Différence de codage = ≠ce de complexité ⇝ Entrée : n entier
Sortie : oui ssi n pair

914

Décidabilité, indécidabilité. Exemples.

AUT p13/92 WOL p3

I Définitions et outils pour la décidabilité

1) Problèmes et codages

Déf Un problème de décision P est la donnée de
→ E un ensemble
→ p un prédicat sur E

Les éléments de E sont appelés instances du problème, ceux vérifiant p instances positives, les autres instances négatives. On notera resp P^+ et P^- leur ensemble.

Rq On donnera plutôt les problèmes sous la forme entrée/sortie.

ex PAIR [entrée : n un entier naturel
sortie : oui s'il est pair, non sinon

ici $P = \mathbb{N}$
$P^+ = 2\mathbb{N}$
$P^- = 2\mathbb{N}+1$.

Déf Un codage pour un problème (de décision) $P = (E, p)$ est une fonction injective de E dans Σ^* où Σ est un alphabet fini. On la notera souvent $<\cdot>$.
↳ "effectif"

Le langage associé au problème est alors $\mathcal{L}_P = \{ <x> \mid x \in P^+ \}$ l'ensemble des codages des instances positives.

⚠ On considère les codages "raisonnables" → pas mettre la réponse ds le codage

ex L'écriture en binaire est un codage possible des entiers naturels sur $\{0,1\}$.
$\mathcal{L}_{PAIR} = \{ \omega \in \{0,1\}^* \mid \omega = \omega_n \dots \omega_1 \text{ où } \omega_1 = 0 \}$

2) Rappels sur les machines de Turing

Déf Une machine de Turing* (MT) $\mathcal{M}$ décide un langage $\mathcal{L} \subset \Sigma^*$ si pour tout mot $\omega \in \Sigma^*$ $\{ \mathcal{M}$ accepte ω si $\omega \in \mathcal{L}$; $\mathcal{M}$ termine et rejette ω si $\omega \notin \mathcal{L}$

Une MT $\mathcal{M}$ accepte $\mathcal{L} \subset \Sigma^*$ si pour tout mot $\omega \in \mathcal{L}$, $\mathcal{M}$ accepte ω.

Pté [$\mathcal{M}$ décide $\mathcal{L}$ ⟹ $\mathcal{M}$ accepte $\mathcal{L}$.

* déterministe.

WOL p108

Déf Un langage est dit récursif s'il existe une MT qui le décide. On note R la classe des langages récursifs.
Un langage est dit récursivement énumérable s'il existe une MT qui l'accepte. On note RE la classe des lang. réc. énum.

WOL p109

Pté [$R \subseteq RE$ **Pté** [R est stable par passage au complémentaire

Soit Σ un alphabet fini. Σ^* est dénombrable, on note $(\omega_i)_{i\in\mathbb{N}}$ ses mots.
L'ensemble des MT sur Σ est dénombrable, on les note $(\mathcal{M}_i)_{i\in\mathbb{N}}$.
On introduit alors $L^0 = \{ \omega \in \Sigma^* \mid \omega = \omega_i \text{ et } \mathcal{M}_i \text{ n'accepte pas } \omega \}$

WOL p142 → 145

Pté [$L^0 \notin RE$. En revanche $\overline{L^0} \in RE$ donc $\overline{L^0} \in RE \setminus R$

Co [• l'inclusion $R \subsetneq RE$ est stricte
• RE n'est pas stable par passage au complémentaire

Rq Si $\mathcal{L}$ est accepté par une MT non dét, il l'est aussi par une MT dét. C'est pourquoi on ne précise pas le genre de MT pour définir RE

WOL p115

3) Décidabilité des problèmes

Déf Un problème P est décidable s'il existe un codage tel que $\mathcal{L}_P \in R$
indécidable sinon
semi-décidable s'il existe un codage tq $\mathcal{L}_P \in RE \setminus R$
cf fig 1

AUT p92

⚠ la décidabilité ne dépend pas du codage

Pté [P décidable ⟹ $\overline{P}$ décidable où $\overline{P}$ désigne le problème

⚠ C'est faux pour semi-décidable, ci-dessous un contre exemple :

PL^0 [entrée $\omega \in \Sigma^*$
sortie oui si $\omega = \omega_i$ et $\omega_i \notin \mathcal{L}_{acc}(\mathcal{M}_i)$
est indécidable non semi-décidable

$\overline{PL^0}$ [entrée $\omega \in \Sigma^*$
sortie oui si $\omega = \omega_i$ et $\omega_i \in \mathcal{L}_{acc}(\mathcal{M}_i)$
est semi-décidable (et donc indécidable)

Déf Soient A et B deux problèmes de décision.
Une réduction de A à B est une procédure effective pour transformer une instance a de A en une instance b de B tq $a \in A^+$ ssi $b \in B^+$
cf fig 2.

AUT p92

Pté [A se réduit à B ; A est indécidable] ⟹ B est indécidable.

AUT p93

Résumé En pratique pour MQ P est décidable on donne une procédure effective, tandis que pour MQ P est indécidable on cherche à réduire un pb indéc connu à P.

(Ajouter des exemples après les définitions)

⚠ Codage = délicat !

⚠ Insister sur le fait que la décidabilité ne dépend pas du codage !

II Problèmes indécidables théoriques

1) Problème de l'arrêt et d'acceptation

AUT p93 KOL p146

ARRET [**entrée**: $\mathcal{M}$ une machine de Turing sur Σ, ω un mot de Σ^*
sortie: oui si $\mathcal{M}$ s'arrête lancée sur ω, non sinon

Pté $\boxed{\text{ARRET est semi décidable}}$

NB: on montre que $\mathcal{L}_{ARRET}$ est dans RE grâce à une machine universelle
on montre que ARRET est indécidable par l'absurde.

ACCEPT [**entrée** $\mathcal{M}$ une machine de Turing sur Σ, ω un mot de Σ^*
sortie oui si $\mathcal{M}$ accepte ω, non sinon.

Pté $\boxed{\text{ACCEPT est semi-décidable}}$

NB: on montre que $\mathcal{L}_{ACCEPT}$ est dans RE grâce à une autre machine-univ.
on montre que ACCEPT est indécidable en réduisant ARRET à ACCEPT.

2) Théorème de Rice

AUT p94 KOL p150

Déf || Soit p un prédicat sur RE (ie p est une partie de RE)
On dit que p est trivial si $p = \emptyset$ ou $p = RE$

P_p [**entrée** M une MT
sortie oui si $\mathcal{L}_{acc}(\mathcal{M}) \in p$, non sinon

Pté [(Théorème de Rice)
Si p est un prédicat non trivial sur RE alors le problème P_p est indécidable

Csq Le théorème de Rice signifie qu'on ne peut pas construire de procédure permettant de vérifier qu'un programme est valide, si on considère tous les programmes possibles.

3) Le problème de correspondance de Post

AUT p96

PCP [**entrée** $(u_i, v_i)_{i \in [1..m]}$ un nombre fini de couples de mots
sortie oui s'il existe $(\alpha_i)_{i \in [1..k]} \in [1..m]^k$ tq $\begin{cases} u_{\alpha_1} \dots u_{\alpha_k} = v_{\alpha_1} \dots v_{\alpha_k} \\ k \geq 1 \end{cases}$.

cf fig3

Pté $\boxed{\text{PCP est semi-décidable.}}$

NB on montre que PCP est indécidable par réduction d'ACCEPT à PCP.

III Problèmes issus de la logique

1) Théories et problèmes associés

Déf || Une théorie est un ensemble d'énoncés (ie formules closes) appelés axiomes. DNR p79

⚠ Ici une théorie n'est pas nécessairement close par conséquence.
Ne pas confondre avec les théorèmes d'une théorie ou d'un modèle.

Pour la suite on fixe $\vdash$ un système de déduction correct et complet.

Déf || Une théorie T est consistante ssi $T \nvdash \perp$.
En particulier pour toute formule F on n'a pas simultanément $T \vdash F$ et $T \vdash \neg F$ DNR p80

Déf || Une théorie T est complète ssi $\begin{cases} T \text{ est consistante et} \\ \text{pour toute formule close } \varphi \text{ on a } T \vdash \varphi \text{ ou } T \vdash \neg\varphi \end{cases}$

Déf || Une théorie T est récursive ssi l'appartenance à T est décidable. APP(T) DNR p125
———— décidable ssi être conséquence de T est décidable. CSQ(T)

⚠ Bien comprendre la différence.

⚠ Connaître les axiomes est différent de connaître les conséquences.
Il ne faut donc pas confondre les deux problèmes suivants

APP(T) [**entrée**: φ une formule close
sortie: oui si $\varphi \in T$, non sinon

$\neq$

CSQ(T) [**entrée**: φ formule close
sortie: oui si $T \vdash \varphi$, non sinon

Pté [Soit T une $\mathcal{L}$-théorie. $\left.\begin{array}{l} T \text{ complète et récursive} \\ \mathcal{L} \text{ dénombrable} \end{array}\right\} \Rightarrow T$ est décidable DNR p126

2) Arithmétique de Péano

DNR p111

Déf || Le langage de l'arithmétique de Péano est $\mathcal{L}_{Péano} = \{0, s, +, \times, =\}$

cst; f unaire; f binaire; relation binaire

PÉANO [**entrée** φ une $\mathcal{L}_{Péano}$ formule close
sortie oui si $\mathbb{N} \models \varphi$, non sinon.

Déf La théorie de l'arithmétique élémentaire est DNR p111

$$P_0 = \left\{ \begin{array}{lll} \forall x \;\; s(x) \neq 0 & (A_1) ; & \forall x, \; x+0 = 0 \quad (A_4) \\ \forall x \;\; (x=0) \vee (\exists y, \; s(y) = x) & (A_2) ; & \forall x \; \forall y \;\; x + s(y) = s(x+y) \quad (A_5) \\ \forall x, \forall y \;\; s(x) = s(y) \rightarrow x = y & (A_3) ; & \forall x \;\; x \times 0 = 0 \quad (A_6) \\ & & \forall x \; \forall y \;\; x \times s(y) = x \times y + x \quad (A_7) \end{array} \right\}$$

DNR p111

Déf La théorie de l'arithmétique de Péano est l'arithmétique élémentaire à laquelle on ajoute le schéma de récurrence

$$PA = P^0 \cup \{ [F[x:=0] \wedge \forall y\, F[x:=y] \to F[x:=s(y)]] \to \forall x F \mid F\ \mathcal{L}_{Peano}\text{-formule} \}$$

NB : P^0 était fini donc récursif. PA est infini mais aussi récursive.

DNR p126

Pté (ADMIS) Soit T une $\mathcal{L}_{Peano}$-théorie. T non contradictoire, $P_0 \subset T$ $\Rightarrow$ T est indécidable.

Csq
- Puisque $\mathbb{N} \models PA$, PA est non contradictoire et elle contient P_0 donc P_0 est indéc.
- En notant $Th(\mathbb{N}) = \{\varphi$ un $\mathcal{L}_{Peano}$-énoncé $\mid \mathbb{N} \models \varphi\}$ les théorèmes de $\mathbb{N}$ on a
  - $\to Th(\mathbb{N})$ est non contradictoire
  - $\to P_0 \subset Th(\mathbb{N})$ car $\mathbb{N} \models P_0$.

  donc $Th(\mathbb{N})$ est indécidable.

Or pour φ un $\mathcal{L}_{Peano}$ énoncé $\mathbb{N} \models \varphi$ ssi $\varphi \in Th(\mathbb{N})$ ssi $Th(\mathbb{N}) \vdash \varphi$
donc PÉANO EST INDÉCIDABLE.

DNR p127

Pté (ADMIS) (Théorème d'incomplétude de Gödel)
Soit T une $\mathcal{L}_{Peano}$-théorie contenant PA et non contradictoire
Si T est récursive alors T n'est pas complète.

3) Arithmétique de Presburger

DNR p136

Déf Le langage de l'arithmétique de Presburger est $\mathcal{L}_{Pres} = \{0, s, +, =\}$

PRES
- entrée : φ une $\mathcal{L}_{Pres}$-formule close.
- sortie : oui si $\mathbb{N} \models \varphi$, non sinon

inspiré de Carton p 178

Déf "locale" On dira qu'une $\mathcal{L}_{Pres}$-formule atomique est simple si elle est de la forme $x_i = 0$ ou $x_i = x_j$ ou $x_i + x_j = x_k$ ou $s(x_i) = x_k$ (pr $i, j, k \neq$).

Lemme Pour F une combinaison booléenne de $\mathcal{L}_{Pres}$ formules atomiques simple il existe un automate — qu'on peut effectivement construire — qui reconnait exactement les codages des k-uplets d'entiers naturels satisfaisant φ, pour un codage bien choisi

Pté PRES est décidable.] DVP 1.

IV Problèmes sur les langages algébriques

1) Problèmes décidables

VIDE
- entrée $G = (\Sigma, \Gamma, R, S)$
- sortie oui si $L_G(S) = \emptyset$, non sinon

MOT_VIDE
- entrée $G = (\Sigma, \Gamma, R, S)$
- sortie oui si $\varepsilon \in L_G(S)$

VARIABLE
- entrée $G = (\Sigma, \Gamma, R, S)$, $T \in \Gamma$ une variable
- sortie oui si T apparait ds $\widehat{L_G(S)}$

MOT
- entrée $G = (\Sigma, \Gamma, R, S)$, w un mot de Σ^*
- sortie oui si $w \in L_G(S)$, non sinon.

Aut94 p80,81,83

Pté VIDE, MOT_VIDE et VARIABLES sont décidables par une méthode de saturation.

Ca On peut effectivement transformer une grammaire en une grammaire réduite équivalente (grâce à VIDE et VARIABLE), ou en une grammaire propre quasi-équivalente (grâce à MOT_VIDE) et donc sous forme normale de Chomsky (FNC)

[FB] p320

Pté L'algorithme C.Y.K, basé sur la prog. dynamique, détermine si un mot appartient à une grammaire sous FNC.

└ **Ca** MOT est aussi décidable.

2) Problèmes indécidables

Carton p 166

INTER_VIDE
- entrée $G = (\Sigma, \Gamma, R, S)$, $G' = (\Sigma, \Gamma', R', S')$
- sortie oui si $L_G(S) \cap L_{G'}(S') = \emptyset$

EGAL
- entrée $G = (\Sigma, \Gamma, R, S)$, $G' = (\Sigma, \Gamma', R', S')$
- sortie oui si $L_G(S) = L_{G'}(S')$

UNIV
- entrée $G = (\Sigma, \Gamma, R, S)$
- sortie oui si $L_G(S) = \Sigma^*$

AMBIGU
- entrée $G = (\Sigma, \Gamma, R, S)$
- sortie oui si G ambigüe.

Pté Ces quatre problèmes sont indécidables. } DVP 2.

Rq En revanche pour des langages rationnels ces problèmes sont décidables puisqu'on sait
- → faire l'automate produit
- → complémenter un automate
- → décider la vacuité d'un langage reconnu par un automate.

fig 1

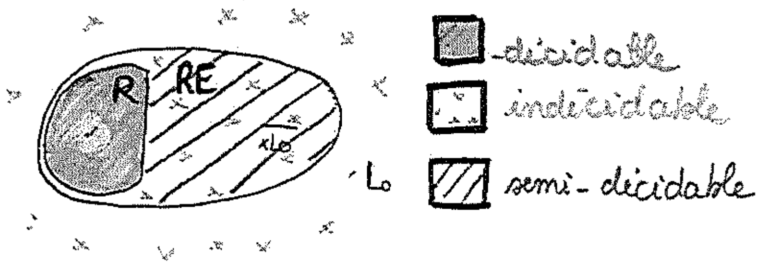
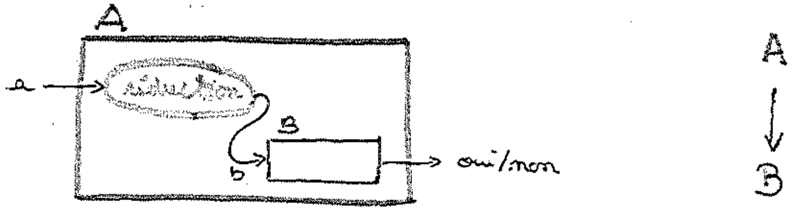


fig 2



réduction du problème A au problème B

fig 3

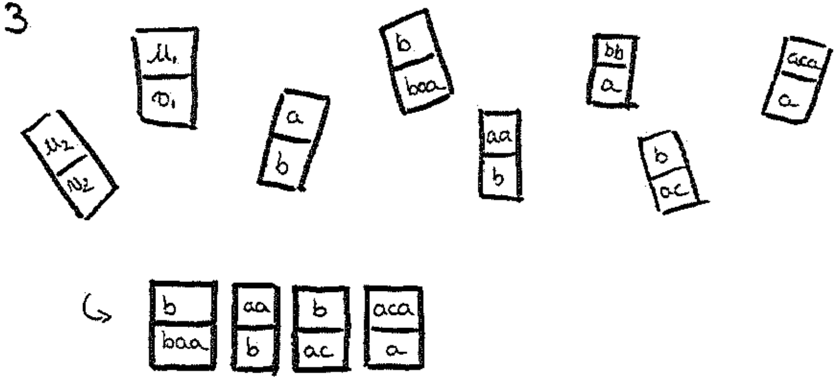
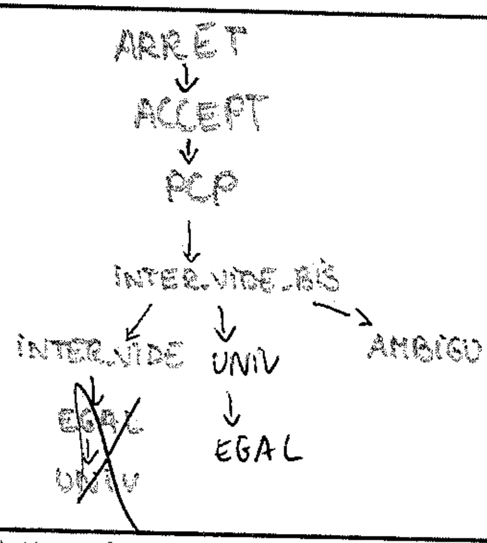


fig 4



Intro : - ne pas utiliser les mots compliqués tt de suite ! Ou alors donner des exemples !

- pb = question à laquelle on doit répondre par oui/non

↳ Un problème décidable ≡ il y a un algo.

Question : Est-ce qu'on peut AUTOMATISER ?

Références.

[Aut94] J.M Autebert
[F-B] Floyd-Biegel
[Cart] Olivier-Carton
[AUT] J.M Autebert. Calculabilité et décidabilité. éd Masson.
[WOL] P. Wolper Introduction à la calculabilité, éd. Dunod
[DNR] David, Nour, Raffali Introduction à la logique, éd Dunod

Décidabilité de l'arithmétique de Presburger

Ref Librement inspiré du Carton p (!!)
Leçons 914, 917, 924 NB : pour 909 faire une autre version.

Déf L'langage de l'arithmétique de Presburger est le langage $\mathcal{L}$ constitué de [DNR] p136.
→ 0 symbole de constante
→ s ——— fonction unaire
→ + ——— binaire
→ = ——— relation binaire

Déf On appellera ici formule atomique simple sur $\mathcal{L}$ une formule de la forme $x_i = 0$ ou $x_i = x_j$ ou $s(x_i) = x_j$ ou $x_i + x_j = x_k$ avec i, j, k dans $\mathbb{Z}$.

Lemme Pour toute formule Ψ combinaison booléenne de formules atomiques simples sur $\mathcal{L}$, on peut construire $\mathcal{A}_\Psi$ un automate qui reconnaît exactement les codages des entiers naturels satisfaisant Ψ pour un codage bien choisi.

ADMIS ici (À développer dans la 909) Cf dernier encadré.

Pté PRES : $\begin{cases} \text{entrée} & \varphi \text{ une } \mathcal{L}\text{ formule close} \\ \text{sortie} & \text{oui si } \mathbb{N} \models \varphi, \text{ non sinon} \end{cases}$ est décidable

Pour démontrer cette propriété il nous faudra dans un premier temps donner un codage des entiers, ou plutôt des k-uplets d'entiers, afin d'utiliser le lemme.

1) Codage des k-uplets d'entiers

Une fois ce codage expliqué on montrera comment ramener notre problème au problème de la vacuité d'un langage, un langage de nos codages même.

2) Se ramener à un problème de vacuité d'un langage

Enfin il restera à montrer qu'on peut construire un automate reconnaissant ce langage, puisqu'on saura alors décider sa vacuité : il suffira de résoudre un pb d'accessibilité dans le graphe qu'est l'automate.

3) Construire un automate reconnaissant ce langage.

1) Codage des k-uplets d'entiers

Soit $(n_i)_{i\in[1..k]} \in \mathbb{N}^k$

Pour $i \in [1..k]$, on peut écrire n_i en binaire $n_i : c_i^{r_i} \cdots c_i^2 c_i^1$

on a écrit n en r_i chiffres ici — bit de poids faible

Quitte à poser $r = \max_{i\in[1..k]} r_i$ on a

$$n_1 = c_1^r \cdots c_1^1$$
$$n_2 = c_2^r \cdots c_2^1$$
$$\vdots$$
$$n_k = c_k^r \cdots c_k^1$$

On pose alors :

$$\langle (n_i)_{i\in[1..k]} \rangle_k^r = \begin{pmatrix} c_1^r \\ \vdots \\ c_k^r \end{pmatrix} \cdots \underbrace{\begin{pmatrix} c_1^2 \\ \vdots \\ c_k^2 \end{pmatrix}}_{\in \Sigma_k := \{0,1\}^k} \begin{pmatrix} c_1^1 \\ \vdots \\ c_k^1 \end{pmatrix} \in \Sigma_k^*$$

codage de k-uplets sur r chiffres

ex $r=3$ $0 \to 000$, $1 \to 001$ $\quad \langle (0,1) \rangle_2^3 = \begin{matrix} 0 & 0 & 0 \\ 0 & 0 & 1 \end{matrix}$

Il nous faut aussi un "décodage" : pour $\omega \in \Sigma_k^*$ on pose

$$[\![\omega]\!]_k = \rangle \omega \langle_k^r \quad \text{où } r = |\omega|$$

Jai pas besoin de préciser quel "décodage" on applique, celui ci est évident, ou plutôt donné par la longueur du mot ω

ex $[\![\begin{matrix} 0&0&0&0 \\ 0&0&0&1 \end{matrix}]\!]_2 = (0,1)$

$[\![\begin{matrix} 0&0 \\ 0&1 \end{matrix}]\!]_2 = (0,1)$

⚠ Il faut néanmoins préciser le "k".
ex $[\![\varepsilon]\!]_1 = 0$ mais $[\![\varepsilon]\!]_3 = (0,0,0)$.

Remarquons qu'on a par construc°,

$$\left[\!\!\left[\frac{\omega}{c} \right]\!\!\right]_{k+1} = (\underbrace{n_1, \dots n_k}_{= [\![\omega]\!]_k}, \underbrace{n_{k+1}}_{= [\![c]\!]_1}) \quad \text{pour } \begin{matrix} c \in \Sigma_1^* \\ \omega \in \Sigma_k^* \end{matrix}$$

nota° pour $\binom{\omega_1}{c_1}\binom{\omega_2}{c_2} \cdots \underbrace{\binom{\omega_r}{c_r}}_{\in \Sigma_{k+1}} \in \Sigma_{k+1}^*$

2) Se ramener à un problème de vacuité

Soit φ une $\mathcal{L}$-formule close.
On se ramène à $\tilde{\varphi} \equiv \varphi$ sous forme prénexe.
Et m̂ quitte à ajouter des variables on se ramène à $\hat{\varphi} \equiv \varphi$, sous forme prénexe et dont les formules atomiques sont simples.

ex φ $\exists x_1 (\forall x_2\ x_1 + x_2 = s(x_2)) \wedge (\exists x_3, s(x_3) = x_1)$
$\tilde{\varphi} = \exists x_1, \forall x_2, \exists x_3, (x_1 + x_2 = s(x_2)) \wedge (s(x_3) = x_1)$
$\hat{\varphi} = \exists x_1, \forall x_2, \exists x_3, \exists x_4, (x_1 + x_2 = x_4) \wedge (s(x_2) = x_4) \wedge (s(x_3) = x_1)$

On écrit $\hat{\varphi} : Q_1 x_1 \dots Q_k x_k \underbrace{Q_{k+1} x_{k+1} \cdots Q_m x_m\, \psi}_{:= \psi_k}$, où ψ sans quantif.

Ainsi $\psi_0 = \hat{\varphi}$, $\psi_m = \psi$ et $\forall k$, $Varlib(\psi_k) = \{x_i \mid i \in [1..k]\}$

On note aussi $\mathcal{N}_k = \{(n_i)_{i\in[1..k]} \mid \mathbb{N}[{}^{x_i := n_i}_{i\in[1..k]}] \models \hat{\varphi}\}$
$\mathcal{L}_k = \{\omega \in \Sigma_k^* \mid [\![\omega]\!]_k = (n_1 - n_k) \in \mathcal{N}_k\}$

Si $\hat{\varphi} : \exists x_1 \psi_1$, alors $\mathbb{N} \models \hat{\varphi}$ ssi $\mathcal{N}_1 \neq \emptyset$ ssi $\mathcal{L}_1 \neq \emptyset$
Si $\hat{\varphi} : \forall x_1 \psi_1$ ——— $\mathcal{N}_1 = \mathbb{N}$ ssi $\mathcal{L}_1 = \Sigma_1^*$ ssi $\mathcal{L}_1^c = \emptyset$.

↳ On est ramené au problème de vacuité de $\mathcal{L}_1$.

3) Montrer par récurrence que $\mathcal{L}_1$ est reconnaissable

On montre par réc. finie, pour k allant de m à 1 la propriété H_k [Il existe un automate $\mathcal{A}_k$ tq $\mathcal{L}(\mathcal{A}_k) = \mathcal{L}_k$]

- Puisque les formules atomique de $\hat{\varphi}$ et donc de ψ sont simples, le lemme assure H_m ($\psi = \psi_m$)
- Soit $k > 1$ tel que H_k. On note $\mathcal{A}_k = (\Sigma_k, Q, q_0, F, T)$.
  On suppose que $\psi_{k-1} : \exists x_k\, \psi_k$.

alors $\mathcal{N}_{k-1} = \{(n_1 - n_{k-1}) \mid \exists n_k \in \mathbb{N}, (n_1 - n_k) \in \mathcal{N}_k\}$
donc $\mathcal{L}_{k-1} = \{\omega \in \Sigma_{k-1}^* \mid \exists n_k \in \mathbb{N}, ([\![\omega]\!]_{k-1}, n_k) \in \mathcal{N}_k\}$
$= \{ \text{———} \mid \exists c \in \Sigma_1^* \ ([\![\omega]\!]_{k-1}, [\![c]\!]_1) \in \mathcal{N}_k\}$
$= \{ \text{———} \mid \exists c \in \Sigma_1^* \ [\![{}^{0-0\,\omega}_{\ c}]\!]_k \in \mathcal{N}_k\}$ ⚠
$= \{ \text{———} \mid \exists u v \in \Sigma_1^* \ ({}^{0-0}_{\ u} | {}^{\omega}_{v}) \in \mathcal{L}_k\}$
$= \{\omega \in \Sigma_{k-1}^* \mid \exists uv \in \Sigma_1^*, (\frac{0-0}{u} | \frac{\omega}{v}) \in \mathcal{L}(\mathcal{A}_k)\}$ par H_k

On introduit alors $p : \begin{pmatrix} \Sigma_k \longrightarrow \Sigma_{k-1} \\ \begin{pmatrix} c_1 \\ \vdots \\ c_k \end{pmatrix} \longmapsto \begin{pmatrix} c_1 \\ \vdots \\ c_{k-1} \end{pmatrix} \end{pmatrix}$

et on pose $\pi(\mathcal{A}_k) = (\Sigma_{k-1}, Q, q_0, F^\pi, T^\pi)$

où $F^\pi = \{q \mid q \xrightarrow[\mathcal{A}_k]{(p^{-1}(0))^+} q_f \in F\}$

i.e l'ens. des états à partir desquels on pouvait rejoindre l'état final en ne lisant que des 0 sur les $k-1$ 1ères composantes.

$T^\pi = \{q \xrightarrow{p(a)} q' \mid (q \xrightarrow{a} q') \in T\}$

Si $w \in \mathcal{L}(\pi(\mathcal{A}_k))$
alors il existe $q_f' \in F^\pi$ tq $q_0 \xrightarrow[\pi(\mathcal{A}_k)]{w} q_f'$.

Par déf de T^π il existe $v \in \Sigma_1^*$ tq $q_0 \xrightarrow[\mathcal{A}_k]{\boxed{w/v}} q_f'$

Par déf de F^π ——— $u \in \Sigma_1^*$ tq $q_f' \xrightarrow[\mathcal{A}_k]{\boxed{0/u}} q_f$
pour un certain $q_f \in F$.

Ainsi $q_0 \xrightarrow[\mathcal{A}_k]{\frac{0|w}{u|v}} q_f \in F$.

Donc $\frac{0|w}{u|v} \in \mathcal{L}(\mathcal{A}_k)$, d'où $w \in \mathcal{L}_{k-1}$.

Réc si $w \in \mathcal{L}_{k-1}$.
Il existe $u, v \in \Sigma_1^*$ tel que $\frac{0|w}{u|v} \in \mathcal{L}(\mathcal{A}_k)$
Donc il existe $q_f' \in Q$, et $q_f \in F$ tq
$q_0 \xrightarrow[\mathcal{A}_k]{\boxed{w/v}} q_f' \xrightarrow[\mathcal{A}_k]{\boxed{0/u}} q_f$

Par déf de T^π, $q_0 \xrightarrow[\pi(\mathcal{A}_k)]{w} q_f'$

Par déf de F^π, $q_f' \in F^\pi$

Donc $w \in \mathcal{L}(\pi(\mathcal{A}_k))$.

On pose alors $\mathcal{A}_{k-1} = \pi(\mathcal{A}_k)$
ainsi $\mathcal{L}(\mathcal{A}_{k-1}) = \mathcal{L}_{k-1}$.

D'où H_{k-1}.

Si $\psi_{k-1}: \forall x_k\, \psi_k$, $\psi_{k-1} \equiv \neg(\exists x_k \neg\psi_k)$.
Il suffit, puisqu'on sait complémenter, de refaire ce qu'on a fait avant.

On pose ici $\mathcal{A}_{k-1} = \overline{\pi(\overline{\mathcal{A}_k})}$.
$\mathcal{N}_{k-1}^c = \{n_1 - n_{k-1} \mid \mathbb{N}[x_i := n_i,\ i \in [1,k-1]] \models \exists x_k, \neg\psi_k\}$
$= \{ ——— \mid \exists n_k \in \mathbb{N},\ \mathbb{N}[x_i := n_i,\ i \in [1,k]] \models \neg\psi_k\}$
$= \{n_1 - n_{k-1} \mid \exists n_k \in \mathbb{N},\ (n_1 - n_k) \in \mathcal{N}_k^c\}$

Par HR $\mathcal{L}(\mathcal{A}_k) = \mathcal{N}_k$, donc $\mathcal{L}(\overline{\mathcal{A}_k}) = \mathcal{N}_k^c$
Comme préc, et d'après cette reformulation de $\mathcal{N}_{k-1}^c$
$\mathcal{L}(\pi(\overline{\mathcal{A}_k})) = \mathcal{N}_{k-1}^c$ (On reconnaît les k-1 uplets projetés des k-uplets reconnus par $\overline{\mathcal{A}_k}$. C'est ce que fait π)

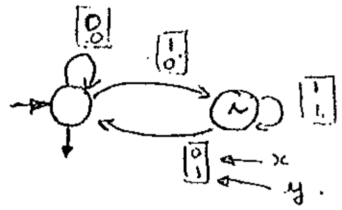
Donc $\mathcal{L}(\overline{\pi(\overline{\mathcal{A}_k})}) = \mathcal{N}_{k-1}$
soit $\mathcal{L}(\mathcal{A}_{k-1}) = \mathcal{N}_{k-1}$, d'où H_{k-1}.

FIN DVP.

Pour comprendre pourquoi cette histoire d'états finals est importante

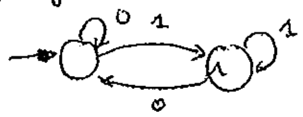
$\varphi \sim \forall x. \exists y,\ y = x + x$

$\psi_2 \sim y = x + x$. $\mathcal{A}_2$:



$\psi_1 \sim \exists y,\ y = x + x$.

$\mathcal{A}_1$

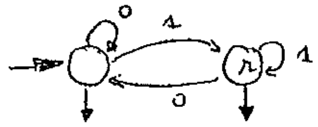


si ici on n'a pas bien déf. les états finals

Pb : Cet automate ne reconnaît pas 1 !

Pourtant il devrait reconnaître 1, puisque pour $x=1$ il existe y tq $y = x+x$, avec $y=2$.
L'idée est que $\mathcal{A}_2$ reconnaissait "(1,2)" codé en $\boxed{\frac{1|0}{0|1}}$, donc $\mathcal{A}_1$ ——— "1" codé en 01.

$\mathcal{A}_1$ le bien défini

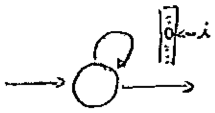


($r \in F_1$ car $r \xrightarrow{\boxed{0/1}} \bigcirc \in F_2$)

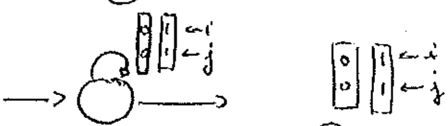
ici $\mathcal{L}(\mathcal{A}_1) = \{0,1\}^* = \Sigma_1^* = \mathcal{L}_1$ puisque $\mathcal{N}_1 = \mathbb{N}$
Donc $\mathbb{N} \models \varphi$, en effet tout entier a un double entier.

Rappels des automates de base pour le lemme

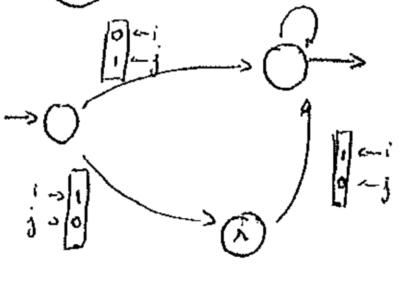
Si $\psi: x_i = 0$ $\mathcal{A}_\psi$:



Si $\psi: x_i = x_j$ $\mathcal{A}_\psi$:

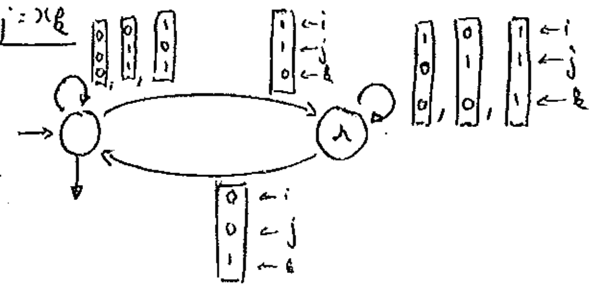


Si $\psi: s(x_i) = x_j$ $\mathcal{A}_\psi$



Si $\psi: x_i + x_j = x_k$

$\mathcal{A}_\psi$



Si $\psi = \neg\psi_a$ $\mathcal{A}_\psi = \overline{\det(\mathcal{A}_{\psi_a})}$
Si $\psi: \psi_a \vee \psi_b$ $\mathcal{A}_\psi = \mathcal{A}_{\psi_a} \cup \mathcal{A}_{\psi_b}$
Si $\psi: \psi_a \wedge \psi_b$ $\mathcal{A}_\psi = \mathcal{A}_{\psi_a} \times \mathcal{A}_{\psi_b}$