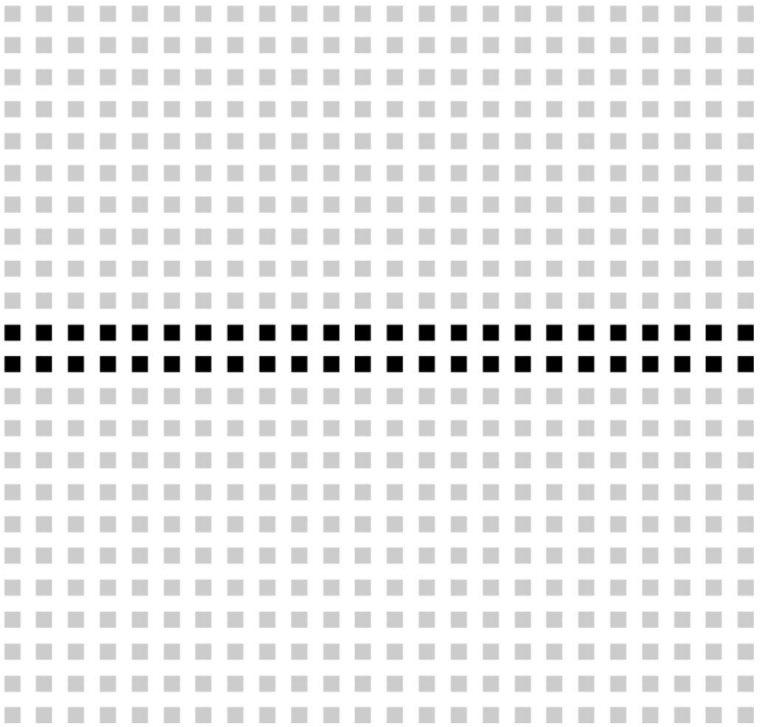
DEUXIÈME PARTIE



CALCULABILITÉ

THÉORIE

Copyright 2012 Cengage Learning. Tous droits réservés. Ne peut être copié, numérisé ou dupliqué, en tout ou en partie. En raison des droits électroniques, certains contenus tiers peuvent être supprimés du livre électronique et/ou des chapitres électroniques. La revue éditoriale a estimé que tout contenu supprimé n'affecte pas matériellement l'expérience d'apprentissage globale. Cengage Learning se réserve le droit de supprimer du contenu supplémentaire à tout moment si des restrictions de droits ultérieures l'exigent.

3

LA THÈSE DE L'ÉGLISE----TURING

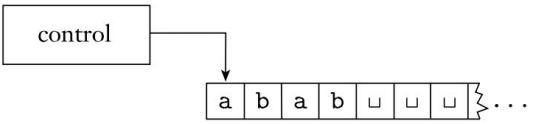
Jusqu'à présent, dans le cadre de notre développement de la théorie du calcul, nous avons présenté plusieurs modèles de dispositifs informatiques. Les automates finis sont de bons modèles pour les dispositifs qui ont une petite quantité de mémoire. Les automates à pile sont de bons modèles pour les dispositifs qui ont une mémoire illimitée qui n'est utilisable que dans le mode dernier entré, premier sorti d'une pile. Nous avons montré que certaines tâches très simples dépassent les capacités de ces modèles. Ils sont donc trop limités pour servir de modèles d'ordinateurs à usage général.

3.1 TURING MACHINES

Nous nous tournons maintenant vers un modèle beaucoup plus puissant, proposé pour la première fois par Alan Turing en 1936, appelé la machine de Turing. Semblable à un automate fini mais doté d'une mémoire illimitée et sans restriction, une machine de Turing est un modèle beaucoup plus précis d'un ordinateur à usage général. Une machine de Turing peut faire tout ce qu'un véritable ordinateur peut faire. Néanmoins, même une machine de Turing ne peut pas résoudre certains problèmes. Dans un sens très réel, ces problèmes dépassent les limites théoriques du calcul.

Le modèle de la machine de Turing utilise une bande infinie comme mémoire illimitée. Il possède une tête de bande qui peut lire et écrire des symboles et se déplacer sur la bande.

Au départ, la bande ne contient que la chaîne d'entrée et est vide partout ailleurs. Si la machine a besoin de stocker des informations, elle peut les écrire sur la bande. Pour lire les informations qu'elle a écrites, la machine peut déplacer sa tête dessus. La machine continue de calculer jusqu'à ce qu'elle décide de produire une sortie. Les sorties d'acceptation et de rejet sont obtenues en entrant dans des états d'acceptation et de rejet désignés. Si elle n'entre pas dans un état d'acceptation ou de rejet, elle continuera indéfiniment, sans jamais s'arrêter.



CHIFFRE 3.1

Schéma d'une machine de Turing

La liste suivante résume les différences entre les automates finis et Machines de Turing.

1. Une machine de Turing peut à la fois écrire sur la bande et la lire.
2. La tête de lecture-écriture peut se déplacer aussi bien vers la gauche que vers la droite.
3. La bande est infinie.
4. Les conditions spéciales de rejet et d'acceptation prennent effet immédiatement.

Introduisons une machine de Turing M1 pour tester l'appartenance au langage B = {w#w| w {0,1} }. Nous voulons que M1 accepte si son entrée est un membre de B et la rejette dans le cas contraire. Pour mieux comprendre M1 , mettez-vous à sa place en imaginant que vous vous trouvez sur une entrée d'un kilomètre de long composée de millions de caractères. Votre objectif est de déterminer si l'entrée est un membre de B, c'est-à-dire si l'entrée comprend deux chaînes identiques séparées par un symbole #. L'entrée est trop longue pour que vous puissiez vous en souvenir dans son intégralité, mais vous êtes autorisé à vous déplacer d'avant en arrière sur l'entrée et à y faire des marques. La stratégie évidente consiste à zigzaguer jusqu'aux emplacements correspondants des deux côtés du # et à déterminer s'ils correspondent. Placez des marques sur la bande pour garder une trace des emplacements correspondants.

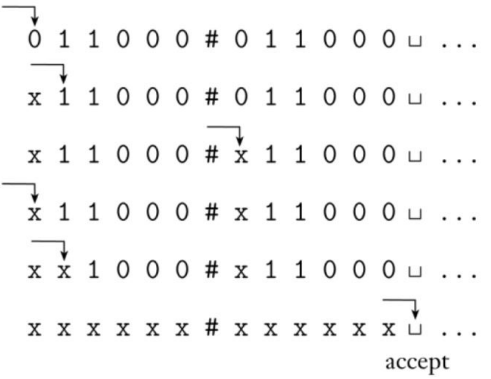
Nous avons conçu M1 pour fonctionner de cette manière. Il effectue plusieurs passages sur la chaîne d'entrée avec la tête de lecture-écriture. À chaque passage, il fait correspondre l'un des caractères de chaque côté du symbole #. Pour garder une trace des symboles qui ont déjà été vérifiés, M1 raye chaque symbole au fur et à mesure de son examen. S'il raye tous les symboles, cela signifie que tout correspond correctement, et M1 passe dans un état d'acceptation. S'il découvre une non-concordance, il entre dans un état de rejet. En résumé, l'algorithme de M1 est le suivant.

M1 = « Sur la chaîne d'entrée w :

1. Zigzaguez sur la bande jusqu'aux positions correspondantes de chaque côté du symbole # pour vérifier si ces positions contiennent le même symbole. Si ce n'est pas le cas, ou si aucun # n'est trouvé, rejetez. Rayez les symboles au fur et à mesure qu'ils sont vérifiés pour garder une trace des symboles correspondants.

2. Lorsque tous les symboles à gauche du # ont été barrés, vérifiez s'il reste des symboles à droite du #. S'il reste des symboles, rejetez-les ; sinon, acceptez-les.

La figure suivante contient plusieurs instantanés non consécutifs de la bande de M1 après son démarrage sur l'entrée 011000#011000.



CHIFFRE 3.2

Instantanés de la machine de Turing M1 calculant sur l'entrée 011000#011000

Cette description de la machine de Turing M1 esquisse son fonctionnement mais ne donne pas tous ses détails. Nous pouvons décrire les machines de Turing de manière très détaillée en donnant des descriptions formelles analogues à celles introduites pour les automates finis et à pile. Les descriptions formelles précisent chacune des parties de la définition formelle du modèle de machine de Turing qui sera présentée sous peu. En réalité, nous ne donnons presque jamais de descriptions formelles des machines de Turing car elles ont tendance à être très grandes.

DÉFINITION FORMELLE D'UNE MACHINE DE TURING

Le cœur de la définition d'une machine de Turing est la fonction de transition δ car elle nous indique comment la machine passe d'une étape à la suivante. Pour une machine de Turing, δ prend la forme : $Q\times\Gamma \longrightarrow Q\times\Gamma\times\{L,R\}$. C'est-à-dire que lorsque la machine est dans un certain état q et que la tête est au-dessus d'un carré de ruban contenant un symbole a, et si $\delta(q,a) = (r,b,L)$,

la machine écrit le symbole b en remplacement du a, et passe à l'état r. Le troisième élément est soit L soit R et indique si la tête se déplace vers la gauche ou vers la droite après l'écriture. Dans ce cas, le L indique un déplacement vers la gauche.

DÉFINITION

Une machine de Turing est un 7-uplet, (Q,Σ,Γ,δ,q0,qaccept,qreject), où Q, Σ, Γ sont tous des ensembles finis et

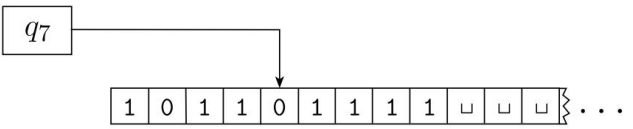
1. Q est l'ensemble des états,
2. Σ est l'alphabet d'entrée ne contenant pas le symbole vide ,
3. Γ est l'alphabet de la bande, où Γ et Σ Γ,
4. d : Q × Γ−→Q × Γ × {L,R} est la fonction de transition,
5. q0 Q est l'état de départ,
6. qaccept Q est l'état d'acceptation, et
7. qreject Q est l'état de rejet, où qreject 6= qaccept.

3.3

Une machine de Turing M = (Q,Σ,Γ,δ,q0,qaccept,qreject) calcule comme suit. Initialement, M reçoit son entrée w = w1w2 ...wn Σ sur les n cases les plus à gauche de la bande, et le reste de la bande est vide (c'est-à-dire rempli de symboles vides). La tête commence sur la case la plus à gauche de la bande. Notez que Σ ne contient pas le symbole vide, donc le premier blanc apparaissant sur la bande marque la fin de l'entrée. Une fois que M a commencé, le calcul se poursuit selon les règles décrites par la transition

fonction. Si M essaie un jour de déplacer sa tête vers la gauche de l'extrémité gauche de la bande, la tête reste au même endroit pour ce déplacement, même si la fonction de transition indique L. Le calcul continue jusqu'à ce qu'il entre dans l'état d'acceptation ou de rejet, auquel cas il s'arrête. Si aucun des deux ne se produit, M continue indéfiniment.

Lorsqu'une machine de Turing effectue des calculs, des changements se produisent dans l'état actuel, le contenu actuel de la bande et l'emplacement actuel de la tête. Un réglage de ces trois éléments est appelé une configuration de la machine de Turing. Les configurations sont souvent représentées d'une manière spéciale. Pour un état q et deux chaînes u et v sur l'alphabet de bande Γ, nous écrivons uqv pour la configuration où l'état actuel est q, le contenu actuel de la bande est uv et l'emplacement actuel de la tête est le premier symbole de v. La bande ne contient que des espaces après le dernier symbole de v. Par exemple, 1011q701111 représente la configuration lorsque la bande est 101101111, l'état actuel est q7 et la tête est actuellement sur le deuxième 0. La figure 3.4 décrit une machine de Turing avec cette configuration.



CHIFFRE 3.4

Une machine de Turing avec la configuration 1011q701111

Nous formalisons ici notre compréhension intuitive de la manière dont une machine de Turing calcule. Supposons que la configuration C1 donne la configuration C2 si la machine de Turing peut légalement passer de C1 à C2 en une seule étape. Nous définissons formellement cette notion comme suit.

Supposons que nous ayons a, b et c dans Γ, ainsi que u et v dans Γ et les états qi et qj. Dans ce cas, uaqi bv et uqj acv sont deux configurations. Disons que

rendements de uaqi bv uqj acv

si dans la fonction de transition δ(qi,b) = (qj,c,L). Cela gère le cas où la machine de Turing se déplace vers la gauche. Pour un déplacement vers la droite, disons que

uaqi bv donne uacqj v

si δ(qi,b) = (qj,c,R).

Des cas particuliers se produisent lorsque la tête se trouve à l'une des extrémités de la configuration. Pour l'extrémité gauche, la configuration qi bv donne qj cv si la transition se déplace vers la gauche (car nous empêchons la machine de sortir de l'extrémité gauche de la bande), et elle donne cqjv pour la transition se déplaçant vers la droite. Pour l'extrémité droite, la configuration uaqi est équivalente à uaqi car nous supposons que les blancs suivent la

partie de la bande représentée dans la configuration. On peut donc traiter ce cas comme précédemment, la tête n'étant plus à l'extrémité droite.

La configuration de départ de M sur l'entrée w est la configuration q0 w, qui indique que la machine est dans l'état de départ q0 avec sa tête à l'extrême gauche de la bande. Dans une configuration d'acceptation, l'état de la configuration est qaccept. Dans une configuration de rejet, l'état de la configuration est qreject. Les configurations d'acceptation et de rejet sont des configurations d'arrêt et ne génèrent pas d'autres configurations. Étant donné que la machine est définie pour s'arrêter lorsqu'elle est dans les états qaccept et qreject, nous aurions pu définir de manière équivalente la fonction de transition pour avoir la forme plus compliquée $\delta : Q0 \times \Gamma \to Q \times \Gamma \times \{L,R\}$, où Q0 est Q sans qaccept et qreject. Une machine de Turing M accepte l'entrée w si une séquence de configurations C1, C2, ...

, Ck

existe, où

1. C1 est la configuration de départ de M sur l'entrée w,
2. chaque At donne At+1, et
3. Ck est une configuration acceptante.

La collection de chaînes que M accepte est le langage de M, ou le langage reconnu par M, noté L(M).

DÉFINITION 3.5

On appellera un langage reconnaissable par Turing si une machine de Turing le reconnaît.[11]

Lorsque nous démarrons une machine de Turing sur une entrée, trois résultats sont possibles. La machine peut accepter, rejeter ou boucler. Par boucle, nous entendons que la machine ne s'arrête tout simplement pas. La boucle peut impliquer n'importe quel comportement simple ou complexe qui ne conduit jamais à un état d'arrêt.

Une machine de Turing M peut ne pas accepter une entrée en entrant dans l' état qreject et en la rejetant, ou en effectuant une boucle. Il est parfois difficile de distinguer une machine qui effectue une boucle d'une autre qui prend simplement beaucoup de temps. Pour cette raison, nous préférons les machines de Turing qui s'arrêtent sur toutes les entrées ; de telles machines ne font jamais de boucle. Ces machines sont appelées des décideurs car elles prennent toujours une décision d'acceptation ou de rejet. On dit également qu'un décideur qui reconnaît une langue décide de cette langue.

DÉFINITION 3.6

[11] On l'appelle langage récursivement énumérable dans certains autres manuels.

On peut dire qu'un langage est décidable par Turing ou simplement décidable si une machine de Turing le décide.[12]

Ensuite, nous donnons des exemples de langages décidables. Tout langage décidable est reconnaissable par Turing. Nous présentons des exemples de langages qui sont reconnaissables par Turing mais non décidables après avoir développé une technique pour prouver l'indécidabilité au chapitre 4.

EXEMPLES DE MACHINES À TOURNER

Comme nous l'avons fait pour les automates finis et à pile, nous pouvons décrire formellement une machine de Turing particulière en spécifiant chacune de ses sept parties. Cependant, aller à ce niveau de détail peut être fastidieux pour toutes les machines de Turing, sauf les plus petites. Par conséquent, nous ne consacrerons pas beaucoup de temps à donner de telles descriptions. La plupart du temps, nous ne donnerons que des descriptions de niveau supérieur, car elles sont suffisamment précises pour nos besoins et sont beaucoup plus faciles à comprendre. Néanmoins, il est important de se rappeler que chaque description de niveau supérieur n'est en fait qu'un raccourci pour son homologue formel. Avec de la patience et du soin, nous pourrions décrire n'importe laquelle des machines de Turing de ce livre dans les moindres détails formels.

Pour vous aider à établir le lien entre les descriptions formelles et les descriptions de niveau supérieur, nous donnons des diagrammes d'état dans les deux exemples suivants. Vous pouvez les ignorer si vous vous sentez déjà à l'aise avec ce lien.

EXEMPLE 3.7

Nous décrivons ici une machine de Turing (TM) M2 qui décide que $A = \{02^n | n \geq 0\}$, la langage constitué de toutes les chaînes de 0 dont la longueur est une puissance de 2.

M2 = « Sur la chaîne d'entrée w :

1. Balayez de gauche à droite sur la bande, en barrant chaque autre 0.
2. Si à l'étape 1 la bande contenait un seul 0, acceptez.
3. Si à l'étape 1 la bande contenait plus d'un seul 0 et que le nombre de 0 était impair, rejetez-la.

[12] On l'appelle langage récursif dans certains autres manuels.

4. Remettez la tête à l'extrémité gauche de la bande.
5. Passez à l'étape 1. »

Chaque itération de l'étape 1 réduit de moitié le nombre de 0. Lorsque la machine parcourt la bande à l'étape 1, elle garde une trace du nombre de 0 observés, pair ou impair. Si ce nombre est impair et supérieur à 1, le nombre initial de 0 dans l'entrée ne peut pas être une puissance de 2. Par conséquent, la machine rejette dans ce cas. Cependant, si le nombre de 0 observés est de 1, le nombre initial doit être une puissance de 2. Dans ce cas, la machine accepte.

Nous donnons maintenant la description formelle de M2 = (Q,Σ,Γ,δ,q1,qaccept,qreject) :

• Q = {q1,q2,q3,q4,q5,qaccepter,qrejeter},

• Σ = {0}, et

• Γ = {0,x, }.

• Nous décrivons δ avec un diagramme d'état (voir Figure 3.8).

• Les états de démarrage, d'acceptation et de rejet sont respectivement q1, qaccept et qreject .

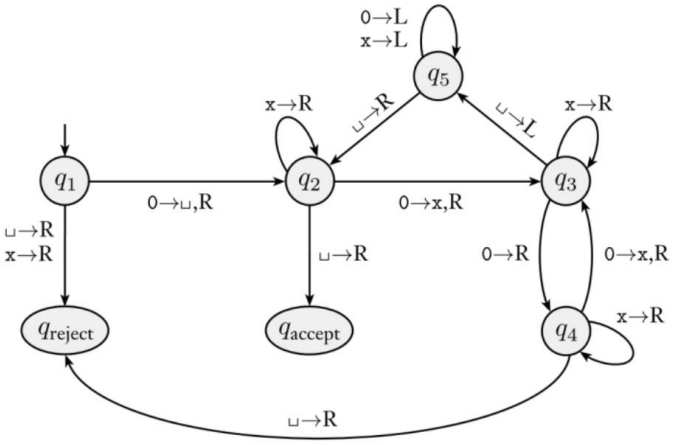


FIGURE 3.8

Diagramme d'état de la machine de Turing M2

Dans ce diagramme d'état, l'étiquette 0→,R apparaît sur la transition de q1 à q2. Cette étiquette signifie que lorsque l'état q1 est atteint avec la tête indiquant 0, la machine se déplace pour indiquer q2, écrit , et déplace la tête vers la droite. En d'autres termes, δ(q1,0) = (q2, ,R). Pour plus de clarté, nous utilisons l'abréviation 0→R dans la transition de q3 à q4, pour signifier que la machine se déplace vers la droite lors de la lecture de 0 dans l'état q3 mais ne modifie pas la bande, donc δ(q3,0) = (q4,0,R).

Cette machine commence par écrire un symbole vide sur le 0 le plus à gauche de la bande afin de pouvoir trouver l'extrémité gauche de la bande à l'étape 4. Alors que nous utiliserions normalement un symbole plus suggestif tel que # pour le délimiteur d'extrémité gauche, nous utilisons ici un espace vide pour garder l'alphabet de la bande, et donc le diagramme d'état, petit.
L'exemple 3.11 donne une autre méthode pour trouver l'extrémité gauche de la bande.

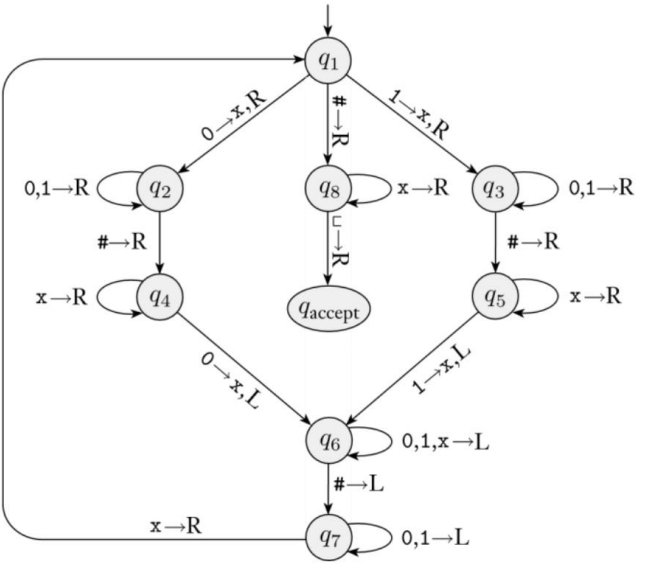
Nous donnons ensuite un exemple d'exécution de cette machine sur l'entrée 0000. La configuration de départ est q10000. La séquence de configurations saisies par la machine apparaît comme suit ; lisez les colonnes de haut en bas et de gauche à droite.

q10000	q5x0x	xq5xx
q2000	q5 x0x	q5xxx
xq300	q2x0x	q5 xxx
x0q40	xq20x	q2xxx
x0xq3	xxq3x	xq2xx
x0q5x	xxxq3	xxq2x
xq50x	xxq5x	xxxq2
		xxx qaccepter

EXEMPLE 3.9

Ce qui suit est une description formelle de M1 = (Q,Σ,Γ,δ,q1,qaccept,qreject), la machine de Turing que nous avons décrite de manière informelle (page 167) pour décider du langage B = {w#w| w {0,1} }.

- Q = {q1,...,q8,qaccepter,qrejeter},
- Σ = {0,1,#} et Γ = {0,1,#,x, }.
- Nous décrivons δ avec un diagramme d'état (voir la figure suivante).
- Les états de démarrage, d'acceptation et de rejet sont respectivement q1, qaccept et qreject .



CHIFFRE 3.10

Diagramme d'état de la machine de Turing M1

Dans la figure 3.10, qui représente le diagramme d'état de TM M1, vous trouverez l'étiquette 0,1→R sur la transition allant de q3 à elle-même. Cette étiquette signifie que la machine reste en q3 et se déplace vers la droite lorsqu'elle lit un 0 ou un 1 dans l'état q3. Cela ne change pas le symbole sur la bande.

L'étape 1 est implémentée par les états q1 à q7, et l'étape 2 par les états restants. Pour simplifier la figure, nous ne montrons pas l'état de rejet ni les transitions allant

EXEMPLE

vers l'état de rejet. Ces transitions se produisent implicitement chaque fois qu'un état ne dispose pas d'une transition sortante pour un symbole particulier. Ainsi, comme dans l'état q5 aucune flèche sortante avec un # n'est présente, si un # apparaît sous la tête lorsque la machine est dans l'état q5, elle passe à l'état qreject. Pour être complet, nous disons que la tête se déplace vers la droite dans chacune de ces transitions vers l'état de rejet. ■

3.11

Ici, un TM M3 effectue une opération d'arithmétique élémentaire. Il décide que le langage C = {aibjck | i × j = k et i,j,k ≥ 1}.

M3 = « Sur la chaîne d'entrée w :

1. Scannez l'entrée de gauche à droite pour déterminer si elle est membre de a + b + c + et rejetez-la si ce n'est pas le cas.
2. Remettez la tête à l'extrémité gauche de la bande.
3. Rayez un a et parcourez vers la droite jusqu'à ce qu'un ab apparaisse. Faites la navette entre les b et les c, en rayant un de chaque jusqu'à ce que tous les b aient disparu. Si tous les c ont été rayés et qu'il reste quelques b, rejetez-les.
4. Remettez les b barrés et répétez l'étape 3 s'il y a un autre a à barrer. Si tous les a ont été barrés, déterminez si tous les c ont également été barrés. Si oui, acceptez ; sinon, rejetez.

Examinons de plus près les quatre étapes de M3 . Au cours de la première étape, la machine fonctionne comme un automate fini. Aucune écriture n'est nécessaire car la tête se déplace de gauche à droite, en gardant une trace en utilisant ses états pour déterminer si l'entrée est sous la forme appropriée.

L'étape 2 semble tout aussi simple mais comporte une subtilité. Comment le TM peut-il trouver l'extrémité gauche de la bande d'entrée ? Trouver l'extrémité droite de l'entrée est facile car elle se termine par un symbole vide. Mais l'extrémité gauche n'a pas de terminateur au départ. Une technique qui permet à la machine de trouver l'extrémité gauche de la bande consiste à marquer le symbole le plus à gauche d'une manière ou d'une autre lorsque la machine démarre avec sa tête sur ce symbole. Ensuite, la machine peut balayer vers la gauche jusqu'à ce qu'elle trouve la marque lorsqu'elle veut réinitialiser sa tête à l'extrémité gauche. L'exemple 3.7 illustre cette technique ; un symbole vide marque l'extrémité gauche.

Une méthode plus délicate pour trouver l'extrémité gauche de la bande tire parti de la façon dont nous avons défini le modèle de la machine de Turing. Rappelons que si la machine essaie de déplacer sa tête au-delà de l'extrémité gauche de la bande, elle reste au même endroit. Nous pouvons utiliser cette fonctionnalité pour créer un détecteur d'extrémité gauche. Pour détecter si la tête est situé à l'extrémité gauche, la machine peut écrire un symbole spécial sur la position actuelle tout en enregistrant le symbole qu'elle a remplacé dans la commande. Elle peut ensuite tenter de déplacer la tête vers la gauche. Si elle est toujours au-dessus du symbole spécial, le déplacement vers la gauche n'a pas réussi, et donc la tête devait être à gauche

EXEMPLE

fin. Si au contraire il s'agit d'un symbole différent, il reste des symboles à gauche de cette position sur la bande. Avant d'aller plus loin, la machine doit s'assurer de rétablir le symbole modifié à l'original.

Les étapes 3 et 4 ont des implémentations simples et utilisent chacune plusieurs états.

3.12

Ici, un TM M4 résout ce qu'on appelle le problème de distinction des éléments. On lui donne une liste de chaînes sur {0,1} séparées par des # et sa tâche est d'accepter si toutes les chaînes sont différents. La langue est

$$E = \{\#x1\#x2\#\cdots\#xl \mid \text{chaque } xi \in \{0,1\} \text{ et } xi \neq xj \text{ pour chaque } i \neq j\}.$$

La machine M4 fonctionne en comparant x1 avec x2 via xl, puis en comparant x2 avec x3 via xl, et ainsi de suite. Une description informelle de la TM M4 qui décide de cette langue suit.

M4 = « Sur l'entrée w :

1. Placez une marque sur le symbole de ruban adhésif le plus à gauche. Si ce symbole est un blanc, acceptez. Si ce symbole est un #, passez à l'étape suivante. Sinon, rejetez.
2. Passez au numéro # suivant et placez une seconde marque dessus. Si aucun numéro n'est rencontré avant un symbole vide, seul x1 était présent, alors acceptez.
3. En zigzaguant, comparez les deux chaînes à droite des # marqués. Si elles sont égales, rejetez-les.
4. Déplacez la marque la plus à droite des deux jusqu'au symbole # suivant à droite. Si aucun symbole # n'est rencontré avant un symbole vide, déplacez la marque la plus à gauche jusqu'au symbole # suivant à sa droite et la marque la plus à droite jusqu'au symbole # suivant celui-ci. Cette fois, si aucun # n'est disponible pour la marque la plus à droite, toutes les chaînes ont été comparées, alors acceptez.
5. Passez à l'étape 3. »

Cette machine illustre la technique de marquage des symboles de bande. À l'étape 2, le La machine place une marque au-dessus d'un symbole, # dans ce cas. Dans l'implémentation réelle

- En fait, la machine a deux symboles différents, # et #, dans son alphabet à bande. Dire que la machine place une marque au-dessus d'un # signifie que la machine écrit
- le symbole # à cet endroit.

La suppression de la marque signifie que la machine écrit le

EXEMPLE

Symbole sans le point. En général, nous souhaitons placer des marques sur divers symboles de la bande. Pour ce faire, nous incluons simplement des versions de tous ces symboles de bande avec des points dans l'alphabet de la bande.

Nous concluons des exemples précédents que les langages décrits A, B, C et E sont décidables. Tous les langages décidables sont reconnaissables par Turing, donc ces langages sont également reconnaissables par Turing. Démontrer un langage reconnaissable par Turing mais indécidable est plus difficile. Nous le ferons au chapitre 4.

3.2 VARIANTES DE MACHINES A TOURNER

Il existe de nombreuses définitions alternatives des machines de Turing, y compris des versions à bandes multiples ou avec non-déterminisme. On les appelle des variantes du modèle de la machine de Turing. Le modèle original et ses variantes raisonnables ont tous la même puissance.
ils reconnaissent la même classe de langages. Dans cette section, nous décrivons certaines de ces variantes et les preuves d'équivalence en puissance. Nous appelons cette invariance à certains changements dans la définition robustesse. Les automates finis et les automates à pile sont des modèles assez robustes, mais les machines de Turing ont un degré de robustesse étonnant.

Pour illustrer la robustesse du modèle de la machine de Turing, faisons varier le type de fonction de transition autorisée. Dans notre définition, la fonction de transition force la tête à se déplacer vers la gauche ou vers la droite après chaque étape ; la tête ne peut pas simplement rester immobile. Supposons que nous ayons permis à la machine de Turing de rester immobile. La fonction de transition aurait alors la forme $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L,R,S\}$. Cette fonction pourrait-elle permettre aux machines de Turing de reconnaître des langues supplémentaires, augmentant ainsi la puissance du modèle ? Bien sûr que non, car nous pouvons convertir n'importe quelle machine de Turing dotée de la fonction « rester immobile » en une machine qui ne l'a pas. Nous le faisons en remplaçant chaque transition « rester immobile » par deux transitions : l'une qui se déplace vers la droite et la seconde qui revient vers la gauche.

Ce petit exemple contient la clé pour montrer l'équivalence des variantes de TM .
Pour montrer que deux modèles sont équivalents, il suffit de montrer que l'un peut simuler l'autre.

MACHINES TOURNANTES MULTIBANDES

Une machine de Turing multibande est comme une machine de Turing ordinaire avec plusieurs bandes. Chaque bande possède sa propre tête de lecture et d'écriture. Au départ, l'entrée apparaît sur la bande 1, et les autres commencent vides. La fonction de transition est modifiée pour permettre la lecture, l'écriture et le déplacement des têtes sur certaines ou toutes les bandes simultanément. Formellement, c'est

$$\delta: Q \times \Gamma^k \to Q \times \Gamma^k \times \{G,D,S\}^k,$$

où k est le nombre de bandes. L'expression

$$\delta(qi,a1,\ldots,ak) = (qj,b1,\ldots,bk,L,R,\ldots,L)$$

Cela signifie que si la machine est dans l'état qi et que les têtes 1 à k lisent les symboles a1 à ak, la machine passe à l'état qj, écrit les symboles b1 à bk et ordonne à chaque tête de se déplacer vers la gauche ou la droite, ou de rester immobile, comme spécifié.

Les machines de Turing multibandes semblent plus puissantes que les machines de Turing ordinaires, mais nous pouvons montrer qu'elles sont équivalentes en puissance. Rappelons que deux machines sont équivalentes si elles reconnaissent le même langage.

THÉORÈME 3.13

Chaque machine de Turing multibande possède une machine de Turing monobande équivalente.

PREUVE Nous montrons comment convertir un TM multibande M en un TM monobande équivalent S. L'idée clé est de montrer comment simuler M avec S.

Supposons que M possède k bandes. S simule alors l'effet de k bandes en stockant leurs informations sur sa bande unique. Il utilise le nouveau symbole # comme délimiteur pour séparer le contenu des différentes bandes. En plus du contenu de ces bandes, S doit garder une trace de l'emplacement des têtes. Il le fait en écrivant un symbole de bande avec un point au-dessus pour marquer l'emplacement où se trouverait la tête sur cette bande. Considérez-les comme des bandes et des têtes « virtuelles ». Comme précédemment, les symboles de bande « en pointillés » sont simplement de nouveaux symboles qui ont été ajoutés à l'alphabet des bandes. La figure suivante illustre comment une bande peut être utilisée pour représenter trois bandes.

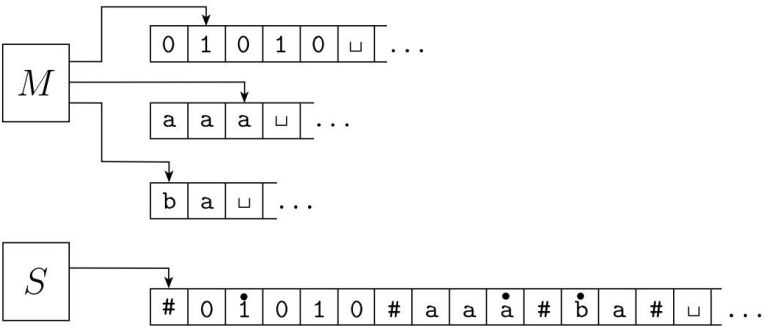


FIGURE 3.14
Représentant trois bandes avec une

S = « En entrée w = w1 ··· wn :

1. Tout d'abord, S met sa bande dans le format qui représente toutes les k bandes de M. La bande formatée contient

   #w•1w2 ··· wn # •# •# ··· #.

2. Pour simuler un seul mouvement, S scanne sa bande depuis le premier #, qui marque l'extrémité gauche, jusqu'au (k + 1)ème #, qui marque l'extrémité droite, afin de déterminer les symboles sous les têtes virtuelles. S fait ensuite un deuxième passage pour mettre à jour les bandes selon la manière dont la fonction de transition de M le dicte.

3. Si à un moment donné S déplace l'une des têtes virtuelles vers la droite sur un #, cette action signifie que M a déplacé la tête correspondante sur la partie vierge non lue précédemment de cette bande. S écrit donc un symbole vierge sur cette cellule de bande et décale le contenu de la bande, de cette cellule jusqu'au # le plus à droite, d'une unité vers la droite. Puis il continue la simulation comme avant.

COROLLAIRE 3.15

Un langage est reconnaissable par Turing si et seulement si une machine de Turing multibande le reconnaît.

PREUVE Un langage reconnaissable par Turing est reconnu par une bande ordinaire (single tape) Machine de Turing, qui est un cas particulier de machine de Turing à bandes multiples. Cela prouve une direction de ce corollaire. L'autre direction découle du théorème 3.13.

MACHINES DE TURING NON DÉTERMINISTES

Une machine de Turing non déterministe est définie de la manière attendue. À tout moment d'un calcul, la machine peut procéder selon plusieurs possibilités. La fonction de transition pour une machine de Turing non déterministe a la forme

$$d : Q \times \Gamma - \rightarrow P (Q \times \Gamma \times \{L, R\}).$$

Le calcul d'une machine de Turing non déterministe est un arbre dont les branches correspondent à différentes possibilités pour la machine. Si une branche du calcul conduit à l'état d'acceptation, la machine accepte son entrée. Si vous ressentez le besoin de revoir le non-déterminisme, reportez-vous à la section 1.2 (page 47). Nous montrons maintenant que le non-déterminisme n'affecte pas la puissance du modèle de la machine de Turing.

THÉORÈME 3.16

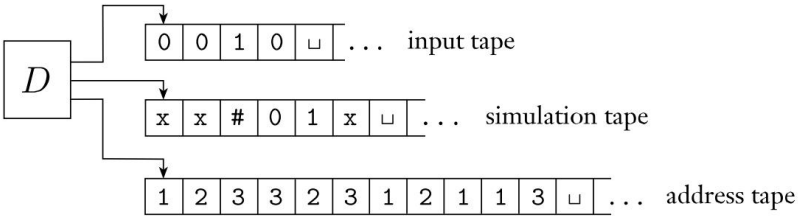
Chaque machine de Turing non déterministe possède une machine de Turing déterministe équivalente. machine.

IDÉE DE DÉMONSTRATION Nous pouvons simuler n'importe quelle TM non déterministe N avec une TM déterministe D. L'idée derrière la simulation est de faire essayer à D toutes les branches possibles du calcul non déterministe de N. Si D trouve un jour l'état d'acceptation sur l'une de ces branches, D accepte. Sinon, la simulation de D ne se terminera pas.

Nous considérons le calcul de N sur une entrée w comme un arbre. Chaque branche de l'arbre représente une des branches du non-déterminisme. Chaque nœud de l'arbre est une configuration de N. La racine de l'arbre est la configuration de départ. La TM D recherche

cet arbre pour une configuration acceptante. Il est crucial de mener cette recherche avec soin, de peur que D ne parvienne pas à visiter l'arbre entier. Une idée tentante, bien que mauvaise, est de faire explorer l'arbre par D en utilisant la recherche en profondeur. La stratégie de recherche en profondeur descend jusqu'à une branche avant de revenir en arrière pour explorer d'autres branches. Si D devait explorer l'arbre de cette manière, D pourrait descendre indéfiniment d'une branche infinie et manquer une configuration acceptante sur une autre branche. Par conséquent, nous concevons D pour explorer l' arbre
arbre en utilisant plutôt une recherche en largeur. Cette stratégie explore toutes les branches à la même profondeur avant de passer à l'exploration de n'importe quelle branche à la profondeur suivante. Cette méthode garantit que D visitera chaque nœud de l'arbre jusqu'à ce qu'il rencontre une configuration d'acceptation.

La machine déterministe simulatrice D possède trois bandes. D'après le théorème 3.13, cette disposition équivaut à n'avoir qu'une seule bande. La machine D utilise ses trois bandes d'une manière particulière, comme illustré dans la figure suivante. La bande 1 contient toujours la chaîne d'entrée et n'est jamais modifiée. La bande 2 conserve une copie de la bande de N sur une branche de son calcul non déterministe. La bande 3 garde une trace de l'emplacement de D dans l'arbre de calcul non déterministe de N.



CHIFFRE 3.17

TM D déterministe simulant TM N non déterministe

Considérons d'abord la représentation des données sur la bande 3. Chaque nœud de l'arbre peut avoir au plus b enfants, où b est la taille du plus grand ensemble de choix possibles donné par la fonction de transition de N. À chaque nœud de l'arbre, nous attribuons une adresse qui est une chaîne sur l'alphabet Γb = {1,2,...,b}. Nous attribuons l'adresse 231 au nœud auquel nous arrivons en commençant par la racine, en allant vers son 2e enfant, en allant vers le 3e enfant de ce nœud et enfin vers le 1er enfant de ce nœud. Chaque symbole de la chaîne nous indique quel choix faire ensuite lors de la simulation d'une étape dans une branche du calcul non déterministe de N. Parfois, un symbole peut ne correspondre à aucun choix si trop peu de choix sont disponibles pour une configuration. Dans ce cas, l'adresse est invalide et ne correspond à aucun nœud. La bande 3 contient une chaîne sur Γb. Elle représente la branche du calcul de N de la racine au nœud adressé par cette chaîne, sauf si l'adresse est invalide. La chaîne vide est l'adresse de la racine de l'arbre. Nous sommes maintenant prêts à décrire D.

1. Initialement, la bande 1 contient l'entrée w, et les bandes 2 et 3 sont vides.

2. Copiez la bande 1 sur la bande 2 et initialisez la chaîne sur la bande 3 pour qu'elle soit ε .
3. Utilisez la bande 2 pour simuler N avec l'entrée w sur une branche de son calcul non déterministe. Avant chaque étape de N, consultez le symbole suivant sur la bande 3 pour déterminer quel choix faire parmi ceux autorisés par la fonction de transition de N. S'il ne reste plus de symboles sur la bande 3 ou si ce choix non déterministe est invalide, abandonnez cette branche en passant à l'étape 4. Passez également à l'étape 4 si une configuration de rejet est rencontrée. Si une configuration d'acceptation est rencontrée, acceptez l'entrée.
4. Remplacez la chaîne de la bande 3 par la chaîne suivante dans l'ordre des chaînes. Simulez la branche suivante du calcul de N en passant à l'étape 2.

COROLLAIRE 3.18

Un langage est reconnaissable par Turing si et seulement si une machine de Turing non déterministe le reconnaît.

PREUVE Toute MT déterministe est automatiquement une MT non déterministe, et donc une direction de ce corollaire découle immédiatement. L'autre direction découle du théorème 3.16.

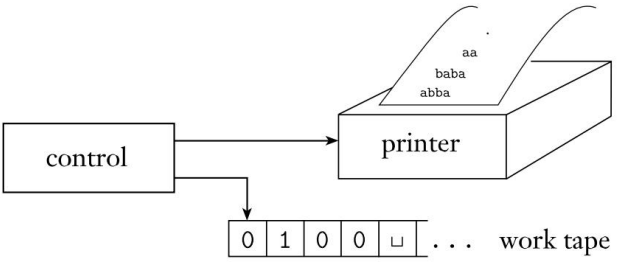
Nous pouvons modifier la preuve du théorème 3.16 de telle sorte que si N s'arrête toujours sur toutes les branches de son calcul, D s'arrêtera toujours. Nous appelons une machine de Turing non déterministe un décideur si toutes les branches s'arrêtent sur toutes les entrées. L'exercice 3.3 vous demande de modifier la preuve de cette manière pour obtenir le corollaire suivant du théorème 3.16.

COROLLAIRE 3.19

Un langage est décidable si et seulement si une machine de Turing non déterministe le décide.

RECENSEURS

Comme nous l'avons mentionné précédemment, certaines personnes utilisent le terme de langage récursivement énumérable pour désigner un langage reconnaissable par Turing. Ce terme provient d'un type de variante de la machine de Turing appelé énumérateur. Au sens large, un énumérateur est une machine de Turing avec une imprimante connectée. La machine de Turing peut utiliser cette imprimante comme périphérique de sortie pour imprimer des chaînes. Chaque fois que la machine de Turing souhaite ajouter une chaîne à la liste, elle envoie la chaîne à l'imprimante. L'exercice 3.4 vous demande de donner une définition formelle d'un énumérateur. La figure suivante illustre un schéma de ce modèle.



CHIFFRE 3.20
Schéma d'un recenseur

Un énumérateur E démarre avec une entrée vide sur sa bande de travail. Si l'énumérateur ne s'arrête pas, il peut imprimer une liste infinie de chaînes. Le langage énuméré par E est la collection de toutes les chaînes qu'il finit par imprimer. De plus, E peut générer les chaînes du langage dans n'importe quel ordre, éventuellement avec des répétitions. Nous sommes maintenant prêts à développer le lien entre les énumérateurs et les langages reconnaissables par Turing.

THÉORÈME 3.21

Une langue est reconnaissable par Turing si et seulement si un énumérateur l'énumère.

PREUVE Nous montrons d'abord que si nous avons un énumérateur E qui énumère une langue A, une MT M reconnaît A. La MT M fonctionne de la manière suivante.

M = « Sur l'entrée w :

1. Exécutez E. Chaque fois que E génère une chaîne, comparez-la avec w.
2. Si jamais w apparaît dans la sortie de E, acceptez.

De toute évidence, M accepte les chaînes qui apparaissent dans la liste de E.

Maintenant, nous faisons l'inverse. Si TM M reconnaît une langue A, nous pouvons construire l'énumérateur E suivant pour A. Supposons que s1,s2,s3,... est une liste de toutes les chaînes possibles dans Σ

E = « Ignorer l'entrée.

1. Répétez ce qui suit pour i = 1,2,3,... .
2. Exécutez M pendant i étapes sur chaque entrée, s1, s2,..., si.
3. Si des calculs l'acceptent, imprimez le sj correspondant. »

Si M accepte une chaîne particulière s, elle finira par apparaître sur la liste générée par E. En fait, elle apparaîtra sur la liste un nombre infini de fois car M s'exécute depuis le début sur chaque chaîne pour chaque répétition de l'étape 1. Cette procédure donne l'effet d'exécuter M en parallèle sur toutes les chaînes d'entrée possibles.

ÉQUIVALENCE AVEC D'AUTRES MODÈLES

Jusqu'à présent, nous avons présenté plusieurs variantes du modèle de la machine de Turing et avons montré qu'elles étaient équivalentes en termes de puissance. De nombreux autres modèles de calcul à usage général ont été proposés. Certains de ces modèles ressemblent beaucoup aux machines de Turing, mais d'autres sont très différents. Tous partagent la caractéristique essentielle des machines de Turing, à savoir un accès illimité à une mémoire illimitée, ce qui les distingue des modèles plus faibles tels que les automates finis et les automates à pile.

Il est remarquable de constater que tous les modèles dotés de cette fonctionnalité s'avèrent équivalents en termes de puissance, à condition qu'ils répondent à des exigences raisonnables.[13]

Pour comprendre ce phénomène, considérons la situation analogue des langages de programmation. Beaucoup d'entre eux, comme Pascal et LISP, sont très différents en termes de style et de structure. Un algorithme peut-il être programmé dans l'un d'eux et pas dans les autres ? Bien sûr que non : nous pouvons compiler LISP en Pascal et Pascal en LISP, ce qui signifie que les deux langages décrivent exactement la même classe d'algorithmes. Il en va de même pour tous les autres langages de programmation raisonnables. L'équivalence généralisée des modèles de calcul est valable précisément pour la même raison. Deux modèles de calcul qui satisfont à certaines exigences raisonnables peuvent se simuler l'un l'autre et sont donc équivalents en termes de puissance.

Ce phénomène d'équivalence a un corollaire philosophique important. Même si nous pouvons imaginer de nombreux modèles informatiques différents, la classe d'algorithmes qu'ils décrivent reste la même. Alors que chaque modèle informatique individuel a une certaine arbitraire dans sa définition, la classe sous-jacente d'algorithmes qu'il décrit est naturelle car les autres modèles aboutissent à la même classe unique. Ce phénomène a eu de profondes implications pour les mathématiques, comme nous le montrons dans la section suivante.

3.3

LA DÉFINITION DE L'ALGORITHME

Au sens courant, un algorithme est un ensemble d'instructions simples permettant d'exécuter une tâche. Courants dans la vie de tous les jours, les algorithmes sont parfois appelés procédures ou recettes. Les algorithmes jouent également un rôle important en mathématiques.
La littérature mathématique ancienne contient des descriptions d'algorithmes pour une variété de tâches, telles que la recherche de nombres premiers et de plus grands diviseurs communs. Dans les mathématiques contemporaines, les algorithmes abondent.

---

[13] Par exemple, l'une des exigences est la capacité d'effectuer seulement une quantité finie de travail en une seule étape.

Même si les algorithmes ont une longue histoire en mathématiques, la notion d'algorithme elle-même n'a été définie avec précision qu'au XXe siècle. Avant cela, les mathématiciens avaient une notion intuitive de ce qu'étaient les algorithmes et s'appuyaient sur cette notion pour les utiliser et les décrire. Mais cette notion intuitive était
insuffisante pour acquérir une compréhension plus approfondie des algorithmes. L'histoire suivante raconte comment la définition précise de l'algorithme s'est avérée cruciale pour résoudre un problème mathématique important.

LES PROBLEMES DE HILBERT

En 1900, le mathématicien David Hilbert prononce un discours aujourd'hui célèbre au Congrès international des mathématiciens à Paris. Dans sa conférence, il identifie 23 problèmes mathématiques et les présente comme un défi pour le siècle à venir. Le dixième problème de sa liste concerne les algorithmes.

Avant de décrire ce problème, discutons brièvement des polynômes. Un polynôme est une somme de termes, où chaque terme est un produit de certaines variables et d'un

constante, appelée coefficient. Par exemple,

$$6 \cdot x \cdot x \cdot x \cdot y \cdot z \cdot z = 6x\ 3yz2$$

est un terme de coefficient 6, et

$$6x\ 3yz2 + 3xy2 - x^{3-10}$$

est un polynôme à quatre termes, sur les variables x, y et z. Pour cette discussion, nous ne considérons que les coefficients qui sont des entiers. Une racine d'un polynôme est une affectation de valeurs à ses variables de telle sorte que la valeur du polynôme soit 0. Ce polynôme a une racine à x = 5, y = 3 et z = 0. Cette racine est une racine intégrale car toutes les variables sont affectées de valeurs entières. Certains polynômes ont une racine intégrale et d'autres non
pas.

Le dixième problème de Hilbert consistait à concevoir un algorithme qui teste si un polynôme possède une racine entière. Il n'a pas utilisé le terme algorithme mais plutôt « un processus selon lequel il peut être déterminé par un nombre fini d'opérations ».14
Il est intéressant de noter que dans la manière dont il a formulé ce problème, Hilbert a explicitement demandé qu'un algorithme soit « conçu ». Il a donc apparemment supposé qu'un tel algorithme devait exister – il suffisait que quelqu'un le trouve.

Comme nous le savons maintenant, il n'existe aucun algorithme pour cette tâche ; elle est algorithmiquement insoluble. Pour les mathématiciens de cette époque, il aurait été pratiquement impossible d'arriver à cette conclusion avec leur concept intuitif d'algorithme. Le concept intuitif aurait pu être suffisant pour donner des algorithmes pour certaines tâches, mais il était inutile pour montrer qu'aucun algorithme n'existe pour une tâche particulière. Prouver qu'un algorithme n'existe pas nécessite d'avoir une définition claire de l'algorithme. Il a fallu attendre cette définition pour progresser sur le dixième problème.

La définition est apparue dans les articles de 1936 d'Alonzo Church et Alan Turing. Church utilisait un système de notation appelé le λ -calcul pour définir les algorithmes. Turing l'a fait avec ses « machines ». Ces deux définitions se sont avérées équivalentes. Ce lien entre la notion informelle d'algorithme et sa définition précise est devenu la thèse de Church-Turing.

La thèse de Church-Turing fournit la définition de l'algorithme nécessaire pour résoudre le dixième problème de Hilbert. En 1970, Yuri Matijasevic, s'appuyant sur les travaux de Martin Davis, Hilary Putnam et Julia Robinson, a montré qu'il n'existe aucun algorithme permettant de tester si un polynôme possède des racines entières. Dans le chapitre 4, nous développons les techniques qui constituent la base pour prouver que ce problème et d'autres sont algorithmiquement insolubles.

14 Traduit de l'original allemand.

Notion intuitive des algorithmes	est égal à	Algorithmes de la machine de Turing

CHIFFRE 3.22

La thèse de Church-Turing

Formulons le dixième problème de Hilbert dans notre terminologie. Cela nous aide à introduire certains thèmes que nous explorerons dans les chapitres 4 et 5. Soit D = {p| p est un polynôme à racine entière}.

Le dixième problème de Hilbert demande essentiellement si l'ensemble D est décidable. La réponse est négative. En revanche, nous pouvons montrer que D est reconnaissable par Turing. Avant de le faire, considérons un problème plus simple. Il s'agit d'un analogue du dixième problème de Hilbert pour les polynômes qui n'ont qu'une seule variable, comme $4x^3 - 2x^2 + x - 7$. Soit

$$D_1 = \{p \mid p \text{ est un polynôme sur } x \text{ avec une racine entière}\}.$$

Voici une TM M1 qui reconnaît D1 :

M1 = « En entrée hpi : où p est un polynôme sur la variable x.

1. Évaluer p avec x fixé successivement aux valeurs 0, 1, −1, 2, −2, 3, −3, ... Si à un moment donné le polynôme est évalué à 0, acceptez.

Si p a une racine entière, M1 finira par la trouver et l'acceptera. Si p n'a pas de racine entière, M1 fonctionnera indéfiniment. Pour le cas multivariable, nous pouvons présenter une TM M similaire qui reconnaît D. Ici, M passe en revue tous les réglages possibles de ses variables à des valeurs entières.

M1 et M sont tous deux des reconnaisseurs mais pas des décideurs. Nous pouvons convertir M1 en décideur pour D1 car nous pouvons calculer les limites dans lesquelles les racines d'un polynôme à variable unique doivent se trouver et restreindre la recherche à ces limites. Dans le problème 3.21, on vous demande de montrer que les racines d'un tel polynôme doivent se trouver entre les valeurs

$$\pm k \frac{c_{max}}{c_1},$$

où k est le nombre de termes du polynôme, cmax est le coefficient ayant la plus grande valeur absolue et c1 est le coefficient du terme d'ordre le plus élevé. Si aucune racine n'est trouvée dans ces limites, la machine rejette. Le théorème de Matijasevic˘ montre que le calcul de telles limites pour les polynômes multivariables est impossible.

TERMINOLOGIE POUR DÉCRIRE LES MACHINES À TOURNER

Nous sommes arrivés à un tournant dans l'étude de la théorie du calcul. Nous continuons à parler des machines de Turing, mais notre véritable centre d'intérêt est désormais les algorithmes. Autrement dit, la machine de Turing ne sert que de modèle précis pour la

Définition de l'algorithme. Nous passons sous silence la théorie extensive des machines de Turing elles-mêmes et ne passons pas beaucoup de temps sur la programmation de bas niveau des machines de Turing. Il nous suffit d'être suffisamment à l'aise avec les machines de Turing pour croire qu'elles capturent tous les algorithmes.

Dans cet esprit, standardisons la façon dont nous décrivons les algorithmes de la machine de Turing. Dans un premier temps, nous nous demandons : quel est le bon niveau de détail à donner lors de la description

De tels algorithmes ? Les étudiants posent souvent cette question, en particulier lorsqu'ils préparent des solutions à des exercices et à des problèmes. Envisageons trois possibilités. La première est la description formelle qui énonce en détail les états de la machine de Turing, sa fonction de transition, etc. Il s'agit du niveau de description le plus bas et le plus détaillé. Le deuxième est un niveau de description plus élevé, appelé description d'implémentation, dans lequel nous utilisons de la prose anglaise pour décrire la manière dont la machine de Turing déplace sa tête et la manière dont elle stocke les données sur sa bande. À ce niveau, nous ne donnons pas de détails sur les états ou la fonction de transition. La troisième est la description de haut niveau, dans laquelle nous utilisons de la prose anglaise pour décrire un algorithme, en ignorant les détails d'implémentation. À ce niveau, nous n'avons pas besoin de mentionner comment la machine gère sa bande ou sa tête.

Dans ce chapitre, nous avons donné des descriptions formelles et de mise en œuvre de divers exemples de machines de Turing. La pratique avec des descriptions de machines de Turing de niveau inférieur vous aide à comprendre les machines de Turing et à gagner en confiance dans leur utilisation. Une fois que vous vous sentez à l'aise, des descriptions de haut niveau sont suffisantes.

Nous avons maintenant défini un format et une notation pour décrire les machines de Turing. L'entrée d'une machine de Turing est toujours une chaîne de caractères. Si nous voulons fournir un objet autre qu'une chaîne de caractères en entrée, nous devons d'abord représenter cet objet sous forme de chaîne de caractères. Les chaînes de caractères peuvent facilement représenter des polynômes, des graphes, des grammaires, des automates et toute combinaison de ces objets. Une machine de Turing peut être programmée pour décoder la représentation afin qu'elle puisse être interprétée de la manière que nous souhaitons. Notre notation pour l'encodage d'un objet O dans sa représentation sous forme de chaîne de caractères est hOi. Si nous avons plusieurs objets O1,O2,...,Ok, nous désignons leur encodage par une seule chaîne de caractères hO1,O2,...,Oki. L'encodage lui-même peut être effectué de plusieurs manières raisonnables. Peu importe celle que nous choisissons, car une machine de Turing peut toujours traduire un tel encodage en un autre.

Dans notre format, nous décrivons les algorithmes de la machine de Turing avec un segment de texte en retrait entre guillemets. Nous décomposons l'algorithme en étapes, chacune impliquant généralement de nombreuses étapes individuelles du calcul de la machine de Turing. Nous indiquons la structure en blocs de l'algorithme avec une indentation supplémentaire. La première ligne de l'algorithme décrit l'entrée de la machine. Si la description de l'entrée est simplement w, l'entrée est considérée comme une chaîne. Si la description de l'entrée est l'encodage d'un objet comme dans hAi, la machine de Turing teste d'abord implicitement si l'entrée encode correctement un objet de la forme souhaitée et la rejette si ce n'est pas le cas.

EXEMPLE 3.23 .....

Soit A le langage constitué de toutes les chaînes représentant des graphes non orientés qui sont connectés. Rappelons qu'un graphe est connecté si chaque nœud peut être atteint à partir de chaque autre nœud en parcourant les arêtes du graphe. On écrit

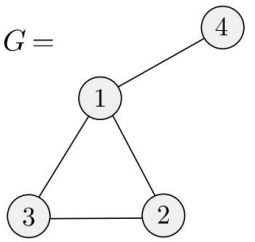
$$A = \{\langle G\rangle \mid G \text{ est un graphe non orienté connexe}\}.$$

Ce qui suit est une description de haut niveau d'une TM M qui décide de A.

M = « Sur l'entrée hGi, le codage d'un graphe G :

1. Sélectionnez le premier nœud de G et marquez-le.
2. Répétez l'étape suivante jusqu'à ce qu'aucun nouveau nœud ne soit marqué :
3. Pour chaque nœud de G, marquez-le s'il est attaché par une arête à un nœud déjà marqué.
4. Analysez tous les nœuds de G pour déterminer s'ils sont tous marqués. Si c'est le cas, acceptez ; sinon, rejetez.

Pour plus de pratique, examinons quelques détails au niveau de l'implémentation de la machine de Turing M. En général, nous ne donnerons pas ce niveau de détail à l'avenir et vous n'en aurez pas besoin non plus, sauf si on vous le demande spécifiquement dans un exercice. Tout d'abord, nous devons comprendre comment hGi encode le graphe G sous forme de chaîne. Considérons un codage qui est une liste des nœuds de G suivie d'une liste des arêtes de G. Chaque nœud est un nombre décimal et chaque arête est la paire de nombres décimaux qui représentent les nœuds aux deux extrémités de l'arête. La figure suivante illustre un tel graphe et son codage.



$\langle G\rangle =$

(1,2,3,4)((1,2),(2,3),(3,1),(1,4))

FIGURE **3.24**
Un graphe G et son codage hGi

Lorsque M reçoit l'entrée hGi, il vérifie d'abord si l'entrée est le codage correct d'un graphe. Pour ce faire, M scanne la bande pour s'assurer qu'il y a deux listes et qu'elles sont sous la forme appropriée. La première liste doit être une liste de nombres décimaux distincts, et la seconde doit être une liste de paires de nombres décimaux. Ensuite, M vérifie plusieurs choses. Tout d'abord, la liste des nœuds ne doit contenir aucune répétition ; et deuxièmement, chaque nœud apparaissant sur la liste des arêtes doit également apparaître sur la liste des nœuds. Pour le premier, nous pouvons utiliser la procédure donnée dans l'exemple 3.12 pour TM M4 qui vérifie la distinction des éléments. Une méthode similaire fonctionne pour le deuxième contrôle. Si le

l'entrée passe ces vérifications, c'est l'encodage d'un graphe G. Cette vérification termine la vérification de l'entrée, et M passe à l'étape 1.

Pour l'étape 1, M marque le premier nœud avec un point sur le chiffre le plus à gauche.

Pour l'étape 2, M parcourt la liste des nœuds pour trouver un nœud n1 non pointé et le signale en le marquant différemment, par exemple en soulignant le premier symbole. Puis M parcourt à nouveau la liste pour trouver un nœud n2 pointé et le souligne également.

Maintenant , M parcourt la liste des arêtes. Pour chaque arête, M teste si les deux nœuds soulignés n1 et n2 sont ceux qui apparaissent dans cette arête. Si c'est le cas, M met des points sur n1, supprime les soulignements et continue depuis le début de l'étape 2. Si ce n'est pas le cas, M vérifie l'arête suivante de la liste. S'il n'y a plus d'arêtes, {n1,n2} n'est pas une arête de G. Ensuite, M déplace le soulignement sur n2 vers le nœud en pointillé suivant et appelle maintenant ce nœud n2. Il répète les étapes de ce paragraphe pour vérifier, comme précédemment, si la nouvelle paire {n1,n2} est une arête. S'il n'y a plus de nœuds en pointillé, n1 n'est attaché à aucun nœud en pointillé.

Ensuite, M définit les soulignements de sorte que n1 soit le prochain nœud non pointé et n2 le premier nœud pointé et répète les étapes de ce paragraphe. S'il n'y a plus de nœuds non pointés, M n'a pas pu trouver de nouveaux nœuds à pointer, il passe donc à l'étape 4.

Pour l'étape 4, M analyse la liste des nœuds pour déterminer si tous sont en pointillés. Si c'est le cas, il passe à l'état d'acceptation ; sinon, il passe à l'état de rejet. Ceci termine la description de TM M.

EXERCICES

3.1 Cet exercice concerne la TM M2, dont la description et le diagramme d'état apparaissent dans l'exemple 3.7. Dans chacune des parties, donner la séquence de configurations dans laquelle M2 entre lorsqu'il est démarré sur la chaîne d'entrée indiquée.

un. 0.

À partir de 00.

environ 000.

d. 000000.

3.2 Cet exercice concerne la TM M1, dont la description et le diagramme d'état apparaissent dans l'exemple 3.9. Dans chacune des parties, donner la séquence de configurations dans laquelle M1 entre lorsqu'il est démarré sur la chaîne d'entrée indiquée.

Aa. 11.

b. 1#1.

c. 1##1.

d. 10#11.

et. 10#10.

A3.3 Modifier la preuve du théorème 3.16 pour obtenir le corollaire 3.19, montrant qu'un langage est décidable ssi une machine de Turing non déterministe le décide. (Vous pouvez supposer que

théorème suivant sur les arbres. Si chaque nœud d'un arbre a un nombre fini d'enfants et que chaque branche de l'arbre a un nombre fini de nœuds, l'arbre lui-même a un nombre fini de nœuds.)

3.4 Donnez une définition formelle d'un énumérateur. Considérez-le comme un type de machine de Turing à deux bandes qui utilise sa deuxième bande comme imprimante. Incluez une définition du langage énuméré.

A3.5 Examiner la définition formelle d'une machine de Turing pour répondre aux questions suivantes, et expliquez votre raisonnement.

a. Une machine de Turing peut-elle un jour écrire le symbole vide sur sa bande ?

b. L'alphabet de bande Γ peut-il être le même que l'alphabet d'entrée Σ ?

c. La tête d'une machine de Turing peut-elle se trouver au même endroit pendant deux mesures?

d. Une machine de Turing peut-elle contenir un seul état ?

3.6 Dans le théorème 3.21, nous avons montré qu'un langage est reconnaissable par Turing ssi L'énumérateur l'énumère. Pourquoi n'avons-nous pas utilisé l'algorithme suivant, plus simple, pour la direction avant de la preuve ? Comme précédemment, s1,s2,... est une liste de toutes les chaînes de Σ .

E = « Ignorer l'entrée.

1. Répétez ce qui suit pour i = 1,2,3,... .
2. Exécutez M sur si.
3. S'il accepte, imprimez si."

3.7 Expliquez pourquoi ce qui suit n'est pas une description d'une machine de Turing légitime.

Mbad = « Sur l'entrée hpi, un polynôme sur les variables x1,...,xk :

1. Essayez tous les paramètres possibles de x1,...,xk sur des valeurs entières.
2. Évaluez p sur tous ces paramètres.
3. Si l'un de ces paramètres est évalué à 0, acceptez ; sinon, rejetez.

3.8 Donnez des descriptions au niveau de l'implémentation des machines de Turing qui décident des langues suivantes sur l'alphabet {0,1}.

Aa. {w| w contient un nombre égal de 0 et de 1}

b. {w| w contient deux fois plus de 0 que de 1}

c. {w| w ne contient pas deux fois plus de 0 que de 1}

PROBLEMES

3.9 Soit un k-PDA un automate à pile qui possède k piles. Ainsi, un 0-PDA est un NFA et un 1-PDA est un PDA conventionnel. Vous savez déjà que les 1-PDA sont plus puissants (ils reconnaissent une plus grande classe de langages) que les 0-PDA.

a. Montrez que les PDA doubles sont plus puissants que les PDA simples.

b. Montrez que les 3-PDA ne sont pas plus puissants que les 2-PDA.
(Astuce : simulez une bande de machine de Turing avec deux piles.)

A3.10 Supposons qu'une machine de Turing à écriture unique est une machine à bande unique qui peut modifier chaque carré de bande au plus une fois (y compris la partie d'entrée de la bande). Montrez que ce modèle de machine de Turing variante est équivalent au modèle de machine de Turing ordinaire. (Conseil : dans un premier temps, considérons le cas où la machine de Turing peut modifier chaque carré de bande au plus deux fois. Utilisez beaucoup de bande.)

PROBLEMES

3.11 Une machine de Turing avec une bande doublement infinie est semblable à une machine de Turing ordinaire, mais sa bande est infinie à gauche comme à droite. La bande est initialement remplie de blancs, à l'exception de la partie qui contient l'entrée. Le calcul est défini comme d'habitude, sauf que la tête ne rencontre jamais une extrémité de la bande lorsqu'elle se déplace vers la gauche. Montrez que ce type de machine de Turing reconnaît la classe des langages reconnaissables par Turing.

3.12 Une machine de Turing avec réinitialisation à gauche est similaire à une machine de Turing ordinaire, mais la fonction de transition a la forme

$$d : Q \times \Gamma \to Q \times \Gamma \times \{R,\text{RESET}\}.$$

Si $\delta(q,a) = (r,b,\text{RESET})$, lorsque la machine est dans l'état q en train de lire un a, la tête de la machine saute à l'extrémité gauche de la bande après avoir écrit b sur la bande et entre dans l'état r. Notez que ces machines n'ont pas la capacité habituelle de déplacer la tête d'un symbole vers la gauche. Montrez que les machines de Turing avec réinitialisation à gauche reconnaissent la classe des langages reconnaissables par Turing.

3.13 Une machine de Turing avec rester en place au lieu de gauche est similaire à une machine de Turing ordinaire, mais la fonction de transition a la forme

$$d : Q \times \Gamma \to Q \times \Gamma \times \{R,S\}.$$

À chaque point, la machine peut déplacer sa tête vers la droite ou la laisser dans la même position. Montrer que cette variante de la machine de Turing n'est pas équivalente à la version habituelle. Quelle classe de langages ces machines reconnaissent-elles ?

3.14 Un automate de file d'attente est comme un automate à poussoir, sauf que la pile est remplacée par une file d'attente. Une file d'attente est une bande permettant d'écrire des symboles uniquement à l'extrémité gauche et de les lire uniquement à l'extrémité droite. Chaque opération d'écriture (nous l'appellerons un push) ajoute un symbole à l'extrémité gauche de la file d'attente et chaque opération de lecture (nous l'appellerons un pull) lit et supprime un symbole à l'extrémité droite. Comme avec un PDA, l'entrée est placée sur une bande d'entrée séparée en lecture seule, et la tête de la bande d'entrée ne peut se déplacer que de gauche à droite. La bande d'entrée contient une cellule avec un symbole vide suivant l'entrée, de sorte que la fin de l'entrée peut être détectée. Un automate de file d'attente accepte son entrée en entrant dans un état d'acceptation spécial à tout moment. Montrez qu'un langage peut être reconnu par un automate de file d'attente déterministe ssi le langage est reconnaissable par Turing.

3.15 Montrer que la collection des langages décidables est fermée sous l'opération de

Défi accepté. union.

b. concaténation.

c. étoile.

d. complémentation.

e. carrefour.

3.16 Montrer que la collection des langages reconnaissables par Turing est fermée sous la fonctionnement de

Défi accepté. union.

b. concaténation.

c. étoile.

d. intersection.

e. homomorphisme.

?3.17 Soit B = {hM1i,hM2i,...} un langage reconnaissable par Turing constitué de descriptions TM.
Démontrer qu'il existe un langage décidable C constitué de descriptions TM telles que chaque machine décrite en B possède une machine équivalente en C et vice versa.

? 3.18 Montrer qu'un langage est décidable ssi un énumérateur énumère le langage dans l'ordre standard des chaînes.

? 3.19 Montrer que tout langage infiniment reconnaissable par Turing possède un nombre infini de décidables sous-ensemble.

? 3.20 Montrez que les mémoires de traduction à bande unique qui ne peuvent pas écrire sur la partie de la bande contenant la chaîne d'entrée ne reconnaissent que les langages normaux.

3.21 Soit c1x n + c2xn−1 +···+ cnx+cn+1 est un polynôme de racine en x = x0. Soit cmax la plus grande valeur absolue d'un ci. Montrer que

$$|x_0| < (n+1)\frac{c_{\max}}{|c_1|}.$$

A3.22 Soit A le langage contenant uniquement la chaîne unique s, où

$$s = \begin{cases} 0 \text{ si la vie ne sera jamais trouvée sur Mars.} \\ 1 \text{ si la vie sera trouvée sur Mars un jour.} \end{cases}$$

La réponse à la question A est -elle décidable ? Pourquoi ou pourquoi pas ? Pour les besoins de ce problème, supposons que la réponse à la question de savoir si la vie sera trouvée sur Mars ait une réponse OUI ou NON sans ambiguïté .

SOLUTIONS SÉLECTIONNÉES

3.1 (b) q100, q20, xq3 , q5x ,q5 x , q2x , xq2 , x qaccepter.

3.2 (a) q111,xq31,x1q3 ,x1 qrejeter.

3.3 Nous prouvons les deux directions de l'hypothèse ssi. Premièrement, si un langage L est décidable, il peut être décidé par une machine de Turing déterministe, et celle-ci est automatiquement une machine de Turing non déterministe.

Deuxièmement, si un langage L est décidé par une TM N non déterministe, nous modifions la TM D déterministe qui a été donnée dans la preuve du théorème 3.16 comme suit.
Déplacer l'étape 4 vers l'étape 5.
Ajouter une nouvelle étape 4 : Rejeter si toutes les branches du non-déterminisme de N ont été rejetées.
Nous soutenons que ce nouveau TM D0 est un décideur pour L. Si N accepte son entrée, D0 finira par trouver une branche acceptante et l'acceptera également. Si N rejette son entrée, toutes ses branches s'arrêtent et rejettent car c'est un décideur. Par conséquent, chacune des branches a un nombre fini de nœuds, où chaque nœud représente une étape du calcul de N le long de cette branche.
Par conséquent, l'arbre de calcul de N sur cette entrée est fini, en vertu du théorème sur les arbres donné dans l'énoncé de l'exercice. Par conséquent, D0 s'arrêtera et rejettera lorsque cet arbre entier aura été exploré.

3.5 (a) Oui. L'alphabet en bande Γ contient . Une machine de Turing peut écrire n'importe quel caractère de Γ sur sa bande.

(b) Non. Σ ne contient jamais , mais Γ contient toujours . Ils ne peuvent donc pas être égaux.

(c) Oui. Si la machine de Turing tente de déplacer sa tête hors de l'extrémité gauche de la bande, elle reste sur la même cellule de bande.

(d) Non. Toute machine de Turing doit contenir deux états distincts : qaccept et qreject. Donc, une La machine de Turing contient au moins deux états.

3.8 (a) « Sur la chaîne d'entrée w :

1. Scannez la bande et marquez le premier 0 qui n'a pas été marqué. Si aucun 0 non marqué n'est trouvé, passez à l'étape 4. Sinon, remettez la tête vers l'avant de la bande.
2. Scannez la bande et marquez le premier 1 qui n'a pas été marqué. Si aucun 1 non marqué n'est trouvé, rejetez-le.
3. Remettez la tête vers l'avant de la bande et passez à l'étape 1.
4. Remettez la tête vers l'avant de la bande. Balayez la bande pour voir s'il reste des 1 non marqués. Si aucun n'est trouvé, acceptez ; sinon, rejetez.

3.10 Nous simulons d'abord une machine de Turing ordinaire à l'aide d'une machine de Turing à écriture double. La machine à écriture double simule une seule étape de la machine d'origine en copiant la bande entière sur une nouvelle portion de la bande à droite de la portion actuellement utilisée.

La procédure de copie fonctionne caractère par caractère, en marquant un caractère au fur et à mesure de sa copie. Cette procédure modifie chaque case de la bande deux fois : une fois pour écrire le caractère pour la première fois, et une autre fois pour marquer qu'il a été copié. La position de la tête de bande de la machine de Turing d'origine est marquée sur la bande. Lors de la copie des cellules situées à la position marquée ou à proximité de celle-ci, le contenu de la bande est mis à jour conformément aux règles de la machine de Turing d'origine.

Pour effectuer la simulation avec une machine à écriture unique, procédez comme précédemment, sauf que chaque cellule de la bande précédente est maintenant représentée par deux cellules. La première de ces cellules contient le symbole de bande de la machine d'origine et la seconde est destinée à la marque utilisée dans la procédure de copie. L'entrée n'est pas présentée à la machine au format avec deux cellules par symbole, donc la toute première fois que la bande est copiée, les marques de copie sont placées directement sur les symboles d'entrée.

3.15 (a) Pour deux langages décidables L1 et L2, soient M1 et M2 les TM qui les décident.
Nous construisons une TM M0 qui décide de l'union de L1 et L2 :

« Sur l'entrée w :

1. Exécutez M1 sur w. S'il accepte, acceptez.
2. Exécutez M2 sur w. S'il accepte, acceptez. Sinon, rejetez.

M0 accepte w si M1 ou M2 l'accepte. Si les deux rejettent, M0 rejette.

3.16 (a) Pour deux langages reconnaissables par Turing L1 et L2, soit M1 et M2 les TM qui les reconnaissent. On construit une TM M0 qui reconnaît l'union de L1 et L2 :

« Sur l'entrée w :

1. Exécutez M1 et M2 en alternance sur w , étape par étape. Si l'un des deux accepte, acceptez. Si les deux s'arrêtent et rejettent, rejetez.

Si M1 ou M2 accepte w, M0 accepte w car le TM acceptant arrive à son état d'acceptation après un nombre fini d'étapes. Notez que si M1 et M2 rejettent tous les deux et que l'un d'eux le fait en boucle, alors M0 bouclera.

3.22 Le langage A est l'un des deux langages {0} ou {1}. Dans les deux cas, le langage est fini et donc décidable. Si vous n'êtes pas en mesure de déterminer lequel de ces deux langages est A, vous ne pourrez pas décrire le décideur de A. Cependant, vous pouvez donner deux machines de Turing, dont l'une est le décideur de A.