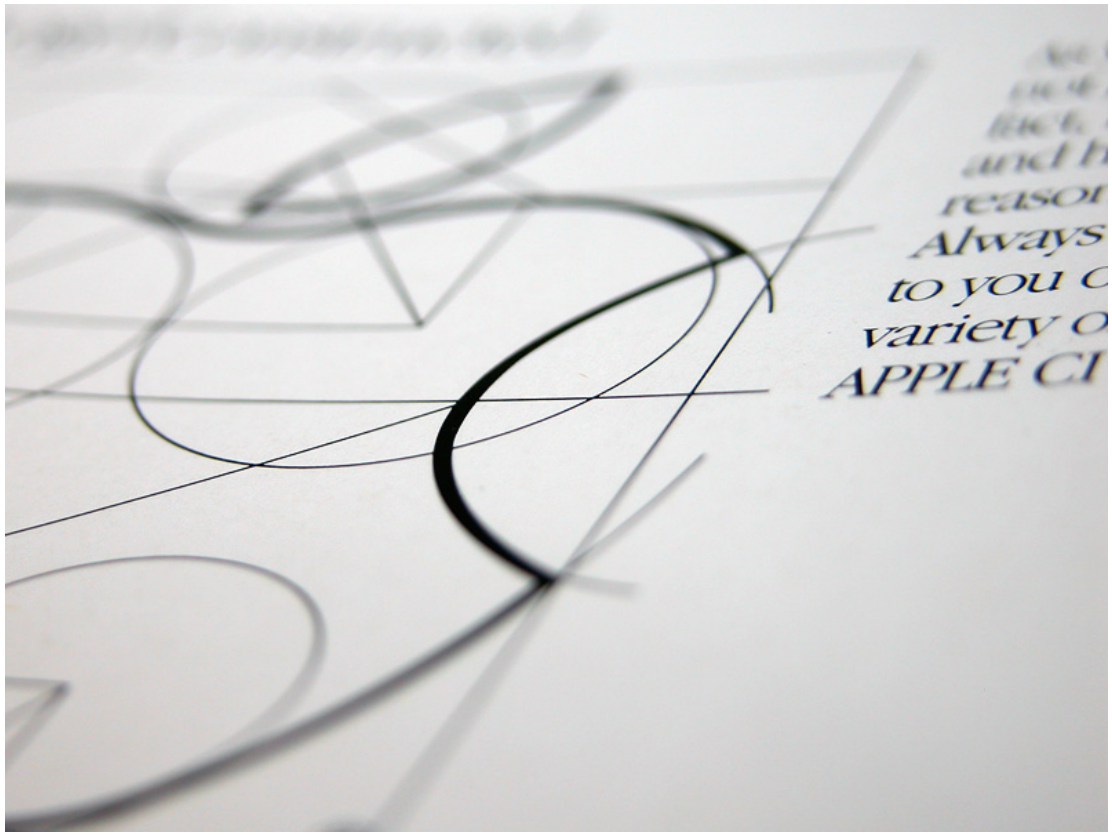
Humbert Fiorino

Humbert.Fiorino@imag.fr
http://membres-liglab.imag.fr/fiorino

Laboratoire d'Informatique de Grenoble (LIG) – Équipe MAGMA

Quelques mots...

À propos du logo d'Apple crée par Rob Janoff en 1977 : hommage à Alan Turing ?



Alan Turing (1912-1954)

- Cryptologie et machine "Enigma"
- Ses travaux sont considérés comme fondateurs pour l'informatique théorique ("*computer science*") et l'IA
- Contributions majeures :
  - Machines de Turing (Turing, A. M., 1936, "*On computable numbers, with an application to the Entscheidungsproblem*", Proc. London Maths. Soc., ser. 2, 42 : 230-265. Papier écrit à 24 ans)
  - Test de Turing (Turing, A. M., 1950b, "*Computing machinery and intelligence*", Mind 50 : 433-460. Cf. [Boden, 1990])

Des problèmes que les machines ne peuvent pas résoudre

- Voici un programme C qui s'arrête en écrivant "Hello, world"[1] :

```
#include <stdio.h>

main()
{
                    printf("hello, world\n");
}
```

- On peut facilement imaginer un autre programme qui prenant un entier positif n s'arrêterait en écrivant "Hello, world" dès qu'il aurait trouvé des entiers x, y et z tels que $x^n + y^n = z^n$

[1] Cf. B. W. Kernighan & D. M. Ritchie, "The C Programming Language, Prentice Hall, 1988

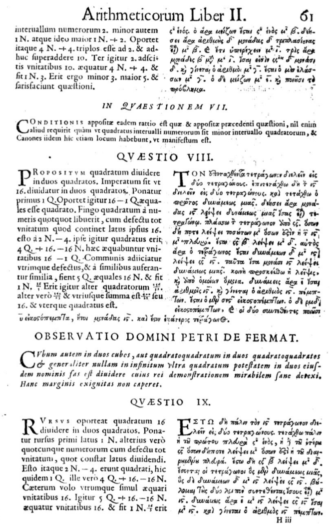
Programme "Hello, world"

```c
#include <stdio.h>

int exp(int i, int n)
/* calcule i puissance n */
{
        int ans, j;
        ans = 1;
        for (j=1; j<=n; j++) ans *= i;
        return(ans);
}

main()
{
        int n, total, x, y, z;
        int loop = 1;
        scanf("%d", &n);
        total = 3;
        while (loop) {
                for (x=1; x<=total-2; x++)
                        for (y=1; y<=total-x-1; y++) {
                                z = total - x - y;
                                if (exp(x,n) + exp(y,n) == exp(z,n))
                                        loop = 0;
                        }
                        total++;
        }
        printf("hello, world\n");
}
```

Dernier théorème de Fermat



Arithmeticorum Liber II. 61

interuallum numerorum 2. minor autem 1 N. atque ideo maior 1 N. + 2. Oportet itaque 4 N. + 4. triplos esse ad 2. & adhuc superaddere 10. Ter igitur 2. adscitis vnitatibus 10. æquatur 4 N. + 4. & fit 1 N. 3. Erit ergo minor 3. maior 5. & satisfaciunt quæstioni.

IN QVAESTIONEM VII.

CONDITIONIS appositæ eadem ratio est quæ & appositæ præcedenti quæstioni, nil enim aliud requirit quàm vt quadratus interualli numerorum sit minor interuallo quadratorum, & Canones iidem hic etiam locum habebunt, vt manifestum est.

QVÆSTIO VIII.

PROPOSITVM quadratum diuidere in duos quadratos. Imperatum sit vt 16. diuidatur in duos quadratos. Ponatur primus 1 Q. Oportet igitur 16 – 1 Q. æquales esse quadrato. Fingo quadratum à numeris quotquot libuerit, cum defectu tot vnitatum quod continet latus ipsius 16. esto à 2 N. – 4. ipse igitur quadratus erit 4 Q. + 16. – 16 N. hæc æquabuntur vnitatibus 16 – 1 Q. Communis adiiciatur vtrimque defectus, & à similibus auferantur similia, fient 5 Q. æquales 16 N. & fit 1 N. $\frac{16}{5}$ Erit igitur alter quadratorum $\frac{256}{25}$. alter verò $\frac{144}{25}$ & vtriusque summa est $\frac{400}{25}$ seu 16. & vterque quadratus est.

OBSERVATIO DOMINI PETRI DE FERMAT.

CVbum autem in duos cubos, aut quadratoquadratum in duos quadratoquadratos & generaliter nullam in infinitum vltra quadratum potestatem in duos eiusdem nominis fas est diuidere cuius rei demonstrationem mirabilem sane detexi. Hanc marginis exiguitas non caperet.

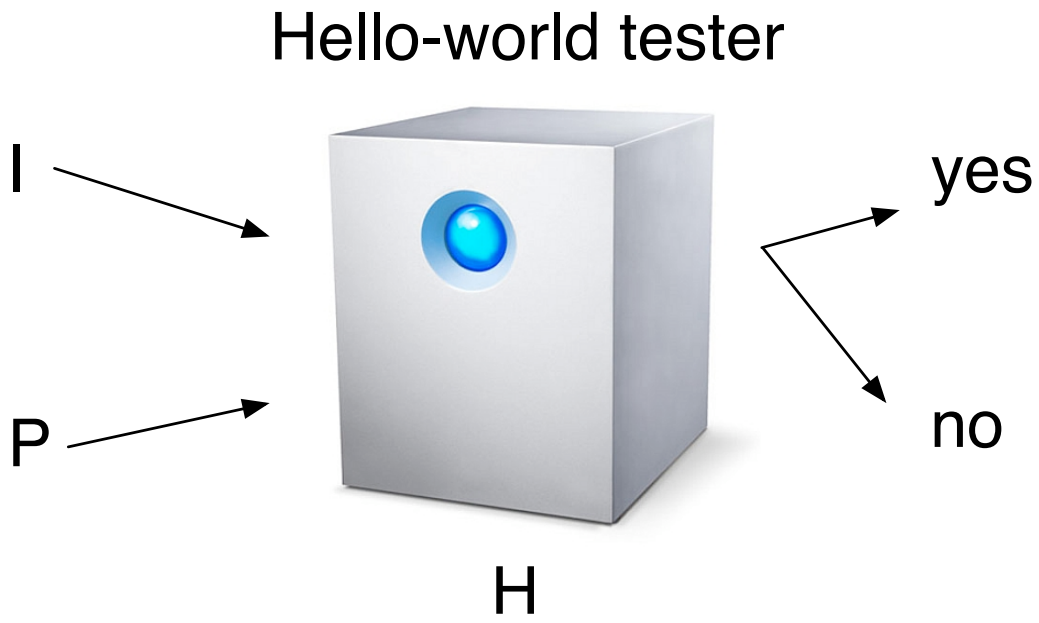
QVÆSTIO IX.

RVRSVS oporteat quadratum 16 diuidere in duos quadratos. Ponatur rursus primi latus 1 N. alterius verò quotcunque numerorum cum defectu tot vnitatum, quot constat latus diuidendi. Esto itaque 2 N. – 4. erunt quadrati, hic quidem 1 Q. ille verò 4 Q. + 16. – 16 N. Cæterum volo vtrumque simul æquari vnitatibus 16. Igitur 5 Q. + 16. – 16 N. æquatur vnitatibus 16. & fit 1 N. $\frac{16}{5}$ erit

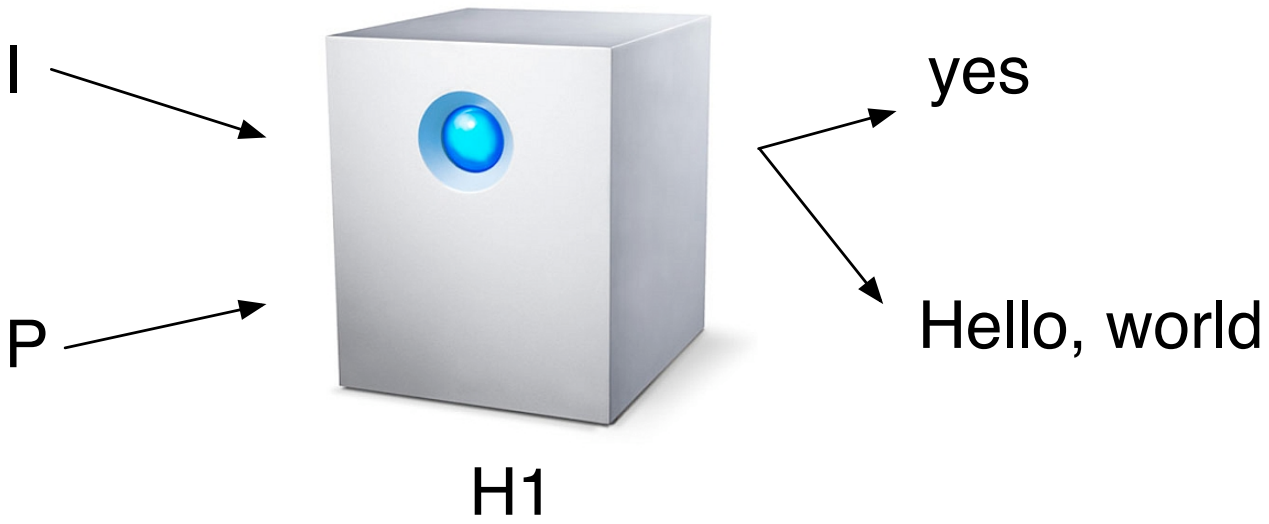
H iii

- Doit son nom à Pierre de Fermat qui écrivit en marge d'une traduction de l'Arithmetica de Diophante (autour de 1630), à côté de l'énoncé de ce problème : "J'ai trouvé une merveilleuse démonstration de cette proposition, mais la marge est trop étroite pour la contenir"
- Le théorème a finalement été démontré en 1994 par le mathématicien Andrew Wiles

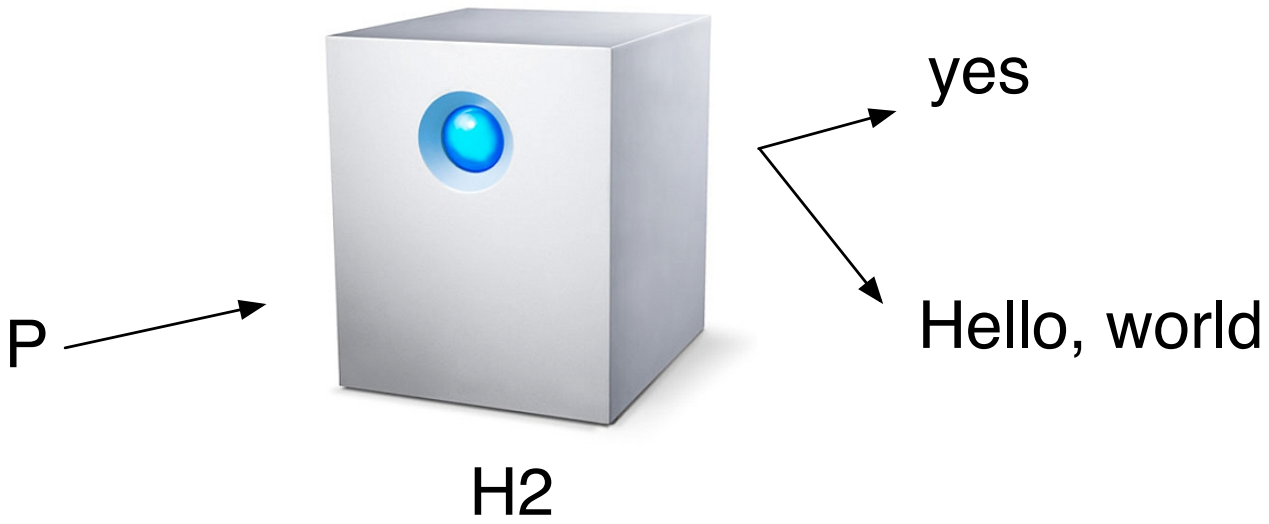
Le problème "Hello, world"



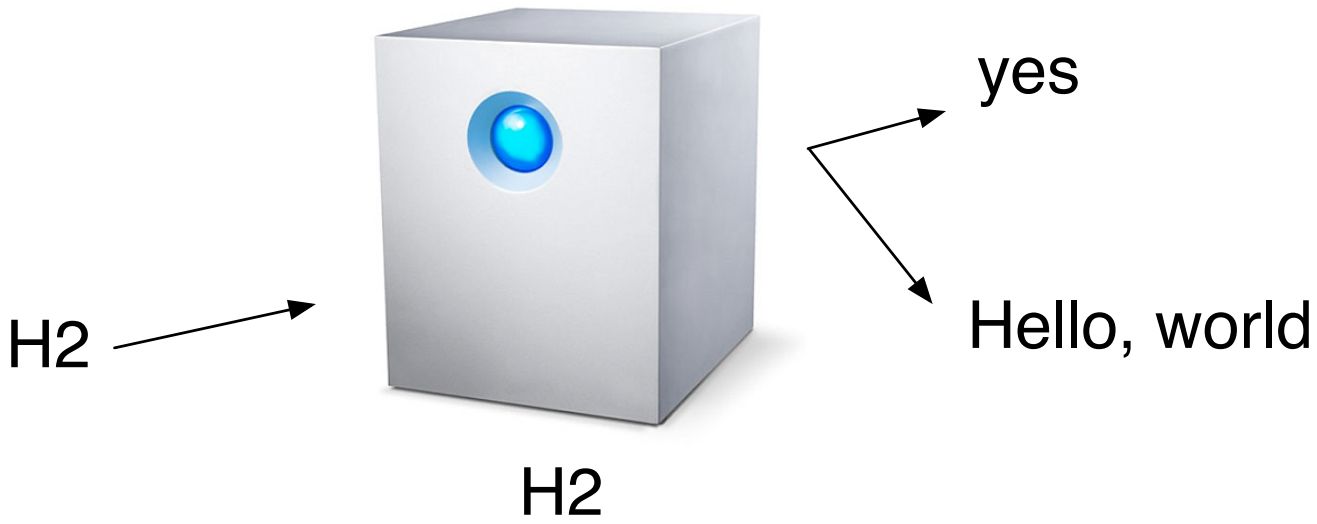
Le problème "Hello, world"



Le problème "Hello, world"



Le problème "Hello, world"



Structure de stockage d'une machine de Turing

- Opère sur une bande linéaire divisée en une infinité de cases. Chacune de ces cases contient un symbole pris dans un ensemble fini de symboles $\Gamma = \{s_0, s_1, \ldots, s_n\}$ (appelé alphabet)
- L'ensemble des mots sur Γ est noté Γ^*
- s_0 est traditionnellement le blanc aussi noté $\sqcup$. À l'instant initial, toutes les cases, sauf éventuellement un nombre fini, contiennent s_0
- N'accède à un moment donné qu'à une seule de ces cases à travers une fenêtre de lecture

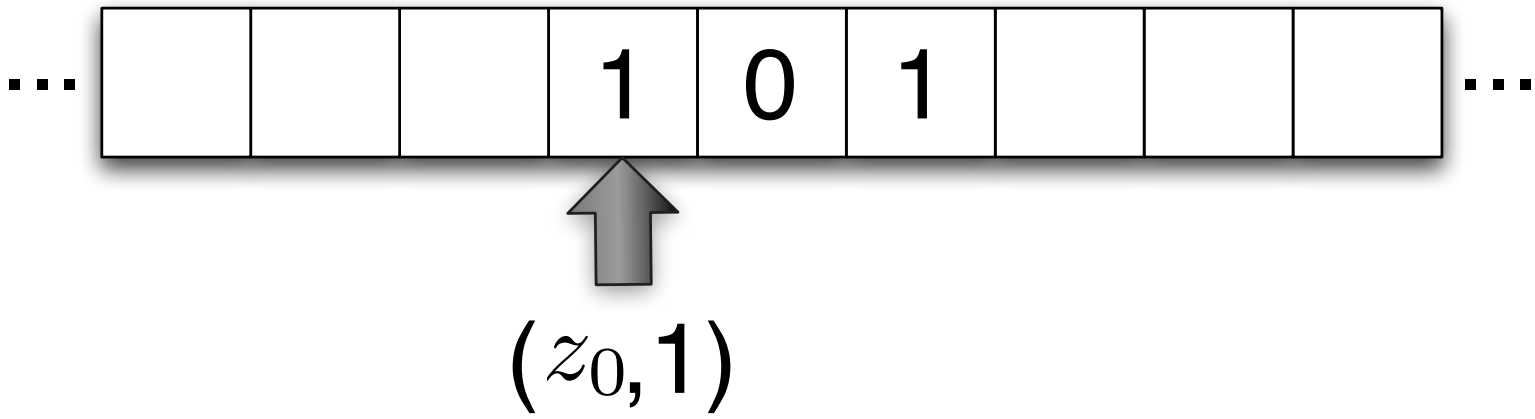
États et fonction de transition

- Une MT se trouve à un instant donné dans un état z_i pris dans un ensemble fini $Z = \{z_0, z_1, \ldots, z_h\}$
- z_0 est l'état initial ou état de démarrage
- z_h est l'état final ou état d'arrêt
- La fonction de transition est définie par une application δ :

$$\delta : (Z - \{z_h\}) \times \Gamma \rightarrow Z \times \Gamma \times \{G, D, I\}$$

- $\delta(z, s) = (z', s', G)$ signifie que **si** la MT est dans l'état z et lit le symbole s **alors** elle passe dans l'état z', écrit dans la case courante s' à la place de s et se place sur la case de gauche (D pour droite, I pour immobile)
- Lorsque la MT passe dans z_h, elle a terminé son calcul
- Un MT est entièrement définie par le n-uplet $(\Gamma, Z, \delta, z_0, z_h)$

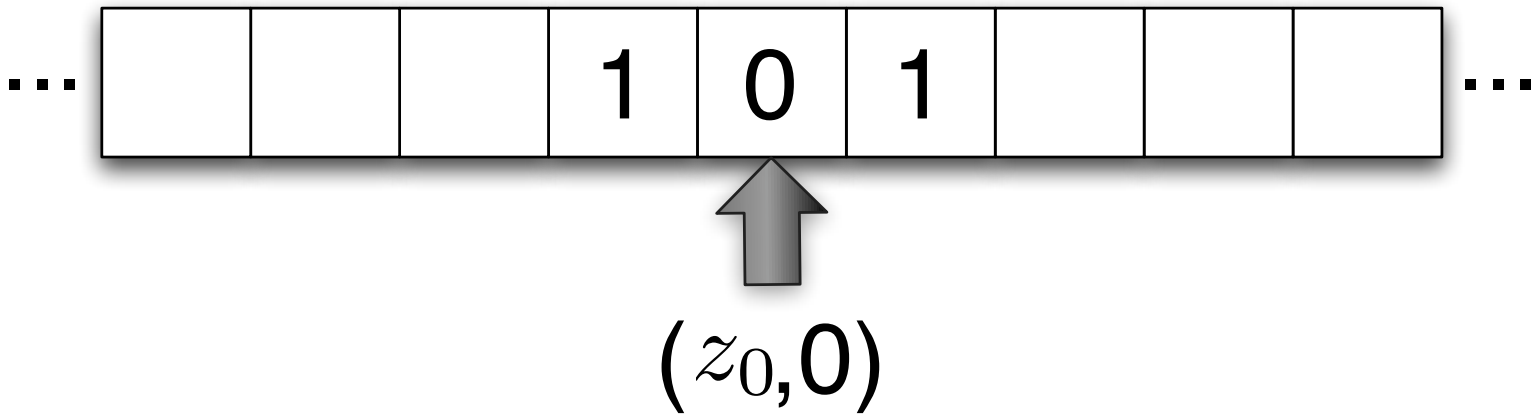
Exemple de machine de Turing



Programme de la MT effectuant $f(x) = x + 1$:

	$\sqcup$	0	1
z_0	$(z_1, \sqcup, G)$	$(z_0, 0, D)$	$(z_0, 1, D)$
z_1	$(z_h, 1, I)$	$(z_h, 1, I)$	$(z_1, 0, G)$

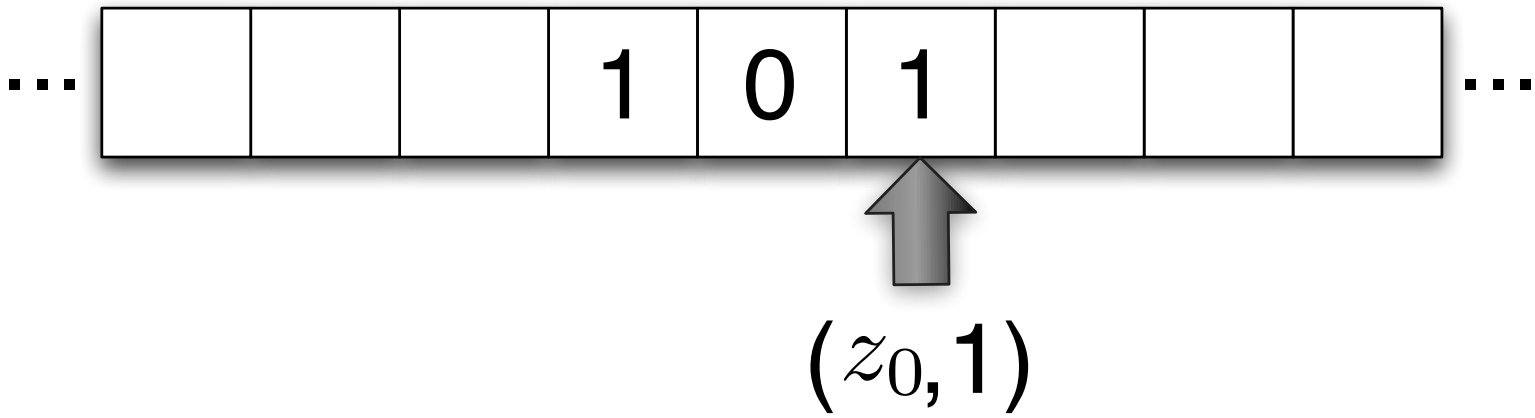
Exemple de machine de Turing



Programme de la MT effectuant $f(x) = x + 1$:

	$\sqcup$	0	1
z_0	$(z_1, \sqcup, G)$	$(z_0, 0, D)$	$(z_0, 1, D)$
z_1	$(z_h, 1, I)$	$(z_h, 1, I)$	$(z_1, 0, G)$

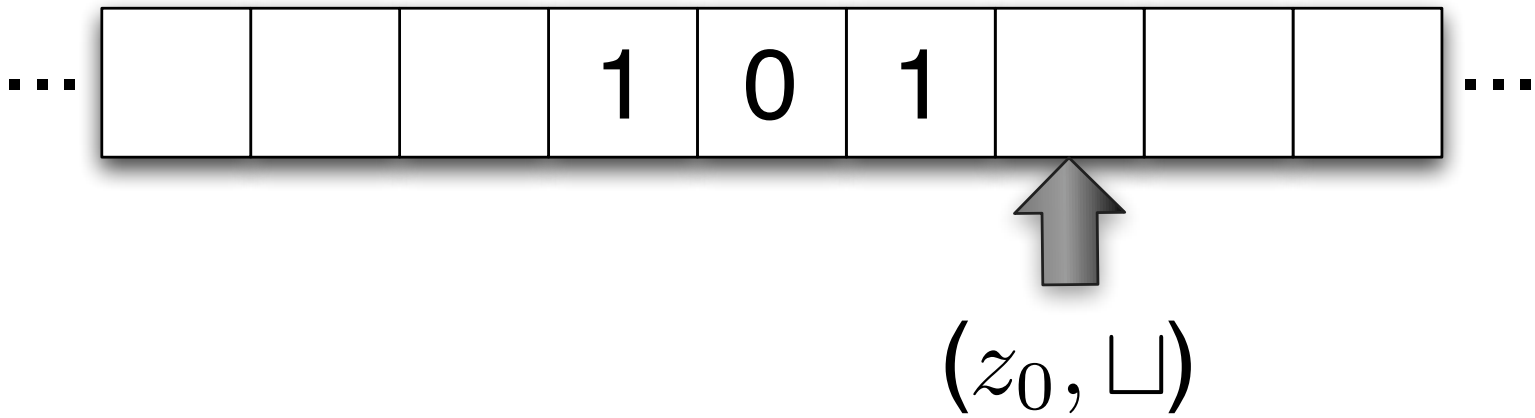
Exemple de machine de Turing



Programme de la MT effectuant $f(x) = x + 1$:

	$\sqcup$	0	1
z_0	$(z_1, \sqcup, G)$	$(z_0, 0, D)$	$(z_0, 1, D)$
z_1	$(z_h, 1, I)$	$(z_h, 1, I)$	$(z_1, 0, G)$

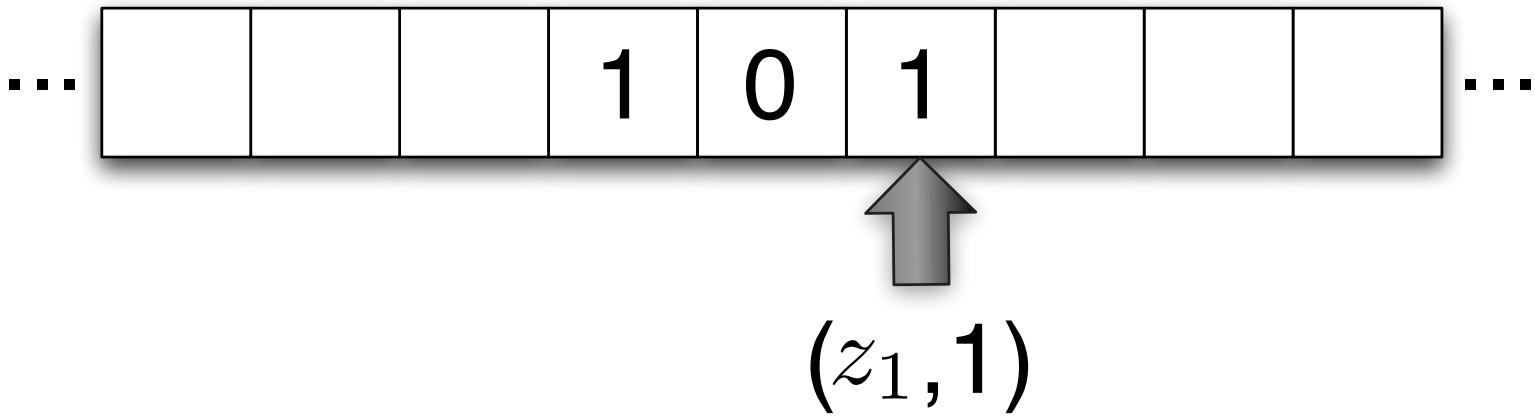
Exemple de machine de Turing



Programme de la MT effectuant $f(x) = x + 1$:

	$\sqcup$	0	1
z_0	$(z_1, \sqcup, G)$	$(z_0, 0, D)$	$(z_0, 1, D)$
z_1	$(z_h, 1, I)$	$(z_h, 1, I)$	$(z_1, 0, G)$

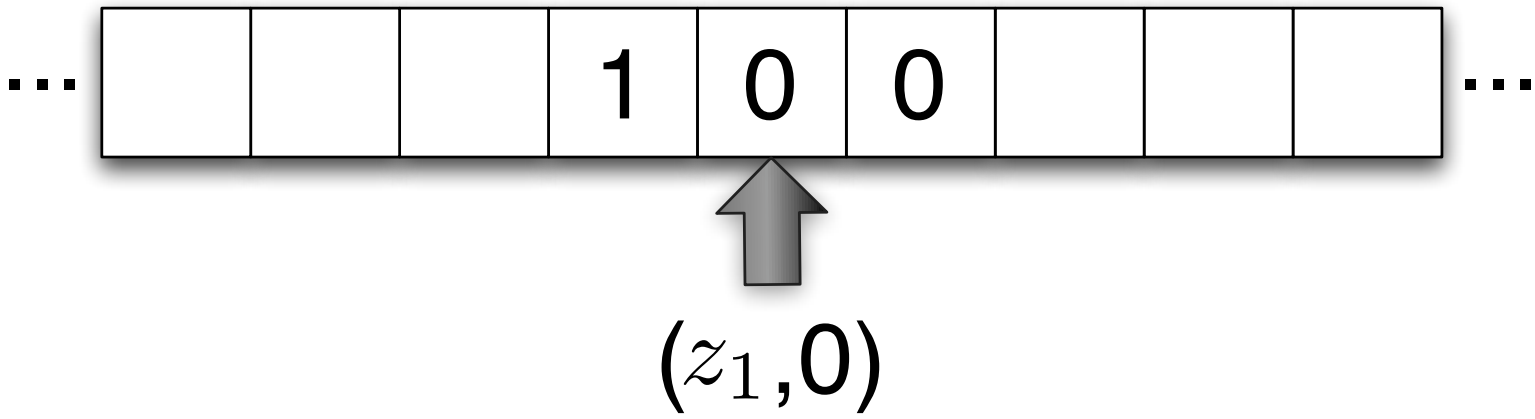
Exemple de machine de Turing



Programme de la MT effectuant $f(x) = x + 1$:

	$\sqcup$	0	1
z_0	$(z_1, \sqcup, G)$	$(z_0, 0, D)$	$(z_0, 1, D)$
z_1	$(z_h, 1, I)$	$(z_h, 1, I)$	$(z_1, 0, G)$

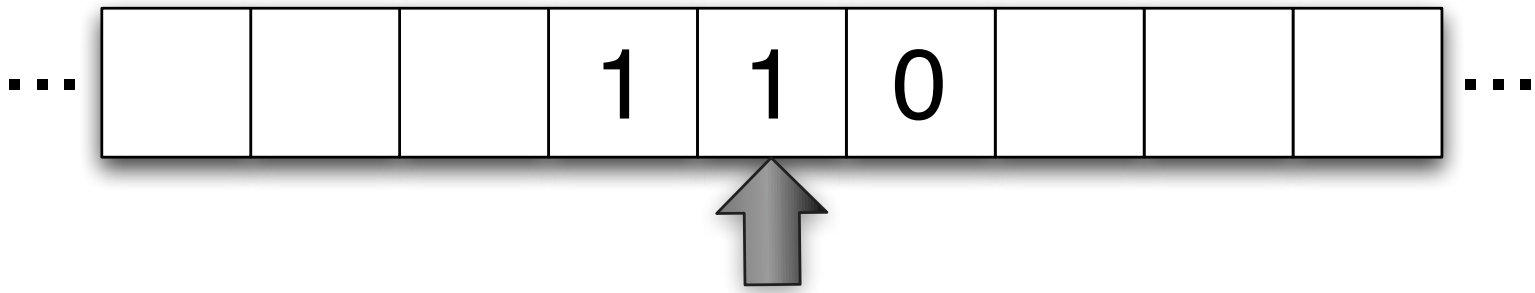
Exemple de machine de Turing



Programme de la MT effectuant $f(x) = x + 1$:

	$\sqcup$	0	1
z_0	$(z_1, \sqcup, G)$	$(z_0, 0, D)$	$(z_0, 1, D)$
z_1	$(z_h, 1, I)$	$(z_h, 1, I)$	$(z_1, 0, G)$

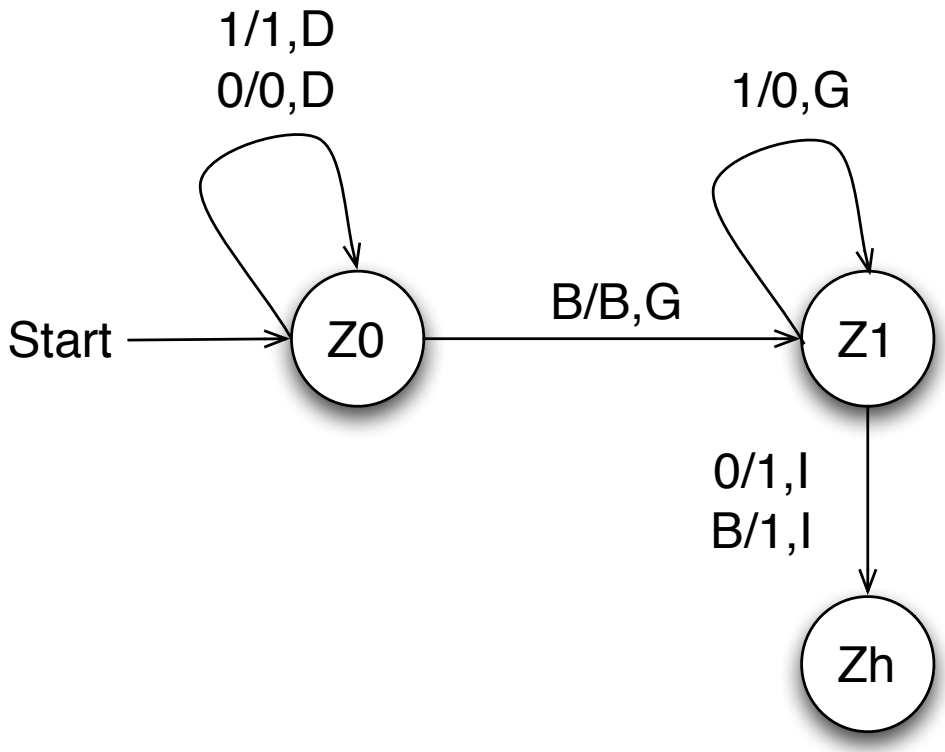
Exemple de machine de Turing



Programme de la MT effectuant $f(x) = x + 1$:

	$\sqcup$	0	1
z_0	$(z_1, \sqcup, \mathrm{G})$	$(z_0, 0, \mathrm{D})$	$(z_0, 1, \mathrm{D})$
z_1	$(z_h, 1, \mathrm{I})$	$(z_h, 1, \mathrm{I})$	$(z_1, 0, \mathrm{G})$

Diagramme de transitions



Thèse de Church-Turing

- 1900 : D. Hilbert se demande s'il est possible de trouver un "algorithme" pour établir la validité de toute proposition mathématique
- 1931 : K. Gödel publie son théorème d'incomplétude. Calcul des prédicats appliqué à des entiers comme moyen d'exprimer "tout calcul possible". D'autres langages sont toutefois possibles.
- 1936 : A. Turing propose son modèle (machine de Turing). Tous les autres modèles (par exemple le λ-calcul de A. Church) ont le même pouvoir expressif.
- **Thèse de Church-Turing : "la notion intuitive d'algorithme = machine de Turing"**

Mesurer la complexité

- Soit le langage $A = \{0^k 1^k | k \geq 0\}$. A est décidable mais en combien de temps ?
- Soit M_1 ="Sur l'entrée w de longueur n :
  1. Parcourir la bande et *rejeter* si 0 à droite de 1
  2. Répéter tant que 0 et 1 sur la bande :
  3. Parcourir la bande en effaçant alternativement 0 et 1.
  4. S'il reste des 0 après avoir effacé tous les 1 ou s'il reste des 1 après avoir effacé tous les 0 alors *rejeter*. Dans le cas contraire (il ne reste ni 0 ni 1 sur la bande) alors *accepter*."
- La complexité de M_1 est $O(n^2)$: $O(n)$ pour l'étape 1 + $O(n^2)$ pour les étapes 2 et 3 + $O(n)$ pour l'étape 4.

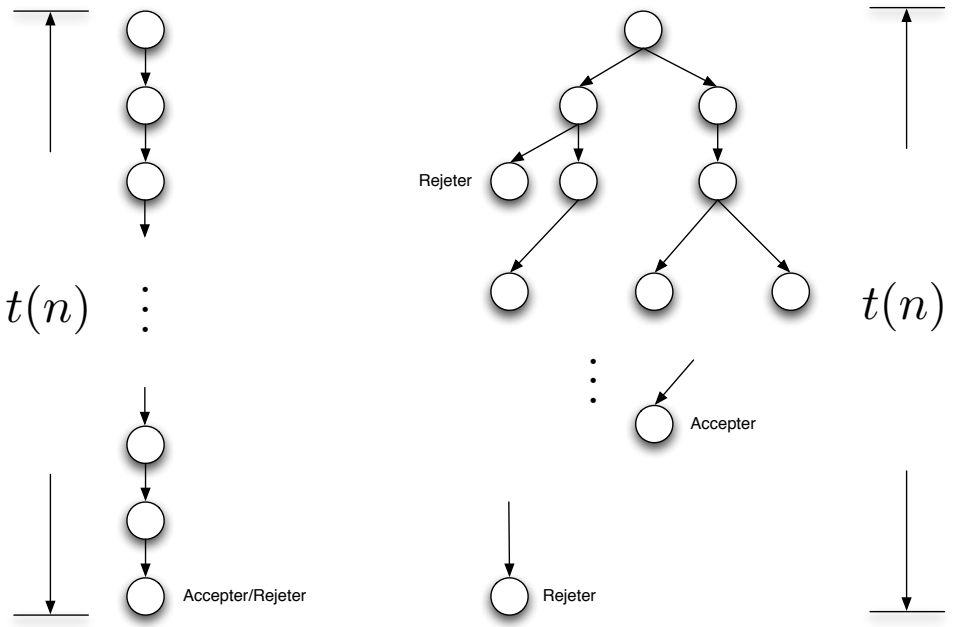
Mesurer la complexité

- Existe-t-il un autre algorithme plus rapide ?
- Soit M_2 ="Sur l'entrée w de longueur n :
  1. Parcourir la bande et *rejeter* si 0 à droite de 1
  2. Répéter tant que 0 et 1 sur la bande :
  3. Parcourir la bande en comptant le nombre total de 0 et 1 restants. Si impair alors *rejeter*
  4. Parcourir la bande à nouveau en effaçant le premier 0 rencontré puis tout 0 suivant un 0. Opérer de même pour les 1
  5. S'il ne reste plus de 0 et de 1 sur la bande alors *accepter* sinon *rejeter*."
- La complexité de M_2 est $O(n \log n)$

MT déterministe vs. non déterministe

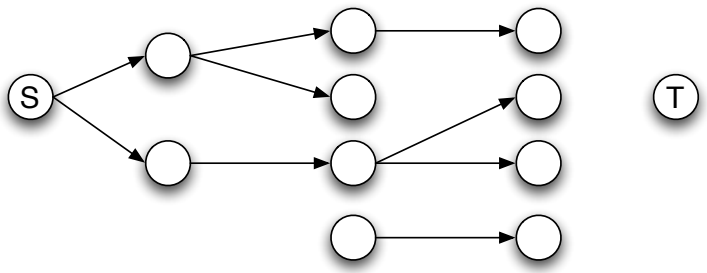
Théorème

Pour toute MT non déterministe à une bande s'exécutant en $t(n)$ ($t(n) \geq n$), il existe une MT déterministe à une bande équivalente et s'exécutant en $2^{O(t(n))}$.



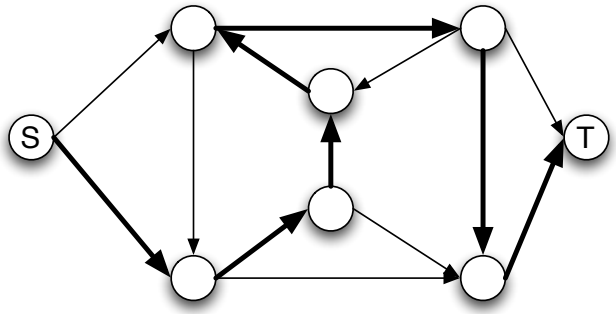
Problèmes polynomiaux

- Les problèmes pouvant être résolus en temps polynomial (sur une MT deterministe à une bande) sont dits "faciles" et, en pratique, résolubles. On note P cette classe de problèmes.
- De ce point de vue, des problèmes admettant des algorithmes en $O(n)$ et $O(n^3)$ sont du même niveau de difficulté !
- Les algorithmes en temps exponentiel sont caractéristiques de méthodes de résolution explorant exhaustivement l'espace des solutions = "force brute"
- Soit $PATH = \{\langle G, s, t\rangle | G$ est un graphe orienté tel qu'il existe un chemin entre s et $t\}$. $PATH \in \mathrm{P}$.



Problèmes NP

- Pour certains problèmes, on ne connait pas d'algorithmes en temps polynomial. Peut être n'existent-ils pas...
- Soit $HAMPATH = \{\langle G, s, t\rangle | G$ est un graphe orienté tel qu'il existe un chemin hamiltonien entre s et $t\}$. $HAMPATH \in \mathrm{NP}$.
- Vérifier l'existence d'un chemin hamiltonien (en temps polynomial) est beaucoup plus facile que déterminer son existence.
- NP est la classe des problèmes admettant un "vérificateur" en temps polynomial. Un vérificateur pour un langage A est un algorithme V tel que $A = \{w | V$ accepte $\langle w, c\rangle\}$: c est un "certificat" ou "preuve".



$\mathrm{P} = \mathrm{NP}$?

- P = classe des langages pour lesquels l'appartenance peut être décidée rapidement
- NP = classe des langages pour lesquels l'appartenance peut être vérifiée rapidement
- À l'heure actuelle, on ne sait pas si $\mathrm{P} = \mathrm{NP}$. La plupart des chercheurs pensent que ce n'est pas le cas !
- Par conséquent, il existe de nombreux problèmes en pratique très importants et impossibles à résoudre si on utilise la force brute !

Comment traiter les problèmes NP-difficiles ?

- Par approximation pour des problèmes d'optimisation. Algorithmes en temps polynomial garantissant de trouver des "bonnes solutions"
- Par des heuristiques = approches "**intelligentes**" donnant souvent des solutions mais pouvant échouer.

Qu'est-ce que l'IA ?

- Traiter de l'information symbolique par des heuristiques pour contenir l'explosion combinatoire
- "Faire exécuter par un ordinateur des tâches pour lesquelles nous sommes, dans un contexte donné, aujourd'hui meilleur que la machine" [Alliot, 2002]
- Construire des agents (artefact agissant sur un environnement et composé d'une boucle fermée de **perception – décision – action**) doté d'une rationalité limitée (prendre les "bonnes" décisions à partir de ressources limitées et d'informations incertaines ou erronées)

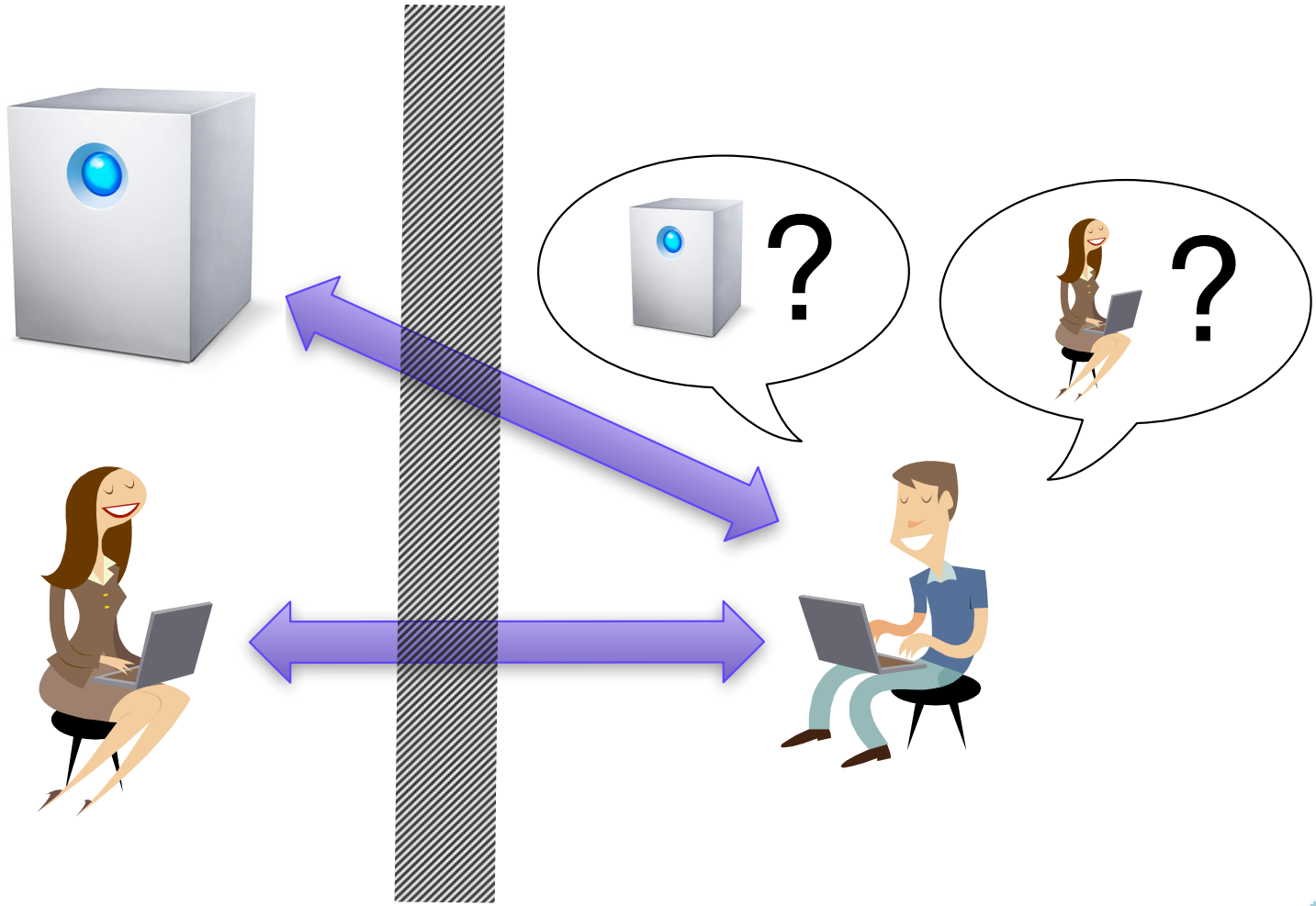
Qu'est-ce que l'IA ?

- "Méthodologie qui doit permettre de rendre les ordinateurs plus intelligents de façon à ce qu'ils montrent des caractéristiques normalement associées aux comportements humains, c'est-à-dire la compréhension du langage, l'apprentissage, la résolution de problèmes, le raisonnement etc." (E. Feigenbaum, Cf. [Alliot, 2002])
- "Étude des facultés mentales à l'aide de modèle de type calculatoire" (D. McDermott, Cf. [Alliot, 2002]). Voir lien entre IA et sciences cognitives

Quelques thématiques

- Résolution de problèmes (A^*, MinMax, CSP)
- Représentation des connaissances et raisonnement automatique (logiques, agents et SMA, réseaux bayésiens, MDP)
- Apprentissage (Apprentissage Symbolique Automatique, algorithmique évolutionnaire, réseaux de neuronnes, Q-learning)
- Perception, action (planification automatique, robotique) etc.

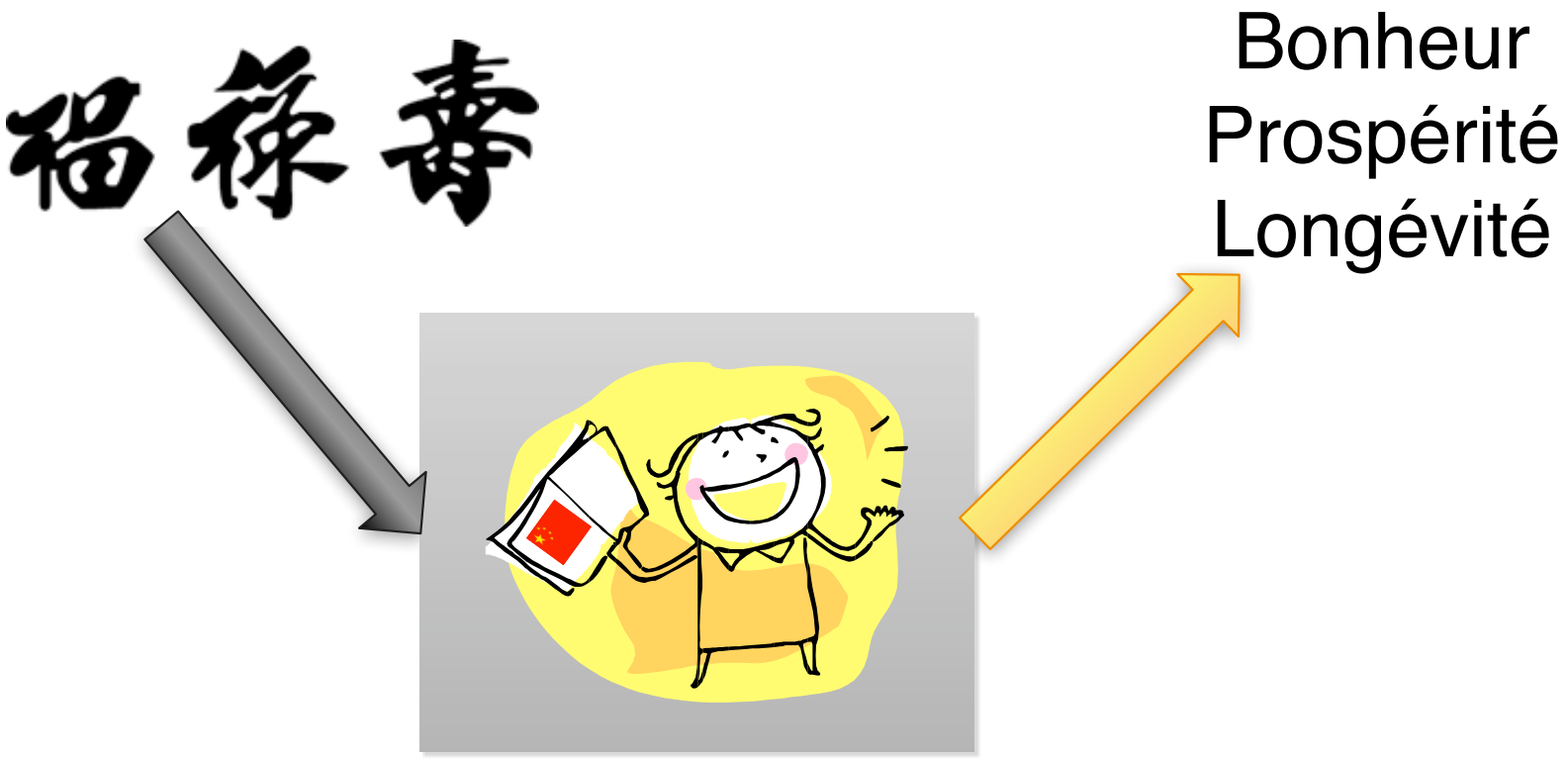
Test de Turing (Alan Turing – 1950)



Test de Turing (Alan Turing – 1950)

- Proposition de test d'intelligence artificielle ayant la faculté d'imiter la conversation humaine
- Mettre en confrontation verbale une personne avec un ordinateur et une autre personne à l'aveugle. Si la personne qui engage les conversations n'est pas capable de dire qui est l'ordinateur et qui est l'autre être humain, on peut considérer que le logiciel de l'ordinateur a passé avec succès le test
- Test "behaviouriste". Identité fondée sur le principe d'indiscernabilité

Chambre chinoise (John Searle – 1980)



Chambre chinoise (John Searle – 1980)

- Expérience de pensée imaginée par John Searle : un programme informatique est-il suffisant pour donner un esprit à une machine ?
- "La sémantique du contenu mental n'est pas intrinsèque à la seule syntaxe du programme informatique"
- "La distinction la plus profonde qu'on puisse effectuer n'est pas entre l'esprit et la matière, mais entre deux aspects du monde : ceux qui existent indépendamment d'un observateur, et que j'appelle intrinsèques, et ceux qui sont relatifs à l'interprétation d'un observateur." Le calcul informatique, pour être qualifié de tel, n'existe que relativement à une interprétation qui assigne une certaine distribution de zéros et de uns à un certain état physique.
- Searle a tenté de démontrer par cette voie que le concept d'intelligence artificielle dite forte devait être abandonné. Toutefois, le raisonnement de Searle part de l'hypothèse que la sémantique ne se réduit pas à la syntaxe, ce que contestent les partisans de l'IA forte, qui suggèrent précisément que le sens peut être une propriété émergente de programmes informatiques ("le tout possède parfois des propriétés qui n'existent dans aucune de ses parties").
- Pour une critique de l'IA au sens "fort", cf. H. L. Dreyfus, "*What Computer Still Can't Do, a Critique of Artificial Reason*", The MIT Press, 1992

Qu'est-ce que l'IA ?

- En pratique... le test de Turing a peu d'influence sur les recherches en Intelligence Artificielle. La philosophie de l'I.A., selon John McCarthy, "a peu de chances d'avoir plus d'effet sur la pratique de la recherche en I.A. que la philosophie de la science en a généralement sur la pratique de la science".

Liens intéressants

Rob Janoff
http ://robjanoff.com

A. Hodges
"Alan Turing", The Stanford Encyclopedia of Philosophy (Fall 2008 Edition), E. N. Zalta (ed.),
http ://plato.stanford.edu/archives/fall2008/entries/turing/

"Chambre chinoise", Wikipédia en français,
http ://fr.wikipedia.org/wiki/Chambre_chinoise

"Test de Turing", Wikipédia en français,
http ://fr.wikipedia.org/wiki/Test_de_Turing

Livres


M. A. Boden
"The Philosophy of Artificial Intelligence", Oxford University Press, 1990


J.-M Alliot, T. Schiex, P. Brisset et F. Garcia
"Intelligence Artificielle & informatique théorique", Cépadues éditions, 2002


S. Russell and P. Norvig
"Artificial Intelligence, A Modern Approach", Prentice Hall, 2003


M. Ghallab, D. Nau and P. Traverso
"Automated Planning, theory and practise", Morgan Kaufmann, 2004


M. Sipser
"Introduction to the Theory of Computation", Thomson Course Technology, 2006


J. E. Hopcroft, R. Motwani and J. D. Ullman
"Introduction to Automata Theory, Languages and Computation", Addison Wesley, 2007


A. Doxiadis , Ch. Papadimitriou and A. Papadatos

"Logicomix", Vuibert, 2010