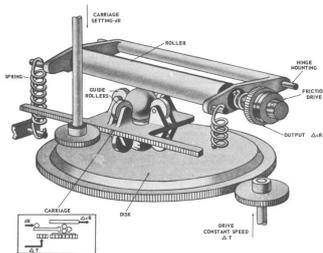


## Cours 4: Incomplétude. Machine de Turing.



Olivier Bournez  
bournez@lix.polytechnique.fr

INF412

Ecole Polytechnique  
CSC\_INF41012\_EP

1

Exprimez vous!!



Page du cours.



Commentaires, avis  
sur les cours et les PCs.

### ■ Séances

- 4: mercredi semaine prochaine: repas avec les délégués.
- 5: mercredi dans deux semaines: PC notée.

- Exprimez des commentaires, avis sur les cours et les PCs:  
email à [bournez@lix.polytechnique.fr](mailto:bournez@lix.polytechnique.fr), ou  
[www.enseignement.polytechnique.fr/informatique/INF412/AVIS](http://www.enseignement.polytechnique.fr/informatique/INF412/AVIS).

2

## Théorème de complétude

- On sait construire un (des) système(s) de déduction qui est valide et complet.
  - ▶ Rappel :  $\mathcal{T} \vdash F$  pour " $F$  se prouve à partir de  $\mathcal{T}$ " dans ce système.
  - ▶ Notons :  $\mathcal{T} \models F$  pour "tout modèle de  $\mathcal{T}$  est un modèle de  $F$ ."
- C'est-à-dire :

### Théorème (Validité)

Soit  $\mathcal{T}$  une théorie. Soit  $F$  une formule close.  
Si  $\mathcal{T} \vdash F$  alors  $\mathcal{T} \models F$ .

### Théorème (Complétude)

Soit  $\mathcal{T}$  une théorie. Soit  $F$  une formule close.  
Si  $\mathcal{T} \models F$  alors  $\mathcal{T} \vdash F$ .

3

Autre façon de comprendre ce qu'on obtient :

- prouvabilité et conséquence (sémantique) sont les mêmes notions.

$$\mathcal{T} \vdash F \text{ si et seulement si } \mathcal{T} \models F.$$

4

Autre façon de le comprendre :

$F$  est prouvable ssi  $F$  est vraie dans tous les modèles

- $F$  est prouvable à partir des axiomes  $\mathcal{T}$  ssi  $F$  est vraie dans tous les modèles de  $\mathcal{T}$ .
- Applications : graphes, groupes, corps, ...

5

## Comment est-ce possible ?

- 3 formulations équivalentes du théorème de Complétude.
  1. Pour toute formule  $F$ ,  $\mathcal{T} \models F$  implique  $\mathcal{T} \vdash F$ .
  2. Pour toute formule  $F$ ,  $F$  n'est pas prouvable à partir de  $\mathcal{T}$  implique que  $\mathcal{T} \cup \{\neg F\}$  possède un modèle.
  3. Si  $\mathcal{T}$  est cohérente, alors  $\mathcal{T}$  possède un modèle.
- (autrement dit : puisque consistant implique cohérent, cohérence et consistance sont des synonymes).

6

## Comment est-ce possible ?

- Tour de la force de la preuve :
  - ▶ On prouve 3. : **Si  $\mathcal{T}$  est cohérente, alors  $\mathcal{T}$  possède un modèle.**
  - ▶ Tour de force et idée de la preuve : construire un modèle
    - Son ensemble de base (domaine) est l'ensemble  $M$  des termes clos sur la signature  $\Sigma$  de la théorie.
    - Et la preuve "construit" une façon d'interpréter les relations pour que cela soit bien un modèle.

7

- Du coup, être cohérent est un synonyme d'être consistant.
- Et comme ce que l'on construit dans la preuve est un modèle qui reste dénombrable (= son ensemble de base est dénombrable) si la signature l'est :

### Théorème (Löwenheim-Skolem)

*Si  $\mathcal{T}$  une théorie sur une signature dénombrable possède un modèle, alors elle possède un modèle dont l'ensemble de base est dénombrable.*

8

## Corps réels clos

- Application du théorème de Löwenheim-Skolem :
  - ▶ il existe des corps réels clos dénombrables.
    - (exemple : les réels algébriques  $\overline{\mathbb{Q}} \cap \mathbb{R}$ ).

9

- Une preuve fait intervenir un nombre fini d'axiomes.
- Et donc :

### Théorème (Compacité)

- ▶ Soit  $\mathcal{T}$  une théorie.
- ▶ Une formule  $F$  est une conséquence de  $\mathcal{T}$  si et seulement si  $F$  est une conséquence d'une partie finie de  $\mathcal{T}$ .
- ▶  $\mathcal{T}$  possède un modèle si et seulement si toute partie finie de  $\mathcal{T}$  possède un modèle.

10

## Axiomes de l'arithmétique

- Considérons une signature constituée du symbole de constante 0, d'une fonction unaire  $s$ , et de deux fonctions binaires  $+$  et  $*$ , et de la relation binaire  $=$ .
- On souhaite que  $(\mathbb{N}, 0, s, +, *, =)$  en soit un modèle.

11

## 1ère tentative

- On considère les axiomes

$$\forall x \neg s(x) = 0 \quad (1)$$

$$\forall x \forall y (s(x) = s(y) \Rightarrow x = y) \quad (2)$$

$$\forall x (x = 0 \vee \exists y s(y) = x) \quad (3)$$

$$\forall x 0 + x = x \quad (4)$$

$$\forall x s(x) + y = s(x + y) \quad (5)$$

$$\forall x 0 * x = 0 \quad (6)$$

$$\forall x s(x) * y = x * y + y \quad (7)$$

12

## Conséquence/Non-conséquence

- Rappel : une formule  $F$  est dite une **conséquence** d'un ensemble de formules  $\mathcal{T}$  si tout modèle de la théorie  $\mathcal{T}$  est un modèle de  $F$ .
  - ▶ se note :  $\mathcal{T} \models F$
- Comment se persuader qu'une formule close n'est pas une conséquence de  $\mathcal{T}$  ?
  - ▶ en exhibant un modèle de  $\mathcal{T}$  qui n'est pas un modèle de  $F$ .

13

## Faits

- Pour chaque entier  $n$  et  $m$ , la formule

$$s^n(0) + s^m(0) = s^{n+m}(0),$$

est une conséquence des axiomes précédents, où  $s^n(0)$  est  $s(s(\dots s(0)))$  avec  $s$  répété  $n$  fois.

- Mais

$$\forall x \forall y \ x + y = y + x$$

n'est pas une conséquence de ces axiomes.

14

## Un modèle où l'addition n'est pas commutative

- Soit  $X$  un ensemble avec au moins deux éléments.
- On considère la structure  $\mathfrak{M}$  dont l'ensemble de base est

$$M = \mathbb{N} \cup (X \times \mathbb{Z}),$$

et où les symboles  $s, +, *, =$  sont interprétés par les conditions suivantes :

- ▶  $=$  est interprété par l'égalité.  $s, +, *$  étendent les fonctions correspondantes sur  $\mathbb{N}$ ;
  - ▶ pour  $a = (x, n)$  :
    - $s(a) = (x, n+1)$ ;
    - $a + m = m + a = (x, n+m)$ ;
    - $a * m = (x, n * m)$  si  $m \neq 0$ , et  $(x, n) * 0 = 0$ ;
    - $m * a = (x, m * n)$ ;
  - ▶ pour  $a = (x, n)$  et  $b = (y, m)$  :
    - $(x, n) + (y, m) = (x, n+m)$ ;
    - $(x, n) * (y, m) = (x, n * m)$ .
- Ce modèle est un modèle des axiomes précédents.
  - L'addition n'y est pas commutative.

15

## Idée des axiomes de Peano

- ajouter une famille (un schéma) d'axiomes pour permettre les preuves par récurrence.
- cette fois l'addition sera bien nécessairement commutative, i.e.

$$\forall x \forall y \ x + y = y + x$$

sera bien toujours satisfaite.

- ▶ et on capture en pratique toutes les propriétés de l'arithmétique.

16

### Axiomes de Peano

■ Les axiomes de l'arithmétique de Peano sont :

▶ les axiomes + ceux de l'égalité ;

$$\forall x \neg s(x) = 0 \quad (8)$$

$$\forall x \forall y (s(x) = s(y) \Rightarrow x = y) \quad (9)$$

$$\forall x (x = 0 \vee \exists y s(y) = x) \quad (10)$$

$$\forall x 0 + x = x \quad (11)$$

$$\forall x s(x) + y = s(x + y) \quad (12)$$

$$\forall x 0 * x = 0 \quad (13)$$

$$\forall x s(x) * y = x * y + y \quad (14)$$

▶ et l'ensemble de toutes les formules de la forme

$$\forall x_1 \dots \forall x_n \left( (F(0, x_1, \dots, x_n) \wedge \forall x_0 (F(x_0, x_1, \dots, x_n) \Rightarrow F(s(x_0), x_1, \dots, x_n))) \Rightarrow \forall x_0 F(x_0, x_1, \dots, x_n) \right)$$

où  $n$  est n'importe quel entier et  $F(x_0, \dots, x_n)$  est n'importe quelle formule de variables libres  $x_0, \dots, x_n$ .

17

■ On note  $Th(\mathbb{N})$  l'ensemble des formules closes  $F$  qui sont vraies sur les entiers.

#### Théorème (Incomplétude)

*Il existe des formules closes de  $Th(\mathbb{N})$  qui ne sont pas prouvables, à partir des axiomes de Peano, ou de toute axiomatisation "raisonnable"<sup>1</sup> des entiers.*

#### Théorème (Second théorème d'incomplétude)

*La cohérence de l'arithmétique (ou sa négation) est un exemple de telle formule.*

1. Formellement, "récurivement énumérable" : voir suite du cours.

18

### Entiers non-standard

■ Il y a d'autres modèles que  $\mathbb{N}$  des axiomes de Peano.

▶ il existe des modèles **non-standard** de l'arithmétique.

19

### Entiers non-standard

■ Il y a d'autres modèles que  $\mathbb{N}$  des axiomes de Peano :

▶ considérons un nouveau symbole de constante  $c$ .

▶ considérons  $\mathcal{T}$  défini comme l'union des axiomes de Peano et de toutes les formules  $\neg c = s^n(0)$ , pour  $n$  un entier.

▶ Tout sous-ensemble fini de  $\mathcal{T}$  admet un modèle, car il est inclus dans l'union des axiomes de Peano et des formules  $\neg c = s^n(0)$  pour  $1 \leq n \leq N$  pour un certain entier  $N$  :

- il suffit d'interpréter  $c$  par  $N+1$ .

▶ Donc  $\mathcal{T}$  admet un modèle.

■ Ce modèle contient donc un élément qui n'est pas un entier standard.

20

⋮  
Analyse non-standard

...  $f$  est continue si et seulement si pour tout  $y$  infiniment petit et pour tout réel  $x$  standard,  $f(x+y) - f(x)$  est infiniment petit ...

⋮

21

## Deux autres utilisations de la logique

- Comment prouver qu'un axiome  $F$  est "indépendant" d'une théorie  $\mathcal{T}$  ?
  - ▶ On utilise un modèle de  $\mathcal{T}$  pour construire un modèle de  $\mathcal{T} \cup \{F\}$ , et un modèle de  $\mathcal{T} \cup \{\neg F\}$ .
  - ▶ On prouve en fait la **cohérence relative** : si  $\mathcal{T}$  cohérent, alors  $\mathcal{T} \cup \{F\}$  cohérent.
  
- Définissabilité :
  - ▶ On se donne un graphe (i.e. modèle d'une théorie sur une signature contenant un symbole de relation  $R$  binaire).
  - ▶ Peut-on définir :
    - la relation "être accessible en deux coups" par une formule ?
    - la relation "être accessible par un chemin" (i.e. dans la même composante connexe) ?
  - ▶ ... applications aux requêtes dans les bases de données ...

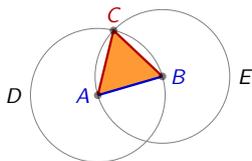
22

## Objectif

- Objectif de ce cours : répondre à la question suivante :

Qu'est-ce qu'un algorithme ?

**Exemple d'algorithme :**  
 Pour construire un *triangle équilatéral* ayant pour côté  $AB$  : tracer le cercle de centre  $A$  de rayon  $AB$  ; tracer le cercle de centre  $B$  de rayon  $AB$ . Nommer  $C$  l'une des intersections de ces deux cercles. Le triangle  $ABC$  est la solution recherchée.



23

## Thèse de Church

- Historiquement, plusieurs modèles :
  - ▶ Fonctions récursives, Kurt Gödel, 1931-34.
  - ▶  $\lambda$ -calcul, Alonzo Church, 1936.
  - ▶ Machines de Turing, Alan Turing, 1936.
  - ▶ Systèmes de Post,
  - ▶ Machines RAM,
  - ▶ Programmes JAVA, C, CAML, ...
  - ▶ ...
  
- Il s'avère que ces modèles, très différents, ont exactement la même puissance
  - ▶ **Thèse de Church/Turing** : une fonction est *calculable* si et seulement si elle est calculable par machine de Turing.
    - (exprimée clairement pour la première fois par Stephen Kleene, étudiant en thèse d'Alonzo Church.)

24

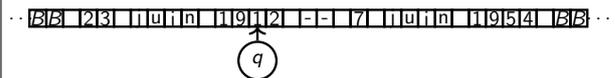
## Parenthèse historique



- Théorie de la relativité : 1907/1915
- Théorème d'incomplétude : 1931

25

## Alan M. Turing



- Machines de Turing : 1936

26

## Parenthèse historique

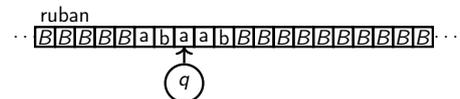
- Ces modèles sont nés des suites du théorème d'incomplétude de Kurt Gödel :
- Et en fait de la question suivante "*Entscheidungsproblem*" :

Peut-on décider mécaniquement  
si un énoncé est démontrable ou non ?

- Il s'avère que préciser "mécaniquement", c'est formaliser la notion d'algorithme.

27

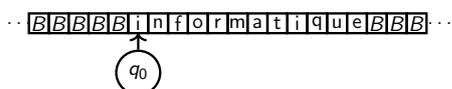
## Machine de Turing



- Une machine de Turing (déterministe) est composée des éléments suivants :
  - ▶ une mémoire infinie sous forme de ruban, divisé en cases ;
  - ▶ chaque case peut contenir un élément d'un alphabet fini fixé ;
  - ▶ une tête de lecture qui se déplace sur le ruban ;
  - ▶ un programme donné par une **fonction de transition** qui pour chaque état de la machine  $q$ , parmi un nombre fini d'états possibles  $Q$ , précise selon le symbole sous la tête de lecture :
    1. l'état suivant  $q' \in Q$  ;
    2. le nouvel élément de  $\Sigma$  à écrire à la place de l'élément de  $\Sigma$  sous la tête de lecture ;
    3. un sens de déplacement  $\leftarrow$ ,  $|$ , ou  $\rightarrow$  pour la tête de lecture.

28

## Exécution d'une machine de Turing sur un mot : configuration initiale



- Initialement :
  - ▶ l'entrée se trouve sur le ruban, et la tête de lecture est positionnée sur la première lettre du mot.
  - ▶ Les cases des rubans qui ne correspondent pas à l'entrée contiennent toutes l'élément  $B$  (symbole de blanc), qui est une lettre particulière.
  - ▶ La machine est positionnée dans son état initial  $q_0$ .

29

## Exécution d'une machine de Turing sur un mot : configurations suivantes

- A chaque étape de l'exécution :
  - ▶ la machine, selon son état, lit le symbole se trouvant sous la tête de lecture, et selon ce symbole :
    - remplace le symbole sous la tête de lecture par celui précisé par sa fonction transition ;
    - déplace possiblement cette tête de lecture d'une case vers la droite ou vers la gauche suivant le sens précisé par la fonction de transition ;
    - change d'état vers l'état suivant.
- Le mot  $w$  est dit accepté lorsque l'exécution de la machine finit par atteindre son état d'acceptation  $q_a$ .

30

## Machine de Turing

- Une machine de Turing est un 8-uplet

$$M = (Q, \Sigma, \Gamma, B, \delta, q_0, q_a, q_r)$$

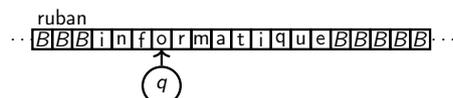
où :

1.  $Q$  est l'ensemble fini des états ;
2.  $\Sigma$  est un alphabet fini ;
3.  $\Gamma$  est l'alphabet de travail fini :  $\Sigma \subset \Gamma$  ;
4.  $B \in \Gamma$ ,  $B \notin \Sigma$  est le caractère blanc ;
5.  $q_0 \in Q$  est l'état initial ;
6.  $q_a \in Q$  est l'état d'acceptation ;
7.  $q_r \in Q$  est l'état de refus (ou d'arrêt) ;
8.  $\delta$  est la fonction de transition :  $\delta$  est une fonction (possiblement partielle) de  $Q \times \Gamma$  dans  $Q \times \Gamma \times \{\leftarrow, |, \rightarrow\}$ . Le symbole  $\leftarrow$  est utilisé pour signifier un déplacement vers la gauche,  $\rightarrow$  un déplacement vers la droite, et  $|$  aucun déplacement.

31

## Configuration

- Une configuration est donnée par la description du ruban, par la position de la tête de lecture/écriture, et par l'état interne.



- ▶ Notation 1 :  $(q, \text{fni}, \text{ormatique})$ 
  - Cas général :  $C = (q, u, v)$ , avec  $u, v \in \Gamma^*$ ,  $q \in Q$  :  $u$  et  $v$  désignent le contenu respectivement à gauche et à droite de la tête de lecture du ruban, la tête de lecture du ruban étant sur la première lettre de  $v$ . On suppose que les dernières lettres de  $u$  et de  $v$  ne sont pas  $B$ , qu'à droite et à gauche de  $u$  et  $v$  sur le ruban il n'y a que des  $B$  et que  $v$  est écrit de gauche à droite, et  $u$  de droite à gauche.
- ▶ Notation 2 :  $\text{infqormatique}$ .
  - Cas général :  $uq_v$ , en gardant  $u$  et  $v$  écrit de gauche à droite.

32

### Dérivations entre configurations

- Une configuration est dite **acceptante** si  $q = q_a$ , **refusante** si  $q = q_r$ .
- Pour  $w \in \Sigma^*$ , la configuration initiale correspondant à  $w$  est la configuration  $C[w] = (q_0, \epsilon, w)$ .
- On note :  $C \vdash C'$  si la configuration  $C'$  est le successeur direct de la configuration  $C$  par le programme  $\delta$  de la machine de Turing.
  - ▶ Formellement, si  $C = (q, u, v)$  et si  $a$  désigne la première lettre<sup>2</sup> de  $v$ , et si  $\delta(q, a) = (q', a', m')$  alors  $C \vdash C'$  si
    - $C' = (q', u', v')$ , et
    - si  $m' = \rightarrow$ ,  $u' = a'u$ , et  $v'$  est obtenu en supprimant la première lettre  $a$  de  $v$ .
    - si  $m' = \leftarrow$ ,  $u'$  est obtenu en supprimant la première lettre  $a''$  de  $u$ , et  $v'$  est obtenu en concaténant  $a''$  et le résultat du remplacement de la première lettre  $a$  de  $v$  par  $a'$ .
    - si  $m' = |$ ,  $u' = u$ , et  $v'$  est obtenu en remplaçant la première lettre  $a$  de  $v$  par  $a'$ .

2. Avec la convention que la première lettre du mot vide est le blanc  $B$ .

33

### Calcul par une machine de Turing

- Un mot  $w \in \Sigma^*$  est dit **accepté** (en temps  $t$ ) par la machine de Turing, s'il existe une suite de configurations  $C_1, \dots, C_t$  avec :
  1.  $C_0 = C[w]$ ;
  2.  $C_i \vdash C_{i+1}$  pour tout  $i < t$ ;
  3. aucune configuration  $C_i$  pour  $i < t$  n'est acceptante ou refusante.
  4.  $C_t$  est acceptante.
- Un mot  $w \in \Sigma^*$  est dit **refusé** (en temps  $t$ ) par la machine de Turing, s'il existe une suite de configurations  $C_1, \dots, C_t$  avec :
  1.  $C_0 = C[w]$ ;
  2.  $C_i \vdash C_{i+1}$  pour tout  $i < t$ ;
  3. aucune configuration  $C_i$  pour  $i < t$  n'est acceptante ou refusante.
  4.  $C_t$  est refusante.
- On dit que la machine de Turing **s'arrête** sur un mot  $w$ , si  $w$  est accepté, ou refusé, **boucle** dans le cas contraire (= n'est ni accepté, ni refusé).

34

### Calcul par une machine de Turing

- Le langage  $L \subset \Sigma^*$  **accepté par  $M$** , noté  $L(M)$  est l'ensemble des mots  $w$  qui sont acceptés par la machine.
- On dit qu'un langage  $L \subset \Sigma^*$  est **décidé par  $\Sigma$**  par la machine si :
  - ▶ pour  $w \in L$ ,  $w$  est accepté par la machine;
  - ▶ pour  $w \notin L$  (=sinon),  $w$  est refusé par la machine.
  - Autrement dit, la machine accepte  $L$  et termine sur toute entrée.

35

### Premier programme

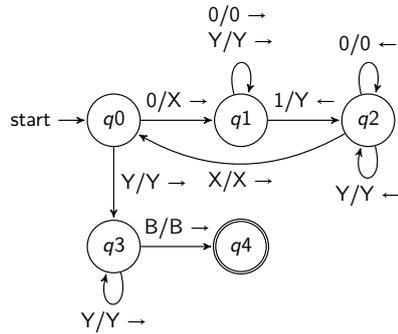
- Construire une machine de Turing qui accepte exactement les mots  $w$  sur l'alphabet  $\Sigma = \{0, 1\}$  de la forme  $0^n 1^n$ ,  $n \in \mathbb{N}, n > 0$ .
  - ▶ Voici une solution :
    - On considère une machine avec  $Q = \{q_0, q_1, q_2, q_3, q_4\}$ ,  $\Gamma = \{0, 1, X, Y, B\}$ ,
    - l'état d'acceptation  $q_4$  et une fonction de transition  $\delta$  telle que :
      - $\delta(q_0, 0) = (q_1, X, \rightarrow)$ ;
      - $\delta(q_0, Y) = (q_3, Y, \rightarrow)$ ;
      - $\delta(q_1, 0) = (q_1, 0, \rightarrow)$ ;
      - $\delta(q_1, 1) = (q_2, Y, \rightarrow)$ ;
      - $\delta(q_1, Y) = (q_1, Y, \rightarrow)$ ;
      - $\delta(q_2, 0) = (q_2, 0, \rightarrow)$ ;
      - $\delta(q_2, X) = (q_0, X, \rightarrow)$ ;
      - $\delta(q_2, Y) = (q_2, Y, \rightarrow)$ ;
      - $\delta(q_3, Y) = (q_3, Y, \rightarrow)$ ;
      - $\delta(q_3, B) = (q_4, B, \rightarrow)$ .

36

## Représentation graphique

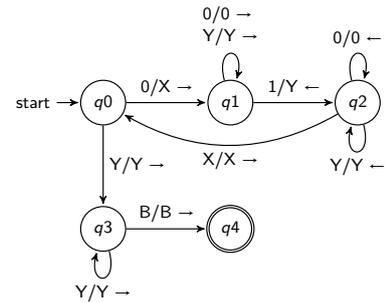
### Convention graphique :

- les sommets du graphe représentent les états de la machine;
- on représente chaque transition  $\delta(q, a) = (q', a', m)$  par un arc de l'état  $q$  vers l'état  $q'$  étiqueté par  $a/a' m$ ;
- l'état initial est marqué par une flèche entrante;
- l'état d'acceptation est marqué par un double cercle.



37

## Comment fonctionne-t-il ?



38

### Principe :

- Le ruban reste de la forme  $X^*0^*Y^*1^*$ .
- A chaque fois que l'on lit un 0, on le remplace par un X, et on rentre dans l'état  $q1$ , ce qui correspond à lancer la sous-procédure suivante :
  - on se déplace à droite tant que l'on lit un 0 ou un Y ;
  - dès qu'on a atteint un 1, on le transforme en un Y, et on revient à gauche jusqu'à revenir sur un X (le X qu'on avait écrit) et s'être déplacé d'une case vers la droite.
  - En faisant ainsi, pour chaque 0 effacé (i.e. X marqué), on aura effacé un 1 (i.e. marqué un Y).
- Si on a marqué tous les 0 et que l'on atteint un Y, on rentre dans l'état  $q3$ ,
  - ce qui a pour objet de vérifier que ce qui est à droite est bien constitué uniquement de Y.
- Lorsqu'on a tout lu, c-à-d atteint un B, on accepte, c'est-à-dire on va dans l'état  $q4$ .

39

## Soustraction en unaire

- Construire un programme de machine de Turing qui réalise une soustraction en unaire : partant d'un mot de la forme  $0^m10^n$ , la machine s'arrête avec  $0^{m \ominus n}$  sur son ruban (entouré de blancs), où  $m \ominus n$  est  $\max(0, m - n)$ .

» Détails

40

## Exercices

- Construire une machine de Turing qui ajoute (respectivement : soustrait) 1 au nombre écrit en binaire sur son ruban.
- Construire une machine de Turing qui renverse son entrée : partant de  $w = a_1 a_2 \dots a_n$ , la machine s'arrête avec le mot  $w^R = a_n \dots a_2 a_1$ .
- Construire une machine de Turing qui partant d'un mot  $w \# w_1 \# w_2$ , avec  $w, w_1, w_2$  écrits sur l'alphabet  $\{0, 1\}$ , le remplace par  $w_1 \# w \# w_2$ .
- ...
- voir la PC de demain.
- ...

41

## Thèse de Church

- Thèse de Church:

Calculable dans un sens intuitif  
correspond à  
calculable par machine de Turing

42

## ANNEXES

### Technique 1 :

Utiliser l'état interne pour stocker une information finie.

- Exercice 1.

- ▶ Construire un programme de machine de Turing qui lit le symbole en face de la tête de lecture et vérifie que ce dernier n'apparaît nulle part ailleurs à droite, sauf sur la toute dernière lettre à droite.

► Details

## Technique 2 : Utiliser des sous-procédures

- Exercice 2 : Multiplication en unaire.
  - ▶ Construire un programme de machine de Turing qui réalise une multiplication en unaire : partant d'un mot de la forme  $0^m 10^n$ , la machine s'arrête avec  $0^{m \cdot n}$  sur son ruban.

[→ Details](#)

## Robustesse du modèle

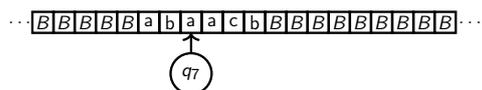
- Le modèle de la machine de Turing est extrêmement robuste.
- On peut en effet envisager de nombreuses variantes autour du concept de machine de Turing,
  - ▶ mais aucune de ces variantes ne change ce que l'on arrive à programmer avec ces machines.

## Restriction à un alphabet binaire

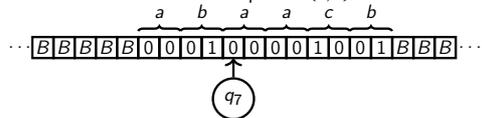
- Toute machine de Turing qui travaille sur un alphabet  $\Sigma$  quelconque peut être simulée par une machine de Turing qui travaille sur un alphabet  $\Sigma$  avec uniquement deux lettres, avec  $\Gamma = \Sigma \cup \{B\}$ .

- Idée de la démonstration :

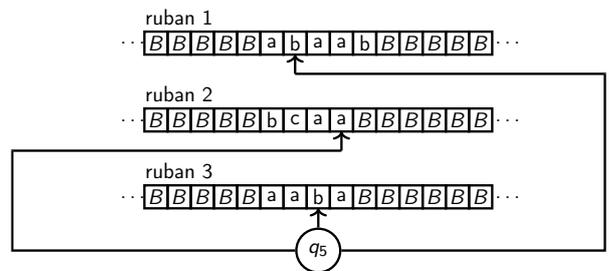
- ▶ Machine  $M$  sur l'alphabet  $\{a, b, c\}$



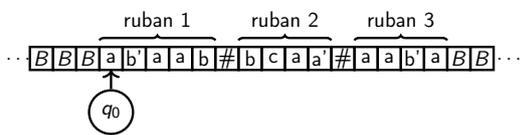
- ▶ Machine  $M'$  simulant  $M$  sur l'alphabet  $\{0, 1\}$ .



## Machines de Turing à plusieurs rubans



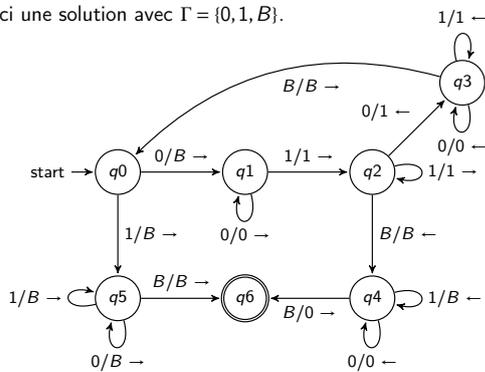
- Toute machine de Turing qui travaille avec  $k$  rubans peut être simulée par une machine de Turing avec un unique ruban.
- Idée de la démonstration :



- Le modèle de la machine de Turing peut paraître extrêmement rudimentaire.
- Il n'en demeure pas moins extrêmement puissant, et capable de capturer la notion de *calculable* en informatique.

### Soustraction

- Voici une solution avec  $\Gamma = \{0, 1, B\}$ .



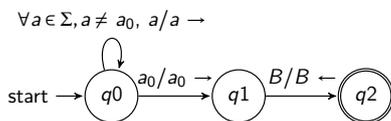
- Principe :

[Retour](#)

- ▶ La machine recherche le 0 le plus à gauche et le remplace par un blanc. Elle cherche alors à droite un 1, quand elle en trouve un elle continue à droite jusqu'à trouver un 0 qu'elle remplace par un 1. La machine retourne alors à gauche pour trouver le 0 le plus à gauche qu'elle identifie en trouvant le premier blanc en se déplaçant à gauche et en se déplaçant depuis ce blanc d'une case vers la droite.
- ▶ On répète le processus jusqu'à ce que :
  - soit en cherchant à droite un 0, on rencontre un blanc. Alors les  $n$  0 dans  $0^m 10^n$  ont été changés en 1 et  $n+1$  des  $m$  0 ont été changés en  $B$ . Dans ce cas, la machine remplace les  $n+1$  1 par un 0 et  $n$  blancs, ce qui laisse  $m-n$  0 sur le ruban. Puisque dans ce cas,  $m \geq n$ ,  $m \ominus n = m - n$ .
  - ou en recommençant le cycle, la machine n'arrive pas à trouver un 0 à changer en blanc, puisque les  $m$  premiers 0 ont déjà été changés en  $B$ . Alors  $n \geq m$ , et donc  $m \ominus n = 0$ . La machine remplace alors tous les 1 et 0 restants par des blancs, et termine avec un ruban complètement blanc.

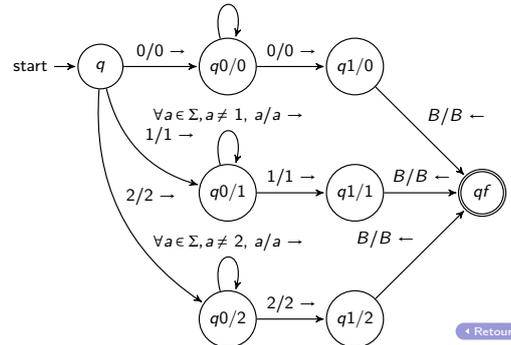
### Utiliser l'état interne pour stocker une information finie

- Si l'on fixe un symbole  $a_0 \in \Sigma$ , il est facile de construire un programme qui vérifie que le symbole  $a_0$  n'apparaît nulle part sauf sur la toute dernière lettre à droite.



- Pour résoudre notre problème, il suffit de lire la première lettre  $a_0$  et de recopier ce programme autant de fois qu'il y a de lettres dans  $\Sigma$ . Si  $\Sigma = \{0, 1, 2\}$  par exemple :

$$\forall a \in \Sigma, a \neq 0, a/a \rightarrow$$



- On utilise le fait que l'on travaille sur des états qui peuvent être des couples : ici des couples  $q_i/j$  avec  $i \in \{1, 2\}$ , et  $j \in \Sigma$ .

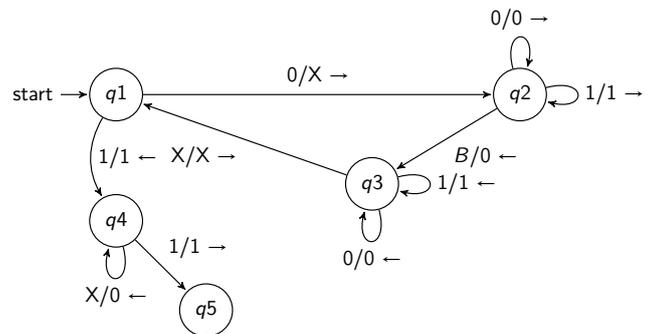
### Technique 2 : Utiliser des sous-procédures

- Voici une stratégie possible :

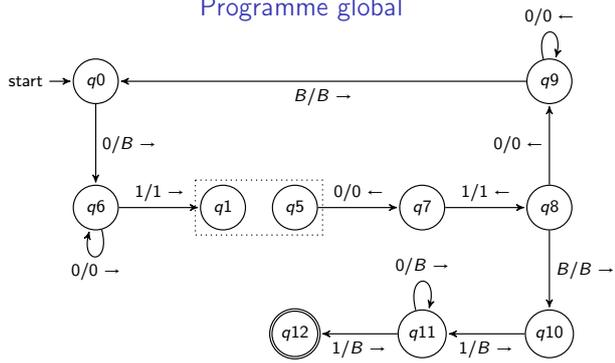
- le ruban contiendra un mot de la forme  $0^i 10^n 10^{kn}$  pour un certain entier  $k$  ;
- dans chaque étape, on change un 0 du premier groupe en un blanc, et on ajoute  $n$  0 au dernier groupe, pour obtenir une chaîne de la forme  $0^{i-1} 10^n 10^{(k+1)n}$  ;
- en faisant ainsi, on copie le groupe de  $n$  0  $m$  fois, une fois pour chaque symbole du premier groupe mis à blanc. Quand il ne reste plus de blanc dans le premier groupe de 0, il y aura donc  $m * n$  0 dans le dernier groupe ;
- la dernière étape est de changer le préfixe  $10^n 1$  en des blancs, et cela sera terminé.

### Implémenter l'étape 2

- Transformer une configuration  $0^{m-k} 1 q_5 0^n 10^{(k-1)n}$  en  $0^{m-k} 1 q_5 0^n 10^{kn}$ .



### Programme global



- On utilise le fait que l'on peut réaliser des sous-procédures en "collant" des programmes correspondant à des sous-procédures.

[← Retour](#)