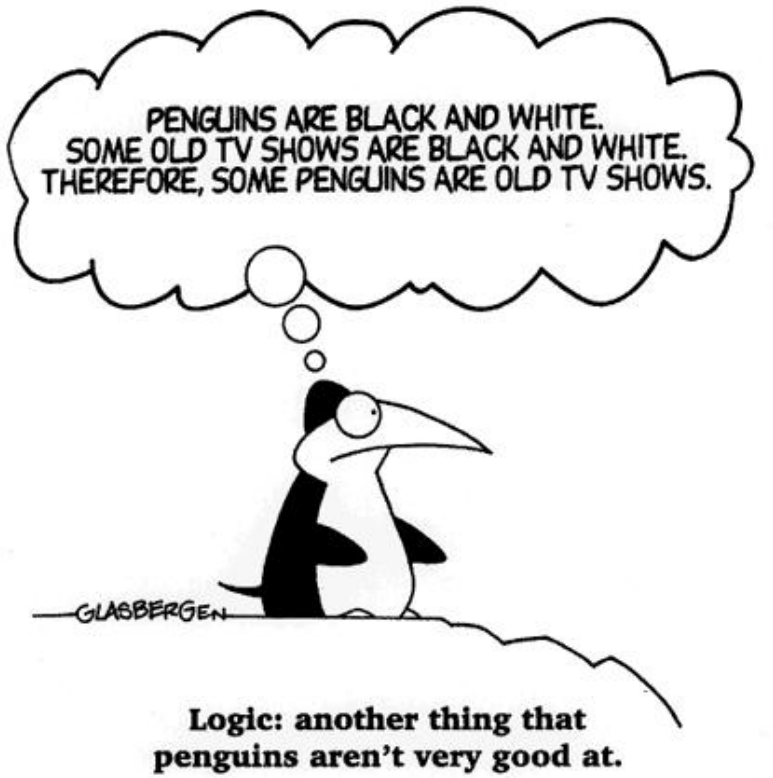
Logique, inférence et calcul

D. Bonnay / M. Cozic / H. Galinon

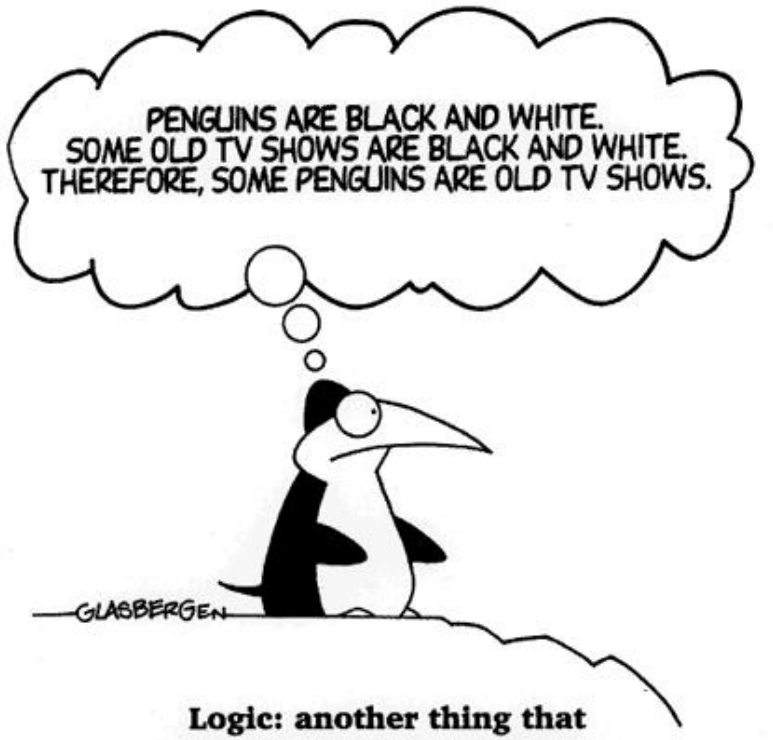
Cogmaster 2007-2008

La logique a d'abord affaire à un certain type d'activités mentales :



les inférences déductives

mais l'histoire ne s'arrête pas là



Logique et Sciences Cognitives

Hypothèse centrale des sciences cognitives

La pensée peut être comprise en termes

- de **représentations** mentales,
- et de **procédures de calcul** qui opèrent sur ces représentations.

Logique et Sciences Cognitives

Hypothèse centrale des sciences cognitives

La pensée peut être comprise en termes

- de **représentations** mentales,
- et de **procédures de calcul** qui opèrent sur ces représentations.

Des outils logiques pour les sciences cognitives

La logique fournit :

- des **langages formels** pour modéliser les représentations mentales,
- et une **théorie du calcul** pour modéliser les procédures de calcul sur ces représentations

Quelques exemples d'applications :

- Théorie normative de la rationalité :
  - Qu'est-il rationnel d'inférer de mes croyances ?
  - Qu'est-il rationnel d'induire de ce que m'apprend l'expérience ?
  - Qu'est-il rationnel de choisir étant donné mes croyances et mes préférences ?

Quelques exemples d'applications :

- Théorie normative de la rationalité :
  - Qu'est-il rationnel d'inférer de mes croyances ?
  - Qu'est-il rationnel d'induire de ce que m'apprend l'expérience ?
  - Qu'est-il rationnel de choisir étant donné mes croyances et mes préférences ?
- Psychologie du raisonnement :
  - Comment les humains raisonnent-ils ?
  - Pourquoi faisons-nous certaines erreurs ?
  - Comment mesurer la difficulté d'un raisonnement ?

Quelques exemples d'applications :

- Théorie normative de la rationalité :
  - Qu'est-il rationnel d'inférer de mes croyances ?
  - Qu'est-il rationnel d'induire de ce que m'apprend l'expérience ?
  - Qu'est-il rationnel de choisir étant donné mes croyances et mes préférences ?
- Psychologie du raisonnement :
  - Comment les humains raisonnent-ils ?
  - Pourquoi faisons-nous certaines erreurs ?
  - Comment mesurer la difficulté d'un raisonnement ?
- Linguistique :
  - Qu'est-ce qu'une grammaire ?
  - Qu'est-ce que la signification d'une phrase ?
  - Quel est le pouvoir expressif des langages que nous utilisons ?

Le programme

- La logique comme théorie du raisonnement correct, qui répond à la question :
  “What follows from what ?”
- La logique comme théorie du calcul, qui répond à la question :
  “Qu’est-ce qui peut calculer quoi ?”

Qu'est-ce qu'un argument correct ?

la notion d'argument

- La logique s'intéresse aux arguments
- Un **argument** est la donnée d'une conclusion et d'un ensemble de prémisses censées la justifier :

$$\frac{\begin{array}{l}\text{Si Marie boit, Pierre trinque}\\ \text{Marie boit}\end{array}}{\text{Pierre trinque}}$$

- Prémisses et conclusions sont des énoncés ; un **énoncé** est une expression susceptible d'être vraie ou fausse.
- La logique cherche à caractériser parmi les arguments ceux qui sont **valides**

quelques arguments

- Valides ou non ?

Si Marie boit, Pierre trinque
Marie boit

Pierre trinque

Tous les philosophes sont des ma
Descartes est un mammifère

Descartes est un philosophe

Si Marie boit, Pierre trinque
Pierre trinque

Marie boit

Tous les logiciens sont marathonie
Proust est un logicien

Proust est marathonien

la notion d'argument valide

- Aristote sur la validité :

  "...un syllogisme est un discours dans lequel, certaines choses étant posées, quelque chose d'autre que ces données en résulte nécessairement par le seul fait de ces données."
  Premiers Analytiques, 24b18, trad. Tricot, Vrin

- Un argument **valide** est un argument dont la vérité des prémisses entraîne celle de la conclusion.
- Un argument **sain** (*sound*) est un argument valide dont les prémisses sont vraies.

substitutions, 1

Si Marie boit, Pierre trinque
Marie boit

Pierre trinque

Si Marie boit, Marc aboit
Marie boit

Marc aboit

Si Marie boit, Marc aboit
Marie boit

Pierre trinque

Si Marie mord, Marc aboit
Marie mord

Marc aboit

Ou bien Marie boit, ou bien
Pierre trinque
Marie boit

Pierre trinque

Si ϕ, ψ
ϕ

ψ

substitutions, 2

Tous les logiciens sont marathoniens
Proust est un logicien

Proust est marathonien

Certains logiciens sont marathoni
Proust est un logicien

Proust est marathonien

Tous les logiciens sont tatoués
Proust est un logicien

Proust est tatoué

Tous les P sont Q
a est P
---
a est Q

Tous les logiciens sont tatoués
Ludwig est un logicien

Ludwig est un tatoué

les constantes logiques

- 2 classes de **constantes logiques** :
  - (i) les connecteurs propositionnels : "si..., alors...", "...et...", "...ou...", "il est faux que..."
  - (ii) les quantificateurs : "tous...", "il existe...", "certains..."
- 2 logiques élémentaires :
  - (i) la **logique propositionnelle** (LP) qui traite exclusivement des connecteurs propositionnels
  - (ii) la **logique du premier ordre** (LPO) qui traite des connecteurs propositionnels et des quantificateurs
- Dans chaque cas, on verra
  - (i) syntaxe : construction d'un langage formel
  - (ii) sémantique : assignation d'une signification aux expressions du langage formel

Logique propositionnelle

introduction

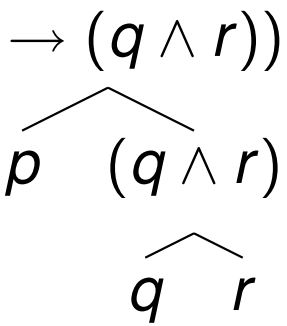
- La LP traite des arguments dont la validité repose exclusivement sur les connecteurs propositionnels
- Dans ces arguments, seule importe la structure des énoncés complexes, pas la structure des énoncés atomiques
- La syntaxe de la LP considère donc les énoncés atomiques comme primitifs et s'intéresse aux combinaisons que l'on peut en faire avec les connecteurs
- La syntaxe de la LP construit la notion de **formule** qui est la contrepartie formelle de la notion d'énoncé (expression susceptible d'être vraie ou fausse)

syntaxe de LP

- L'**alphabet** du langage contient des *formules atomiques* $p, q, r...$ des symboles de *connecteurs* $\vee, \wedge, \neg, \rightarrow, \leftrightarrow$ et des *parenthèses*), (
- Les **formules** sont définies ainsi :
  - (i) les formules atomiques sont des formules
  - (ii) si ϕ et ψ sont des formules, alors $(\phi \vee \psi)$, $(\phi \wedge \psi)$, $(\phi \rightarrow \psi)$, $(\phi \leftrightarrow \psi)$ et $\neg\phi$ sont des formules
  - (iii) seules les expressions engendrées par un nombre fini d'application de (i)-(ii) sont des formules

arbre de formation

- Une formule se décompose de manière unique jusqu'aux formules atomiques qui la constituent. On peut le représenter *via* l'**arbre de formation** d'une formule.
- Exemple : $(p \rightarrow (q \wedge r))$



- La sémantique est la partie de la logique qui s'occupe de l'*interprétation* du langage formel
- Dans le cas propositionnel, on assigne aux formules du langage une **valeur de vérité** : le vrai (1) ou le faux (0)
- La logique classique obéit au **principe de bivalence** : toute formule a une et une seule valeur de vérité
- La valeur de vérité des formules atomiques est arbitraire ; en revanche, une fois ces valeurs fixées, la valeur de vérité des formules complexes dépend de la façon dont elles sont composées.

la négation

- Supposons que $V(p) = 1$ ("il pleut" est vrai). Quelle est la valeur de vérité de $\neg p$ ("il est faux qu'il pleuve") ?

$$V(\neg p) = 0$$

- Inversement, si $V(p) = 0$, alors

$$V(\neg p) = 1$$

- De manière générale, la négation est un opérateur qui inverse la valeur de vérité de la formule à laquelle elle s'applique

$V(\phi)$	$V(\neg\phi)$
1	0
0	1

la conjonction et la disjonction

- Une conjonction $(\phi \wedge \psi)$ est vraie uniquement quand les deux membres sont vrais
- Une disjonction $(\phi \vee \psi)$ est vraie dès que l'un des membres est vrai (disjonction "inclusive")

$V(\phi)$	$V(\psi)$	$V((\phi \wedge \psi))$	$V((\phi \vee \psi))$
1	1	1	1
1	0	0	1
0	1	0	1
0	0	0	0

le conditionnel

- Le conditionnel reflète (plus ou moins) le "si..., alors..." de la langue naturelle.
- Un conditionnel $(\phi \rightarrow \psi)$ n'est faux que lorsque l'antécédent est vrai et le conséquent faux.

$V(\phi)$	$V(\psi)$	$V((\phi \rightarrow \psi))$
1	1	1
1	0	0
0	1	1
0	0	1

valuation

- Une **valuation atomique** est une fonction qui assigne à chaque formule atomique une valeur de vérité.
- Une **valuation** est une fonction qui assigne à chaque formule une valeur de vérité conformément à l'interprétation des connecteurs.
- Il existe une et une seule valuation qui étend une valuation atomique donnée. Concrètement, si ϕ est une formule complexe qui contient les formules atomiques p, q, r, alors une fois que la valeur de vérité de p, q, r est fixée, celle de ϕ est déterminée.

exemple

- Considérons la formule $\phi := \neg(p \rightarrow \neg q)$ et supposons que $V(p) = V(q) = 1$. Quelle est la valeur de vérité de ϕ ?

p	q	$\neg q$	$(p \rightarrow \neg q)$	$\neg(p \rightarrow \neg q)$
1	1	0	0	1

- On peut représenter les valeurs de vérités de ϕ pour toute combinaison possible de valeurs de vérité de ses propositions atomiques en une **table de vérité** :

p	q	$\neg q$	$(p \rightarrow \neg q)$	$\neg(p \rightarrow \neg q)$
1	1	0	0	1
1	0	1	1	0
0	1	0	1	0
0	0	1	1	0

tautologies & antilogies

- Une formule est **satisfiable** s'il existe une valuation qui la rend vraie.
  Exemple : $(p \vee q)$
- Une formule est **tautologique** ou **valide** si toute valuation la rend vraie.
  Exemple : $(p \vee \neg p)$
- Une formule est **antilogique** ou **contradictoire** si aucune valuation ne la rend vraie.
  Exemple : $(p \wedge \neg p)$

exemples

- Une tautologie : $(p \vee \neg p)$

p	$\neg p$	$(p \vee \neg p)$
1	0	1
0	1	1

- Une antilogie : $(p \wedge \neg p)$

p	$\neg p$	$(p \wedge \neg p)$
1	0	0
0	1	0

conséquence logique

- Retour à l'intuition de départ : un argument est valide si la vérité des prémisses entraîne celle de la conclusion.
- On dit qu'une valuation satisfait un ensemble de formules Γ si elle satisfait toutes les formules de Γ.
- Γ a pour **conséquence logique** ϕ si toute valuation qui satisfait Γ satisfait ϕ

exemple

- $\Gamma = \{p, (p \rightarrow q)\}$ et $\phi = q$.

Si Marie boit, Pierre trinque
Marie boit

Pierre trinque

p	q	$(p \rightarrow q)$
1	1	1
1	0	0
0	1	1
0	0	1

La logique du premier ordre

limites de la LP

- Comment traiter l'argument suivant en LP ?

  Tous les logiciens sont malins
  Proust est un logicien

  Proust est malin

- Il faut associer à chaque énoncé une formule atomique - disons, respectivement, p, q, r. Mais r n'est pas conséquence logique de $\{p, q\}$.
- Il faut une analyse plus fine de la structure des énoncés.

prédication & quantification

- La prédication :
  "Rudolf est logicien" $\rightsquigarrow L(r)$
  "Rudolf est plus grand que Ludwig" $\rightsquigarrow G(r, l)$
  $\Rightarrow$ deux classes de symboles :
  - les symboles de constantes : $a, b, c, ...$
  - les symboles de prédicat
  $P^{(1)}, Q^{(1)}, R^{(1)}, ..., P^{(n)}, Q^{(n)}, R^{(n)}, ...$
- La quantification
  "Il existe un logicien" $\rightsquigarrow \exists x L(x)$
  "Tout le monde est plus grand que Ludwig" $\rightsquigarrow \forall x G(x, l)$
  "Tout le monde aime quelqu'un" $\rightsquigarrow \forall x \exists y A(x, y)$
  $\Rightarrow$ deux classes de symboles :
  - symboles de variables $x, y, z, ...$
  - quantificateurs : $\forall, \exists$

paraphrases, 1

- Les propositions aristotéliciennes :

A	Tout S est P	Universelle affirmative
E	Nul S n'est P	Universelle négative
I	Quelque S est P	Particulière affirmative
O	Quelque S n'est pas P	Particulière négative

- Paraphrases :

"Tous les philosophes viendront"	$\rightsquigarrow$	$\forall x(Px \rightarrow Sx)$
"Quelque philosophe viendra"	$\rightsquigarrow$	$\exists x(Px \wedge Sx)$
"Aucun philosophe ne viendra"	$\rightsquigarrow$	$\forall x(Px \rightarrow \neg Sx)$
"Quelque philosophe ne viendra pas"	$\rightsquigarrow$	$\exists x(Px \wedge \neg Sx)$

paraphrases, 2

"Paul aime quelqu'un"	$\rightsquigarrow$	$\exists x Apx$
"Paul est aimé de quelqu'un"	$\rightsquigarrow$	$\exists x Axp$
"Tout le monde aime quelqu'un"	$\rightsquigarrow$	$\forall x \exists y Axy$
"Tous ceux qui aiment Paul aiment Marie"	$\rightsquigarrow$	$\forall x (Axp \rightarrow Ax$
"Paul n'aime personne"	$\rightsquigarrow$	$\forall x \neg Apx$
"Paul aime quelqu'un qui aime Marie"	$\rightsquigarrow$	$\exists x (Apx \wedge Axm$
"Paul aime tous ceux qui aiment tout le monde"	$\rightsquigarrow$	$\forall x (\forall y Axy \rightarrow$

syntaxe de LPO

- L'**alphabet** de LPO contient
  - (i) des symboles de constantes $a, b, c...$
  - (ii) des symboles de prédicats $P^{(1)}, Q^{(1)}, R^{(1)}, ..., P^{(n)}, Q^{(n)}, R^{(n)}, ...$
  - (iii) de variables $x, y, z, ...$
  - (iv) les connecteurs propositionnels $\vee, \wedge, \neg, \rightarrow, \leftrightarrow$
  - (v) les quantificateurs $\forall, \exists$
  - (vi) des parenthèses
- Un langage du premier ordre est une spécification de (i)-(ii). (iii)-(vi) sont communs à tous les langages du premier ordre.

syntaxe de LPO, suite

- Un **terme** est une variable ou un symbole de constante.
- Les **formules** sont définies ainsi :
  - (i) Si $P^{(n)}$ est un symbole de prédicat n-aire et $t_1, ..., t_n$ sont des termes, alors $P(t_1, ..., t_n)$ est une formule
  - (ii) Si ϕ et ψ sont des formules, alors $\neg\phi$, $(\phi \vee \psi)$, etc. sont des formules
  - (iii) Si ϕ est une formule, alors $\forall x \phi$ et $\exists x \phi$ sont des formules
  - (iv) Seules les expressions engendrées par un nombre fini d'applications de (i)-(iii) sont des formules.

variables libres vs. liées

- Différence importante entre les deux types de formules :

$\forall x B(x)$
$\forall x \forall y A(x, y)$

$B(x)$
$\forall x A(x, y)$

- Les variables des formules de gauche sont toutes "sous la portée" d'un quantificateur ou **liées**. Certaines occurences de variables sont au contraire **libres** dans les formules de de droite.
- On appelle **énoncé** (parfois **formule close**) une formule où toutes les variables sont liées. Les énoncés en ce sens correspondent aux énoncés au sens informel i.e. ils sont susceptibles d'être vrai ou faux.

- En LP, on ne traite que de valeurs de vérité : la valeur de vérité d'une formule est déterminée par celles des formules qui la composent. La sémantique de la LPO doit être enrichie car les parties d'une formule ne sont pas nécessairement des formules elles-mêmes
  $\Rightarrow$ il faut interpréter les symboles de constantes et les symboles de prédicats
- 1ère idée fondamentale : on se donne un domaine de discours M
  - les symboles de constantes dénotent des éléments du domaine de de discours ;
  - les symboles de prédicats n-aires dénotent des relations n-aires sur M.
- Intuitivement : une formule Pa est vraie si l'élément du domaine dénoté par a appartient à l'ensemble d'éléments du domaines dénoté par P

$\mathcal{L}$-structures

- Soit $\mathcal{L}$ un langage du premier ordre. Une **$\mathcal{L}$-structure** $\mathcal{M} = (M, I^{\mathcal{M}})$ est une paire constituée
  - (i) d'un domaine (non vide) d'individus M
  - (ii) d'une fonction d'interprétation $I^{\mathcal{M}}$
    - si c est un symbole de constante, alors $I^{\mathcal{M}}(c) = c^{\mathcal{M}} \in M$
    - si $P^{(n)}$ est un symbole de prédicat, alors $I^{\mathcal{M}}(P) = P^{\mathcal{M}} \subseteq M^n$

exemple 1

- Soit le langage $\mathcal{L} = (H, F, P, p, j, m, a)$.
  $\mathcal{M} = (\{Paul, Jacques, Marie, Anne\}, I^{\mathcal{M}})$ où
  - $I^{\mathcal{M}}(H) = \{Paul, Jacques\}$
  - $I^{\mathcal{M}}(F) = \{Marie, Anne\}$
  - $I^{\mathcal{M}}(P) = \{Jacques, Anne\}$
  - $I^{\mathcal{M}}(p) =$ Paul, $I^{\mathcal{N}}(j) =$ Jacques, $I^{\mathcal{N}}(m) = Marie$, $I^{\mathcal{N}}(a) = Anne$
- Intuitivement,
  $\mathcal{M} \models H(j)$, $\mathcal{M} \not\models H(m)$
  $\mathcal{M} \models \forall x(H(x) \vee F(x))$, $\mathcal{M} \not\models \forall x P(x)$

exemple 2

- Soit le langage $\mathcal{L} = (O, E, c_0, c_1)$. $\mathcal{N} = (\mathbb{N}, I^{\mathcal{M}})$ où
  - $I^{\mathcal{N}}(E) = \{0, 2, 4, 6...\}$
  - $I^{\mathcal{N}}(O) = \{1, 3, 5, 7...\}$
  - $I^{\mathcal{N}}(c_0) = 0$
  - $I^{\mathcal{N}}(c_1) = 1$
- Intuitivement,
  $\mathcal{N} \models E(c_0), \mathcal{N} \not\models E(c_1)$
  $\mathcal{N} \models \forall x(O(x) \vee E(x)), \mathcal{N} \not\models \forall x E(x)$

assignations

- Problème : si l'interprétation d'une formule est déterminée par l'interprétation de ses parties, comment construire l'interprétation d'une formule comme $\forall x O(x)$?
- De manière générale, le problème est d'interpréter les formules comportant des variables libres.
- 2nde idée fondamentale : l'**assignation** : une assignation $g(.)$ associe à chaque variable x un élément du domaine d'invididus $g(x) \in M$.
- On définit ensuite la notion de satisfaction d'une formule par une assignation. $P(x)$ est par exemple satisfaite par toutes les assignations qui assignent à x l'un des élements de $P^{\mathcal{I}}$.

satisfaction

- Soit $\mathcal{L}$ un langage du premier ordre, $\mathcal{M}$ une $\mathcal{L}$-structure et g une assignation. Pour une formule ϕ, on définit la **satisfaction** de ϕ par l'assignation g :

(i) si $\phi := Rt_1...t_k$ pour un symbole de prédicat R d'arité k et des termes $t_1, ..., t_k$, alors $\mathcal{M}, g \vDash \phi$ ssi $(t_1^{\mathcal{M},g}, ..., t_k^{\mathcal{M},g}) \in R^{\mathcal{M}}$

(ii) si $\phi := \neg\psi$, alors $\mathcal{M}, g \vDash \phi$ ssi $\mathcal{M}, g \nvDash \psi$

(iv) $\phi := (\psi \wedge \chi)$, alors $\mathcal{M}, g \vDash \phi$ ssi $\mathcal{M}, g \vDash \psi$ et $\mathcal{M}, g \vDash \chi$

(v) si $\phi := \forall x_n \psi$, alors $\mathcal{M}, g \vDash \phi$ ssi pour tout $a \in M$, $\mathcal{M}, g[x_n \to a] \vDash \psi$

(vi) si $\phi := \exists x_n \psi$, alors $\mathcal{M}, g \vDash \phi$ ssi il existe $a \in M$ tq. $\mathcal{M}, g[x_n \to a] \vDash \psi$

vérité et validité

- Si toutes les assignations satisfont ϕ, on dit que ϕ est **vraie** dans $\mathcal{M}$ ou que $\mathcal{M}$ est un **modèle** de ϕ. On le note

$$\mathcal{M} \models \phi$$

- ϕ est **valide** si elle est satisfaite par toute assignation dans toute $\mathcal{L}$-structure. On le note

$$\models \phi$$

exemple

Soit le langage $\mathcal{L} = (H, F, P, p, j, m, a)$.
$\mathcal{M} = (\{Paul, Jacques, Marie, Anne\}, I^{\mathcal{M}})$ où

- $I^{\mathcal{M}}(H) = \{Paul, Jacques\}$
- $I^{\mathcal{M}}(F) = \{Marie, Anne\}$
- $I^{\mathcal{M}}(P) = \{Jacques, Anne\}$
- $I^{\mathcal{M}}(p) =$ Paul, $I^{\mathcal{N}}(j) =$ Jacques, $I^{\mathcal{N}}(m) = Marie$, $I^{\mathcal{N}}(a) = Anne$

$g_1(x) = Paul$	$g_2(x) = Anne$
$\mathcal{M}, g_1 \models Fm$, $\mathcal{M}, g_1 \not\models Fj$	$\mathcal{M}, g_2 \models Fm$, $\mathcal{M}, g_2 \not\models Fj$
$\mathcal{M}, g_1 \not\models Fx$	$\mathcal{M}, g_2 \models Fx$
$\mathcal{M}, g_1 \not\models (Fx \vee Px)$	$\mathcal{M}, g_2 \not\models (Fx \wedge Px)$
$\mathcal{M}, g_1 \models Hx$	$\mathcal{M}, g_2 \not\models Hx$
$\mathcal{M}, g_1 \models \exists x Fx$	$\mathcal{M}, g_2 \models \exists x Fx$
$\mathcal{M}, g_1 \models \forall x(Fx \vee Hx)$	$\mathcal{M}, g_2 \models \forall x(Fx \vee Hx)$

exemple

Soit le langage $\mathcal{L}_1 = (P^{(1)}, R^{(2)}, c_0, c_1)$. On peut l'interpréter dans la $\mathcal{L}_1$-structure $\mathcal{N} = (\mathbb{N}, I^{\mathcal{N}})$ où

- $I^{\mathcal{N}}(P) = \{0, 2, 4, 6...\}$
- $I^{\mathcal{N}}(R) =>$ (l'ordre strict usuel sur les entiers)
- $I^{\mathcal{N}}(c_0) = 0$
- $I^{\mathcal{N}}(c_1) = 1$

$g_1(x_1) = 5, g_1(x_2) = 2$	$g_2(x_1) = 0, g_2(x_2) = 3$
$\mathcal{N}, g_1 \not\models Rc_1x_1$	$\mathcal{N}, g_2 \models Rc_1x_1$
$\mathcal{N}, g_1 \models Rx_1x_2$	$\mathcal{N}, g_2 \not\models Rx_1x_2$
$\mathcal{N}, g_1 \models \forall x_1 \exists x_2 Rx_2x_1$	$\mathcal{N}, g_2 \models \forall x_1 \exists x_2 Rx_2x_1$
$\mathcal{N}, g_1 \not\models \exists x_1 (Rx_1c_0 \wedge Rc_1x_1)$	$\mathcal{N}, g_2 \not\models \exists x_1 (Rx_1c_0 \wedge Rc_1x_1)$

exemple

Soit le langage $\mathcal{L}_2 = (H^{(1)}, F^{(1)}, A^{(2)}, p, j, m, a)$. On peut l'interpréter dans la $\mathcal{L}_2$-structure $\mathcal{M} = (\{Paul, Jacques, Marie, Anne\}, I^{\mathcal{M}})$ où

- $I^{\mathcal{M}}(H) = \{Paul, Jacques\}$
- $I^{\mathcal{M}}(F) = \{Marie, Anne\}$
- $I^{\mathcal{M}}(A) =$ $\{(Marie, Paul), (Paul, Anne), (Jacques, Jacques), (Anne, Marie)\}$
- $I^{\mathcal{M}}(p) =$ Paul, $I^{\mathcal{M}}(j) =$ Jacques, $I^{\mathcal{M}}(m) = Marie$, $I^{\mathcal{M}}(a) = Anne$

$g_1(x_1) = Paul, g_1(x_2) = Marie$	$g_2(x_1) = Anne, g_2(x_2) = Paul$
$\mathcal{M}, g_1 \not\models Apx_1$	$\mathcal{M}, g_2 \models Apx_1$,
$\mathcal{M}, g_1 \models \exists x_1 Ajx_1$	$\mathcal{M}, g_2 \models \exists x_1 Ajx_1$
$\mathcal{M}, g_1 \not\models \forall x_1 Ax_1x_2$	$\mathcal{M}, g_2 \not\models \forall x_1 Ax_1x_2$

La logique modale

Au-delà de la logique pure

Logique propositionnelle : et, ou, non...
Logique du premier ordre : tous, il existe

Au-delà de la logique pure

Logique propositionnelle : et, ou, non...
Logique du premier ordre : tous, il existe

$\rightarrow$ l'analyse logique peut être étendue à d'autres opérateurs :
logique modale : il est possible que, il est nécessaire que
logique épistémique : l'agent *i* sait que

Au-delà de la logique pure

Logique propositionnelle : et, ou, non...
Logique du premier ordre : tous, il existe

$\rightarrow$ l'analyse logique peut être étendue à d'autres opérateurs :
logique modale : il est possible que, il est nécessaire que
logique épistémique : l'agent *i* sait que

L'agent *i* sait que *p*	$K_i p$
L'agent *i* ne sait pas si *p*	$\neg K_i q \wedge \neg K_i q$
Il est nécessaire que *p*	$\Box p$
Il est possible que *q*	$\Diamond q$

Le paradoxe de Fitch

La logique épistémique permet de représenter la connaissance des agents, elle a des implications notamment en théorie des jeux.

Le paradoxe de Fitch

La logique épistémique permet de représenter la connaissance des agents, elle a des implications notamment en théorie des jeux.

Elle vise d'abord à étudier les propriétés de la connaissance. Un 'test' crucial pour la logique est sa capacité à résoudre les paradoxes, comme le paradoxe de Fitch.

Le paradoxe de Fitch

La logique épistémique permet de représenter la connaissance des agents, elle a des implications notamment en théorie des jeux.

Elle vise d'abord à étudier les propriétés de la connaissance. Un 'test' crucial pour la logique est sa capacité à résoudre les paradoxes, comme le paradoxe de Fitch.

Principe de vérifiabilité
Tout ce qui est vrai peut être connu.

$$\phi \rightarrow \Diamond K \phi$$

Le paradoxe de Fitch

La logique épistémique permet de représenter la connaissance des agents, elle a des implications notamment en théorie des jeux.

Elle vise d'abord à étudier les propriétés de la connaissance. Un 'test' crucial pour la logique est sa capacité à résoudre les paradoxes, comme le paradoxe de Fitch.

Principe de vérifiabilité
Tout ce qui est vrai peut être connu.

$$\phi \rightarrow \Diamond K\phi$$

Supposons qu'il existe un certain énoncé p qui n'est pas actuellement connu.

$$p \wedge \neg Kp$$

Prenons pour ϕ l'énoncé $p \wedge \neg Kp$,

$$p \wedge \neg Kp \rightarrow \Diamond K(p \wedge \neg Kp)$$

Prenons pour ϕ l'énoncé $p \wedge \neg Kp$,

$$p \wedge \neg Kp \rightarrow \Diamond K(p \wedge \neg Kp)$$

Considérons $K(p \wedge \neg Kp)$, si je sais que ϕ et ψ, alors je sais que ϕ et je sais que ψ :

$$Kp \wedge K\neg Kp$$

Prenons pour ϕ l'énoncé $p \wedge \neg Kp$,

$$p \wedge \neg Kp \rightarrow \Diamond K(p \wedge \neg Kp)$$

Considérons $K(p \wedge \neg Kp)$, si je sais que ϕ et ψ, alors je sais que ϕ et je sais que ψ :

$$Kp \wedge K\neg Kp$$

or ce qui est connu est vrai

$$Kp \wedge \neg Kp$$

En remplaçant $K(p \wedge \neg Kp)$ par $Kp \wedge \neg Kp$ dans l'instance du principe de vérifiabilité qui nous intéresse, on obtient :

$$p \wedge \neg Kp \rightarrow \Diamond(Kp \wedge \neg Kp)$$

En remplaçant $K(p \wedge \neg Kp)$ par $Kp \wedge \neg Kp$ dans l'instance du principe de vérifiabilité qui nous intéresse, on obtient :

$$p \wedge \neg Kp \rightarrow \Diamond(Kp \wedge \neg Kp)$$

Mais on a supposé que $p \wedge \neg K$, donc

$$\Diamond(Kp \wedge \neg Kp)$$

En remplaçant $K(p \wedge \neg Kp)$ par $Kp \wedge \neg Kp$ dans l'instance du principe de vérifiabilité qui nous intéresse, on obtient :

$$p \wedge \neg Kp \rightarrow \Diamond(Kp \wedge \neg Kp)$$

Mais on a supposé que $p \wedge \neg K$, donc

$$\Diamond(Kp \wedge \neg Kp)$$

or $Kp \wedge \neg Kp$ est une contradiction ($\bot$), d'où :

$$\Diamond \bot$$

Que faire ?

La calculabilité

Logique et calcul

Deux branches de la logique formelle :

- la théorie de la **calculabilité**
  - Qu'est-ce qu'une procédure de calcul / un algorithme / une méthode effective ?
  - Quels sont les problèmes qui admettent une solution selon une méthode effective ?

Logique et calcul

Deux branches de la logique formelle :

- la théorie de la **calculabilité**
  - Qu'est-ce qu'une procédure de calcul / un algorithme / une méthode effective ?
  - Quels sont les problèmes qui admettent une solution selon une méthode effective ?
- la théorie de la **complexité**.
  - Qu'est-ce qu'une procédure de calcul *efficace* ?
  - Comment peut-on mesurer la difficulté d'un problème ?

Remarque

On va s'intéresser essentiellement à des calculs sur les entiers naturels.

Mais notre notion de calcul s'applique, moyennant un codage adéquat, à la résolution de tout problème qui peut être encodé comme un problème à propos des entiers.

Remarque

On va s'intéresser essentiellement à des calculs sur les entiers naturels.

Mais notre notion de calcul s'applique, moyennant un codage adéquat, à la résolution de tout problème qui peut être encodé comme un problème à propos des entiers.

Exemple

- Problème : décider si une suite d'expressions constitue une phrase dans une grammaire donnée.

Remarque

On va s'intéresser essentiellement à des calculs sur les entiers naturels.

Mais notre notion de calcul s'applique, moyennant un codage adéquat, à la résolution de tout problème qui peut être encodé comme un problème à propos des entiers.

Exemple

- Problème : décider si une suite d'expressions constitue une phrase dans une grammaire donnée.
- Codage des suites d'expressions dans les entiers, grammaire comme fonction qui envoie les suites d'expressions dans $\{0, 1\}$ (correct / pas correct)

Remarque

On va s'intéresser essentiellement à des calculs sur les entiers naturels.

Mais notre notion de calcul s'applique, moyennant un codage adéquat, à la résolution de tout problème qui peut être encodé comme un problème à propos des entiers.

Exemple

- Problème : décider si une suite d'expressions constitue une phrase dans une grammaire donnée.
- Codage des suites d'expressions dans les entiers, grammaire comme fonction qui envoie les suites d'expressions dans $\{0, 1\}$ (correct / pas correct)
- Le problème initial se réduit à un problème à propos des entiers (calcul de la 'fonction' grammaire)

Modèles de calcul

De nombreux modèles différents :

- Machines de Turing,
  Notion de machine abstraite proposée par Alan Turing,

Modèles de calcul

De nombreux modèles différents :

- Machines de Turing,
  Notion de machine abstraite proposée par Alan Turing,
- Un langage de programmation élémentaire,
  par exemple langage impératif,

Modèles de calcul

De nombreux modèles différents :

- Machines de Turing,
  Notion de machine abstraite proposée par Alan Turing,
- Un langage de programmation élémentaire,
  par exemple langage impératif,
- Fonctions récursives,
  $\approx$ fonctions élémentaires sur les entiers + récursion

Modèles de calcul

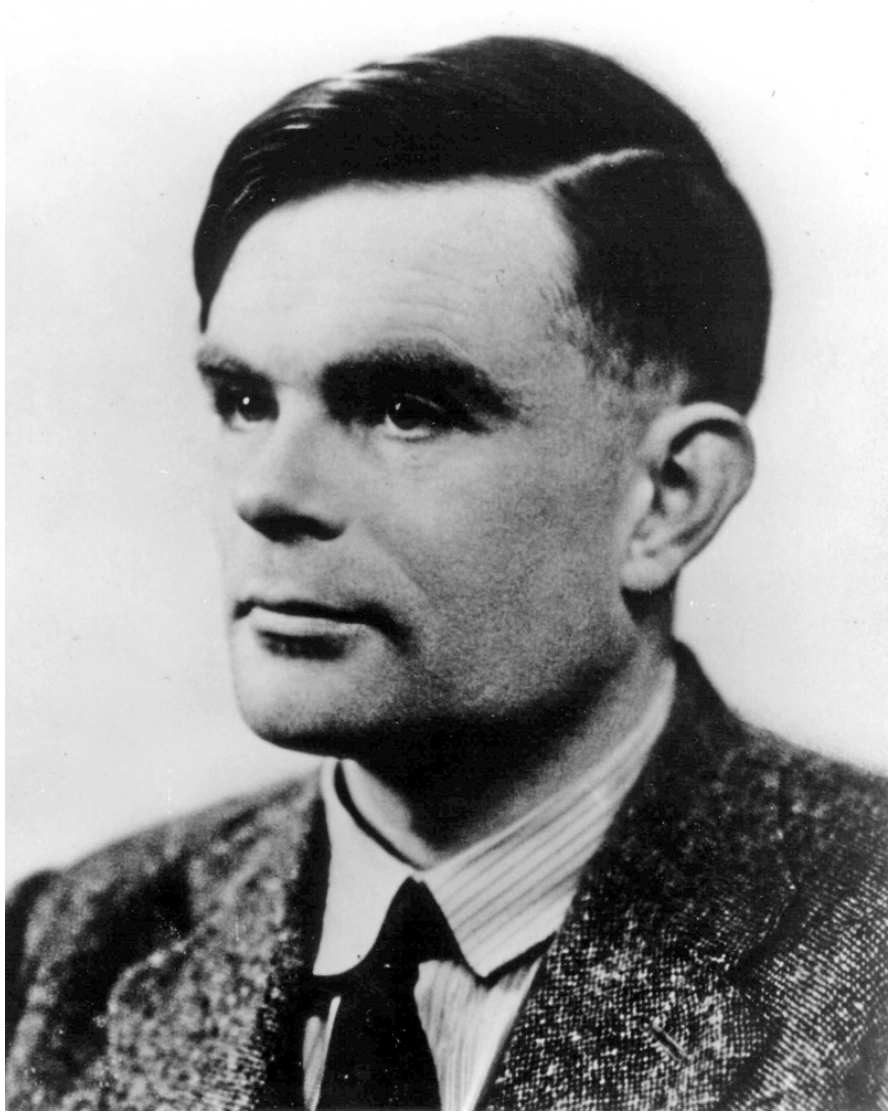
De nombreux modèles différents :

- Machines de Turing,
  Notion de machine abstraite proposée par Alan Turing,
- Un langage de programmation élémentaire,
  par exemple langage impératif,
- Fonctions récursives,
  $\approx$ fonctions élémentaires sur les entiers + récursion
- Lambda-calcul
  $\approx$ langage de programmation fonctionnelle (Lisp)

Modèles de calcul

De nombreux modèles différents :

- Machines de Turing,
  Notion de machine abstraite proposée par Alan Turing,
- Un langage de programmation élémentaire,
  par exemple langage impératif.
- Fonctions récursives,
  $\approx$ fonctions élémentaires sur les entiers + récursion
- Lambda-calcul
  $\approx$ langage de programmation fonctionnelle (Lisp)



Alan Turing (1912–1954)

Machines de Turing (1)

Machine abstraite : modèle mathématique des machines réelles

Machines de Turing (1)

Machine abstraite : modèle mathématique des machines réelles

Trois ingrédients :

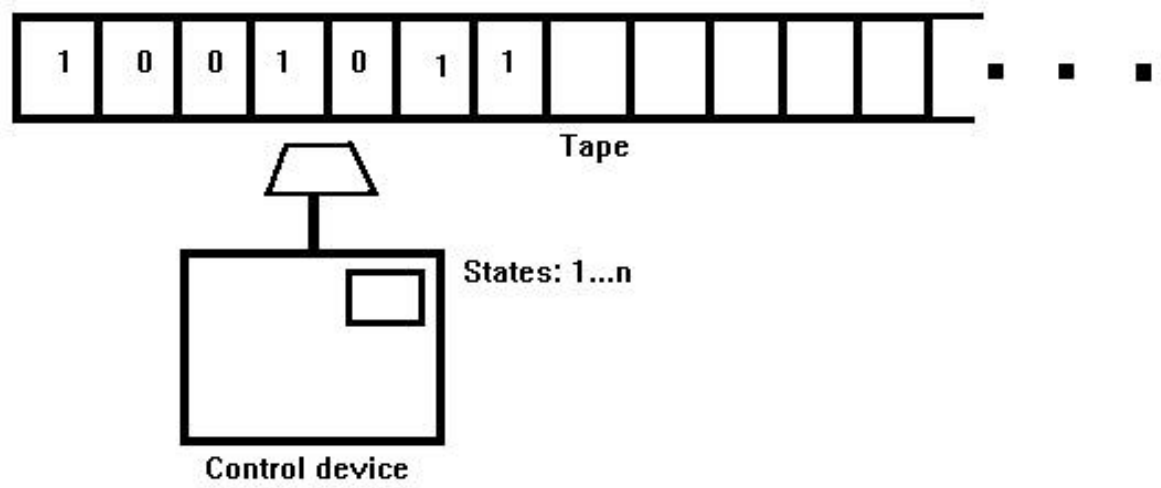
- Un ruban, avec des cases remplies de 0 ou de 1.
- Une tête de lecture / écriture
- Des instructions qui déterminent ce que va faire la machine.

Machines de Turing (1)

Machine abstraite : modèle mathématique des machines réelles

Trois ingrédients :

- Un ruban, avec des cases remplies de 0 ou de 1.
- Une tête de lecture / écriture
- Des instructions qui déterminent ce que va faire la machine.



Machine de Turing (2)

Une machine est définie par :

- un ensemble fini d'états $q_1, ..., q_n$, auxquels sont associés des instructions de la
- pour chaque état q_i, des instructions de la forme :
  si la case du ruban est marquée de 0 (resp. 1), accomplir l'action X et passer à l'état q_j.

Machine de Turing (2)

Une machine est définie par :

- un ensemble fini d'états $q_1, ..., q_n$, auxquels sont associés des instructions de la
- pour chaque état q_i, des instructions de la forme :
  si la case du ruban est marquée de 0 (resp. 1), accomplir l'action X et passer à l'état q_j.

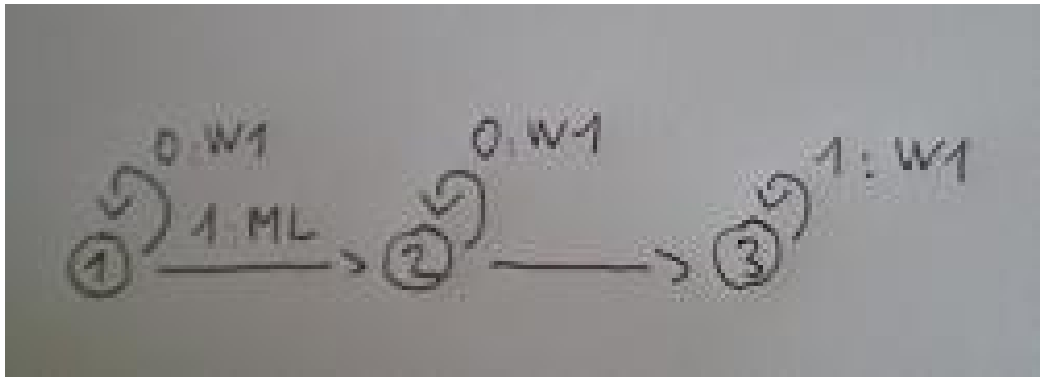
Les actions X possibles sont :

- écrire 0 [W0],
- écrire 1 [W2],
- bouger la tête d'une case vers la gauche [ML],
- bouger la tête d'une case vers la droite [MR],
- s'arrêter [pas d'instruction].

Exemple de machine (1)

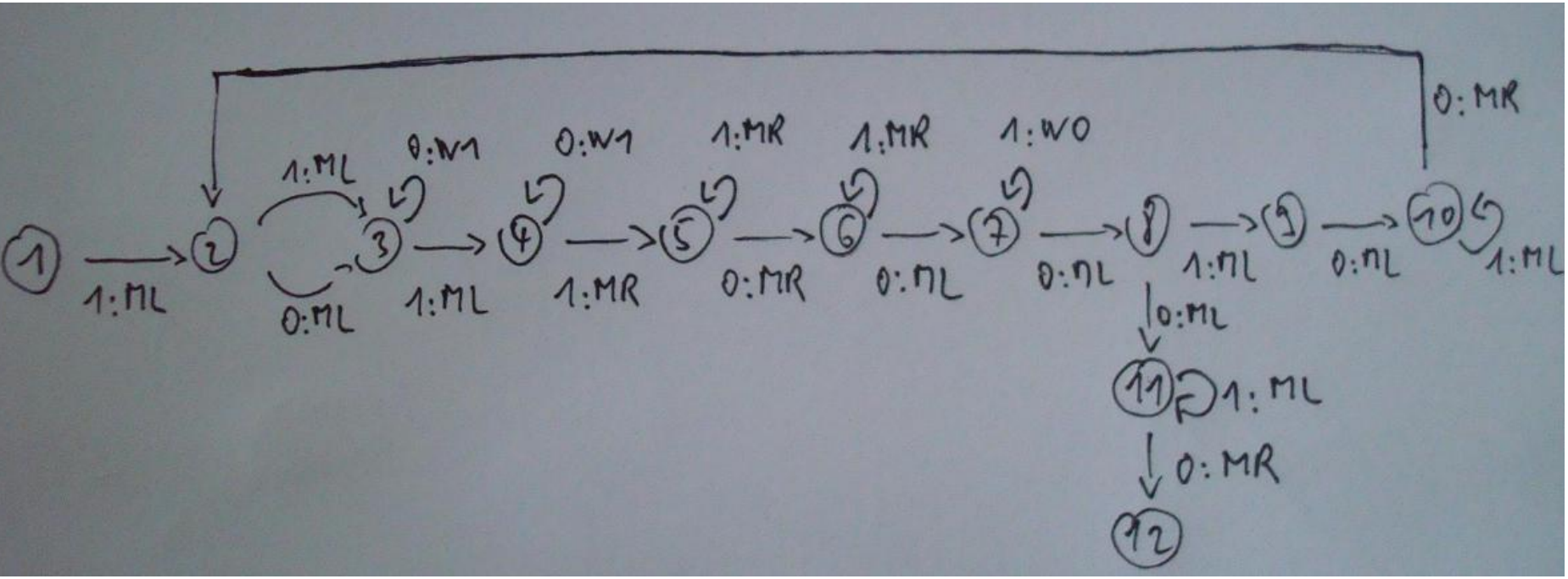
La machine à écrire trois "1" :

q_1	Si 0, exécuter W1 et passser à l'état q_1 Si 1, exécuter ML et à l'état q_2
q_2	Si 0, exécuter W1 et passser à l'état q_2 Si 1, exécuter ML et passer à l'état q_3
q_3	Si 0, exécuter W1 et passer à l'état q_3



Exemple de machine (2)

La machine à multiplier par deux :



Limites de la calculabilité

Fait

Il existe des fonctions qui ne sont pas calculables par machines de Turing.

Limites de la calculabilité

Fait

Il existe des fonctions qui ne sont pas calculables par machines de Turing.

Exemples de problèmes indécidables :

- Est-ce qu'une certaine machine de Turing va s'arrêter ?
- Est-ce qu'une certaine équation diophantienne a une solution ?
- Est-ce qu'un certain énoncé arithmétique est vrai ?

Programme WHILE (1)

On se donne des variables $X_0, ..., X_n$ qui prennent leur valeur dans les entiers naturels.

La valeur initiale de X_0 (et éventuellement X_1...) correspond à l'input du programme, sa valeur finale à son output.

Un programme peut réassigner des valeurs aux variables, et l'on dispose également de deux types d'instructions pour avoir des boucles.

```
type intarray = array of int;
var target:intarray, k:int;

func partition(A:intarray, p:int, r:int) : int
      var x:int, i:int, j:int, temp:int;
      x = A[r];
      i = p - 1;
      j = p;
      while(j < r) do {
            if(A[j] <= x) then {
                  i = i + 1;
                  temp = A[i];
                  A[i] = A[j];
                  A[j] = temp;
            }
            j = j + 1;
      }
      temp = A[i+1];
      A[i+1] = A[r];
      A[r] = temp;
      return i + 1;
end partition;
```

Programme WHILE (2)

Le langage de programmation contient cinq types d'instructions :

- $X_i \leftarrow 0$, 'clear' : cette commande assigne la valeur 0 à la variable X_i.
- $X_i \leftarrow X_i + 1$, 'increment' : cette commande augmente de 1 la valeur assignée à X_i.
- $X_i \leftarrow X_j$, 'copy' : cette commande assigne à X_i la valeur de X_j et laisse la valeur de X_j inchangée.

Programme WHILE (3)

LOOP X_n

- P

ENDLOOP X_n

Exécute le programme P
k fois où k est
la valeur **initiale** de X_n

P peut changer la valeur de X_n mais cela ne change pas le nombre d'itérations de la boucle

WHILE $X_n \neq 0$

- P

ENDWHILE $X_n \neq 0$

Exécute le programme P
tant que la valeur **actuelle**
de X_n est différente de 0

Un programme qui multiplie par 2 :

Un programme qui multiplie par 2 :

LOOP X_0
$X_0 \leftarrow X_0 + 1$
ENDLOOP X_0

Un programme qui multiplie par 2 :

LOOP X_0
$X_0 \leftarrow X_0 + 1$
ENDLOOP X_0

Un programme qui soustrait 1 si $X_0 > 0$ et rend 0 sinon :

Un programme qui multiplie par 2 :

LOOP X_0
$X_0 \leftarrow X_0 + 1$
ENDLOOP X_0

Un programme qui soustrait 1 si $X_0 > 0$ et rend 0 sinon :

$X_1 \leftarrow X_1 + 1$
LOOP X_0
$X_2 \leftarrow X_2 + 1$
WHILE $X_1 \neq 0$
$X_2 \leftarrow 0$
$X_1 \leftarrow 0$
ENDWHILE $X_1 \neq 0$
ENDLOOP X_0
$X_0 \leftarrow X_2$

Equivalence

Fait

Une fonction f est calculable est par une machine de Turing
ssi
elle est calculable par un programme WHILE

On dispose également de résultats d'équivalences pour les autres modèles de calcul évoqués.

Machine de Turing et calculabilité

Mais ces modèles rendent-ils compte de notre notion intuitive de calcul ?

Thèse de Turing :

Une fonction est calculable
ssi
elle est calculable par une machine de Turing.

Perspectives

- Les limitations de la calculabilité sont-elles des limitations pour nous ou seulement des limitations pour les / certaines machines ?
- L'esprit peut-il être assimilé à une machine de Turing / un ordinateur ?

Perspectives

- Les limitations de la calculabilité sont-elles des limitations pour nous ou seulement des limitations pour les / certaines machines ?
- L'esprit peut-il être assimilé à une machine de Turing / un ordinateur ?
- Quel puissance de calcul est nécessaire à l'esprit pour quel type de processus de calcul ?

Perspectives

- Les limitations de la calculabilité sont-elles des limitations pour nous ou seulement des limitations pour les / certaines machines ?
- L'esprit peut-il être assimilé à une machine de Turing / un ordinateur ?
- Quel puissance de calcul est nécessaire à l'esprit pour quel type de processus de calcul ?
- Quelles sont les propriétés cognitivement pertinentes des procédures de calcul ?