Décidabilité et indécidabilité

François Schwarzentruber

ENS Rennes

Table des matières

1 Problèmes de décision

Un problème de décision est souvent présenté de manière informelle entrée (description d'une instance) / sortie (oui / non).

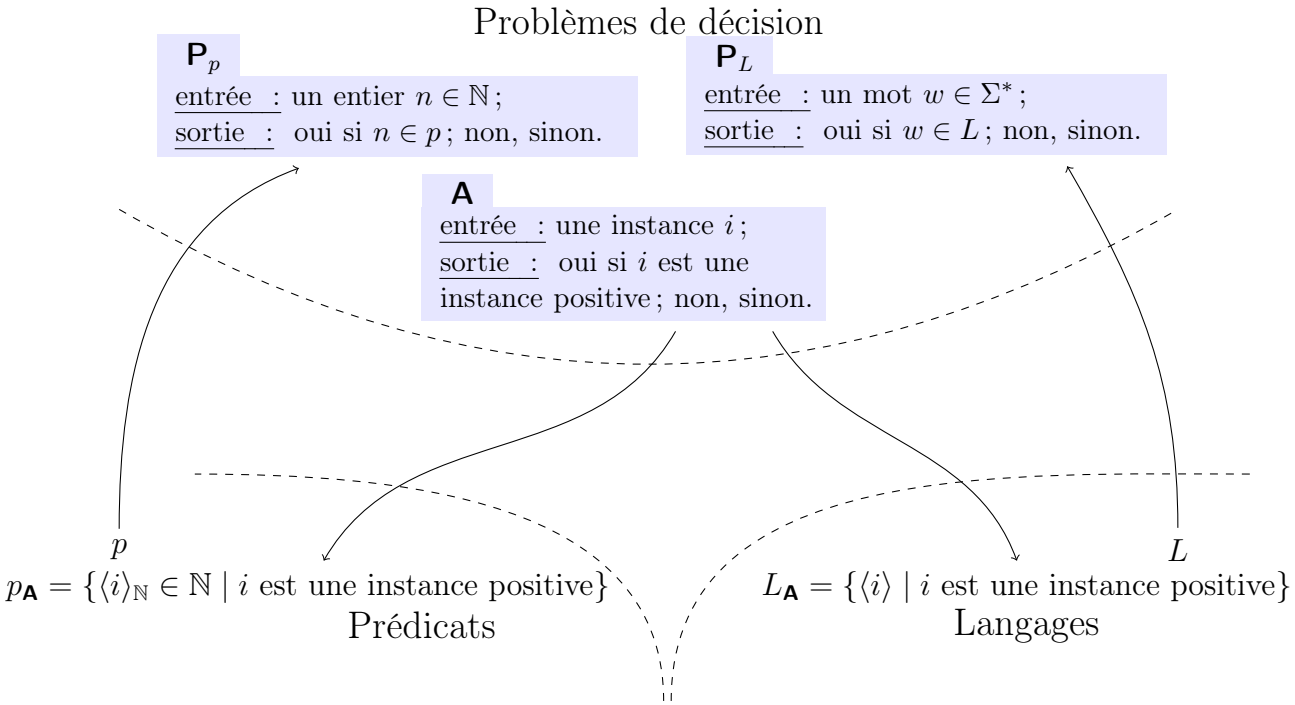
Exemple 1 *Le problème de décision* ***Connexe*** *qui teste si un graphe non orienté G est connexe :*

Connexe
entrée : *un graphe fini non orienté G ;*
sortie : *oui si G est connexe ; non, sinon.*

- *Un graphe fini non orienté G est une* ***instance*** *de* ***Connexe****.*
- *Un graphe fini non orienté G connexe est une* ***instance positive*** ***Connexe****.*
- *Un graphe fini non orienté G non connexe est une* ***instance négative*** ***Connexe****.*

1.1 Différentes présentations

Se donner un problème de décision est équivalent à se donner un langage $L \subseteq \Sigma^*$ sur un alphabet fini Σ ou à un prédicat $p \subseteq \mathbb{N}$ (i.e. une fonction $p : \mathbb{N} \to \{0, 1\}$).



1.2 Dual

Connexe
entrée : un graphe fini non orienté G ;
sortie : oui si G est connexe ; non, sinon.

$\overline{\text{Connexe}}$
entrée : un graphe fini non orienté G ;
sortie : oui si G est non connexe ; non, sinon.

1.3 Décidabilité

Un problème de décision **A** est **décidable** si (au choix)

1. il existe un **algorithme** qui décide **A**. L'algorithme A s'arrête sur toutes les instances de **A** et :
   — Si i est une instance positive, A accepte i ;
   — Si i est une instance négative, A refuse i.
2. le langage associé $L_{\mathbf{A}}$ est récursif (dans R), i.e. il existe une machine de Turing M qui **décide** $L_{\mathbf{A}}$. La machine de Turing s'arrête quelque soit le mot d'entrée w et :
   — Si w représente une instance positive, M accepte w ;
   — Si w ne représente pas une instance positive (i.e. ne représente pas une instance ou représente une instance négative), M refuse w.
3. le prédicat $\mathbb{N} \to \{0,1\}$ qui représente **A** est μ-récursive totale.

1.4 Récursivement énumérable

Un problème de décision **A** est **récursivement énumérable**/**semi-décidable** si (au choix)

1. il existe un **semi-algorithme** qui accepte **A** :
   — Si i est une instance positive, A s'arrête et accepte i ;
   — Si i est une instance négative, A refuse i ou boucle.
2. le langage associé $L_{\mathbf{A}}$ est récursivement énumérable (dans RE), i.e. il existe une machine de Turing qui **accepte** $L_{\mathbf{A}}$:
   — Si w représente une instance positive, M accepte w ;
   — Si w ne représente pas une instance positive (i.e. ne représente pas une instance ou représente une instance négative), M refuse w ou M ne s'arrête pas w.
3. le prédicat $p : \mathbb{N} \to \{0,1\}$ qui représente **A** est telle qu'il existe une fonction f **μ-récursive partielle** tel que pour tout $n \in \mathbb{N}$,
   — Si $p(n) = 1$, $f(n) = 1$;
   — Si $p(n) = 0$, alors $f(n) = 0$ ou $f(n)$ n'est pas défini.

1.5 Trois façons d'être indécidables

Théorème 1 *Soit* **A** *un problème indécidable. Alors soit :*

1. $\mathbf{A} \in RE$ *et* $\overline{\mathbf{A}} \notin RE$ *;*
2. *ou alors* $\mathbf{A} \notin RE$ *et* $\overline{\mathbf{A}} \in RE$ *;*
3. *ou alors* $\mathbf{A} \notin RE$ *et* $\overline{\mathbf{A}} \notin RE$.

Démonstration.
Supposons $\mathbf{A} \in RE$ et $\overline{\mathbf{A}} \in RE$. Montrons que **A** est décidable. Soit A et $\overline{A}$ des semi-algorithmes respectifs pour **A** et $\overline{\mathbf{A}}$. L'algorithme B défini ci-dessous décide **A** :

procedure $B(i)$
Lancer en parallèle $A(i)$ et $\overline{A}(i)$ et s'arrêter quand l'un des proccessus a accepté i
accepter si A a accepté i ; **rejeter** si $\overline{A}$ a accepté i.

■

2 Problème de l'arrêt

On s'intéresse au **problème de l'arrêt** :

Arrêt
entrée : une machine de Turing M, un mot w ;
sortie : oui si $M(w)$ s'arrête ; non, sinon.

2.1 ... est récursivement énumérable

Théorème 2 ***Arrêt*** *est récursivement énumérable.*

Démonstration.

On donne une machine U qui accepte l'entrée M, w ssi M s'arrête sur w. La machine U est dite **universelle** car elle est capable de simuler l'exécution de n'importe quel machine de Turing M.

```
procedure U(M, w)
    simuler M(w)
    accepter
```

■

2.2 ... est indécidable

Théorème 3 ***Arrêt*** *est indécidable.*

Démonstration.

Par l'absurde. Supposons que **Arrêt** est décidable. Il existe donc une machine de Turing A qui s'arrête sur toute entrée M, w et

- A accepte M, w, alors $M(w)$ termine ;
- A refuse M, w, alors $M(w)$ ne termine pas.

```
procedure A(M, w)
    ...
    ...
    ...
```

On construit une machine *paradoxe* comme suit :

```
procedure paradoxe(M)
    if A accepte (M, ⟨M⟩)
        boucler ↻
    else
        accepter
```

$$\begin{gathered} paradoxe(paradoxe) \text{ ne termine pas} \\ \text{ssi} \\ A \text{ accepte } (paradoxe, \langle paradoxe \rangle) \\ \text{ssi} \\ paradoxe(paradoxe) \text{ termine.} \end{gathered}$$

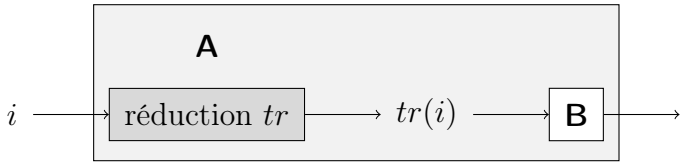
Contradiction.

■

3 Montrer l'indécidabilité par réduction

3.1 Réduction

Définition 1 (Réduction)
Une réduction d'un problème **A** à un problème **B** est une fonction tr calculable telle que pour toute instance i de **A**, i est instance positive de **A** ssi $tr(i)$ est instance positive de **B**.



On dit que **A** se réduit à **B** s'il existe une réduction de **A** à **B**.

Théorème 4 *Si **A** se réduit à **B**, alors **A** indécidable implique **B** indécidable.*

Démonstration.
Montrons la contraposée. Supposons que **A** se réduit à **B** et **B** est décidable puis montrons que **A** est décidable. Nous donnons un algorithme A qui décide **A** à partir d'une réduction tr de **A** à **B** et d'un algorithme B qui décide **B**. Donc **A** est décidable.
■

```
procedure A(i)
    if B accepte tr(i)
        accepter
    else
        rejeter
```

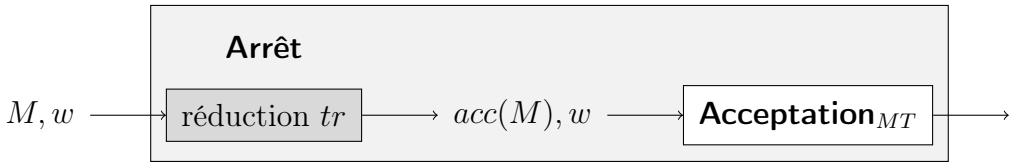
3.2 Exemple : acceptation par une machine de Turing

On s'intéresse au problème de l'acceptation d'un mot par une machine de Turing :

Acceptation$_{MT}$
entrée : une machine de Turing M, un mot w ;
sortie : oui si M accepte w ; non, sinon.

Théorème 5 ***Acceptation***$_{MT}$ *est indécidable.*

Démonstration.
On va réduire **Arrêt** dans **Acceptation**$_{MT}$.



À partir d'une description d'une machine M, on peut construire effectivement une machine $acc(M)$ où les transitions qui pointent vers q_{rej} sont redirigés vers q_{acc}. Ainsi,

$$M(w) \text{ s'arrête ssi } acc(M) \text{ accepte } w$$

On pose $tr(M, w) = (acc(M), w)$.
tr est une réduction de **Arrêt** dans **Acceptation**$_{MT}$. En effet :
— tr est calculable ;
— M, w est une instance positive de **Arrêt**
ssi $tr(M, w)$ est une instance positive de **Acceptation**$_{MT}$.
Comme **Arrêt** est indécidable, **Acceptation**$_{MT}$ est indécidable. ■

4 Théorème de Rice

4.1 Enoncé

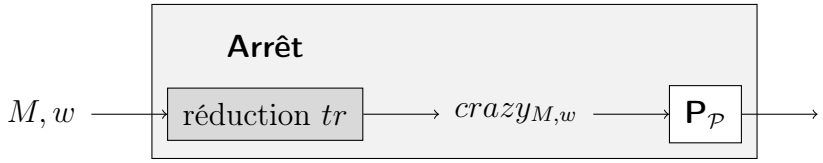
Théorème 6 (de Rice) *[Wol06] Soit $\mathcal{P}$ telle $\emptyset \subsetneq \mathcal{P} \subsetneq RE$. Le problème $\mathbf{P}_\mathcal{P}$ défini par*

$\mathbf{P}_\mathcal{P}$
entrée : une machine de Turing M
sortie : oui si $L(M) \in \mathcal{P}$; non, sinon.

est indécidable.

Démonstration.

Sans perte de généralité, on peut supposer que $\emptyset \notin \mathcal{P}$ (sinon prendre $RE \setminus \mathcal{P}$ à la place de $\mathcal{P}$ dans la suite de la démonstration). Réduisons **Arrêt** à $\mathbf{P}_\mathcal{P}$.



Soit $\mathbf{G} \in \mathcal{P}$. Comme $\mathbf{G} \in RE$, il existe une machine G qui accepte $\mathbf{G}$. La réduction tr est définie par

$$tr(M, w) = crazy_{M,w}$$

où $crazy_{M,w}$ est la machine décrite à droite.

procedure $crazy_{M,w}(x)$
  Simuler $M(w)$
  if G accepte x
    accepter
  else
    rejeter

1. tr est une fonction calculable : on construit effectivement $crazy_{M,w}$ à partir de M et w ;
2. Comme $L(crazy_{M,w}) = \begin{cases} \mathbf{G} \text{ si } M \text{ s'arrête sur } w \\ \emptyset \text{ sinon} \end{cases}$, on a :

$$M, w \text{ est une instance positive de } \textbf{Arrêt}$$
$$\text{ssi}$$
$$M \text{ s'arrête sur } w$$
$$\text{ssi}$$
$$L(crazy_{M,w}) \in \mathcal{P}$$
$$\text{ssi}$$
$$tr(M, w) = crazy_{M,w} \text{ est une instance positive de } \mathbf{P}_\mathcal{P}.$$

■

4.2 Exemple

Exemple 2 *Avec $\mathcal{P} = \{\emptyset\}$, le théorème de Rice donne que*

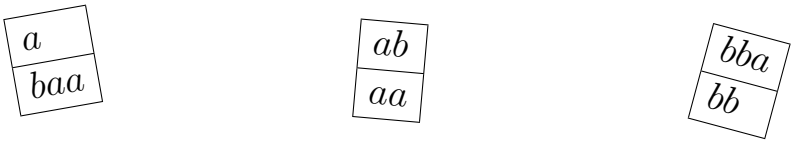
Langagevide
entrée : une machine de Turing M
sortie : oui si $L(M) = \emptyset$; non, sinon.

est indécidable.

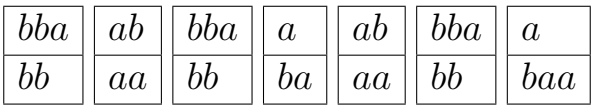
5 Problèmes de correspondance de Post

5.1 Définitions

Le problème de Post prend en entrée un système de tuiles comme



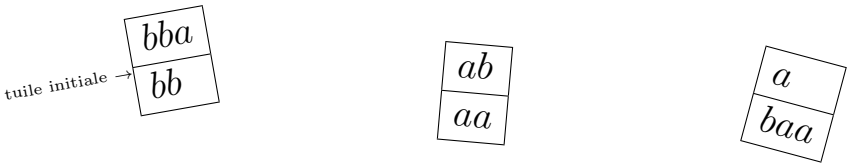
et répond oui si on peut mettre bout à bout des tuiles du système (on peut réutiliser plusieurs fois la même tuile) pour que les mots du haut et du bas soient identiques.



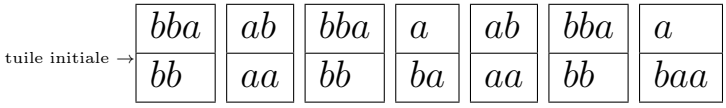
Définition 2 (Problème de correspondance de Post)

Post
entrée : un alphabet fini Σ, une famille finie de couples de mots $((h_i, b_i))_{i=1..n}$ sur Σ ;
sortie : oui s'il existe $i_1, \ldots, i_p \in \{1, \ldots, n\}$ tels que $p \geq 1$ et $h_{i_1} \ldots h_{i_p} = b_{i_1} \ldots b_{i_p}$; non, sinon.

Le problème de Post marqué est similaire mais il impose de commencer la suite de tuiles par la tuile initiale. Si l'entrée est :



et répond oui si on peut mettre bout à bout des tuiles du système (on peut réutiliser plusieurs fois la même tuile) pour que les mots du haut et du bas soient identiques en partant de la tuile initiale.



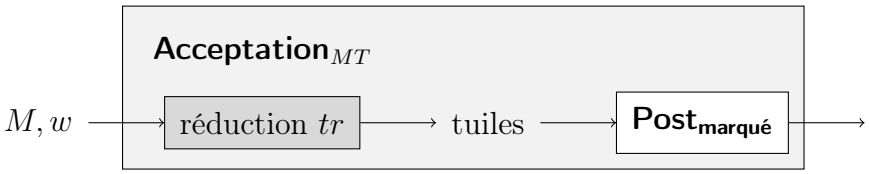
Définition 3 (Problème de correspondance de Post marqué)

Post$_{\text{marqué}}$
entrée : un alphabet fini Σ, une famille finie de couples de mots $((h_i, b_i))_{i=1..n}$ sur Σ ;
sortie : oui s'il existe $i_2, \ldots, i_p \in \{1, \ldots, n\}$ tels que $p \geq 1$ et $h_{\mathbf{1}} h_{i_2} \ldots h_{i_p} = b_{\mathbf{1}} b_{i_2} \ldots b_{i_p}$; non, sinon.

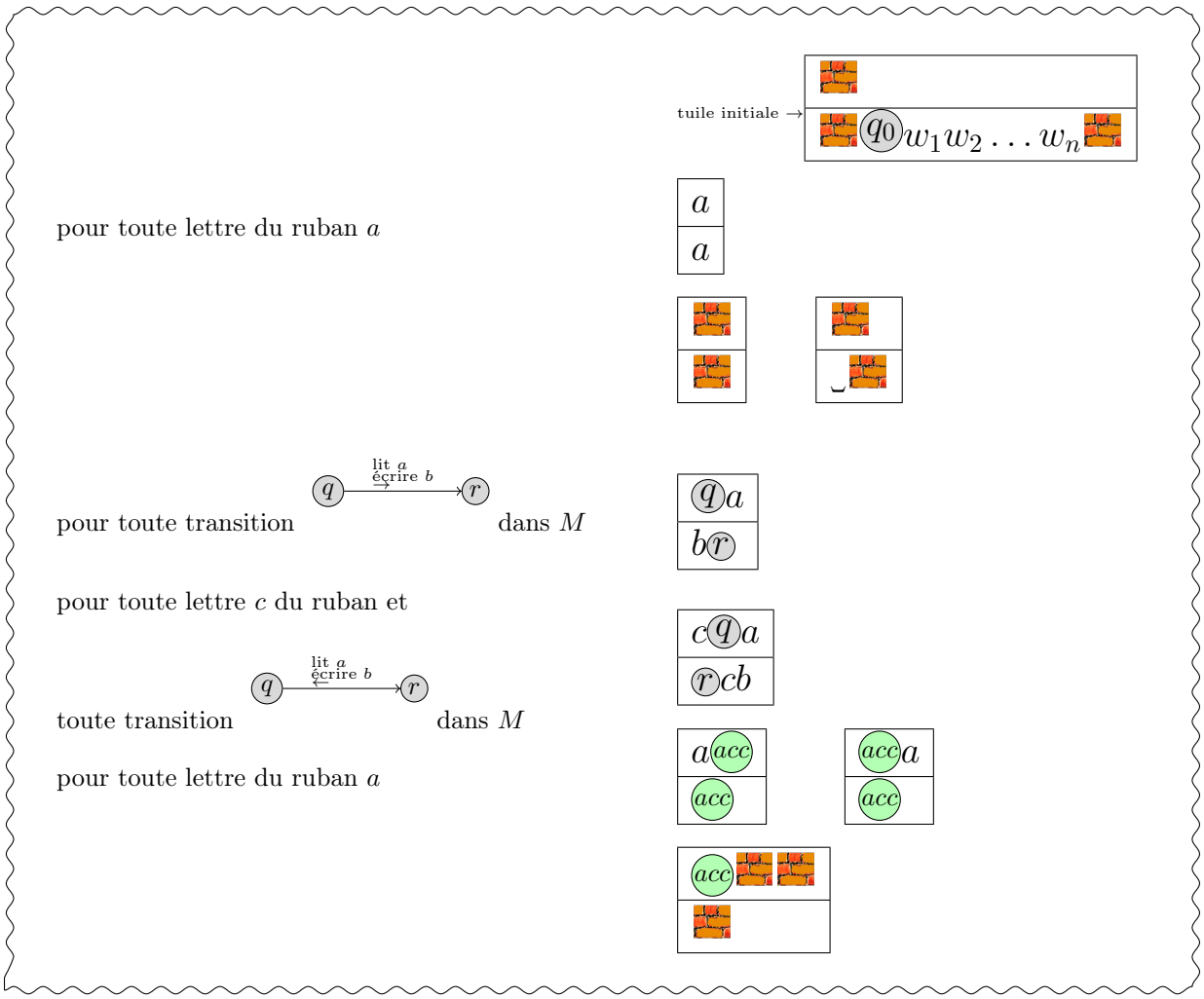
5.2 Démonstrations d'indécidabilité

Théorème 7 ***Post*$_{\text{marqué}}$** *est indécidable.*

DÉMONSTRATION.
[Sip06]
Définissons une réduction tr de **Acceptation**$_{MT}$ dans **Post**$_{\text{marqué}}$.



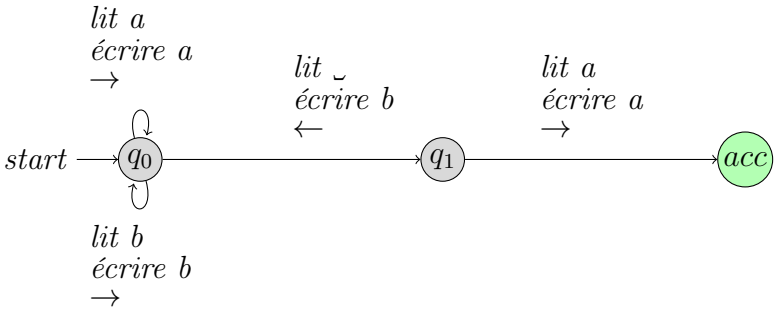
$tr(M, w_1 w_2 \dots w_n)$ est le système de tuiles ci-dessous :



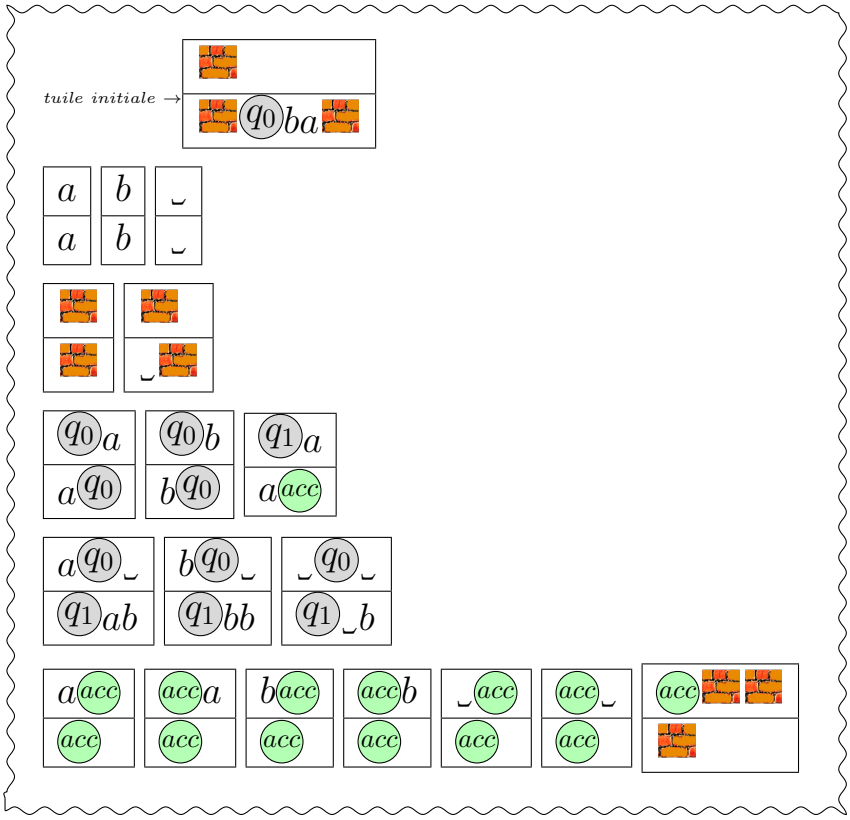
1. tr est une fonction calculable ;
2. M accepte w ssi $tr(M, w)$ est une instance positive de **Post**$_{\text{marqué}}$.

Comme **Acceptation**$_{MT}$ est indécidable, **Post**$_{\text{marqué}}$ est indécidable.
■

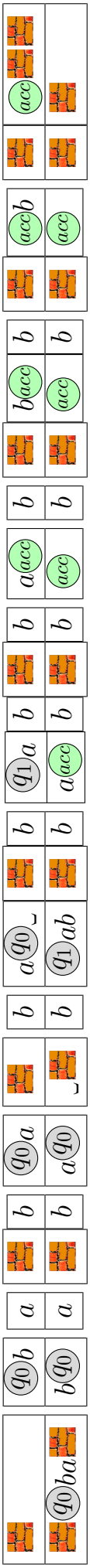
Exemple 3 *Considérons la machine de Turing M suivante qui accepte les mots qui finissent par un a :*



et le mot $w = ba$.

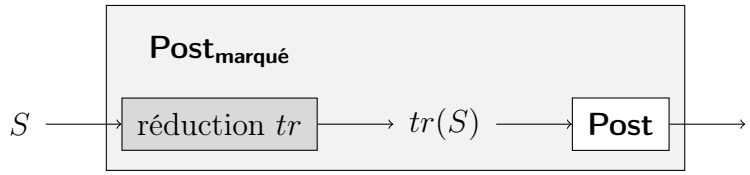


L'exécution acceptante de M sur aba est représentée par la suite de tuiles sur le bord droit de la page.



Théorème 8 ***Post*** *est indécidable.*

DÉMONSTRATION.
[Sip06] Définissons une réduction de $\textbf{Post}_{\textbf{marqué}}$ dans **Post**.



Pour tout système de tuiles S, $tr(S)$ est définie comme :

Si la tuile initiale de S est $\begin{array}{|c|}\hline h_1h_2\dots h_\ell \\ \hline b_1b_2\dots b_k \\ \hline \end{array}$ alors $tr(S)$ contient $\begin{array}{|c|}\hline \star h_1\star h_2\dots\star h_\ell \\ \hline \star b_1\star b_2\star\dots\star b_k\star \\ \hline \end{array}$

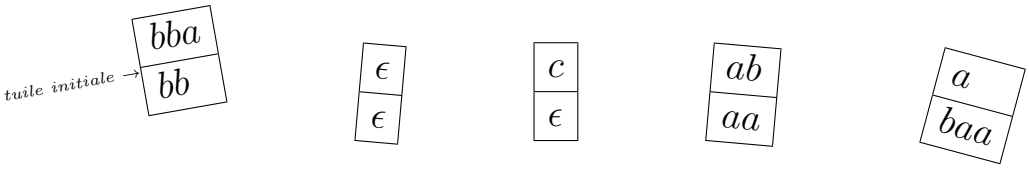
Si S contient $\begin{array}{|c|}\hline h_1h_2\dots h_\ell \\ \hline b_1b_2\dots b_k \\ \hline \end{array}$ alors $tr(S)$ contient $\begin{array}{|c|}\hline \star h_1\star h_2\dots\star h_\ell \\ \hline b_1\star b_2\star\dots\star b_k\star \\ \hline \end{array}$

$\begin{array}{|c|}\hline \star\,\blacksquare \\ \hline \blacksquare \\ \hline \end{array}$

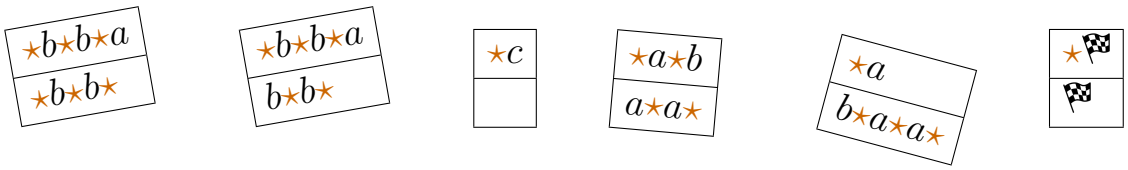
1. tr est une fonction calculable ;
2. S instance positive de $\textbf{Post}_{\textbf{marqué}}$ ssi $tr(S)$ instance positive de **Post**.

Comme $\textbf{Post}_{\textbf{marqué}}$ est indécidable, **Post** est indécidable.
■

Exemple 4 *L'instance de* ***Post***$_{\textbf{marqué}}$



est transformée en l'instance de ***Post*** *suivante :*



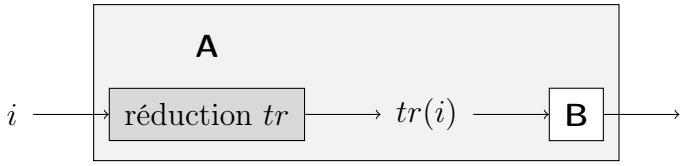
6 Notes bibliographiques

6.1 Problème de l'arrêt

Dans [Wol06], il introduit le langage LU, etc. La présentation est longue mais intéressante avant d'arriver... au problème de l'arrêt. J'ai préféré ici une démonstration plus directe. Dans [Sip06], le problème présenté dans la section "problème de l'arrêt" est le problème de l'acceptation. Le véritable problème de l'arrêt est présenté après.

6.2 Réduction

Je me suis inspiré de [DPV06] pour les schémas de réductions :



Références

[DPV06] Sanjoy Dasgupta, Christos H Papadimitriou, and Umesh Virkumar Vazirani. Algorithms. 2006.

[Sip06] M. Sipser. *Introduction to the Theory of Computation*, volume 27. Thomson Course Technology Boston, MA, 2006.

[Wol06] Pierre Wolper. *Introduction à la calculabilité*. Dunod, 2006.