Numéro d'anonymat : .....

MCAL – MT – Examen

Durée : 1h30, sans document

- N'oubliez pas d'indiquer votre numéro d'anonymat sur le sujet et glissez le dans votre copie à la fin de l'épreuve.
- Répondez sur votre copie sauf pour les questions avec pointillés.
- Commencez par lire tout le sujet pour repérer les questions faciles.
- Respectez les notations du cours.
- Le sujet est sur 20 points et comporte 5 exercices indépendants.

Le barème est donné à titre indicatif.

Tous les appareils électroniques sont interdits à l'exception des montres.

Exercice 1 : Connaissez-vous les définitions du cours ? (4 pt)

Complétez les pointillés.

Q1. (0.5 pt) Deux MT M_1 et M_2 sont équivalentes si et seulement si $\mathscr{L}(M_1) = \mathscr{L}(M_2)$ *ie.* si elles reconnaissent le même langage.

Q2. (0.5 pt) Un langage L est décidable si et seulement si le langage et son complémentaire sont récursivement énumérables *ie.* reconnaissable par une MT

Q3. (0.5 pt) Un langage L est récursivement énumérable si et seulement si il existe une MT M qui le reconnaît, *ie.* $\mathscr{L}(M) = L$

Q4. (0.5 pt) Un langage L est indécidable si et seulement si L **ou** $\overline{L}$ n'est pas reconnaissable par une MT

Q5. (0.5 pt) Le langage reconnu par une machine de Turing M est l'ensemble des mots tels que l'exécution de M sur ω termine dans l'état accepteur ◎ .

Q6. (0.5 pt) Si m est le codage binaire de la MT M, ω un mot binaire et U est la machine de Turing universelle alors $U(m) = M$ et $M(\omega) = U(m)(\omega)$

Q7. (1 pt) Le langage reconnu par la machine universelle U est

$\mathscr{L}(U) = \{(m, \omega) \in \mathcal{M} \times \{0,1\}^* \mid U(m, \omega) = \mathbb{V}\}$

C'est l'ensemble des couples (m, ω) formés d'une MT et d'un mot tel que m accepte le mot ω.

Exercice 2 : Utilisation des machines de Turing (2 pt)

Q8. (0.25 pt) Expliquez en français le fonctionnement d'une instruction « do M_{body} while M_{cond} ».

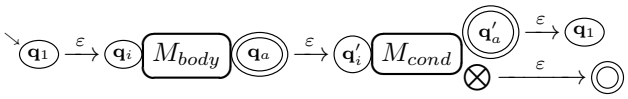
SOLUTION

- *L'instruction exécute M_{body} tant que la condition M_{cond} est satisfaite. L'instruction commence par exécuter M_{body} (au moins une fois).*
- *L'instruction exécute M_{body} puis évalue la condition M_{cond} ; si la condition est satisfaite, elle recommence ; sinon l'exécution de l'instruction s'achève.*

Q9. (0.75 pt) Dessinez la MT qui correspond à l'instruction do M_{body} while M_{cond}. Vous représenterez les machines de Turing M_{body} et M_{cond} par des nuages avec un état initial, un état accepteur et si nécessaire un état de rejet.

SOLUTION

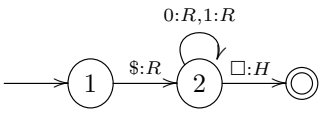
Soit q_i *l'état initial et* q_a *l'état accepteur de* M_{body}. *Soit* q'_i *l'état initial,* q'_a *l'état accepteur et* $\otimes$ *l'état rejet de* M_{cond}. *On constuit la* MT [*do* M_{body} *until* M_{cond}] *de la manière suivante :*



où on supprime le statut initial, accepteur et rejet des états de M_{body} *et* M_{cond} *car une* MT *ne doit avoir un unique état intial, un unique état accepteur et au plus un état rejet. et* $\xrightarrow{\varepsilon}$ *est une notation qui représente* $\xrightarrow{\Sigma:H}$.

Q10. (0.25 pt) Donnez une MT qui reconnaît le langage $\{0,1\}^*$.

SOLUTION

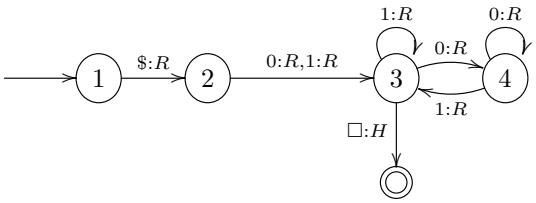


Q11. (0.75 pt) Donnez une MT déterministe qui reconnaît les écritures binaires (little endian = bits de poids faible à gauche) des entiers naturels sans 0 inutile.

Exemples :

- 3 est représenté en binaire par le mot | ∞□ | $ | 1 | 1 | □∞ |, **pas** par | ∞□ | $ | 1 | 1 | 0 | … | 0 | □∞ |
- En particulier l'entier 0 est représenté par le mot | ∞□ | $ | 0 | □∞ |

SOLUTION



Exercice 3 : Langages réguliers et machines de Turing (3.5 pt)

Justifiez soigneusement vos réponses.

Q12. (.75 pt) Expliquez comment traduire un automate (à nombre) d'états fini A en une machine de Turing M équivalente. Que signifie « équivalente » ?

SOLUTION

(0.25 pt) $M \equiv A \iff \mathscr{L}(M) = \mathscr{L}(A)$ *: il suffit que la* MT M *reconnaisse* $\mathscr{L}(A)$. *Il est donc inutile de s'occuper des cas où* A *rejette le mot. On construit* M *de la manière suivante :*

- (0.25 pt) *une transition* $q \xrightarrow{\ell} q'$ *de* A *se traduit par une transition* $q \xrightarrow{\ell/\ell:R} q'$ *dans* M
- (0.25 pt) *un état accepteur* q *se traduit par une transition* $q \xrightarrow{\square:H}$ ◎ *vers l'unique état accepteur de* M
- *Remarque : on peut toujours se rammener au cas d'un* AEF *à un seul état initial et un seul état accepteur en ajoutant des* ε*-transitions et une* ε*-transition* $q \xrightarrow{\varepsilon} q'$ *de* A *se traduit par* $q \xrightarrow{\Sigma:H} q'$

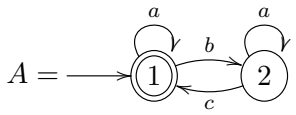
Q13. (0.75 pt) Que faut-il faire en plus si on souhaite que la MT M décide le langage $\mathscr{L}(A)$? Que signifie « décide » ?

SOLUTION

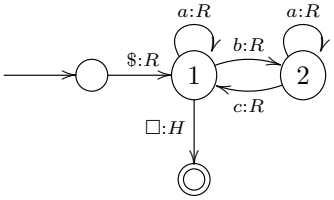
(0.25 pt) *pour que la MT M décide $\mathscr{L}(A)$ il faut qu'elle reconnaisse $\mathscr{L}(A)$ mais aussi qu'elle rejette les mots de $\overline{\mathscr{L}(A)}$, ie. les mots non reconnus par A. Pour cela :*

- (0.25 pt) *on complète l'automate A, ce qui a pour effet d'ajouter un état puit* ⊙
- (0.25 pt) *l'état puit de A se traduit par une transition* ⊙ $\xrightarrow{\square:H}$ ⊗ *vers l'unique état rejet de M*

Q14. (0.5 pt) **Application :** On considère l'alphabet $\Sigma = \{a, b, c\}$. Dessinez la machine de Turing M équivalente à l'automate (à nombre) d'états fini suivant



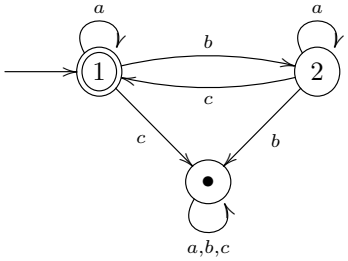
SOLUTION



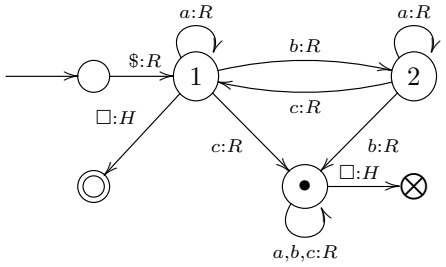
Q15. (0.5 pt) **Application (suite) :** Complétez (avec un crayon d'une autre couleur) la machine de Turing M de la question précédente afin qu'elle **décide** le langage $\mathscr{L}(A)$.

SOLUTION

- *On complète l'automate A*



- *On applique la traduction définie dans les deux questions précédentes*



Q16. (0.5 pt) Existe-t'il des **langages réguliers** qui ne sont pas reconnaissables par les machines de Turing ?
Justifiez soigneusement votre réponse ; une réponse « oui/non » ne donne pas de point.

SOLUTION

Non. Par définition, un langage régulier L correspond au langage d'un automate (à nombre) d'états fini, disons A. On a donc $L = \mathscr{L}(A)$.
Or on a montré dans une question précédente qu'on peut constuire une machine M équivalente à A au sens où $\mathscr{L}(M) = \mathscr{L}(A) = L$.
Conclusion *Pour chaque langage régulier L, on a montré qu'il existe donc un* MT *M qui le reconnaît, ie. $\mathscr{L}(M) = L$.*

Q17. (0.5 pt) Existe-t'il des **langages** qui ne sont pas reconnaissables par les machines de Turing ?
Justifiez soigneusement votre réponse ; une réponse « oui/non » ne donne pas de point.

SOLUTION

Oui.

(a) *En fait, pour tout langage indécidable L, le langage L ou son complémentaire $\overline{L}$ n'est pas reconnaissable par une machine de Turing. Et il existe des langages indécidables : citons les langage qui correspondent aux ensembles de Rice non-triviaux.*

(b) *Le langage des exécutions infinies formé des couples (m, ω) tel que $U(m, \omega) \to \infty$ n'est pas reconnaissable par une machine de Turing.*

Exercice 4 : Formalisation et Théorème de Rice (4 pt)

On note $\mathcal{M}$ l'ensemble des codages binaires de machines de Turing.

Q18. (0.25 pt) **Complétez :** Un ensemble de machines de Turing est un ensemble de Rice s'il s'écrit $\{m \in \mathcal{M} \mid \mathcal{C}(\mathscr{L}(U(m)))\}$ où la condition $\mathcal{C}$ porte sur le langage reconnu par m

Q19. (0.25 pt) **Complétez :** Tout ensemble de Rice non-trivial c'est-à-dire différent de $\mathcal{M}$ et de $\emptyset$, est indécidable.

Q20. (1 pt) **Formalisez en termes mathématiques les ensemble suivants**

1. L'ensemble des machines de Turing qui n'acceptent pas le mot m correspondant à leur codage en binaire :
   $$L_1 = \{m \in \mathcal{M} \mid U(m)(m) \neq \mathbb{V}\}$$
2. L'ensemble des machines de Turing dont l'exécution sans paramètre est infinie :
   $$L_2 = \{m \in \mathcal{M} \mid U(m)(\varepsilon) \to \infty\}$$
3. L'ensemble des machines de Turing qui acceptent tous les mots binaires :
   $$L_3 = \{m \in \mathcal{M} \mid \mathscr{L}(U(m)) = \{0,1\}^*\}$$
4. L'ensemble des machines de Turing équivalentes à la MT $M_\emptyset$ qui reconnaît le langage vide :
   $$L_4 = \{m \in \mathcal{M} \mid \mathscr{L}(U(m)) = \emptyset\}$$

Q21. (1.5 pt) Parmi les ensembles L_1 à L_4, trois sont des ensembles de Rice. Lesquels ? Justifiez votre réponse en définissant le critère de Rice, $\mathcal{C}$, correspondant.

- L_2 est un ensemble de Rice car ce langage correspond à $\{m \in \mathcal{M} \mid \varepsilon \notin \mathscr{L}(U(m))\}$ qui est de la forme $\{m \in \mathcal{M} \mid \mathcal{C}(\ \mathscr{L}(U(m))\ )\}$ avec $\mathcal{C}(\mathscr{L}) \stackrel{def}{=} \varepsilon \notin \mathscr{L}$
- L_3 est un ensemble de Rice car il peut s'écrire $\{m \in \mathcal{M} \mid \mathcal{C}(\ \mathscr{L}(U(m))\ )\}$ avec $\mathcal{C}(\mathscr{L}) \stackrel{def}{=} \mathscr{L} = \{0,1\}^*$
- L_4 est un ensemble de Rice car il peut s'écrire $\{m \in \mathcal{M} \mid \mathcal{C}(\ \mathscr{L}(U(m))\ )\}$ avec $\mathcal{C}(\mathscr{L}) \stackrel{def}{=} \mathscr{L} = \emptyset$

Q22. (1 pt) Expliquez pourquoi l'ensemble restant n'est pas un ensemble de Rice. Que peut-on en déduire ?

SOLUTION

— *L_1 n'est pas un ensemble de Rice. C'est un exemple piège car on peut écrire L_1 sous la forme*

$$L_1 = \{ m \in \mathcal{M} \mid m \notin \mathscr{L}(U(m)) \}$$

*On est alors tenté d'exprimer L_1 sous la forme $\{ m \in \mathcal{M} \mid \mathcal{C}(\mathscr{L}(U(m))) \}$ en prenant $\mathcal{C}(\mathscr{L}) \stackrel{def}{=} m \notin \mathscr{L}$ et de conclure que c'est un ensemble de Rice. C'est incorrect car avec $\mathcal{C}$ ainsi définie le terme m **est fixe** dans la définition de $\mathcal{C}$.*

— *Le théorème de Rice donne un critère pour détecter des ensembles indécidables. Si un ensemble ne satisfait pas ce critère, on ne peut rien dire car le théorème ne s'applique pas.*

Exercice 5 : Réduction du PCP au problème de terminaison d'un programme Gamma (6.5 pt)

Le but de cet exercice est de démontrer que la terminaison d'un programme Gamma est un problème indécidable. Pour le prouver on va réduire le Problème de Correspondance de Post (PCP) – connu pour être indécidable – au problème de la terminaison de programmes Gamma.

Problèmes de Correspondance de Post et exécutions de programmes Gamma (3.5 pt)

On considère l'alphabet $\{ a, c, f, i, \ell \}$ et ε désigne le mot vide (*ie.* le `""` des chaînes de caractères).

Q23. (0.25 pt) **Complétez :** Soit D un ensemble de domino PCP(D) ℯ D telle que le mot obtenu par concaténation de la partie supérieure de inférieure de

Exemple : Considérons l'ensemble de dominos $D_1 = \left\{ \underbrace{\boxed{\frac{f}{fac}}}_{d_1}, \underbrace{\boxed{\frac{i\ell}{i\ell}}}_{d_2}, \underbrace{\boxed{\frac{ac}{\varepsilon}}}_{d_3} \right\}$. La séquence $d_1.d_3.d_2$ est une solution puisqu'on obtient $f.ac.i\ell$ en haut et $fac.\varepsilon.i\ell$ en bas. La séquence réduite à d_2 est aussi une solution puisqu'on obtient $i\ell$ en haut et en bas.

Exécutions de programmes Gamma On considère les constructeurs Gamma suivants A(.), I(.), C(.), F(.), L(.) qui prennent tous un argument, ainsi qu'un constructeur Gamma sans argument (*ie.* une constante) ε.
À l'aide des constantes et constructeurs précédents on peut représenter les mots écrits sur l'alphabet $\Sigma = \{ a, c, f, i, \ell \}$. On désigne par $[\, \omega \,]_\Gamma$ la représentation en Gamma d'un mot $\omega \in \Sigma^*$.

Q24. (0.25 pt) Donnez la représentation en Γ du mot $facil$: $[\, facil \,]_\Gamma = \mathrm{F}(\mathrm{A}(\mathrm{C}(\mathrm{I}(\mathrm{L}(\varepsilon)))))$

Q25. (0.25 pt) Donnez l'exécution du programme Gamma Γ_{D_1} ci-dessous sur l'ensemble réduit à l'élément $\mathrm{G}(\, [\, facil \,]_\Gamma \,, [\, facil \,]_\Gamma \,)$

$$\Gamma_{D_1} \stackrel{def}{=} \begin{cases} \mathrm{G}(\, \mathrm{F}(x) \,, \mathrm{F}(\mathrm{A}(\mathrm{C}(y))) \,) & \xrightarrow[r_1]{} \quad \mathrm{G}(\, x \,, y \,) \\ \mathrm{G}(\, \mathrm{I}(\mathrm{L}(x)) \,, \mathrm{I}(\mathrm{L}(y)) \,) & \xrightarrow[r_2]{} \quad \mathrm{G}(\, x \,, y \,) \\ \mathrm{G}(\, \mathrm{A}(\mathrm{C}(x)) \,, y \,) & \xrightarrow[r_3]{} \quad \mathrm{G}(\, x \,, y \,) \end{cases}$$

SOLUTION

$$\mathrm{G}(\ [\ \mathit{facil}\ ]_\Gamma\ ,\ [\ \mathit{facil}\ ]_\Gamma\ ) = \mathrm{G}(\ \textsc{facil}(\varepsilon)\ ,\ \textsc{facil}(\varepsilon)\ ) \xrightarrow[r_1]{} \mathrm{G}(\ \textsc{acil}(\varepsilon)\ ,\ \textsc{il}(\varepsilon)\ ) \xrightarrow[r_3]{} \mathrm{G}(\ \textsc{il}(\varepsilon)\ ,\ \textsc{il}(\varepsilon)\ ) \xrightarrow[r_2]{} \mathrm{G}(\ \varepsilon\ ,\ \varepsilon\ )$$

Q26. (0.25 pt) Donnez l'exécution du programme Γ_{D_1} sur l'ensemble réduit à l'élément $\mathrm{G}(\ [\ \mathit{acfacil\ell}\ ]_\Gamma\ ,\ [\ \mathit{acfacil\ell}\ ]_\Gamma$

SOLUTION

$$\mathrm{G}(\ [\ \mathit{acfacil\ell}\ ]_\Gamma\ ,\ [\ \mathit{acfacil\ell}\ ]_\Gamma\ ) = \mathrm{G}(\ \textsc{acfacil}(\varepsilon)\ ,\ \textsc{acfacil}(\varepsilon)\ ) \xrightarrow[r_3]{} \mathrm{G}(\ \textsc{facil}(\varepsilon)\ ,\ \textsc{acfacil}(\varepsilon)\ )$$

Q27. (0.5 pt) On souhaite ajouter une règle (r_0) au programme Γ_{D_1} afin qu'il **ne termine pas** lorsqu'on l'exécute sur $\mathrm{G}(\ [\ \mathit{facil}\ ]_\Gamma\ ,\ [\ \mathit{facil}\ ]_\Gamma\ )$. Complétez la régle (r_0).

$$\mathrm{G}(\ \varepsilon\ ,\ \varepsilon\ ) \quad \xrightarrow[r_0]{} \quad \mathrm{G}(\ [\ \mathit{facil\ell}\ ]_\Gamma\ ,\ [\ \mathit{facil\ell}\ ]_\Gamma\ )$$

Généralisation Dans les questions précédentes on a raisonné sur *la solution* « $\mathit{facil\ell}$ » du $\textsc{pcp}(D_1)$; on veut maintenant généraliser le raisonnement à toute solution du $\textsc{pcp}(D_1)$.

Q28. (1 pt) En vous inspirant fortement du programme Γ_{D_1} donnez un programme Γ'_{D_1} qui **ne termine pas** lorsqu'on l'exécute sur $\mathrm{G}'(\ \varepsilon\ ,\ \varepsilon\ ,\ [\ \omega\ ]_\Gamma\ )$ quel que soit le mot ω correspondant à une solution du $\textsc{pcp}(D_1)$. Pour cela on utilise un constructeur $\mathrm{G}'(\ .\ ,\ .\ ,\ .\ )$ à trois arguments au lieu de $\mathrm{G}(\ .\ ,\ .\ )$.

SOLUTION

$$\Gamma'_{D_1} \stackrel{def}{=} \left\{ \begin{array}{rcl} \mathrm{G}'(\ \varepsilon\ ,\ \varepsilon\ ,\ w\ ) & \xrightarrow[r_0]{} & \mathrm{G}'(\ w\ ,\ w\ ,\ w\ ) \\ \mathrm{G}'(\ \textsc{f}(x)\ ,\ \textsc{f}(\textsc{a}(\textsc{c}(y)))\ ,\ w\ ) & \xrightarrow[r_1]{} & \mathrm{G}'(\ x\ ,\ y\ ,\ w\ ) \\ \mathrm{G}'(\ \textsc{i}(\textsc{l}(x))\ ,\ \textsc{i}(\textsc{l}(y))\ ,\ w\ ) & \xrightarrow[r_2]{} & \mathrm{G}'(\ x\ ,\ y\ ,\ w\ ) \\ \mathrm{G}'(\ \textsc{a}(\textsc{c}(x))\ ,\ y\ ,\ w\ ) & \xrightarrow[r_3]{} & \mathrm{G}'(\ x\ ,\ y\ ,\ w\ ) \end{array} \right.$$

Programme Gamma associé à un PCP : application à un autre ensemble de dominos

Q29. (0.5 pt) Donnez une séquence de dominos de l'ensemble D_2 ci-dessous qui soit une solution du $\textsc{pcp}(D_2)$ et donnez le mot correspondant à cette solution.

$$D_2 = \left\{ \underbrace{\boxed{\frac{ia}{a}}}_{d_1}, \underbrace{\boxed{\frac{acc}{ci}}}_{d_2}, \underbrace{\boxed{\frac{f}{fac}}}_{d_3} \right\}$$

SOLUTION

$d_3.d_2.d_1$ est une solution du PCP *sur D_2 correspondant au mot $\omega \stackrel{def}{=} \mathit{faccia}$. En effet,* $\boxed{\frac{f}{fac}}\boxed{\frac{acc}{ci}}\boxed{\frac{ia}{a}}$ *donne*

$$u_3.u_2.u_1 = f.acc.ia = faccia = fac.ci.a = v_3.v_2.v_1$$

Q30. (0.5 pt) Donnez un programme Γ'_{D_2} qui **ne termine pas** lorsqu'on l'exécute sur $\mathrm{G}'(\ \varepsilon\ ,\ \varepsilon\ ,\ [\ \omega\ ]_\Gamma\ )$ où ω est le mot correspondant à une solution du problème $\textsc{pcp}(D_2)$.

SOLUTION

$$\Gamma'_{D_2} \stackrel{def}{=} \begin{cases} \mathrm{G}'(\ \varepsilon\ ,\ \varepsilon\ ,\ w\ ) & \xrightarrow[r_0]{} & \mathrm{G}'(\ w\ ,\ w\ ,\ w\ ) \\ \mathrm{G}'(\ \mathrm{F}(x)\ ,\ \mathrm{F}(\mathrm{A}(\mathrm{C}(y)))\ ,\ w\ ) & \xrightarrow[r_1]{} & \mathrm{G}'(\ x\ ,\ y\ ,\ w\ ) \\ \mathrm{G}'(\ \mathrm{A}(\mathrm{C}(\mathrm{C}(x)))\ ,\ \mathrm{C}(\mathrm{I}(y))\ ,\ w\ ) & \xrightarrow[r_2]{} & \mathrm{G}'(\ x\ ,\ y\ ,\ w\ ) \\ \mathrm{G}'(\ \mathrm{I}(\mathrm{A}(x))\ ,\ \mathrm{A}(y)\ ,\ w\ ) & \xrightarrow[r_3]{} & \mathrm{G}'(\ x\ ,\ y\ ,\ w\ ) \end{cases}$$

Indécidabilité de la terminaison de programmes Gamma (3 pt)

- On note $\mathscr{D}$ l'ensemble de tous les ensembles de dominos possibles sur l'alphabet Σ. Ainsi, $D \in \mathscr{D}$ est un ensemble de dominos.
- PCPSAT désigne l'ensemble des ensembles D de dominos pour lesquels $\textsc{pcp}(D)$ a une solution. Autrement dit, $D \in \textsc{pcpsat}$ si et seulement si $\textsc{pcp}(D)$ admet une solution.
- *Gamma* désigne l'ensemble de tous les programmes Gamma possibles.
- On note GTT l'ensemble de programmes $\Gamma \in \mathit{Gamma}$ dont l'exécution se termine quel que soit l'ensemble E sur lequel on l'exécute. Autrement dit, $\Gamma \in \textsc{Gtt}$ signifie que Γ termine toujours.

$$\textsc{Gtt} = \{\Gamma \in \mathit{Gamma} \mid \forall E,\ \Gamma(E) \not\to \infty\} \quad \textit{et} \quad \overline{\textsc{Gtt}} = \mathit{Gamma} \setminus \textsc{Gtt}.$$

Q31. (0.25 pt) **Complétez :** $\Gamma \in \overline{\textsc{Gtt}}$ signifie qu'il existe (au moins) un ensemble E tel que $\Gamma(E) \to \infty$ autrement dit l'exécution de Γ sur E ne termine pas .

Q32. (0.25 pt) **Complétez le diagramme de réduction et les « on doit montrer »**

$$\begin{array}{rcl} D \in \mathscr{D} & \xrightarrow[\textit{traduction}]{M_R} & \Gamma'_D \in \mathit{Gamma} \\ \underbrace{D \in \textsc{pcpsat}} & \underset{(\ddagger)}{\iff} & \underbrace{\Gamma'_D \in \overline{\textsc{Gtt}}} \\ \textit{indécidable} & \implies & \textit{indécidable} \end{array}$$

Pour montrer l'indécidabilité de la terminaison de programmes Gamma que doit-on faire ?

(a) On doit montrer qu'il existe une MT qui traduit tout ensemble de domino D en un programme Γ'_D

(b) On doit montrer l'équivalence $(\ddagger)$.

Q33. (0.5 pt) **(a) Complétez la définition de la traduction**

Un domino $\boxed{\frac{u}{v}} \in D$ peut être vu comme un couple (u, v) que l'on peut inscrire sur la bande 1 d'une MT. Un ensemble D de domino
cette bande par une séquence de couple
MT M_R à deux bande
bande 1 et pour chaque domino d_i re (u, v), elle inscrit la règle Gamma $\mathrm{G}'(\ u(x)\ ,\ v(x)\ ,\ w\ ) \xrightarrow[r_i]{} \mathrm{G}'(\ x\ ,\ y\ ,\ w\ )$ sur la bande 2. On ajoute ensuite à la bande 2, la règle (r_0) ; ainsi la bande 2 contient le programme Γ'_D tel qu'on l'a défini dans la première partie de cet exercice.

Q34. (0.5 pt) **(b) Complétez la preuve de** (‡)

($\Rightarrow$) On doit montrer que si $D \in$ PCPSAT alors le programme Γ'_D construit par $M_R(D)$ ne termine pas toujours . Il suffit donc de trouver un ensemble E sur lequel l'exécution de Γ'_D ne termine pas . Puisque $D \in$ PCPSAT, le problème PCP(D) admet une solution ; notons ω le mot corre
de l'excercice que Γ'_D ne termine pas si on l'exécute sur l'ensemble $\{G'(\ \epsilon, \epsilon, [\,\omega\,]_\Gamma\ )\}$. Il suffit donc de prendre cet ensemble pour E.

Q35. (0.5 pt) **(b) Complétez la preuve de** (‡)

($\Leftarrow$) Montrons la réciproque : On doit montrer que si il existe un E tel que $\Gamma'_D(E)$ ne termine pas alors il existe une solution au PCP(D) (et donc $D \in$ PCPSAT).
À l'excc (r_0), toute Γ'_D appliquée $G'(\ \omega_1, \omega_2, \omega\ )$ font diminuer le mot ω_1 ou le mot ω_2 (ou le
règle $\Gamma_D(E)$ a une exécution infinie il e
qu'elle utilise la règle (r_0) régulièrement car c'e
mot ω_1 et ω_2. Une exécution infinie contient donc plusieurs applications de (r_0). Elle aura donc forcément la forme $\ldots \xrightarrow[r_0]{} G'(\ \omega, \omega, \omega\ ) \xrightarrow[r_{i_1}]{} \ldots \xrightarrow[r_{i_k}]{} G'(\ \epsilon, \epsilon, \omega\ ) \xrightarrow[r_0]{} \ldots$
On constate sur cette portion d'exécution que le $r_{i_1}, \ldots, r_{i_k}$ consomment intégralement le ω. Puisque chacune
cela signifie que la séquence $d_{i_1}, \ldots, d_{i_k}$ e PCP(D).

Q36. (0.5 pt) **Complétez la conclusion :** On a montré par réduction que la terminaison de programmes Gamma est indécidable.

SUPPOSONS LE CONTRAIRE ET MONTRONS UNE CONTRADICTION : Suppo
existe un algorithme capable de décider la terminaison de
alors on pourrait savoir si un PCP(D) a une solution. Pour cela, étant donné un ensemble D il suffirait de construire le programme Γ'_D au moyen de M_R
e

- si Γ'_D termine toujours
  alors PCP(D) n'a pas de solution
- s'il existe un cas où Γ'_D ne termine pas
  alors PCP(D) admet une solution

On serait donc capable de décider le
qu'on sait impo CONTRADICTION.