UPEC
UNIVERSITÉ PARIS-EST CRÉTEIL
EPISEN

Décidabilité et Calculabilité

Calculabilité et Complexité

Paul Brunet

Décidabilité et Calculabilité
Plan

1. Langages récursivement énumérables
2. Langages décidables
3. Calculer avec une machine de Turing
4. Machine universelle

Décidabilité et Calculabilité
Plan

1. Langages récursivement énumérables
2. Langages décidables
3. Calculer avec une machine de Turing
4. Machine universelle

Machine à énumérer

Parfois, on souhaite énumérer les mots d'un langage.

Machine à énumérer

- Parfois, on souhaite énumérer les mots d'un langage.
- Pour cela, on peut utiliser une machine qui s'exécute à partir d'un ruban vide, et qui périodiquement « imprime » le contenu de son ruban sur une sortie observable.

Machine à énumérer

- Parfois, on souhaite énumérer les mots d'un langage.
- Pour cela, on peut utiliser une machine qui s'exécute à partir d'un ruban vide, et qui périodiquement « imprime » le contenu de son ruban sur une sortie observable.

définition (Machine avec impression)

Une machine avec impression est une structure $\mathcal{M} := \langle Q, \Sigma, \Gamma, \Delta, q_0, q_{\text{print}} \rangle$, où $q_{\text{print}} \in Q$ est un **état d'impression**.
Le langage énuméré par la machine est l'ensemble $\mathcal{L}_{\text{print}}(\mathcal{M})$ défini par :

$$\mathcal{L}_{\text{print}}(\mathcal{M}) := \left\{ w \in \Sigma^\star \mid q_0\, \varepsilon \to^*_{\mathcal{M}} q_{\text{print}}\, w \right\}.$$

Un langage L est récursivement énumérable si il existe une machine $\mathcal{M}$ telle que $L = \mathcal{L}_{\text{print}}(\mathcal{M})$.

Machine à énumérer

- Parfois, on souhaite énumérer les mots d'un langage.
- Pour cela, on peut utiliser une machine qui s'exécute à partir d'un ruban vide, et qui périodiquement « imprime » le contenu de son ruban sur une sortie observable.

définition (Machine avec impression)

Une machine avec impression est une structure $\mathcal{M} := \langle Q, \Sigma, \Gamma, \Delta, q_0, q_{\text{print}} \rangle$, où $q_{\text{print}} \in Q$ est un **état d'impression**.
Le langage énuméré par la machine est l'ensemble $\mathcal{L}_{\text{print}}(\mathcal{M})$ défini par :

$$\mathcal{L}_{\text{print}}(\mathcal{M}) := \left\{ w \in \Sigma^\star \mid q_0\, \varepsilon \to^*_{\mathcal{M}} q_{\text{print}}\, w \right\}.$$

Un langage L est récursivement énumérable si il existe une machine $\mathcal{M}$ telle que $L = \mathcal{L}_{\text{print}}(\mathcal{M})$.

exercice

Montrer qu'un langage est récursivement énumérable si et seulement si il est reconnaissable.

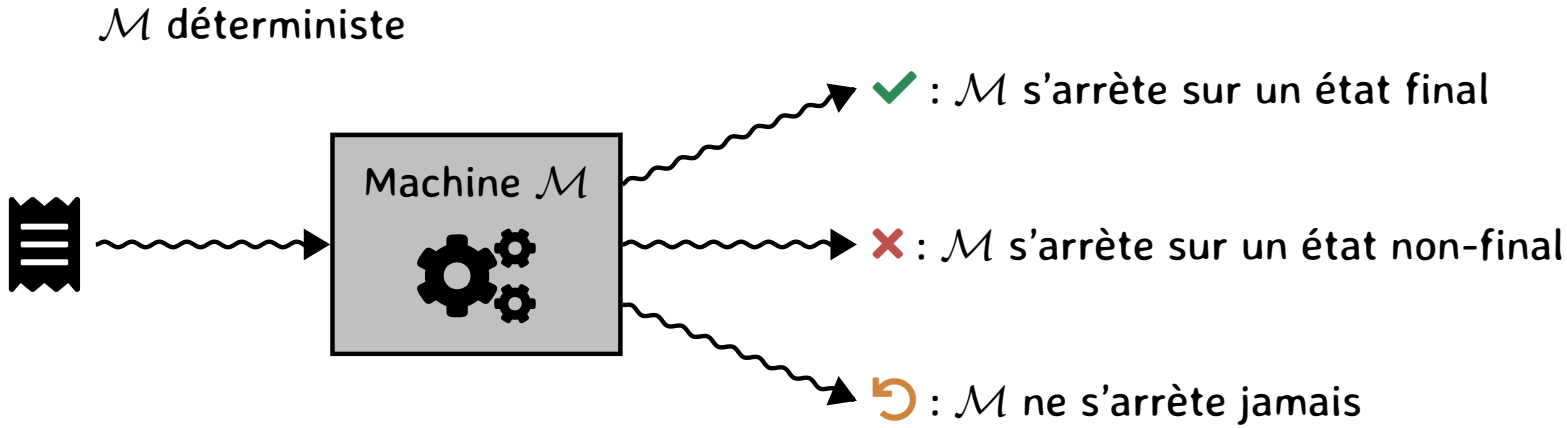
Décidabilité et Calculabilité
Plan

1. Langages récursivement énumérables
2. Langages décidables
3. Calculer avec une machine de Turing
4. Machine universelle

Langages reconnaissables

☞ Jusqu'ici, on a utilisé les machines de Turing pour reconnaître des langages :

$$\mathcal{L}(\mathcal{M}) := \{w \in \Sigma^\star \mid q_0\, w \to^*_{\mathcal{M}} C \text{ acceptante}\}\,.$$

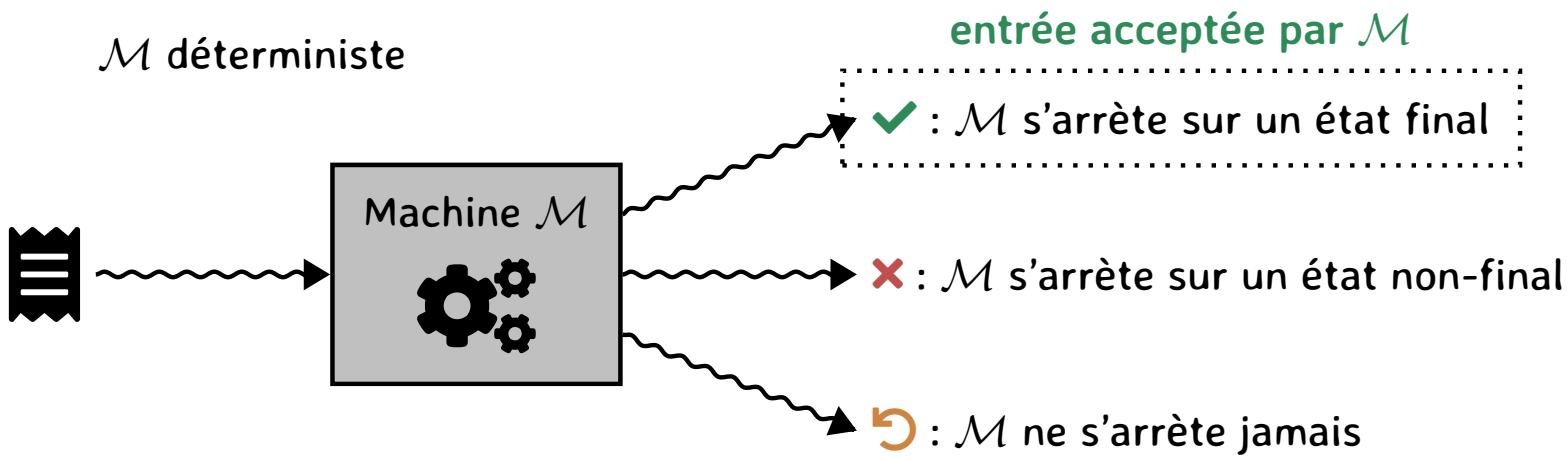


Langages reconnaissables

☞ Jusqu'ici, on a utilisé les machines de Turing pour reconnaître des langages :

$$\mathcal{L}(\mathcal{M}) := \{w \in \Sigma^\star \mid q_0\, w \to^*_{\mathcal{M}} C \text{ acceptante}\}\,.$$

☞ Si un mot est dans le langage ciblé, alors il existe une exécution qui l'accepte.



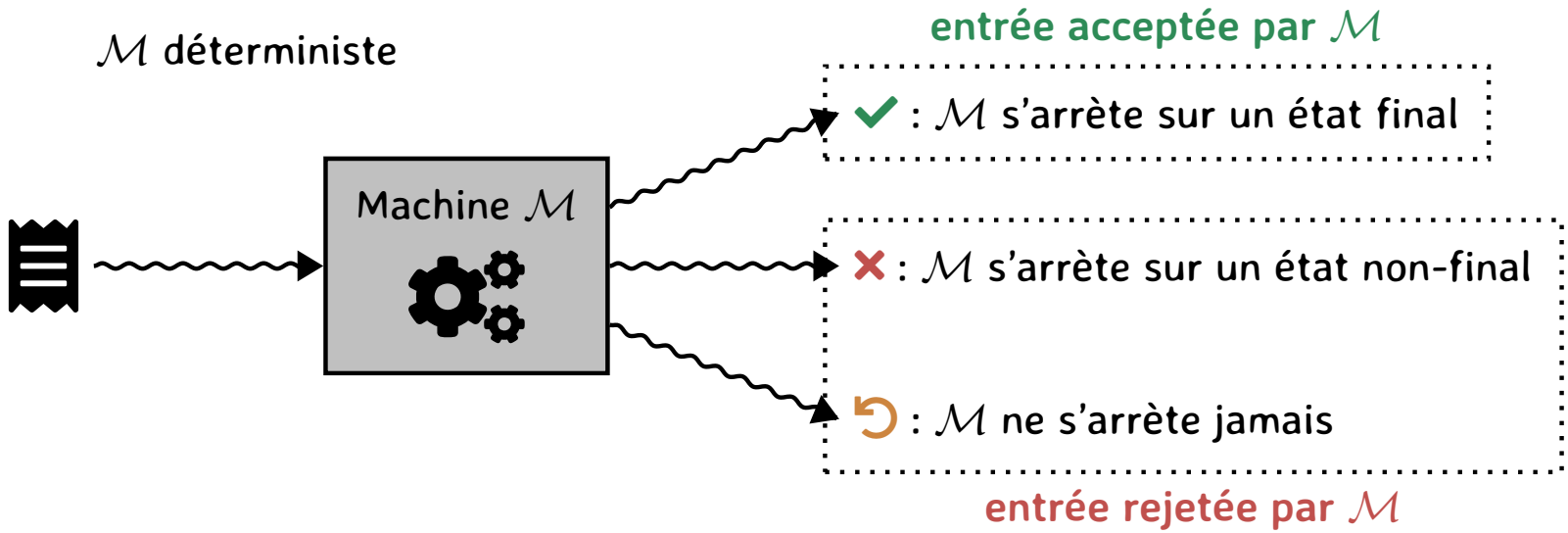
Langages reconnaissables

☞ Jusqu'ici, on a utilisé les machines de Turing pour reconnaître des langages :

$$\mathcal{L}(\mathcal{M}) := \{w \in \Sigma^\star \mid q_0\, w \to^*_{\mathcal{M}} C \text{ acceptante}\}\,.$$

☞ Si un mot est dans le langage ciblé, alors il existe une exécution qui l'accepte.

☞ En revanche, si le mot n'appartient pas au langage, c'est plus compliqué.



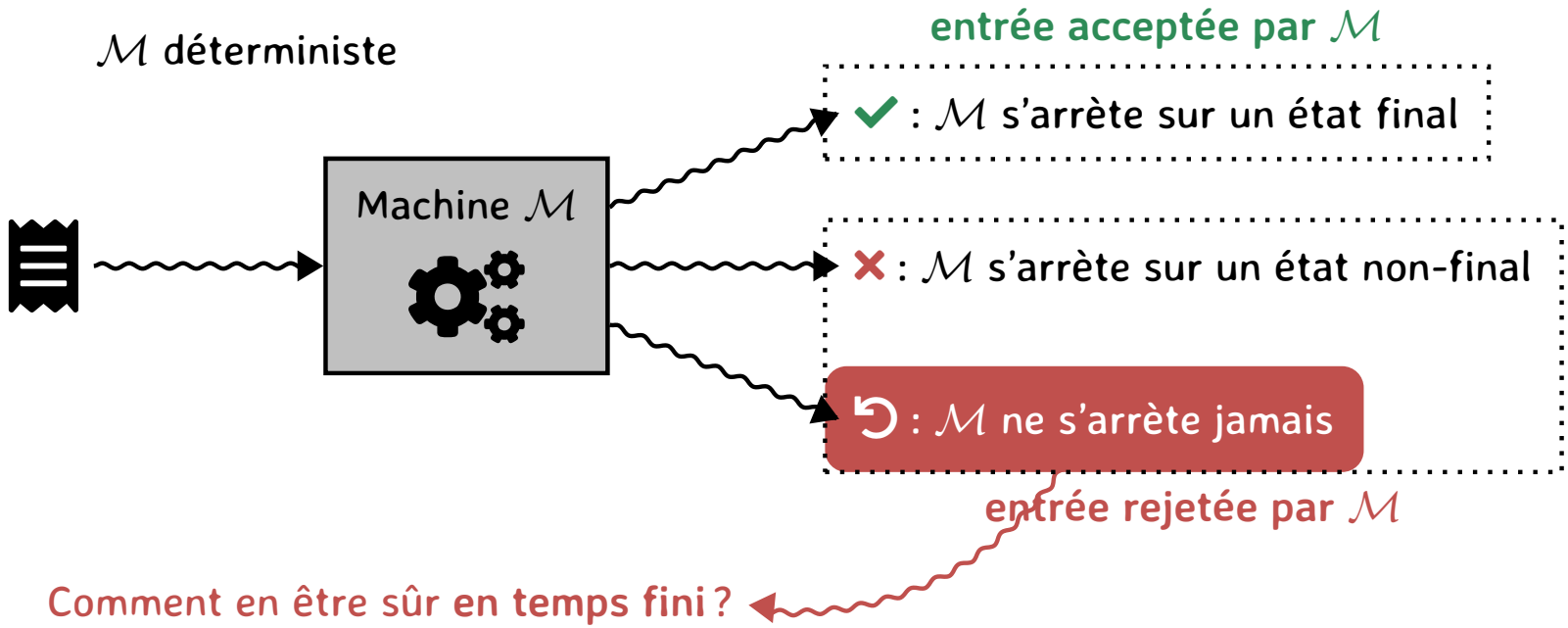
Langages reconnaissables

☞ Jusqu'ici, on a utilisé les machines de Turing pour reconnaître des langages :

$$\mathcal{L}(\mathcal{M}) := \{w \in \Sigma^\star \mid q_0\, w \to^*_{\mathcal{M}} C \text{ acceptante}\}\,.$$

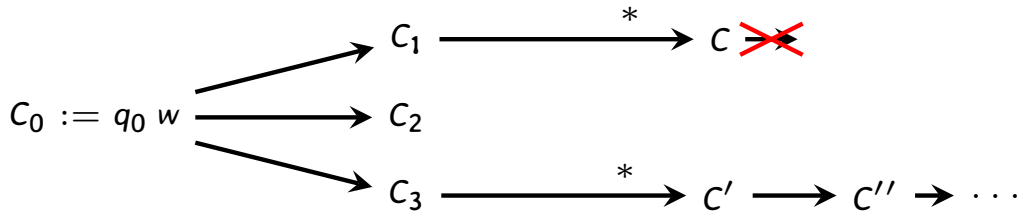
☞ Si un mot est dans le langage ciblé, alors il existe une exécution qui l'accepte.

☞ En revanche, si le mot n'appartient pas au langage, c'est plus compliqué.



Langages reconnaissables

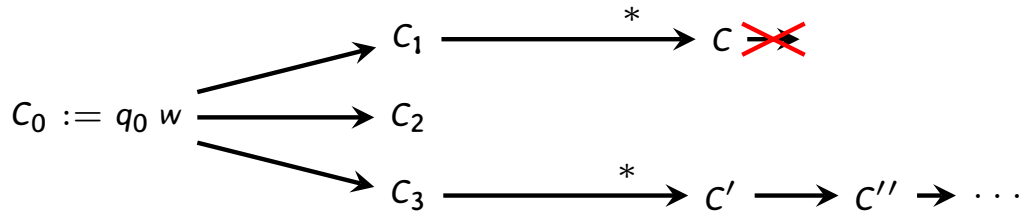
On fixe un mot $w \in \Sigma^\star$.



1. Pour tout n il existe une configuration C telle que $q_0 w \to^n C$.

Langages reconnaissables

On fixe un mot $w \in \Sigma^\star$.

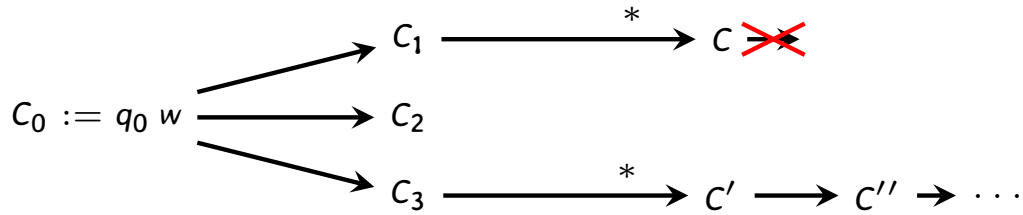


☞ **si toutes les exécutions s'arrêtent** : alors il y a un nombre fini de configurations accessibles, et on peut vérifier si l'une d'elles est acceptantes.

1. Pour tout n il existe une configuration C telle que $q_0 w \to^n C$.

Langages reconnaissables

On fixe un mot $w \in \Sigma^\star$.

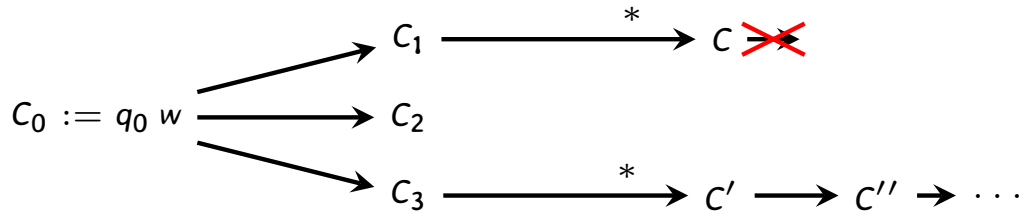


- **si toutes les exécutions s'arrêtent** : alors il y a un nombre fini de configurations accessibles, et on peut vérifier si l'une d'elles est acceptantes.
- **si le mot est dans le langage reconnu** : alors il existe une configuration C acceptante et accessible ;

1. Pour tout n il existe une configuration C telle que $q_0 w \to^n C$.

Langages reconnaissables

On fixe un mot $w \in \Sigma^\star$.

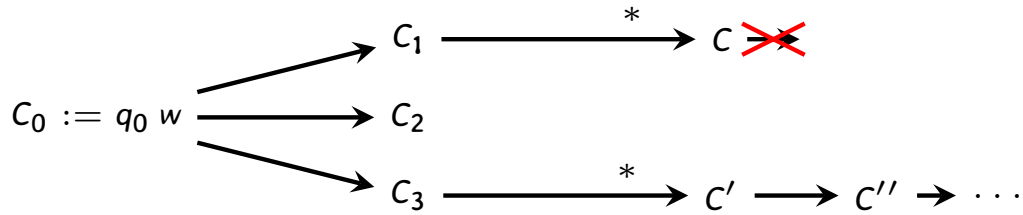


- **si toutes les exécutions s'arrêtent** : alors il y a un nombre fini de configurations accessibles, et on peut vérifier si l'une d'elles est acceptantes.
- **si le mot est dans le langage reconnu** : alors il existe une configuration C acceptante et accessible ;
  - $\forall k \in \mathbb{N}$, il y a un nombre fini de configurations accessibles en moins de k étapes ($\leqslant |\Delta|^n$) ;

1. Pour tout n il existe une configuration C telle que $q_0 w \to^n C$.

Langages reconnaissables

On fixe un mot $w \in \Sigma^\star$.

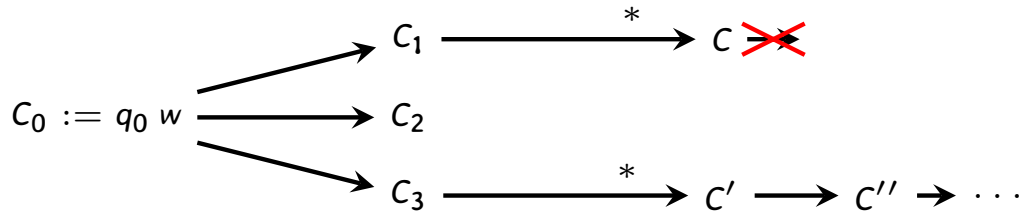


- **si toutes les exécutions s'arrêtent** : alors il y a un nombre fini de configurations accessibles, et on peut vérifier si l'une d'elles est acceptantes.
- **si le mot est dans le langage reconnu** : alors il existe une configuration C acceptante et accessible ;
  - $\forall k \in \mathbb{N}$, il y a un nombre fini de configurations accessibles en moins de k étapes ($\leqslant |\Delta|^n$) ;
  - on peut les énumérer toutes pour $k = 0, 1, 2, \ldots$;

1. Pour tout n il existe une configuration C telle que $q_0 w \to^n C$.

Langages reconnaissables

On fixe un mot $w \in \Sigma^\star$.



☞ **si toutes les exécutions s'arrêtent** : alors il y a un nombre fini de configurations accessibles, et on peut vérifier si l'une d'elles est acceptantes.

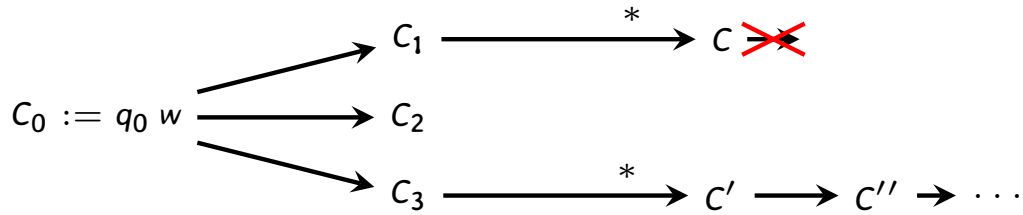
☞ **si le mot est dans le langage reconnu** : alors il existe une configuration C acceptante et accessible ;

- $\forall k \in \mathbb{N}$, il y a un nombre fini de configurations accessibles en moins de k étapes ($\leqslant |\Delta|^n$) ;
- on peut les énumérer toutes pour $k = 0, 1, 2, \ldots$;
- comme C est accessible, elle est atteinte en un certain nombre n d'étapes. Donc on va trouver C lorsqu'on considère $k = n$.

1. Pour tout n il existe une configuration C telle que $q_0 w \to^n C$.

Langages reconnaissables

On fixe un mot $w \in \Sigma^\star$.



- **si toutes les exécutions s'arrêtent** : alors il y a un nombre fini de configurations accessibles, et on peut vérifier si l'une d'elles est acceptantes.
- **si le mot est dans le langage reconnu** : alors il existe une configuration C acceptante et accessible ;
  - $\forall k \in \mathbb{N}$, il y a un nombre fini de configurations accessibles en moins de k étapes ($\leqslant |\Delta|^n$) ;
  - on peut les énumérer toutes pour $k = 0, 1, 2, \ldots$;
  - comme C est accessible, elle est atteinte en un certain nombre n d'étapes. Donc on va trouver C lorsqu'on considère $k = n$.
- **sinon** : le mot n'appartient pas au langage reconnu, et la machine ne termine pas forcément [1] sur ce mot, alors on ne sait pas quoi dire !

1. Pour tout n il existe une configuration C telle que $q_0 w \to^n C$.

Machines à décider

On veut de machines qui peuvent nous dire si le mot appartient ou non au langage ciblé.

définition (Machine à décision)

Une machine à décision est une structure $\mathcal{M} := \langle Q, \Sigma, \Gamma, \Delta, q_0, q_{\text{oui}}, q_{\text{non}} \rangle$ avec :

- Q un ensemble fini d'états,Σ et Γ des alphabets d'entrée et de ruban, avec $\Sigma \cup \{\#\} \subseteq \Gamma$;
- $q_0, q_{\text{oui}}, q_{\text{non}} \in Q$ trois états, respectivement initial, acceptant et rejettant ;
- $\Delta \subseteq Q \times \Gamma \times \Gamma \times \{\blacktriangleleft, \blacktriangledown, \blacktriangleright\} \times Q$ une relation de transitions.

Machines à décider

On veut de machines qui peuvent nous dire si le mot appartient ou non au langage ciblé.

définition (Machine à décision)

Une machine à décision est une structure $\mathcal{M} := \langle Q, \Sigma, \Gamma, \Delta, q_0, q_{\text{oui}}, q_{\text{non}} \rangle$ avec :

- Q un ensemble fini d'états,Σ et Γ des alphabets d'entrée et de ruban, avec $\Sigma \cup \{\#\} \subseteq \Gamma$;
- $q_0, q_{\text{oui}}, q_{\text{non}} \in Q$ trois états, respectivement initial, acceptant et rejettant ;
- $\Delta \subseteq Q \times \Gamma \times \Gamma \times \{\blacktriangleleft, \blacktriangledown, \blacktriangleright\} \times Q$ une relation de transitions.

Cela nous permet de définir les langages décidables :

définition (Langages décidables)

Un langage $L \subseteq \Sigma^\star$ est décidable si il existe une machine à décision $\mathcal{M}$ telle que pour tout mot $w \in \Sigma^\star$ on a :

$$w \in L \Leftrightarrow \exists u, v \in \Gamma^\star : \; q_0\, w \to_{\mathcal{M}}^* u\, q_{\text{oui}}\, v$$
$$w \notin L \Leftrightarrow \exists u, v \in \Gamma^\star : \; q_0\, w \to_{\mathcal{M}}^* u\, q_{\text{non}}\, v$$

Langages décidables vs. reconnaissables

définition (Langages décidables)

Un langage $L \subseteq \Sigma^\star$ est décidable si il existe une machine à décision $\mathcal{M}$ telle que pour tout mot $w \in \Sigma^\star$ on a :

$$w \in L \Leftrightarrow \exists u, v \in \Gamma^\star : \; q_0\, w \to_{\mathcal{M}}^* u\, q_{\text{oui}}\, v$$
$$w \notin L \Leftrightarrow \exists u, v \in \Gamma^\star : \; q_0\, w \to_{\mathcal{M}}^* u\, q_{\text{non}}\, v$$

théorème

Soit $L \subseteq \Sigma^\star$ un langage. Les propositions suivantes sont équivalentes :

1) L est décidable ;
2) L et L^c sont tous les deux reconnaissables.

L^c : complément de L, autrement dit $L^c := \Sigma^\star \setminus L$.

Preuve du théorème

Prouvons que si L est décidable, alors L et L^c sont reconnaissables.

☞ L décidable signifie qu'il existe une machine à décision $\mathcal{M} = \langle Q, \Sigma, \Gamma, \Delta, q_0, q_{\text{oui}}, q_{\text{non}} \rangle$ telle que pour tout mot $w \in \Sigma^\star$ on a :

$$w \in L \Leftrightarrow \exists u, v \in \Gamma^\star : \; q_0\, w \to^*_{\mathcal{M}} u\, q_{\text{oui}}\, v$$
$$w \notin L \Leftrightarrow \exists u, v \in \Gamma^\star : \; q_0\, w \to^*_{\mathcal{M}} u\, q_{\text{non}}\, v$$

☞ On construit deux machines comme suit :
- $\mathcal{M}_{\text{oui}} := \langle Q, \Sigma, \Gamma, \Delta, q_0, \{q_{\text{oui}}\} \rangle$
- $\mathcal{M}_{\text{non}} := \langle Q, \Sigma, \Gamma, \Delta, q_0, \{q_{\text{non}}\} \rangle$

☞ Comme ces deux machines ont les mêmes états et les mêmes transitions que $\mathcal{M}$, on a pour $\mathcal{X} \in \{\mathcal{M}_{\text{oui}}, \mathcal{M}_{\text{non}}\}$:

$$\forall u, u', v, v' \in \Gamma^\star, \forall q_1, q_2 \in Q : \; u\, q_1\, v \to^*_{\mathcal{M}} u'\, q_2\, v' \Leftrightarrow u\, q_1\, v \to^*_{\mathcal{X}} u'\, q_2\, v'$$

☞ On en déduit :

$$\begin{aligned} w \in L &\Leftrightarrow \exists u, v \in \Gamma^\star : \; q_0\, w \to^*_{\mathcal{M}} u\, q_{\text{oui}}\, v \\ &\Leftrightarrow \exists u, v \in \Gamma^\star : \; q_0\, w \to^*_{\mathcal{M}_{\text{oui}}} u\, q_{\text{oui}}\, v \\ &\Leftrightarrow w \in \mathcal{L}(\mathcal{M}_{\text{oui}}) \\ w \in L^c &\Leftrightarrow w \notin L \\ &\Leftrightarrow \exists u, v \in \Gamma^\star : \; q_0\, w \to^*_{\mathcal{M}} u\, q_{\text{non}}\, v \\ &\Leftrightarrow \exists u, v \in \Gamma^\star : \; q_0\, w \to^*_{\mathcal{M}_{\text{non}}} u\, q_{\text{non}}\, v \\ &\Leftrightarrow w \in \mathcal{L}(\mathcal{M}_{\text{non}}) \end{aligned}$$

□

Preuve du théorème

Prouvons que si L et L^c sont tous les deux reconnaissables, alors L est décidable.

☞ On suppose connues $\mathcal{M}_+$ et $\mathcal{M}_-$, deux machines telles que :

$$\mathcal{L}(\mathcal{M}_+) = L \qquad\qquad \mathcal{L}(\mathcal{M}_-) = L^c.$$

On note $\mathcal{M}_+ = \langle Q_+, \Sigma, \Gamma_+, \Delta_+, q_+, F_+ \rangle$, et similairement pour $\mathcal{M}_-$. De plus, on suppose (sans perte de généralité) que Q_+ et Q_- sont disjoints.

☞ On choisit trois nouveaux états $q_0, q_{\text{oui}}, q_{\text{non}} \notin Q_+ \cup Q_-$.

☞ On peut alors construire une machine $\mathcal{M}$ comme suit :

$$\mathcal{M} = \left\langle \{q_0, q_{\text{oui}}, q_{\text{non}}\} \cup Q_+ \cup Q_-, \Sigma, \Gamma_+ \cup \Gamma_-, \Delta' \cup \Delta_+ \cup \Delta_-, q_0, q_{\text{oui}}, q_{\text{non}} \right\rangle$$

avec :

$$\begin{aligned}\Delta' := \; & \{q_0 \xrightarrow{a/a,\blacktriangledown} q_+ \mid a \in \Sigma\} \cup \{q_0 \xrightarrow{a/a,\blacktriangledown} q_- \mid a \in \Sigma\} \\ & \cup \left\{ q_f^+ \xrightarrow{a/a,\blacktriangledown} q_{\text{oui}} \;\middle|\; a \in \Gamma,\, q_f^+ \in F_+ \right\} \\ & \cup \left\{ q_f^- \xrightarrow{a/a,\blacktriangledown} q_{\text{non}} \;\middle|\; a \in \Gamma,\, q_f^- \in F_- \right\}.\end{aligned}$$

On notera $Q = \{q_0, q_{\text{oui}}, q_{\text{non}}\} \cup Q_+ \cup Q_-$.

Suite de la preuve du théorème

☞ Avec cette définition on peut facilement établir :

- $\forall u, u', v, v' \in \Gamma^\star,\ \forall q_1, q_2 \in Q_+ :\ u\, q_1\, v \to^*_{\mathcal{M}} u'\, q_2\, v' \Leftrightarrow u\, q_1\, v \to^*_{\mathcal{M}_+} u'\, q_2\, v'$
- $\forall w, u', v' \in \Gamma^\star,\ \forall q \in Q_+ :\ q_0\, w \to^*_{\mathcal{M}} u'\, q\, v' \Leftrightarrow q_+\, w \to^*_{\mathcal{M}} u'\, q\, v'$
- $\forall u, u', v, v' \in \Gamma^\star,\ \forall q \in Q :\ u\, q\, v \to^*_{\mathcal{M}} u'\, q_{\text{oui}}\, v' \Leftrightarrow \exists q_f^+ \in F_+,\ u\, q\, v \to^*_{\mathcal{M}} u'\, q_f^+\, v'$

On peut également obtenir des propriétés similaires pour $\mathcal{M}_-$, q_- et q_{non}.

☞ On obtient ainsi :

$$
\begin{aligned}
w \in L &\Leftrightarrow \exists u, v \in \Gamma^\star,\ \exists q_f^+ \in F_+ :\ q_+\, w \to^*_{\mathcal{M}_+} u\, q_f^+\, v \\
&\Leftrightarrow \exists u, v \in \Gamma^\star,\ \exists q_f^+ \in F_+ :\ q_+\, w \to^*_{\mathcal{M}} u\, q_f^+\, v \\
&\Leftrightarrow \exists u, v \in \Gamma^\star,\ \exists q_f^+ \in F_+ :\ q_0\, w \to^*_{\mathcal{M}} u\, q_f^+\, v \\
&\Leftrightarrow \exists u, v \in \Gamma^\star :\ q_0\, w \to^*_{\mathcal{M}} u\, q_{\text{oui}}\, v \\
w \notin L \Leftrightarrow w \in L^c &\Leftrightarrow \exists u, v \in \Gamma^\star,\ \exists q_f^- \in F_- :\ q_-\, w \to^*_{\mathcal{M}_-} u\, q_f^-\, v \\
&\Leftrightarrow \exists u, v \in \Gamma^\star,\ \exists q_f^- \in F_- :\ q_-\, w \to^*_{\mathcal{M}} u\, q_f^-\, v \\
&\Leftrightarrow \exists u, v \in \Gamma^\star,\ \exists q_f^- \in F_- :\ q_0\, w \to^*_{\mathcal{M}} u\, q_f^-\, v \\
&\Leftrightarrow \exists u, v \in \Gamma^\star :\ q_0\, w \to^*_{\mathcal{M}} u\, q_{\text{non}}\, v.
\end{aligned}
$$

☞ On a donc bien montré que $\mathcal{M}$ décide le langage L. □

Langages décidables vs. reconnaissables

définition (Langages décidables)

Un langage $L \subseteq \Sigma^\star$ est décidable si il existe une machine à décision $\mathcal{M}$ telle que pour tout mot $w \in \Sigma^\star$ on a :

$$w \in L \Leftrightarrow \exists u, v \in \Gamma^\star : \; q_0\, w \to^*_{\mathcal{M}} u\, q_{\text{oui}}\, v$$
$$w \notin L \Leftrightarrow \exists u, v \in \Gamma^\star : \; q_0\, w \to^*_{\mathcal{M}} u\, q_{\text{non}}\, v$$

théorème

Soit $L \subseteq \Sigma^\star$ un langage. Les propositions suivantes sont équivalentes :

1) L est décidable ;
2) L et L^c sont tous les deux reconnaissables.

L^c : complément de L, autrement dit $L^c := \Sigma^\star \setminus L$.

Exercice

définition

Une machine $\mathcal{M}$ (non-déterministe) s'arrête toujours si pour tout mot w, toutes les exécutions de $\mathcal{M}$ sur w sont de longueur bornée :

$$\forall w \in \Sigma^\star,\ \exists N_w \in \mathbb{N} :\ \forall\, (C_i)_{0 \leqslant i \leqslant n} :\ (C_0 = q_0\, w\ \wedge\ \forall i,\ C_i \to_{\mathcal{M}} C_{i+1}) \Rightarrow n \leqslant N_w.$$

exercice

Montrer qu'un langage est décidable si et seulement si il est reconnaissable par une machine qui s'arrête toujours.

Décidabilité et Calculabilité
Plan

1. Langages récursivement énumérables
2. Langages décidables

3. Calculer avec une machine de Turing
4. Machine universelle

Calculer avec une machine de Turing

☞ Jusqu'ici, nous avons utilisé les machines de Turing pour **tester** si un mot w appartient à un langage L.

Calculer avec une machine de Turing

- Jusqu'ici, nous avons utilisé les machines de Turing pour **tester** si un mot w appartient à un langage L.
- Autrement dit, nos machines implémentent des **prédicats booléens**, soit des fonctions des mots vers l'ensemble à deux éléments :

$$f : \Sigma^\star \to \mathbb{B} = \{\top, \bot\}$$

$$w \mapsto \begin{cases} \top & w \in L \\ \bot & w \notin L \end{cases}$$

Calculer avec une machine de Turing

- Jusqu'ici, nous avons utilisé les machines de Turing pour **tester** si un mot w appartient à un langage L.
- Autrement dit, nos machines implémentent des **prédicats booléens**, soit des fonctions des mots vers l'ensemble à deux éléments :

$$f : \Sigma^\star \to \mathbb{B} = \{\top, \bot\}$$

$$w \mapsto \begin{cases} \top & w \in L \\ \bot & w \notin L \end{cases}$$

- On va maintenant voir comment on peut utiliser ces machines pour calculer des fonctions de type $\Sigma^\star \to \Gamma^\star$.

Calculer avec une machine de Turing

- Jusqu'ici, nous avons utilisé les machines de Turing pour **tester** si un mot w appartient à un langage L.
- Autrement dit, nos machines implémentent des **prédicats booléens**, soit des fonctions des mots vers l'ensemble à deux éléments :

$$f : \Sigma^\star \to \mathbb{B} = \{\top, \bot\}$$

$$w \mapsto \begin{cases} \top & w \in L \\ \bot & w \notin L \end{cases}$$

- On va maintenant voir comment on peut utiliser ces machines pour calculer des fonctions de type $\Sigma^\star \to \Gamma^\star$.
- Les codages nous permettaient d'utiliser les langages pour parler des problèmes de décision.

Calculer avec une machine de Turing

- Jusqu'ici, nous avons utilisé les machines de Turing pour **tester** si un mot w appartient à un langage L.
- Autrement dit, nos machines implémentent des **prédicats booléens**, soit des fonctions des mots vers l'ensemble à deux éléments :

$$f : \Sigma^\star \to \mathbb{B} = \{\top, \bot\}$$

$$w \mapsto \begin{cases} \top & w \in L \\ \bot & w \notin L \end{cases}$$

- On va maintenant voir comment on peut utiliser ces machines pour calculer des fonctions de type $\Sigma^\star \to \Gamma^\star$.
- Les codages nous permettaient d'utiliser les langages pour parler des problèmes de décision.
- Ils nous permettront ici d'utiliser ces fonctions pour parler de fonctions de type (presque) arbitraire $A \to B$.

Fonctions calculables

Soient Σ et Σ' deux alphabets, et $\varphi : \Sigma^\star \to \Sigma'^\star$.

définition (Fonction directement calculable)

φ est **directement calculable** si il existe une machine $\mathcal{M}$ avec un alphabet de ruban $\Gamma \supseteq \Sigma \cup \Sigma'$ telle que :

$$\forall w \in \Sigma^\star,\ \exists u, v \in \Gamma^\star,\ \exists q \in F :\ q_0\, w \to^*_{\mathcal{M}} u\, q\, v \ \wedge\ \varphi(w) = u \cdot v.$$
$$\forall w \in \Sigma^\star,\ \forall u, v \in \Gamma^\star,\ \forall q \in F :\ q_0\, w \to^*_{\mathcal{M}} u\, q\, v \Rightarrow \varphi(w) = u \cdot v.$$

Fonctions calculables

Soient Σ et Σ' deux alphabets, et $\varphi : \Sigma^\star \to \Sigma'^\star$.

définition (Fonction directement calculable)

φ est **directement calculable** si il existe une machine $\mathcal{M}$ avec un alphabet de ruban $\Gamma \supseteq \Sigma \cup \Sigma'$ telle que :

$$\forall w \in \Sigma^\star,\ \exists u, v \in \Gamma^\star,\ \exists q \in F :\ q_0\, w \to^*_{\mathcal{M}} u\, q\, v \ \wedge\ \varphi(w) = u \cdot v.$$
$$\forall w \in \Sigma^\star,\ \forall u, v \in \Gamma^\star,\ \forall q \in F :\ q_0\, w \to^*_{\mathcal{M}} u\, q\, v \Rightarrow \varphi(w) = u \cdot v.$$

Soient A et B deux ensembles, τ, τ' deux **codages**, et φ une fonction tels que :

$$\tau : A \to \Sigma^\star \qquad \tau' : B \to \Sigma'^\star \qquad \varphi : A \to B.$$

définition (Fonction calculable)

φ est **calculable** si il existe une machine $\mathcal{M}$ avec un alphabet de ruban $\Gamma \supseteq \Sigma \cup \Sigma'$ telle que :

$$\forall a \in A,\ \exists u, v \in \Gamma^\star,\ \exists q \in F :\ q_0\, \tau(a) \to^*_{\mathcal{M}} u\, q\, v \ \wedge\ \tau'(\varphi(a)) = u \cdot v.$$
$$\forall a \in A,\ \forall u, v \in \Gamma^\star,\ \forall q \in F :\ q_0\, \tau(a) \to^*_{\mathcal{M}} u\, q\, v \Rightarrow \tau'(\varphi(a)) = u \cdot v.$$

Fonctions calculables

Soient Σ et Σ' deux alphabets, et $\varphi : \Sigma^\star \to \Sigma'^\star$.

définition (Fonction directement calculable)

φ est **directement calculable** si il existe une machine $\mathcal{M}$ avec un alphabet de ruban $\Gamma \supseteq \Sigma \cup \Sigma'$ telle que :

$$\forall w \in \Sigma^\star,\ \exists u, v \in \Gamma^\star,\ \exists q \in F :\ q_0\, w \to^*_{\mathcal{M}} u\, q\, v \ \wedge\ \varphi(w) = u \cdot v.$$

$$\forall w \in \Sigma^\star,\ \forall u, v \in \Gamma^\star,\ \forall q \in F :\ q_0\, w \to^*_{\mathcal{M}} u\, q\, v \Rightarrow \varphi(w) = u \cdot v.$$

fonction injective $\varphi : I \to \Sigma^\star$, c'est à dire que :

$$\forall i, j \in I,\ \varphi(i) = \varphi(j) \Rightarrow i = j.$$

Soient A et B deux ensembles, τ, τ' deux **codages**, et φ une fonction tels que :

$$\tau : A \to \Sigma^\star \qquad \tau' : B \to \Sigma'^\star \qquad \varphi : A \to B.$$

définition (Fonction calculable)

φ est **calculable** si il existe une machine $\mathcal{M}$ avec un alphabet de ruban $\Gamma \supseteq \Sigma \cup \Sigma'$ telle que :

$$\forall a \in A,\ \exists u, v \in \Gamma^\star,\ \exists q \in F :\ q_0\, \tau(a) \to^*_{\mathcal{M}} u\, q\, v \ \wedge\ \tau'(\varphi(a)) = u \cdot v.$$

$$\forall a \in A,\ \forall u, v \in \Gamma^\star,\ \forall q \in F :\ q_0\, \tau(a) \to^*_{\mathcal{M}} u\, q\, v \Rightarrow \tau'(\varphi(a)) = u \cdot v.$$

Exercice

Soient Σ et Σ' deux alphabets, A et B deux ensembles, τ, τ' deux codages, et φ une fonction tels que :

$$\tau : A \to \Sigma^\star \qquad \tau' : B \to \Sigma'^\star \qquad \varphi : A \to B.$$

On suppose que $L_A = \{\tau(a) \in \Sigma^\star \mid a \in A\}$ est un langages reconnaissable.
On définit la fonction $\psi : \Sigma^\star \to \Sigma'^\star$ comme suit :

$$\psi(w) := \begin{cases} \tau'(\varphi(a)) & \text{si } \tau(a) = w \\ \varepsilon & \text{si } \forall a \in A,\ \tau(a) \neq w. \end{cases}$$

exercice

1) Montrer que ψ est bien définie, c'est à dire que chaque mot w a exactement une image par ψ.
2) Montrer que $\forall a \in A,\ \psi(\tau(a)) = \tau'(\varphi(a))$.
3) En déduire que φ est calculable si et seulement si la fonction ψ est directement calculable.

Lien avec les langages reconnaissables

Soient A et B deux ensembles, τ, τ' deux codages, et φ une fonctions, tels que :

$$\tau : A \to \Sigma^\star \qquad \tau' : B \to \Sigma'^\star \qquad \varphi : A \to B.$$

On suppose de plus que $L_A = \{\tau(a) \in \Sigma^\star \mid a \in A\}$ et $L_B = \{\tau'(b) \mid b \in B\}$ sont des langages reconnaissables.
Pour une paire $\langle a, b\rangle \in A \times B$, on définit le codage de $\langle a, b\rangle$ sur l'alphabet $\Sigma \cup \Sigma' \cup \{\S\}$ comme suit :

$$[\langle a, b\rangle]_{code} := \S\tau(a)\S\tau'(b) \in \left(\Sigma \cup \Sigma' \cup \{\S\}\right)^\star$$

théorème

φ est calculable si et seulement si le langage L_φ défini ci-dessous est reconnaissable.

$$L_\varphi := \left\{[\langle a, \varphi(a)\rangle]_{code} \in \left(\Sigma \cup \Sigma' \cup \{\S\}\right)^\star \mid a \in A\right\}$$

Calculabilité implique reconnaissabilité

$$L_\varphi := \left\{ [\langle a, \varphi(a)\rangle]_{code} \in \left(\Sigma \cup \Sigma' \cup \{\S\}\right)^\star \;\middle|\; a \in A \right\}$$

Preuve. Supposons φ calculable. On a donc une machine $\mathcal{M}$ telle que

$$\forall a \in A,\ \forall u, v \in \Gamma^\star,\ \left(\tau'(\varphi(a)) = u \cdot v\right) \Leftrightarrow \left(\exists q \in F :\ q_0\, \tau(a) \to_{\mathcal{M}}^{*} u\, q\, v\right).$$

Alors on peut construire une machine qui procède comme suit :

1) Elle vérifie que son entrée est bien formattée c'est à dire de la forme $\S u \S v$ avec $u \in L_A$ et $v \in L_B$.
2) comme $u \in L_A$, on sait que $u = \tau(a)$ pour un certain a, et $\mathcal{M}$ peut calculer le mot $w = \tau'(\varphi(a))$.
3) on vérifie que $w = v$.

□

Reconnaissabilité implique calculabilité

$$L_\varphi := \left\{ [\langle a, \varphi(a) \rangle]_{code} \in \left(\Sigma \cup \Sigma' \cup \{\S\}\right)^\star \;\middle|\; a \in A \right\}$$

Preuve. Supposons L_φ reconnaissable. On a donc une machine $\mathcal{M}$ telle que :

$$\forall w \in \left(\Sigma \cup \Sigma' \cup \{\S\}\right)^\star, \left(\exists q \in F : \exists u, v \in \Gamma^\star : q_0\, w \to^*_{\mathcal{M}} u\, q\, v\right) \\ \Leftrightarrow \left(\exists a \in A : w = [\langle a, \varphi(a) \rangle]_{code}\right).$$

On a également supposé que L_B était reconnaissable, et donc récursivement énumérable. On peut donc construire une machine qui fonctionne comme suit :

1) sur une entrée $u := \tau(a)$, on va énumérer les mots de L_B, c'est à dire les mots de la forme $\tau'(b)$, pour $b \in B$;
2) on choisit l'un d'eux, c'est à dire $v := \tau'(b)$, de manière non déterministe ;
3) on teste si $\S u \S v \in L_\varphi$;
4) si oui, on écrit v sur le ruban et on accepte.

Pour tout $a \in A$, on sait que $\tau'(\varphi(a)) \in L_B$, et donc dans la machine à imprimer on a une exécution :

$$q^B \to^* q_{\text{print}}\, \tau'(\varphi(a))$$

Comme $\S\tau(a)\S\tau'(\varphi(a)) = [\langle a, \varphi(a) \rangle]_{code} \in L_\varphi$, on a une exécution de $\mathcal{M}$ telle que

$$\exists q \in F : \exists u, v \in \Gamma^\star : q_0\, \S\tau(a)\S\tau'(\varphi(a)) \to^*_{\mathcal{M}} u\, q\, v$$

On calcule donc bien la fonction φ. □

Décidabilité et Calculabilité
Plan

1. Langages récursivement énumérables
2. Langages décidables
3. Calculer avec une machine de Turing
4. Machine universelle

Une machine programmable

☞ Un des attraits des ordinateurs est que ce sont des machines **programmables**.

Une machine programmable

- Un des attraits des ordinateurs est que ce sont des machines **programmables**.
- Ainsi, une même machine peut accomplir une grande variété de tâches, suivant le programme qu'elle exécute.

Une machine programmable

- Un des attraits des ordinateurs est que ce sont des machines **programmables**.
- Ainsi, une même machine peut accomplir une grande variété de tâches, suivant le programme qu'elle exécute.
- Les machines de Turing ont également cette possibilité.

Une machine programmable

- Un des attraits des ordinateurs est que ce sont des machines **programmables**.
- Ainsi, une même machine peut accomplir une grande variété de tâches, suivant le programme qu'elle exécute.
- Les machines de Turing ont également cette possibilité.
- En effet, on va maintenant montrer qu'il existe une machine **déterministe** $\mathcal{U}$ qui prend en entrée une machine (non-déterministe) $\mathcal{M}$ et un mot $w \in \Sigma^\star$, et qui simule l'exécution de $\mathcal{M}$ sur l'entrée w.

Une machine programmable

- Un des attraits des ordinateurs est que ce sont des machines **programmables**.
- Ainsi, une même machine peut accomplir une grande variété de tâches, suivant le programme qu'elle exécute.
- Les machines de Turing ont également cette possibilité.
- En effet, on va maintenant montrer qu'il existe une machine **déterministe** $\mathcal{U}$ qui prend en entrée une machine (non-déterministe) $\mathcal{M}$ et un mot $w \in \Sigma^\star$, et qui simule l'exécution de $\mathcal{M}$ sur l'entrée w.
- Conceptuellement, $\mathcal{U}$ est un genre d'interpréteur, comme par exemple la JVM.

À l'entrée de la machine universelle
Codage de $\mathcal{M}$

$$\mathcal{M} := \langle Q, \Sigma, \Gamma, \Delta, q_0, F \rangle \, .$$

À l'entrée de la machine universelle

Codage de $\mathcal{M}$

états : entiers consécutifs $0, \ldots, n$.

$$\mathcal{M} := \langle Q, \Sigma, \Gamma, \Delta, q_0, F \rangle .$$

☞ On code les entiers en binaire : $[n]_{code} \in \{0,1\}^\star$.

À l'entrée de la machine universelle

Codage de $\mathcal{M}$

états : entiers consécutifs $0, \ldots, n$.

$$\mathcal{M} := \langle Q, \Sigma, \Gamma, \Delta, q_0, F \rangle .$$

alphabets : on fixe $\Sigma = \{0, 1\}$ et $\Gamma = \{0, 1, \boxplus\}$.

- ☞ On code les entiers en binaire : $[n]_{code} \in \{0, 1\}^\star$.
- ☞ Pour éviter les ambiguités, on distingue le symbole case vide $\boxplus$ de $\mathcal{M}$ du symbole case vide # de la machine universelle.

À l'entrée de la machine universelle

Codage de $\mathcal{M}$

états : entiers consécutifs $0, \ldots, n$.

état initial : on fixe $q_0 = 0$.

$$\mathcal{M} := \langle Q, \Sigma, \Gamma, \Delta, q_0, F \rangle .$$

alphabets : on fixe $\Sigma = \{0, 1\}$ et $\Gamma = \{0, 1, \boxplus\}$.

- On code les entiers en binaire : $[n]_{code} \in \{0, 1\}^\star$.
- Pour éviter les ambiguités, on distingue le symbole case vide $\boxplus$ de $\mathcal{M}$ du symbole case vide # de la machine universelle.

À l'entrée de la machine universelle

Codage de $\mathcal{M}$

états : entiers consécutifs $0, \ldots, n$.

état initial : on fixe $q_0 = 0$.

$$\mathcal{M} := \langle Q, \Sigma, \Gamma, \Delta, q_0, F \rangle .$$

alphabets : on fixe $\Sigma = \{0, 1\}$ et $\Gamma = \{0, 1, \boxplus\}$.

états finaux : liste d'entiers $q_1^f, \ldots, q_m^f$.

- On code les entiers en binaire : $[n]_{code} \in \{0, 1\}^\star$.
- Pour éviter les ambiguités, on distingue le symbole case vide $\boxplus$ de $\mathcal{M}$ du symbole case vide # de la machine universelle.

À l'entrée de la machine universelle

Codage de $\mathcal{M}$

Relation de transition : liste de transitions $t_1, \ldots, t_k$.

états : entiers consécutifs $0, \ldots, n$.

état initial : on fixe $q_0 = 0$.

$$\mathcal{M} := \langle Q, \Sigma, \Gamma, \Delta, q_0, F \rangle .$$

alphabets : on fixe $\Sigma = \{0, 1\}$ et $\Gamma = \{0, 1, \boxplus\}$.

états finaux : liste d'entiers $q_1^f, \ldots, q_m^f$.

- On code les entiers en binaire : $[n]_{code} \in \{0, 1\}^\star$.
- Pour éviter les ambiguités, on distingue le symbole case vide $\boxplus$ de $\mathcal{M}$ du symbole case vide # de la machine universelle.
- Chaque transition : on encode les directions $\{\blacktriangleleft, \blacktriangleright\}$ par les symboles $\{\mathtt{G}, \mathtt{D}\}$, et la transition $q_1 \xrightarrow{a/b,dir} q_2$ par le mot

$$[q_1]_{code} \; \S \; a \; \S \; b \; \S \; [dir]_{code} \; \S \; [q_2]_{code} \in \{0, 1, \boxplus, \mathtt{D}, \mathtt{G}, \S\}^\star .$$

À l'entrée de la machine universelle

Codage de $\mathcal{M}$

Relation de transition : liste de transitions $t_1, \ldots, t_k$.

états : entiers consécutifs $0, \ldots, n$.

état initial : on fixe $q_0 = 0$.

$$\mathcal{M} := \langle Q, \Sigma, \Gamma, \Delta, q_0, F \rangle .$$

alphabets : on fixe $\Sigma = \{0, 1\}$ et $\Gamma = \{0, 1, \boxplus\}$.

états finaux : liste d'entiers $q_1^f, \ldots, q_m^f$.

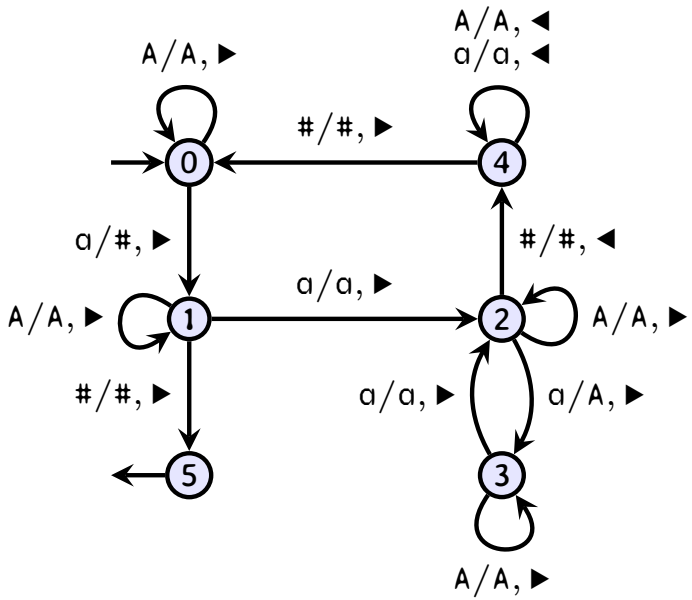
- On code les entiers en binaire : $[n]_{code} \in \{0, 1\}^\star$.
- Pour éviter les ambiguités, on distingue le symbole case vide $\boxplus$ de $\mathcal{M}$ du symbole case vide # de la machine universelle.
- Chaque transition : on encode les directions $\{\blacktriangleleft, \blacktriangleright\}$ par les symboles $\{\mathtt{G}, \mathtt{D}\}$, et la transition $q_1 \xrightarrow{a/b,dir} q_2$ par le mot

$$[q_1]_{code} \; \S \; a \; \S \; b \; \S \; [dir]_{code} \; \S \; [q_2]_{code} \in \{0, 1, \boxplus, \mathtt{D}, \mathtt{G}, \S\}^\star .$$

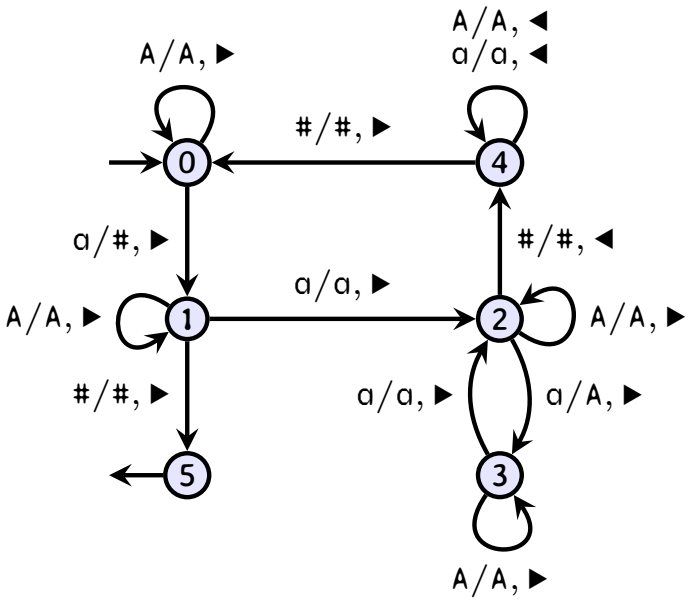
- Finalement on obtient le code de la machine $\mathcal{M}$:

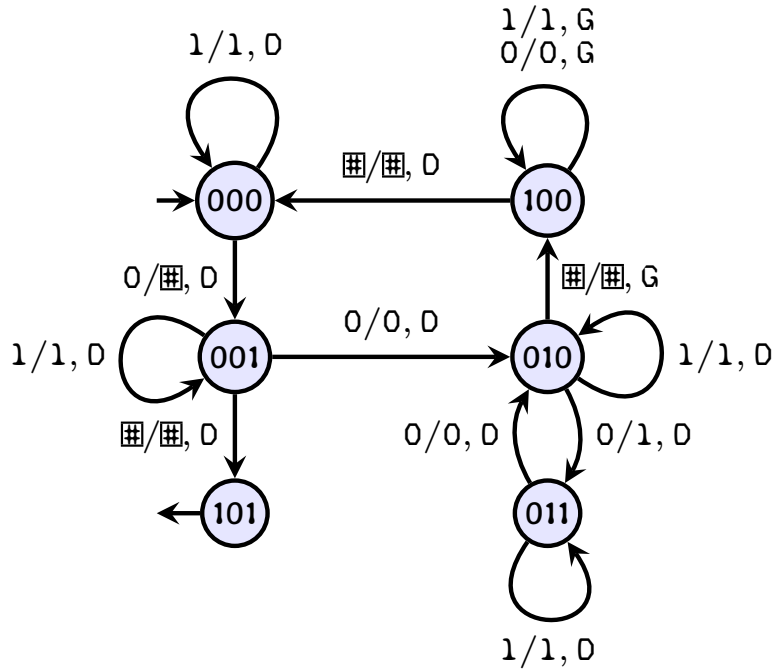
$$[\mathcal{M}]_{code} = \$ \; [t_1]_{code} \mid \ldots \mid [t_k]_{code} \; \$ \; \left[q_1^f\right]_{code} \mid \ldots \mid \left[q_m^f\right]_{code}$$
$$\in \{0, 1, \boxplus, \mathtt{D}, \mathtt{G}, \S, \$, \mid\}^\star$$

Codage de $\mathcal{M}$
Exemple

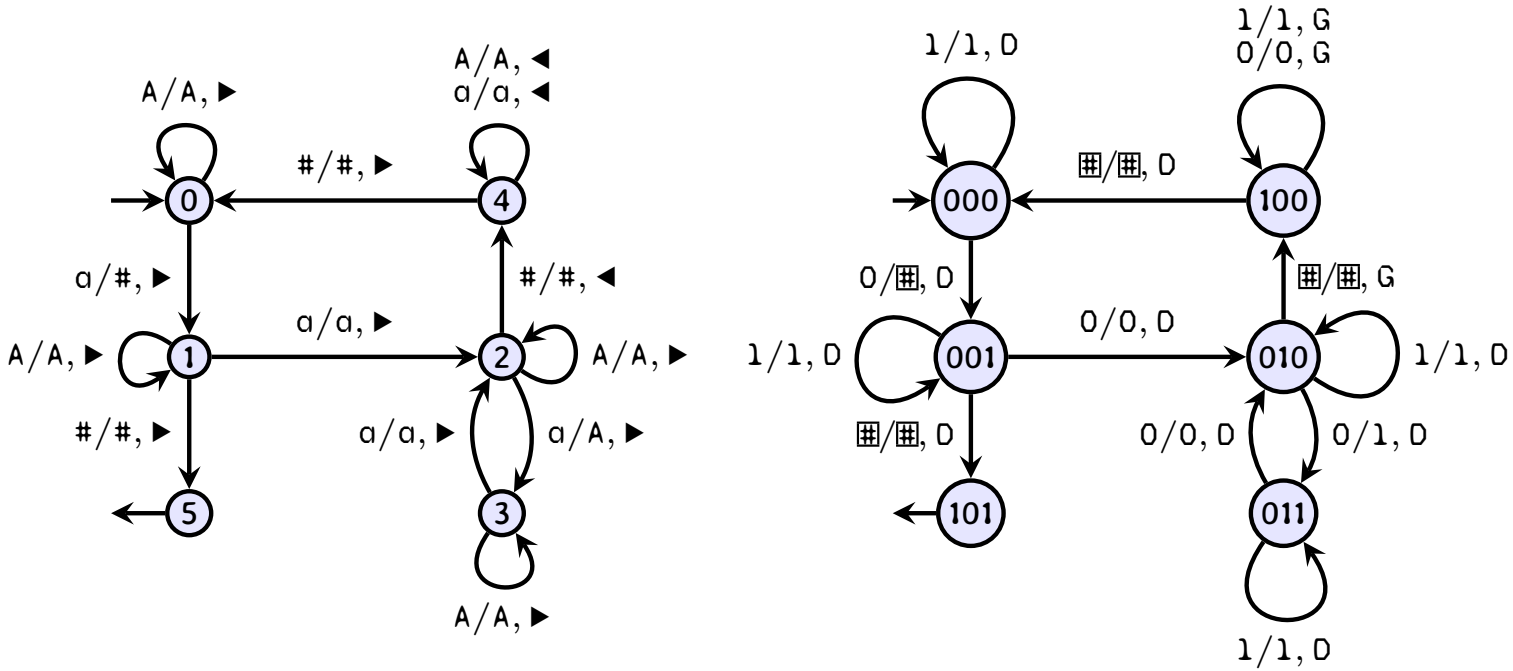


Codage de $\mathcal{M}$
Exemple



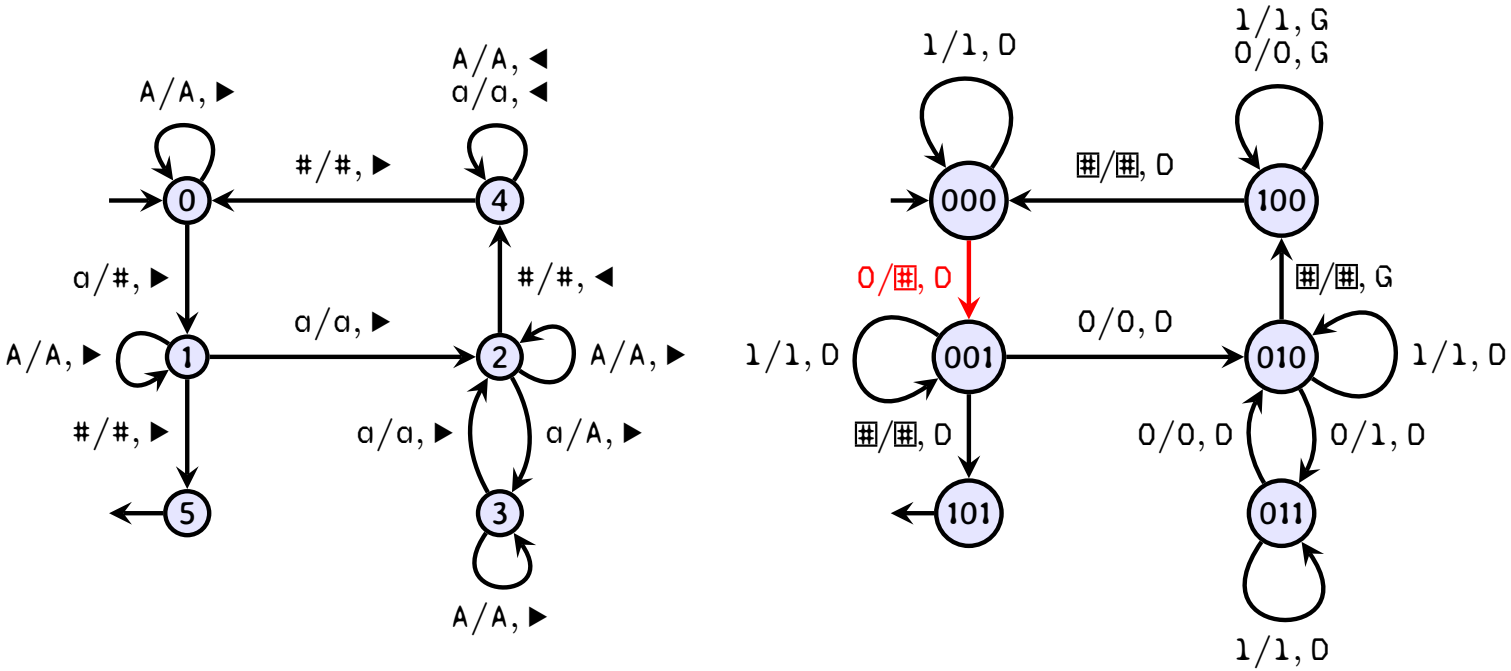


Codage de $\mathcal{M}$
Exemple



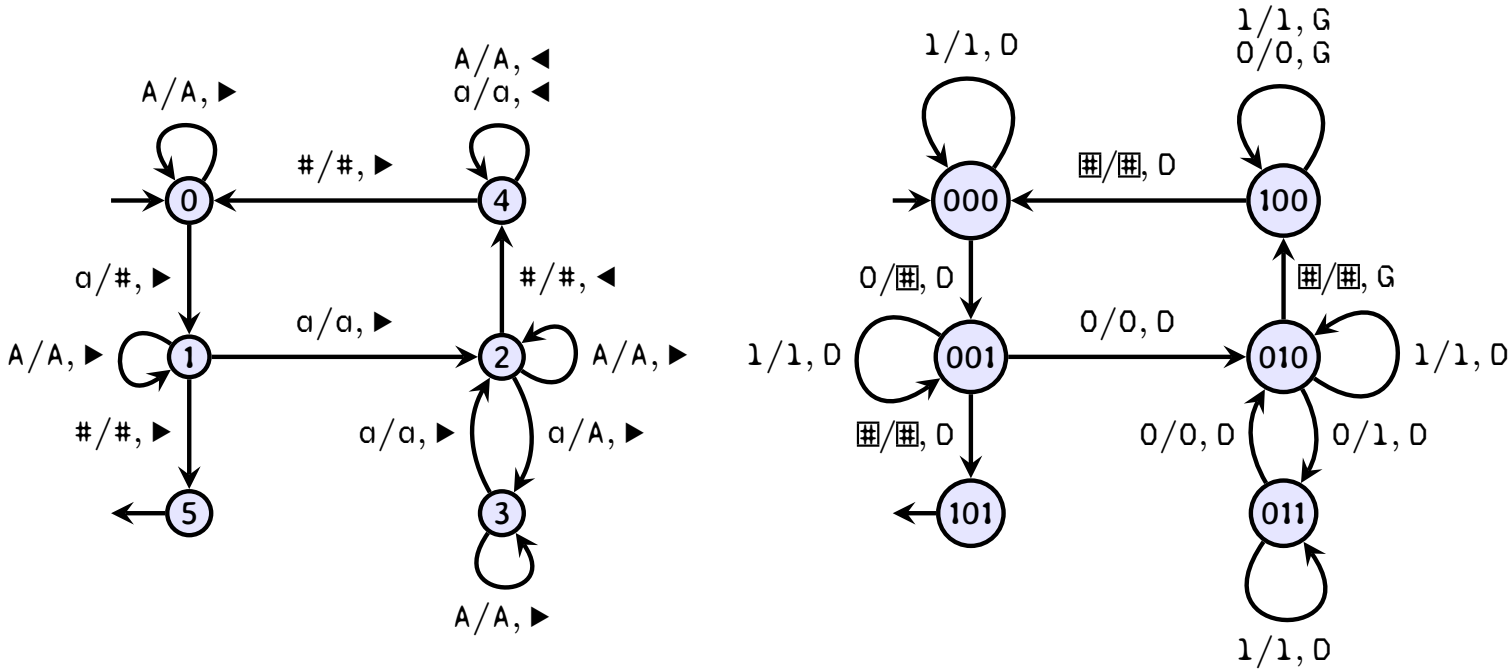
$$[\mathcal{M}]_{code} = \$\,000\S1\S1\S D\S000 \mid 000\S0\S\boxplus\S D\S001 \mid \ldots\ \$101$$

Codage de $\mathcal{M}$
Exemple



$$[\mathcal{M}]_{code} = \$\,000\S1\S1\S D\S000 \mid 000\S0\S\boxplus\S D\S001 \mid \ldots\ \$101$$

Codage de $\mathcal{M}$
Exemple



$$[\mathcal{M}]_{code} = \$\,000\S1\S1\S D\S000 \mid 000\S0\S\boxplus\S D\S001 \mid \ldots\ \$101$$

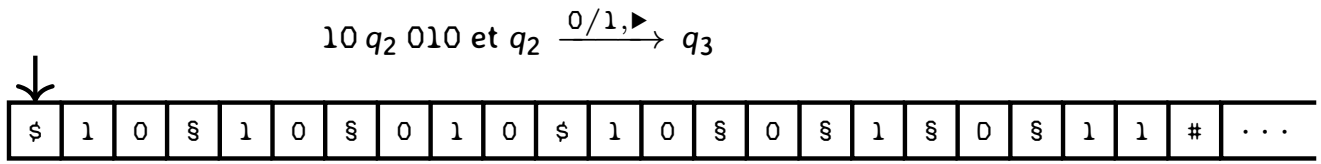
exercice

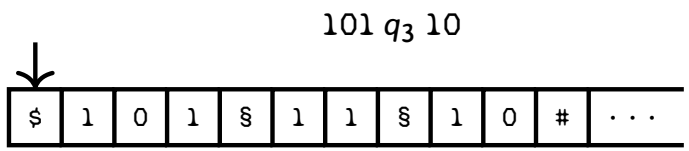
Montrer que le langage $L = \{[\mathcal{M}]_{code} \mid \mathcal{M} \text{ machine de Turing}\}$ est décidable.

Simuler l'exécution de $\mathcal{M}$
Exécuter une transition

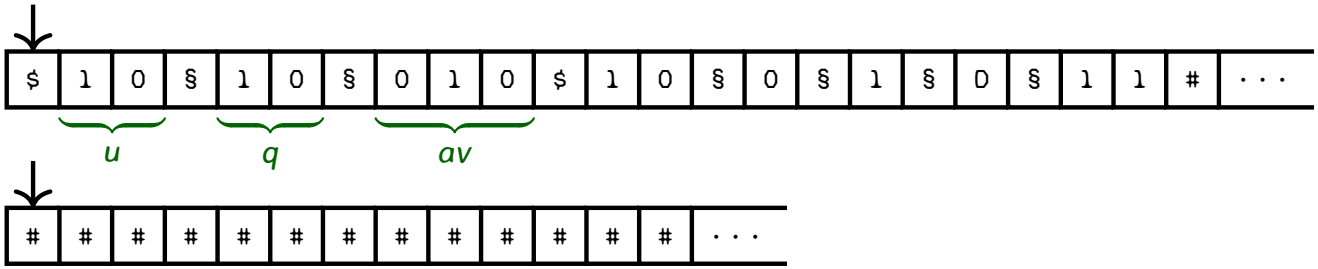
On encode une configuration $u\,q\,v$ par le mot $u\S\,[q]_{code}\,\S v$.
Supposons qu'on nous donne une configuration et une transition, on peut construire un machine $\mathcal{M}_\Delta$ à deux rubans qui :

- rejette si la transition ne peut pas être activée depuis cette configuration ;
- sinon, elle écrit sur son ruban d'entrée la configuration après avoir exécuté la transition, et elle accepte.



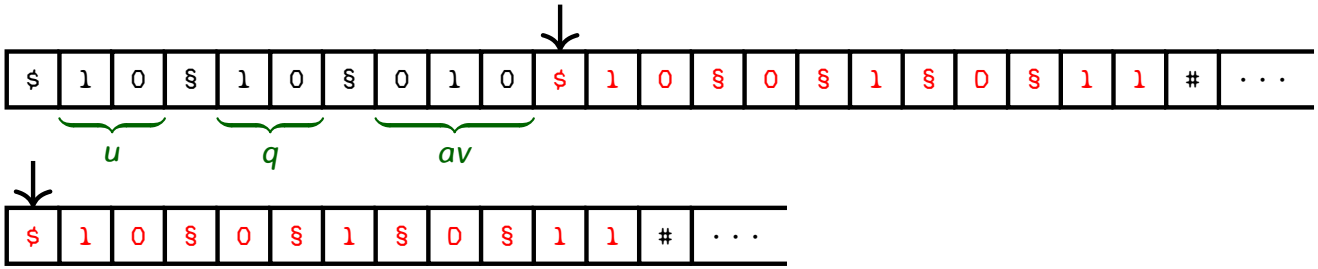


Simuler l'exécution de $\mathcal{M}$
Exécuter une transition



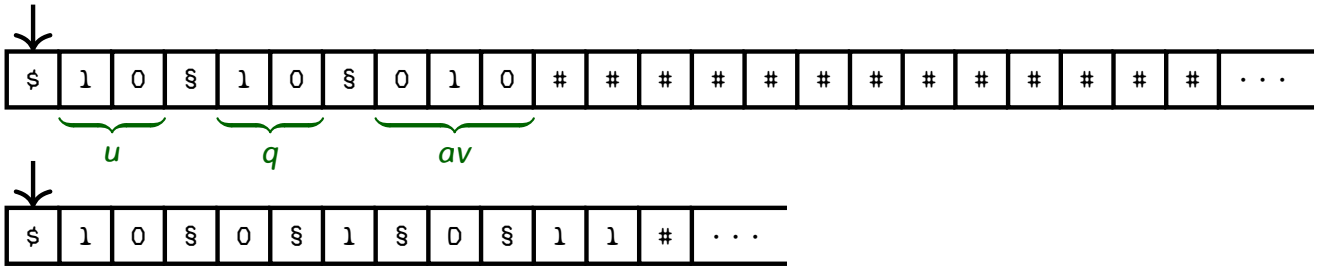
☞ On commence avec le codage de la paire $\langle u\,q\,av, q_1 \xrightarrow{a_1/a_2,dir} q_2 \rangle$ sur le premier ruban.

Simuler l'exécution de $\mathcal{M}$
Exécuter une transition



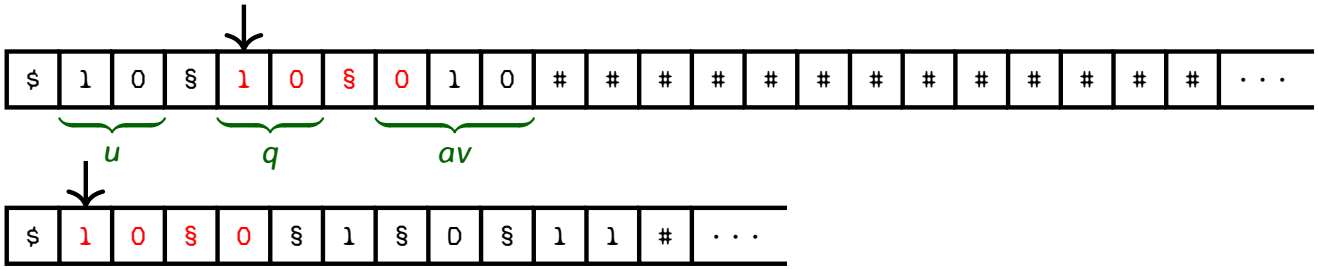
- On commence avec le codage de la paire $\langle u\,q\,av, q_1 \xrightarrow{a_1/a_2,dir} q_2 \rangle$ sur le premier ruban.
- On déplace la transition sur le deuxième ruban.

Simuler l'exécution de $\mathcal{M}$
Exécuter une transition



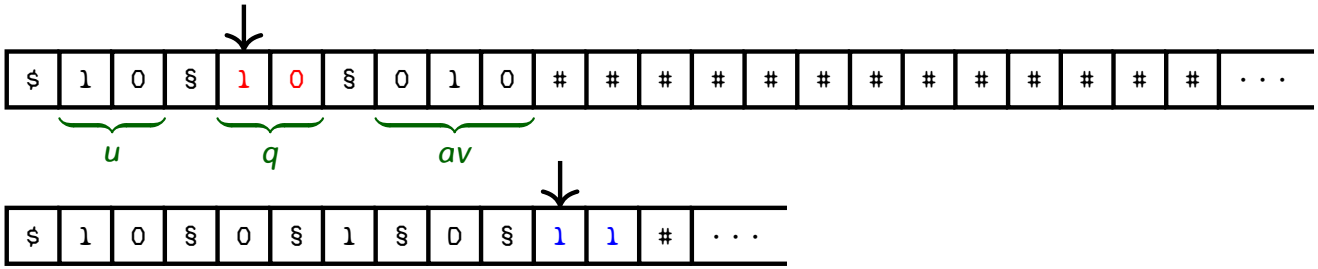
- On commence avec le codage de la paire $\langle u\,q\,av, q_1 \xrightarrow{a_1/a_2,dir} q_2 \rangle$ sur le premier ruban.
- On déplace la transition sur le deuxième ruban.
- On l'efface du premier ruban, et on replace les deux têtes au début de leurs rubans.

Simuler l'exécution de $\mathcal{M}$
Exécuter une transition



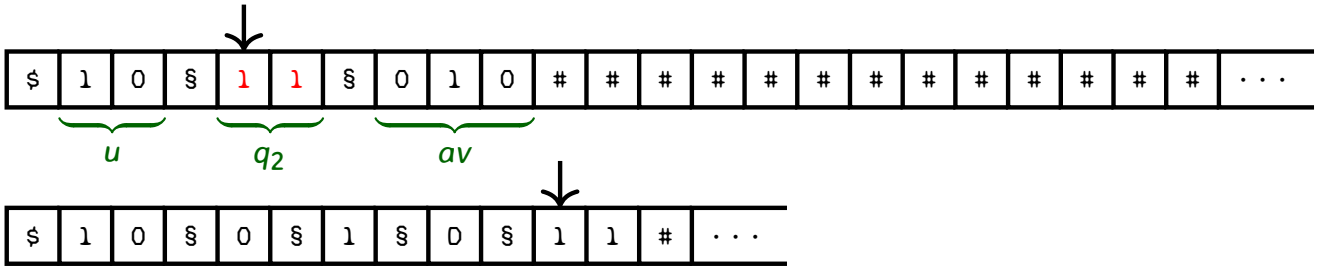
- On commence avec le codage de la paire $\langle u\,q\,av, q_1 \xrightarrow{a_1/a_2,dir} q_2 \rangle$ sur le premier ruban.
- On déplace la transition sur le deuxième ruban.
- On l'efface du premier ruban, et on replace les deux têtes au début de leurs rubans.
- On vérifie que $q = q_1$ et $a = a_1$:
  - si ça n'est pas le cas, on efface les deux rubans et on va sur l'état de rejet (q_{non}) ;
  - sinon on poursuit.

Simuler l'exécution de $\mathcal{M}$
Exécuter une transition



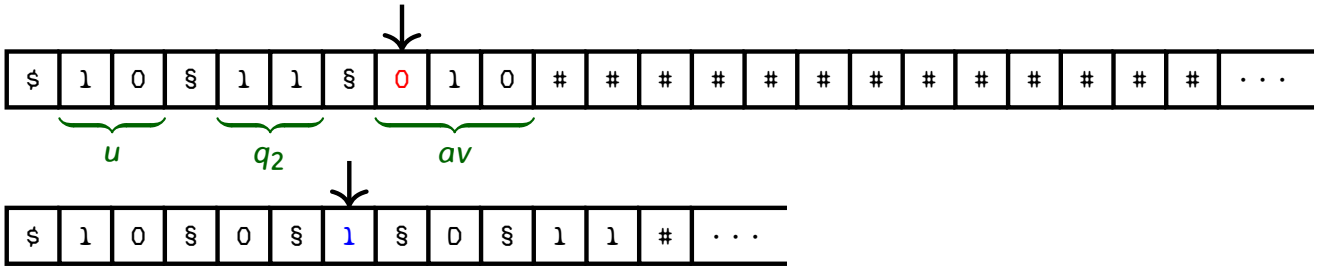
- On commence avec le codage de la paire $\langle u\,q\,av, q_1 \xrightarrow{a_1/a_2,dir} q_2 \rangle$ sur le premier ruban.
- On déplace la transition sur le deuxième ruban.
- On l'efface du premier ruban, et on replace les deux têtes au début de leurs rubans.
- On vérifie que $q = q_1$ et $a = a_1$:
  - si ça n'est pas le cas, on efface les deux rubans et on va sur l'état de rejet (q_{non}) ;
  - sinon on poursuit.
- On remplace q par q_2,

Simuler l'exécution de $\mathcal{M}$
Exécuter une transition



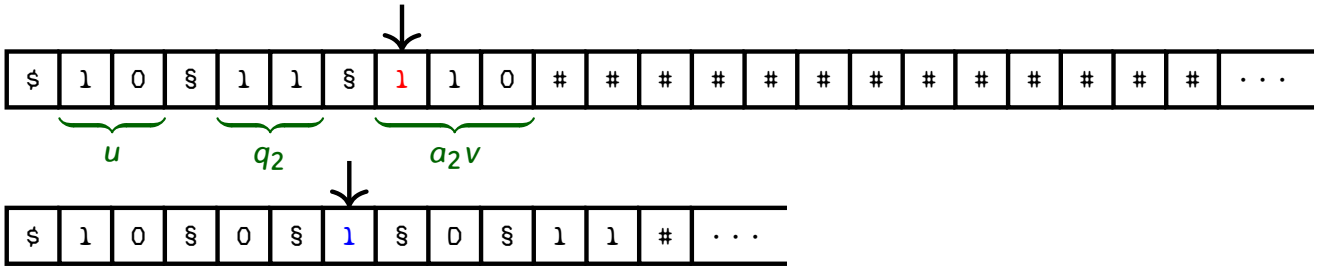
- On commence avec le codage de la paire $\langle u\,q\,av, q_1 \xrightarrow{a_1/a_2,dir} q_2 \rangle$ sur le premier ruban.
- On déplace la transition sur le deuxième ruban.
- On l'efface du premier ruban, et on replace les deux têtes au début de leurs rubans.
- On vérifie que $q = q_1$ et $a = a_1$:
  - si ça n'est pas le cas, on efface les deux rubans et on va sur l'état de rejet (q_{non}) ;
  - sinon on poursuit.
- On remplace q par q_2,

Simuler l'exécution de $\mathcal{M}$
Exécuter une transition



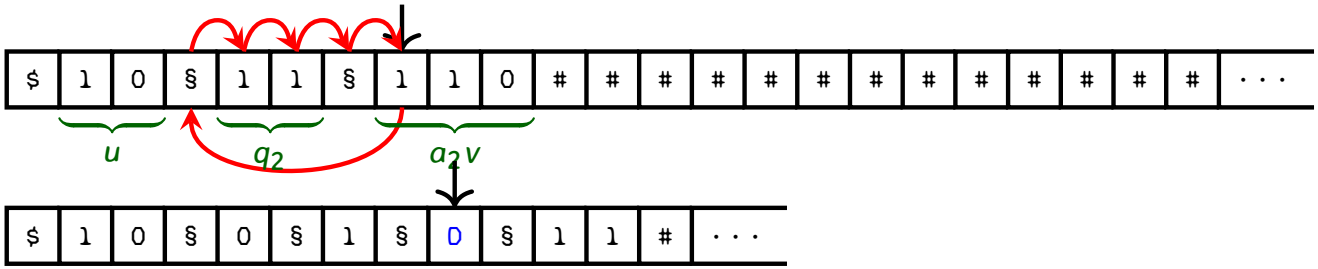
- On commence avec le codage de la paire $\langle u\,q\,av, q_1 \xrightarrow{a_1/a_2,dir} q_2 \rangle$ sur le premier ruban.
- On déplace la transition sur le deuxième ruban.
- On l'efface du premier ruban, et on replace les deux têtes au début de leurs rubans.
- On vérifie que $q = q_1$ et $a = a_1$:
  - si ça n'est pas le cas, on efface les deux rubans et on va sur l'état de rejet (q_{non}) ;
  - sinon on poursuit.
- On remplace q par q_2, et a par a_2.

Simuler l'exécution de $\mathcal{M}$
Exécuter une transition



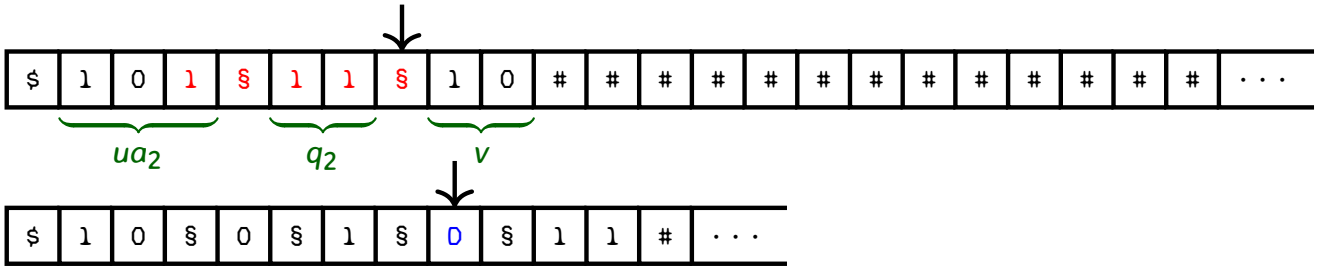
- On commence avec le codage de la paire $\langle u\,q\,av, q_1 \xrightarrow{a_1/a_2,dir} q_2\rangle$ sur le premier ruban.
- On déplace la transition sur le deuxième ruban.
- On l'efface du premier ruban, et on replace les deux têtes au début de leurs rubans.
- On vérifie que $q = q_1$ et $a = a_1$:
  - si ça n'est pas le cas, on efface les deux rubans et on va sur l'état de rejet (q_{non}) ;
  - sinon on poursuit.
- On remplace q par q_2, et a par a_2.

Simuler l'exécution de $\mathcal{M}$
Exécuter une transition



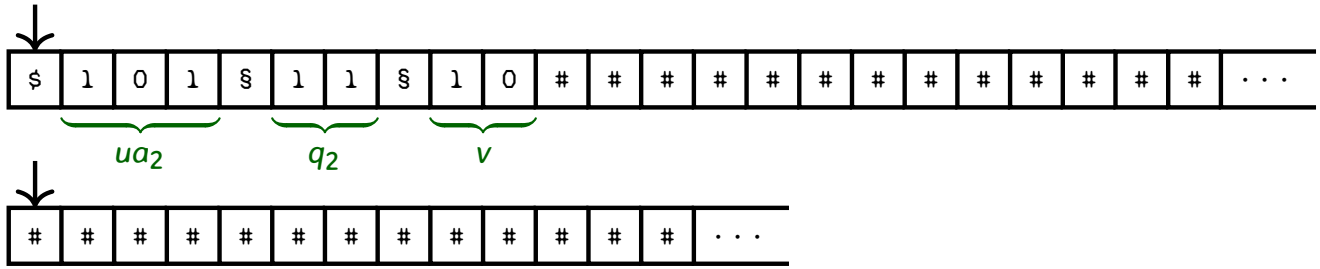
- On commence avec le codage de la paire $\langle u\,q\,av, q_1 \xrightarrow{a_1/a_2,dir} q_2\rangle$ sur le premier ruban.
- On déplace la transition sur le deuxième ruban.
- On l'efface du premier ruban, et on replace les deux têtes au début de leurs rubans.
- On vérifie que $q = q_1$ et $a = a_1$:
  - si ça n'est pas le cas, on efface les deux rubans et on va sur l'état de rejet (q_{non}) ;
  - sinon on poursuit.
- On remplace q par q_2, et a par a_2.
- On regarde quelle est la direction de déplacement,

Simuler l'exécution de $\mathcal{M}$
Exécuter une transition



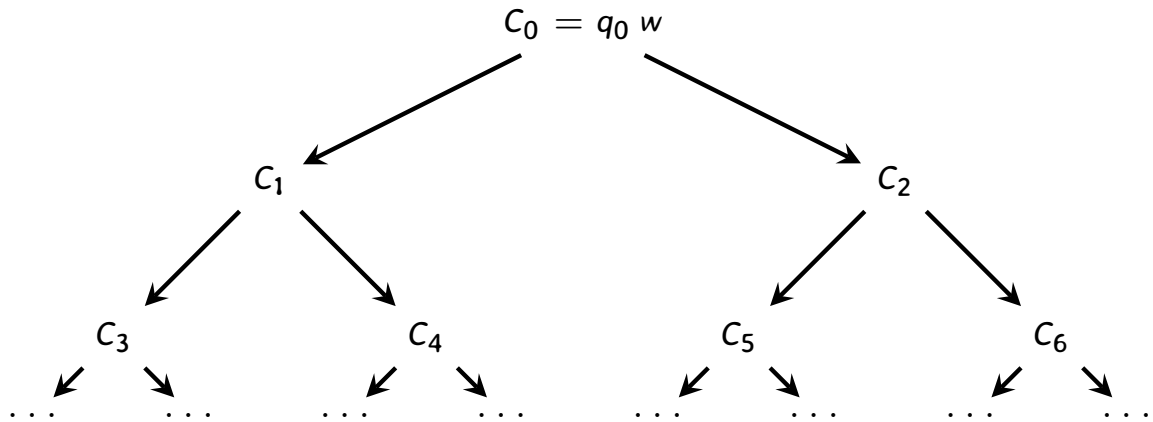
- On commence avec le codage de la paire $\langle u\,q\,av, q_1 \xrightarrow{a_1/a_2,dir} q_2 \rangle$ sur le premier ruban.
- On déplace la transition sur le deuxième ruban.
- On l'efface du premier ruban, et on replace les deux têtes au début de leurs rubans.
- On vérifie que $q = q_1$ et $a = a_1$:
  - si ça n'est pas le cas, on efface les deux rubans et on va sur l'état de rejet (q_{non}) ;
  - sinon on poursuit.
- On remplace q par q_2, et a par a_2.
- On regarde quelle est la direction de déplacement, et on déplace la tête de lecture.

Simuler l'exécution de $\mathcal{M}$
Exécuter une transition



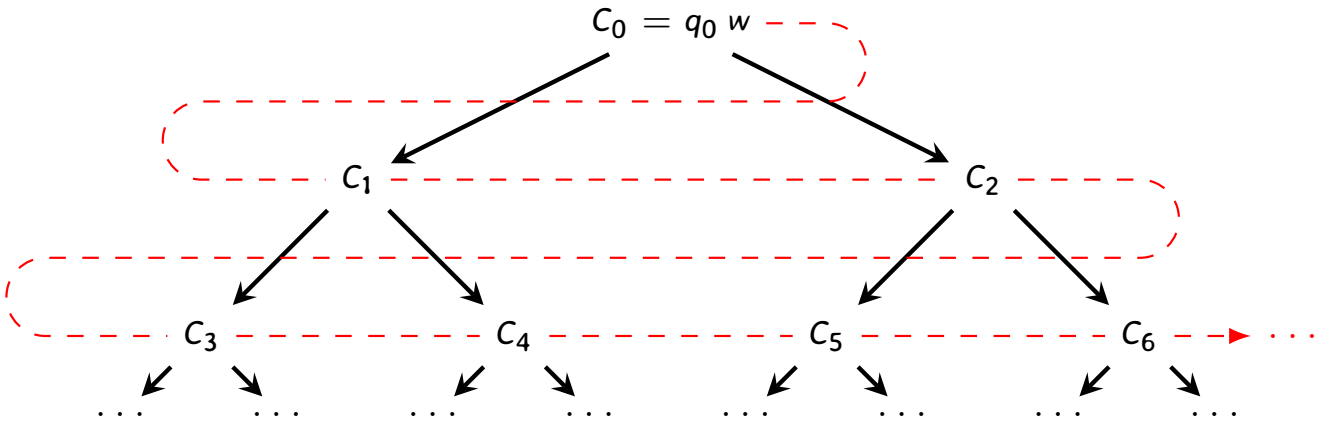
- On commence avec le codage de la paire $\langle u\,q\,av, q_1 \xrightarrow{a_1/a_2,dir} q_2 \rangle$ sur le premier ruban.
- On déplace la transition sur le deuxième ruban.
- On l'efface du premier ruban, et on replace les deux têtes au début de leurs rubans.
- On vérifie que $q = q_1$ et $a = a_1$:
  - si ça n'est pas le cas, on efface les deux rubans et on va sur l'état de rejet (q_{non}) ;
  - sinon on poursuit.
- On remplace q par q_2, et a par a_2.
- On regarde quelle est la direction de déplacement, et on déplace la tête de lecture.
- Enfin, on nettoie le second ruban, on repositionne la tête de lecture sur le premier, et on va sur l'état d'acceptation q_{oui}.

Simuler l'exécution de $\mathcal{M}$
Déterminisation : idée de l'algorithme



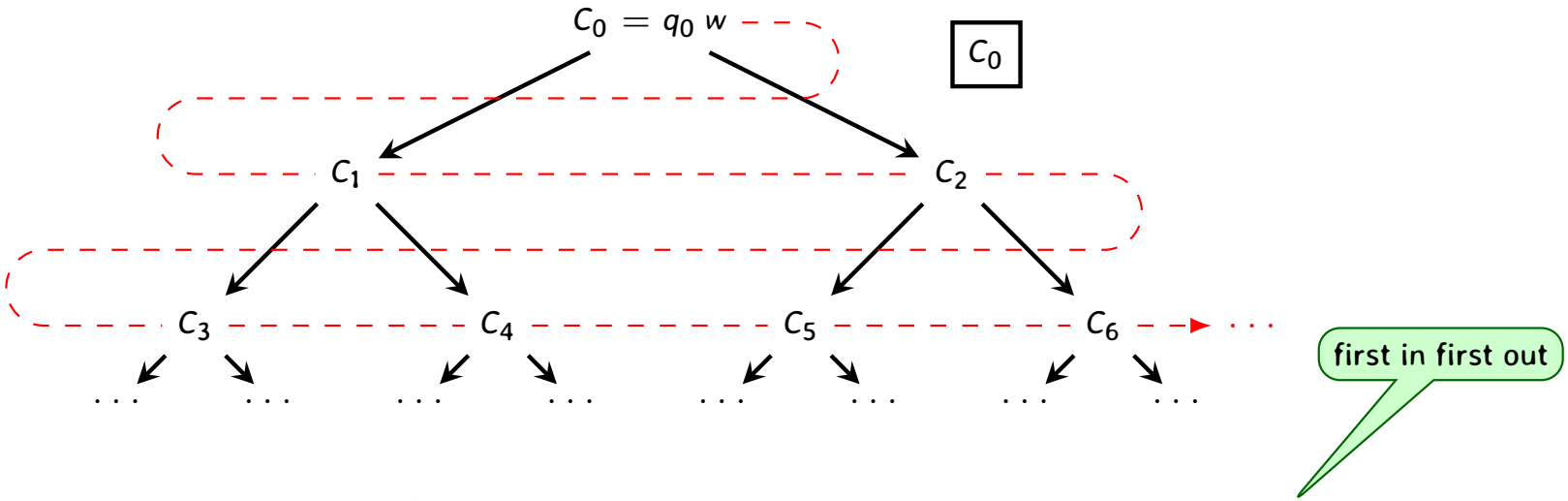
On va parcourir l'arbre d'exécution **en largeur**. Pour cela, on maintient une **file** de configurations.

Simuler l'exécution de $\mathcal{M}$
Déterminisation : idée de l'algorithme



On va parcourir l'arbre d'exécution **en largeur**. Pour cela, on maintient une **file** de configurations.

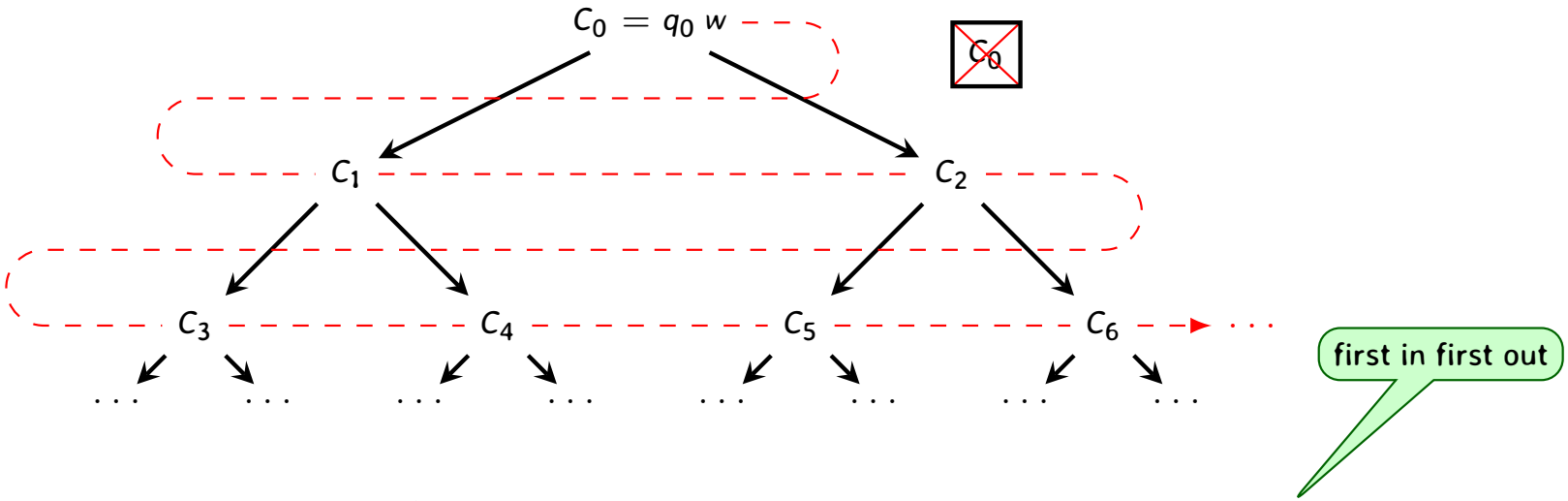
Simuler l'exécution de $\mathcal{M}$
Déterminisation : idée de l'algorithme



On va parcourir l'arbre d'exécution **en largeur**. Pour cela, on maintient une **file** de configurations.

1) On initialise la file avec $C_0 = \S\, [q_0]_{code}\, \S w$.

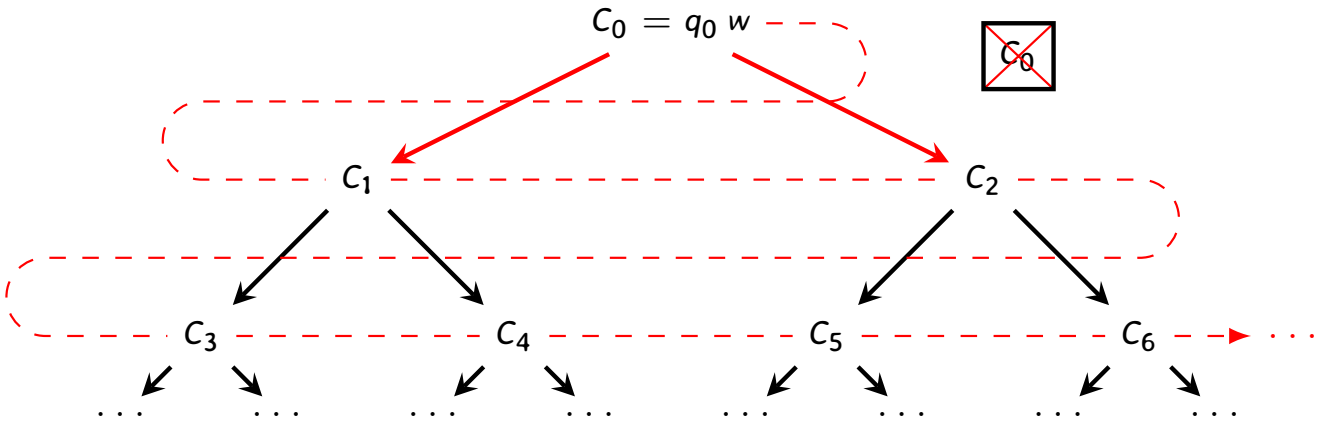
Simuler l'exécution de $\mathcal{M}$
Déterminisation : idée de l'algorithme



On va parcourir l'arbre d'exécution **en largeur**. Pour cela, on maintient une **file** de configurations.

1) On initialise la file avec $C_0 = \S\, [q_0]_{code}\, \S w$.
2) Tant que la file n'est pas vide :
   - 2.1 On enlève la première configuration C de la file ;
   - 2.2 Si C est acceptante, on **accepte** l'entrée.

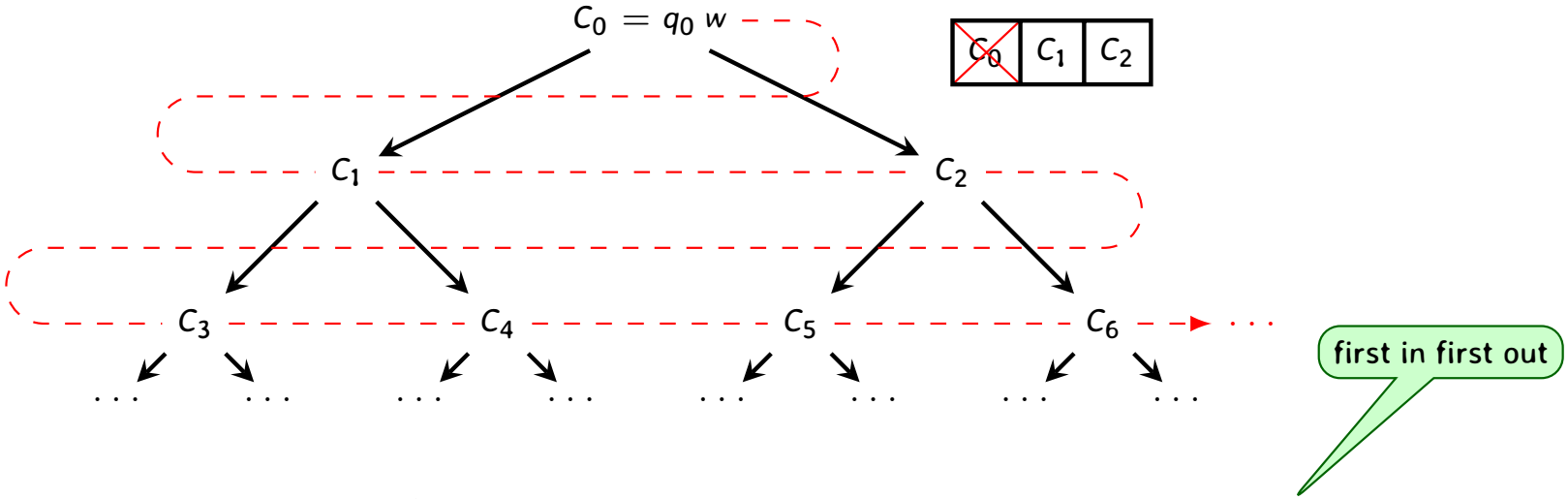
Simuler l'exécution de $\mathcal{M}$
Déterminisation : idée de l'algorithme



On va parcourir l'arbre d'exécution **en largeur**. Pour cela, on maintient une **file** de configurations.

1) On initialise la file avec $C_0 = \S\, [q_0]_{code}\, \S w$.
2) Tant que la file n'est pas vide :
   - 2.1 On enlève la première configuration C de la file ;
   - 2.2 Si C est acceptante, on **accepte** l'entrée.
   - 2.3 Sinon, pour toutes les transitions t de la machine :
     - 2.3.1 On essaie d'exécuter t à partir de C ;
     - 2.3.2 Si c'est impossible, on continue ;

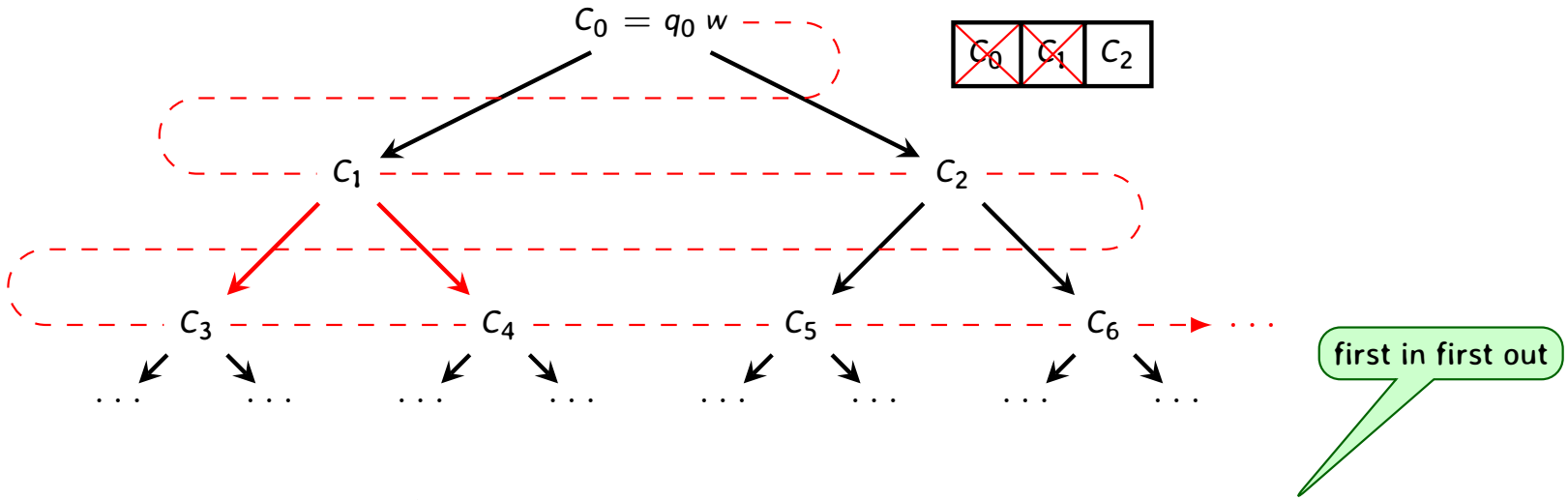
Simuler l'exécution de $\mathcal{M}$
Déterminisation : idée de l'algorithme



On va parcourir l'arbre d'exécution **en largeur**. Pour cela, on maintient une **file** de configurations.

1) On initialise la file avec $C_0 = \S\, [q_0]_{code}\, \S w$.
2) Tant que la file n'est pas vide :
   - 2.1 On enlève la première configuration C de la file ;
   - 2.2 Si C est acceptante, on **accepte** l'entrée.
   - 2.3 Sinon, pour toutes les transitions t de la machine :
     - 2.3.1 On essaie d'exécuter t à partir de C ;
     - 2.3.2 Si c'est impossible, on continue ;
     - 2.3.3 Sinon, on obtient C' telle que $C \to_{\mathcal{M}} C'$, et on l'ajoute à la fin de la file.

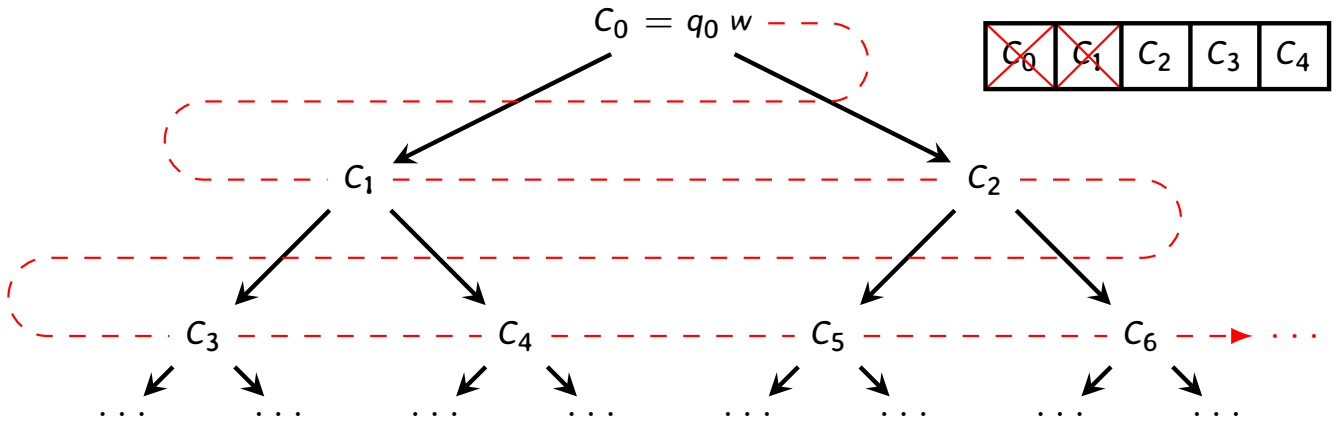
Simuler l'exécution de $\mathcal{M}$
Déterminisation : idée de l'algorithme



On va parcourir l'arbre d'exécution **en largeur**. Pour cela, on maintient une **file** de configurations.

1) On initialise la file avec $C_0 = \S\, [q_0]_{code}\, \S w$.
2) Tant que la file n'est pas vide :
   - 2.1 On enlève la première configuration C de la file ;
   - 2.2 Si C est acceptante, on **accepte** l'entrée.
   - 2.3 Sinon, pour toutes les transitions t de la machine :
     - 2.3.1 On essaie d'exécuter t à partir de C ;
     - 2.3.2 Si c'est impossible, on continue ;
     - 2.3.3 Sinon, on obtient C' telle que $C \to_{\mathcal{M}} C'$, et on l'ajoute à la fin de la file.

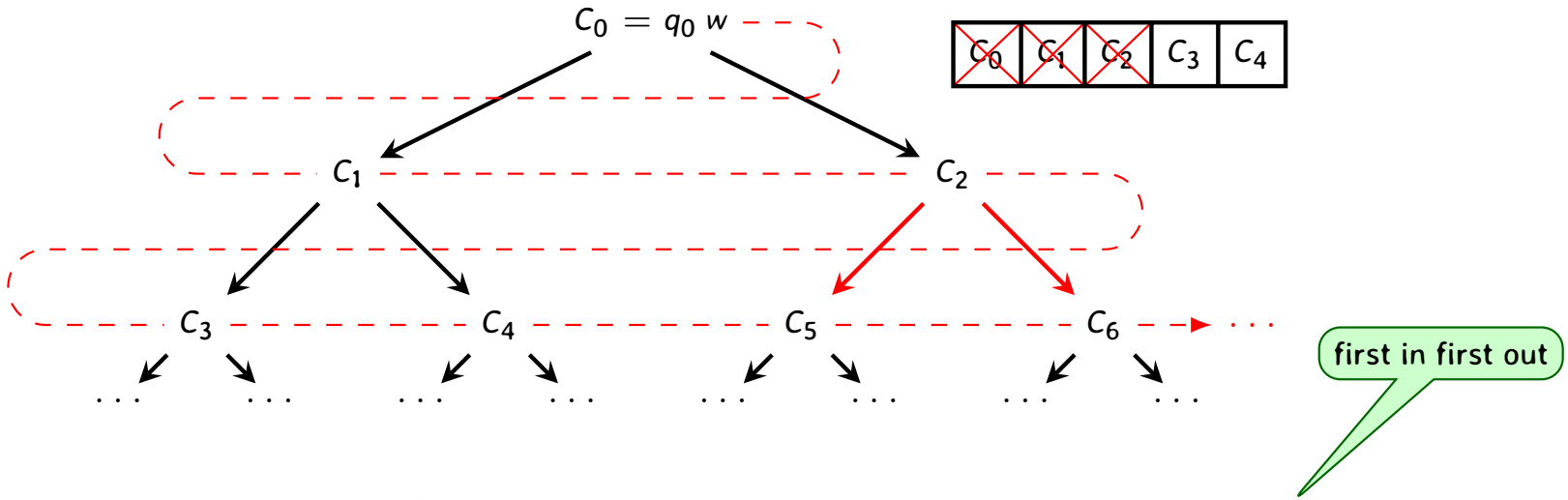
Simuler l'exécution de $\mathcal{M}$
Déterminisation : idée de l'algorithme



On va parcourir l'arbre d'exécution **en largeur**. Pour cela, on maintient une **file** de configurations.

1) On initialise la file avec $C_0 = \S\, [q_0]_{code}\, \S w$.
2) Tant que la file n'est pas vide :
   - 2.1 On enlève la première configuration C de la file ;
   - 2.2 Si C est acceptante, on **accepte** l'entrée.
   - 2.3 Sinon, pour toutes les transitions t de la machine :
     - 2.3.1 On essaie d'exécuter t à partir de C ;
     - 2.3.2 Si c'est impossible, on continue ;
     - 2.3.3 Sinon, on obtient C' telle que $C \to_{\mathcal{M}} C'$, et on l'ajoute à la fin de la file.

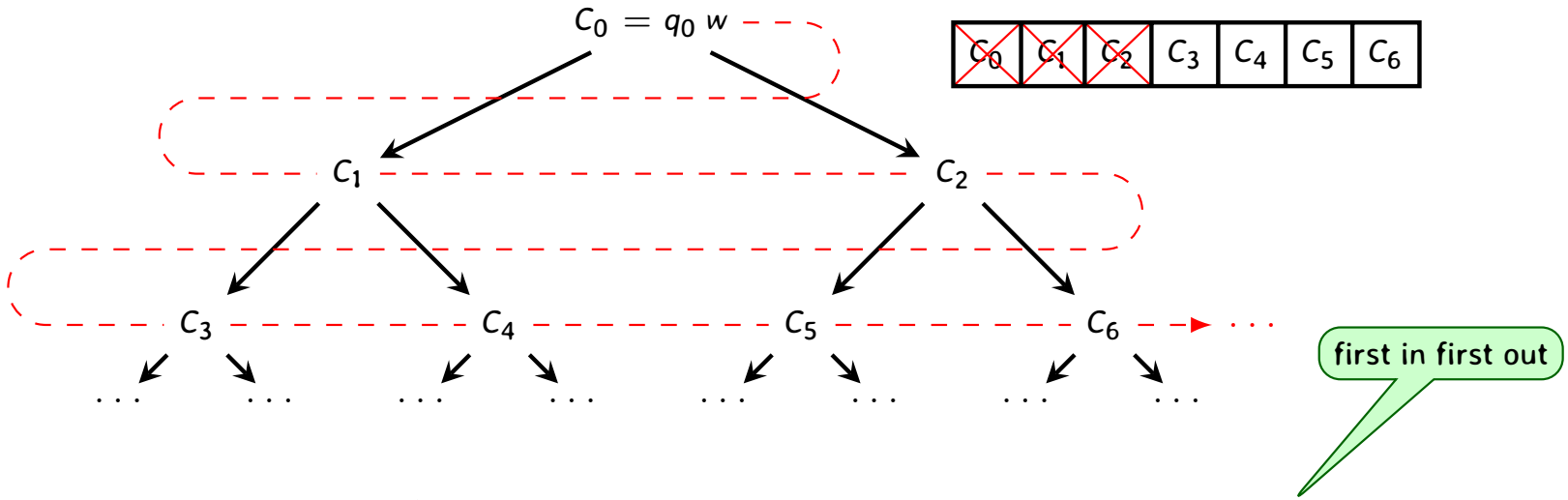
Simuler l'exécution de $\mathcal{M}$
Déterminisation : idée de l'algorithme



On va parcourir l'arbre d'exécution **en largeur**. Pour cela, on maintient une **file** de configurations.

1) On initialise la file avec $C_0 = \S\, [q_0]_{code}\, \S w$.
2) Tant que la file n'est pas vide :
   - 2.1 On enlève la première configuration C de la file ;
   - 2.2 Si C est acceptante, on **accepte** l'entrée.
   - 2.3 Sinon, pour toutes les transitions t de la machine :
     - 2.3.1 On essaie d'exécuter t à partir de C ;
     - 2.3.2 Si c'est impossible, on continue ;
     - 2.3.3 Sinon, on obtient C' telle que $C \to_{\mathcal{M}} C'$, et on l'ajoute à la fin de la file.

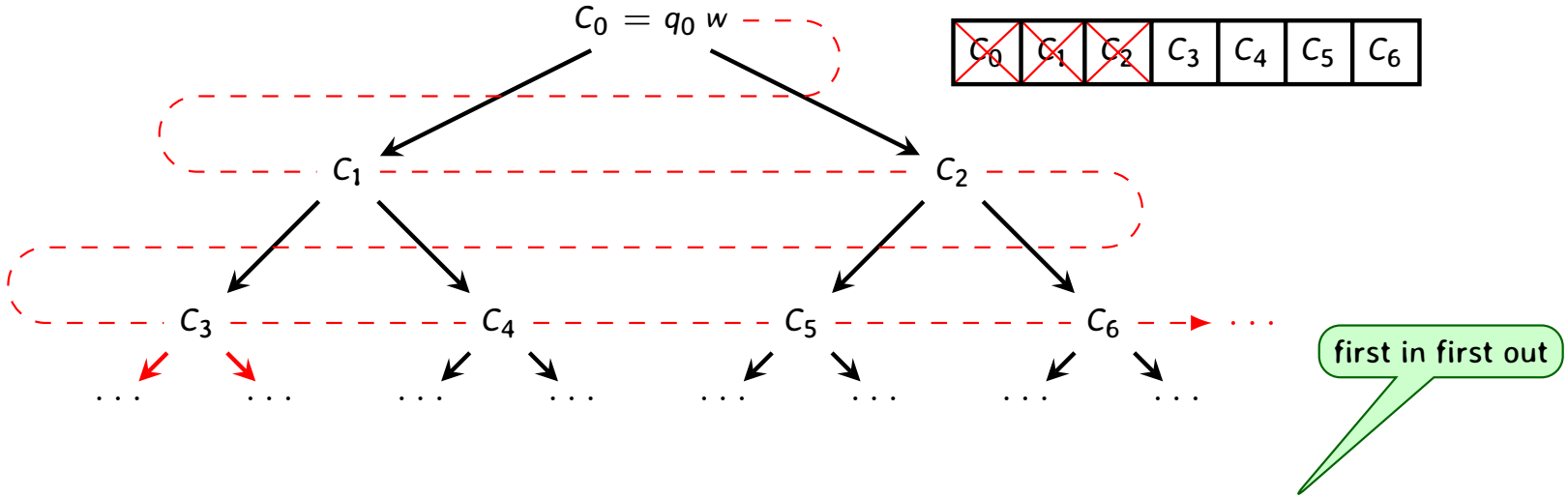
Simuler l'exécution de $\mathcal{M}$
Déterminisation : idée de l'algorithme



On va parcourir l'arbre d'exécution **en largeur**. Pour cela, on maintient une **file** de configurations.

1) On initialise la file avec $C_0 = \S\, [q_0]_{code}\, \S w$.
2) Tant que la file n'est pas vide :
   - 2.1 On enlève la première configuration C de la file ;
   - 2.2 Si C est acceptante, on **accepte** l'entrée.
   - 2.3 Sinon, pour toutes les transitions t de la machine :
     - 2.3.1 On essaie d'exécuter t à partir de C ;
     - 2.3.2 Si c'est impossible, on continue ;
     - 2.3.3 Sinon, on obtient C' telle que $C \to_{\mathcal{M}} C'$, et on l'ajoute à la fin de la file.

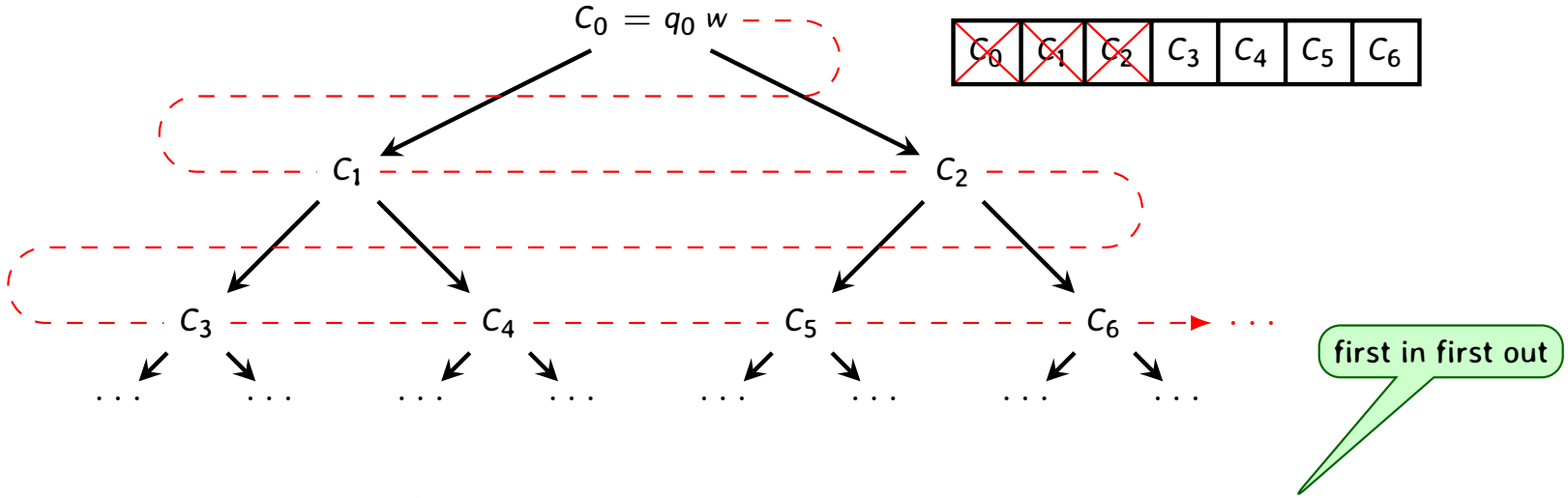
Simuler l'exécution de $\mathcal{M}$
Déterminisation : idée de l'algorithme



On va parcourir l'arbre d'exécution **en largeur**. Pour cela, on maintient une **file** de configurations.

1) On initialise la file avec $C_0 = \S\, [q_0]_{code}\, \S w$.
2) Tant que la file n'est pas vide :
   - 2.1 On enlève la première configuration C de la file ;
   - 2.2 Si C est acceptante, on **accepte** l'entrée.
   - 2.3 Sinon, pour toutes les transitions t de la machine :
     - 2.3.1 On essaie d'exécuter t à partir de C ;
     - 2.3.2 Si c'est impossible, on continue ;
     - 2.3.3 Sinon, on obtient C' telle que $C \to_{\mathcal{M}} C'$, et on l'ajoute à la fin de la file.

Simuler l'exécution de $\mathcal{M}$
Déterminisation : idée de l'algorithme



On va parcourir l'arbre d'exécution **en largeur**. Pour cela, on maintient une **file** de configurations.

1) On initialise la file avec $C_0 = \S\, [q_0]_{code}\, \S w$.
2) Tant que la file n'est pas vide :
   - 2.1 On enlève la première configuration C de la file ;
   - 2.2 Si C est acceptante, on **accepte** l'entrée.
   - 2.3 Sinon, pour toutes les transitions t de la machine :
     - 2.3.1 On essaie d'exécuter t à partir de C ;
     - 2.3.2 Si c'est impossible, on continue ;
     - 2.3.3 Sinon, on obtient C' telle que $C \to_{\mathcal{M}} C'$, et on l'ajoute à la fin de la file.
3) Si la file est vide et si on n'a pas trouvé de configuration acceptante, on **rejette**.

Langage accepté par la machine universelle

☞ Les entrées de la machine $\mathcal{U}$ doivent être de la forme $[\langle \mathcal{M}, w \rangle]_{code}$.

Langage accepté par la machine universelle

☞ Les entrées de la machine $\mathcal{U}$ doivent être de la forme $[\langle \mathcal{M}, w \rangle]_{code}$.
  - Si on propose une entrée mal formattée, i.e. un mot u tel que $\forall \mathcal{M}, \forall w, u \neq [\langle \mathcal{M}, w \rangle]_{code}$, $\mathcal{U}$ rejette son entrée en temps fini.

Langage accepté par la machine universelle

☞ Les entrées de la machine $\mathcal{U}$ doivent être de la forme $[\langle \mathcal{M}, w \rangle]_{code}$.

- Si on propose une entrée mal formattée, i.e. un mot u tel que $\forall \mathcal{M}, \forall w, u \neq [\langle \mathcal{M}, w \rangle]_{code}$, $\mathcal{U}$ rejette son entrée en temps fini.

- Sinon, on passe à la simulation de $\mathcal{M}$.

Langage accepté par la machine universelle

- Les entrées de la machine $\mathcal{U}$ doivent être de la forme $[\langle \mathcal{M}, w \rangle]_{code}$.
  - Si on propose une entrée mal formattée, i.e. un mot u tel que $\forall \mathcal{M}, \forall w, u \neq [\langle \mathcal{M}, w \rangle]_{code}$, $\mathcal{U}$ rejette son entrée en temps fini.
  - Sinon, on passe à la simulation de $\mathcal{M}$.
- $\mathcal{U}$ explore l'arbre d'exécution de $\mathcal{M}$ sur l'entrée w.

Langage accepté par la machine universelle

- Les entrées de la machine $\mathcal{U}$ doivent être de la forme $[\langle \mathcal{M}, w \rangle]_{code}$.
  - Si on propose une entrée mal formattée, i.e. un mot u tel que $\forall \mathcal{M}, \forall w, u \neq [\langle \mathcal{M}, w \rangle]_{code}$, $\mathcal{U}$ rejette son entrée en temps fini.
  - Sinon, on passe à la simulation de $\mathcal{M}$.
- $\mathcal{U}$ explore l'arbre d'exécution de $\mathcal{M}$ sur l'entrée w.
  - Si il existe une configuration acceptante accessible depuis la configuration initiale de $\mathcal{M}$ sur w, c'est à dire $q_0\, w$, on **accepte**.

Langage accepté par la machine universelle

- ☞ Les entrées de la machine $\mathcal{U}$ doivent être de la forme $[\langle \mathcal{M}, w \rangle]_{code}$.
  - Si on propose une entrée mal formattée, i.e. un mot u tel que $\forall \mathcal{M}, \forall w, u \neq [\langle \mathcal{M}, w \rangle]_{code}$, $\mathcal{U}$ rejette son entrée en temps fini.
  - Sinon, on passe à la simulation de $\mathcal{M}$.
- ☞ $\mathcal{U}$ explore l'arbre d'exécution de $\mathcal{M}$ sur l'entrée w.
  - Si il existe une configuration acceptante accessible depuis la configuration initiale de $\mathcal{M}$ sur w, c'est à dire $q_0\, w$, on **accepte**.
  - Si l'arbre d'exécution est fini, i.e. il n'y a pas d'exécutions infinie, et si aucune configuration acceptante n'est accessible, on **rejette**.

Langage accepté par la machine universelle

- Les entrées de la machine $\mathcal{U}$ doivent être de la forme $[\langle \mathcal{M}, w \rangle]_{code}$.
  - Si on propose une entrée mal formattée, i.e. un mot u tel que $\forall \mathcal{M}, \forall w, u \neq [\langle \mathcal{M}, w \rangle]_{code}$, $\mathcal{U}$ rejette son entrée en temps fini.
  - Sinon, on passe à la simulation de $\mathcal{M}$.
- $\mathcal{U}$ explore l'arbre d'exécution de $\mathcal{M}$ sur l'entrée w.
  - Si il existe une configuration acceptante accessible depuis la configuration initiale de $\mathcal{M}$ sur w, c'est à dire $q_0\, w$, on **accepte**.
  - Si l'arbre d'exécution est fini, i.e. il n'y a pas d'exécutions infinie, et si aucune configuration acceptante n'est accessible, on **rejette**.
  - Si l'arbre d'exécution est infini mais qu'il ne contient aucune configuration acceptante, $\mathcal{U}$ ne s'arrête jamais.

Langage accepté par la machine universelle

☞ Les entrées de la machine $\mathcal{U}$ doivent être de la forme $[\langle \mathcal{M}, w \rangle]_{code}$.

- Si on propose une entrée mal formattée, i.e. un mot u tel que $\forall \mathcal{M}, \forall w, u \neq [\langle \mathcal{M}, w \rangle]_{code}$, $\mathcal{U}$ rejette son entrée en temps fini.

- Sinon, on passe à la simulation de $\mathcal{M}$.

☞ $\mathcal{U}$ explore l'arbre d'exécution de $\mathcal{M}$ sur l'entrée w.

- Si il existe une configuration acceptante accessible depuis la configuration initiale de $\mathcal{M}$ sur w, c'est à dire $q_0\, w$, on **accepte**.

- Si l'arbre d'exécution est fini, i.e. il n'y a pas d'exécutions infinie, et si aucune configuration acceptante n'est accessible, on **rejette**.

- Si l'arbre d'exécution est infini mais qu'il ne contient aucune configuration acceptante, $\mathcal{U}$ ne s'arrête jamais.

☞ On en déduit :

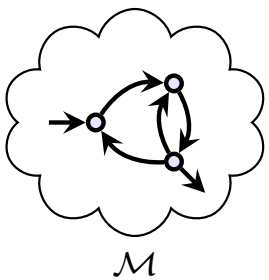
Langage accepté par la machine universelle

- Les entrées de la machine $\mathcal{U}$ doivent être de la forme $[\langle \mathcal{M}, w \rangle]_{code}$.
  - Si on propose une entrée mal formattée, i.e. un mot u tel que $\forall \mathcal{M}, \forall w, u \neq [\langle \mathcal{M}, w \rangle]_{code}$, $\mathcal{U}$ rejette son entrée en temps fini.
  - Sinon, on passe à la simulation de $\mathcal{M}$.
- $\mathcal{U}$ explore l'arbre d'exécution de $\mathcal{M}$ sur l'entrée w.
  - Si il existe une configuration acceptante accessible depuis la configuration initiale de $\mathcal{M}$ sur w, c'est à dire $q_0\, w$, on **accepte**.
  - Si l'arbre d'exécution est fini, i.e. il n'y a pas d'exécutions infinie, et si aucune configuration acceptante n'est accessible, on **rejette**.
  - Si l'arbre d'exécution est infini mais qu'il ne contient aucune configuration acceptante, $\mathcal{U}$ ne s'arrête jamais.
- On en déduit :
  - $\mathcal{L}(\mathcal{U}) := \{w \mid q_0\, w \to_{\mathcal{U}}^* C \text{ acceptante}\} = \{[\langle \mathcal{M}, w \rangle]_{code} \mid w \in \mathcal{L}(\mathcal{M})\}$.

Langage accepté par la machine universelle

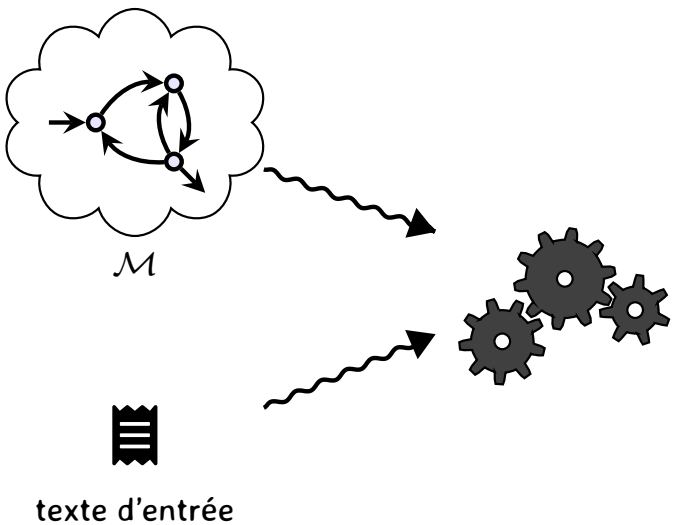
- Les entrées de la machine $\mathcal{U}$ doivent être de la forme $[\langle \mathcal{M}, w \rangle]_{code}$.
  - Si on propose une entrée mal formattée, i.e. un mot u tel que $\forall \mathcal{M}, \forall w, u \neq [\langle \mathcal{M}, w \rangle]_{code}$, $\mathcal{U}$ rejette son entrée en temps fini.
  - Sinon, on passe à la simulation de $\mathcal{M}$.
- $\mathcal{U}$ explore l'arbre d'exécution de $\mathcal{M}$ sur l'entrée w.
  - Si il existe une configuration acceptante accessible depuis la configuration initiale de $\mathcal{M}$ sur w, c'est à dire $q_0\, w$, on **accepte**.
  - Si l'arbre d'exécution est fini, i.e. il n'y a pas d'exécutions infinie, et si aucune configuration acceptante n'est accessible, on **rejette**.
  - Si l'arbre d'exécution est infini mais qu'il ne contient aucune configuration acceptante, $\mathcal{U}$ ne s'arrête jamais.
- On en déduit :
  - $\mathcal{L}(\mathcal{U}) := \{w \mid q_0\, w \to_{\mathcal{U}}^* C \text{ acceptante}\} = \{[\langle \mathcal{M}, w \rangle]_{code} \mid w \in \mathcal{L}(\mathcal{M})\}$.
  - $\mathcal{U}$ s'arrête sur l'entrée $[\langle \mathcal{M}, w \rangle]_{code}$ si et seulement si $\mathcal{M}$ n'admet pas d'exécution infinie sur l'entrée w.

La machine universelle
Utilisation habituelle d'une machine de Turing

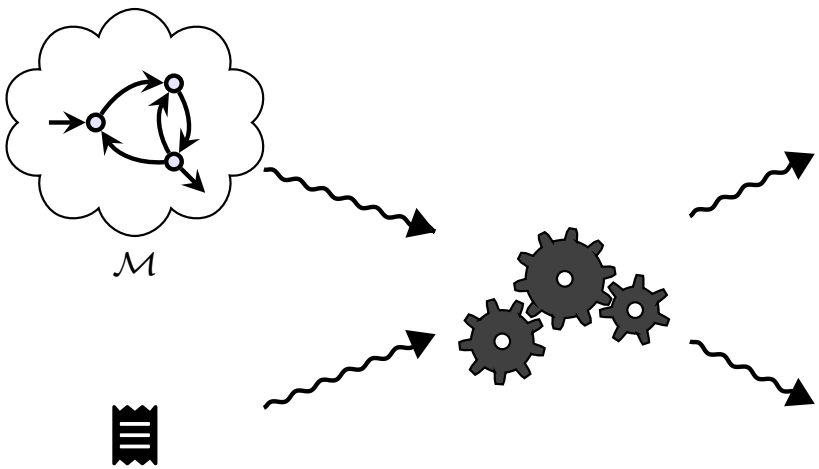


texte d'entrée

La machine universelle
Utilisation habituelle d'une machine de Turing



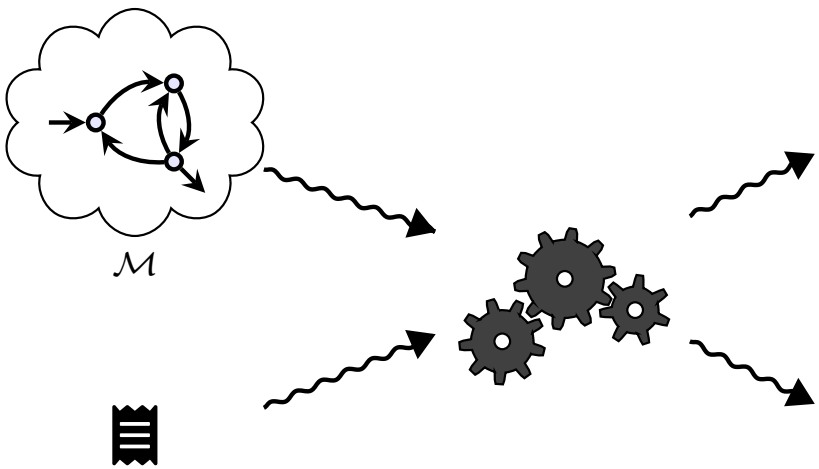
La machine universelle
Utilisation habituelle d'une machine de Turing



il existe une exécution acceptante de $\mathcal{M}$

il n'existe pas d'exécution acceptante de $\mathcal{M}$

La machine universelle
Utilisation habituelle d'une machine de Turing

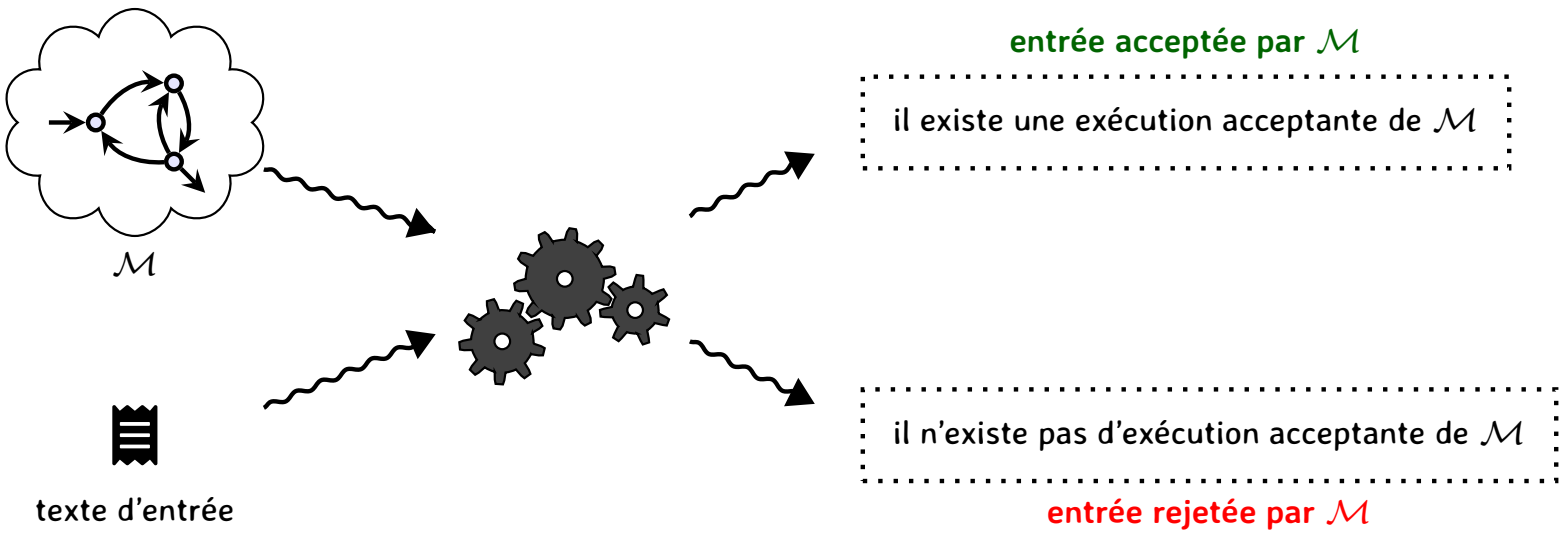


entrée acceptée par $\mathcal{M}$

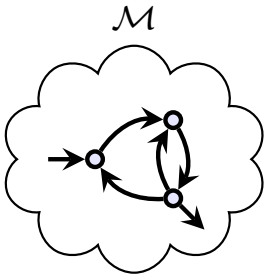
il existe une exécution acceptante de $\mathcal{M}$

il n'existe pas d'exécution acceptante de $\mathcal{M}$

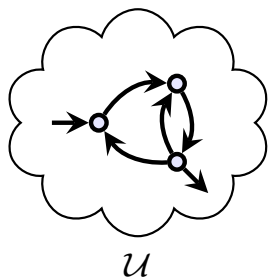
La machine universelle
Utilisation habituelle d'une machine de Turing



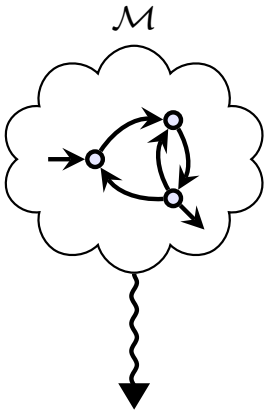
La machine universelle
Utilisation d'une machine de Turing via la machine universelle

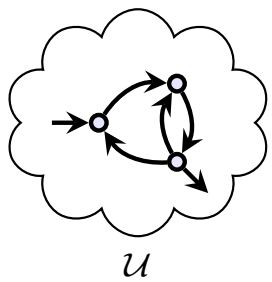


texte d'entrée

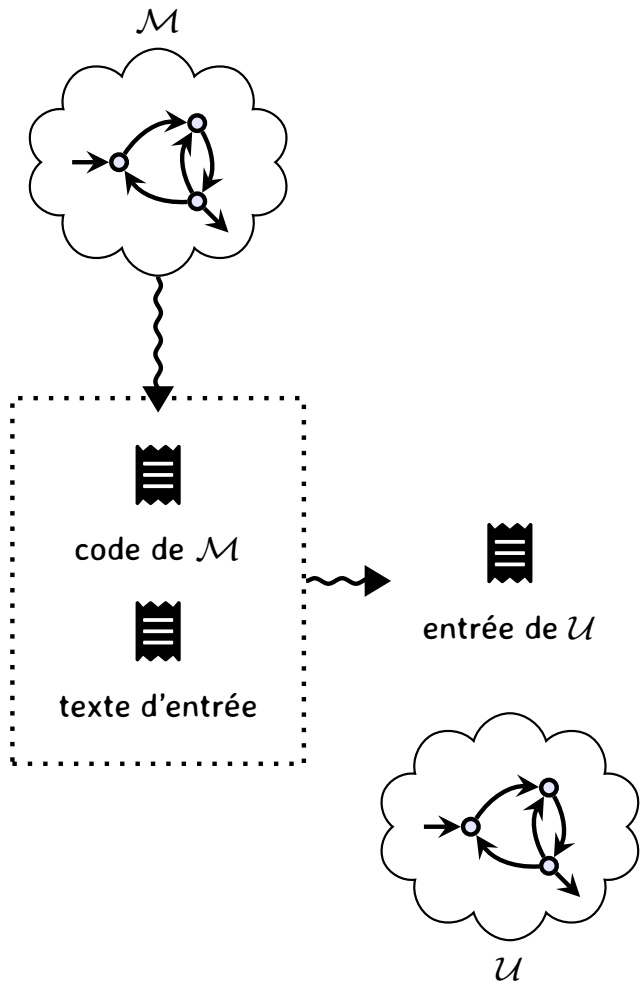


La machine universelle
Utilisation d'une machine de Turing via la machine universelle



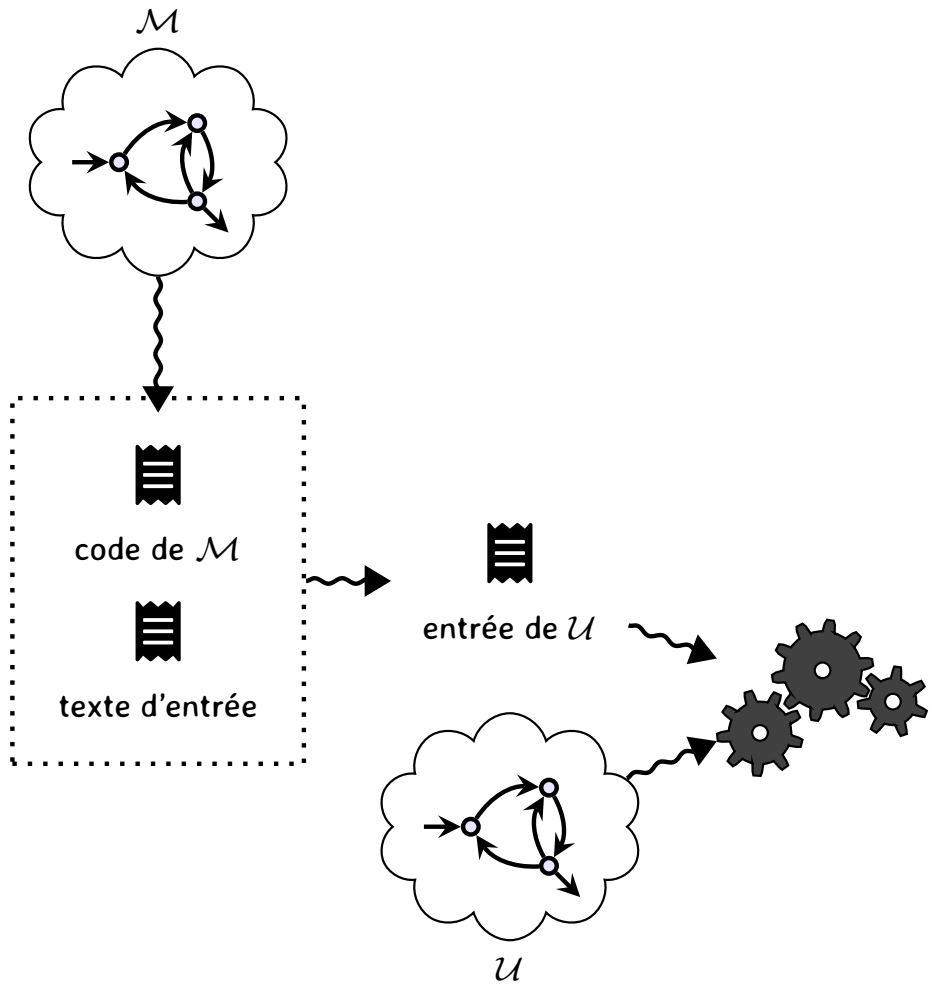


La machine universelle
Utilisation d'une machine de Turing via la machine universelle

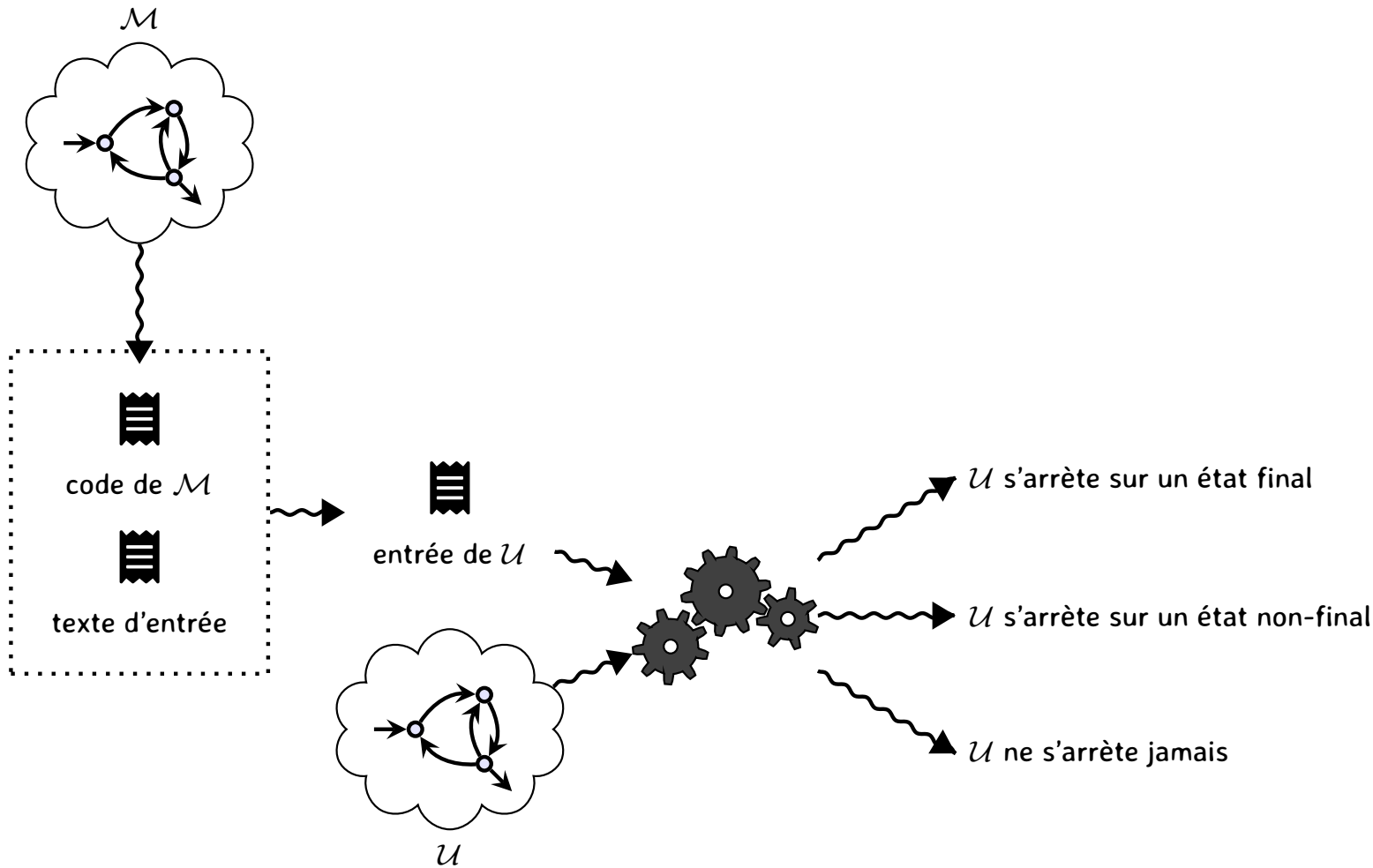


La machine universelle

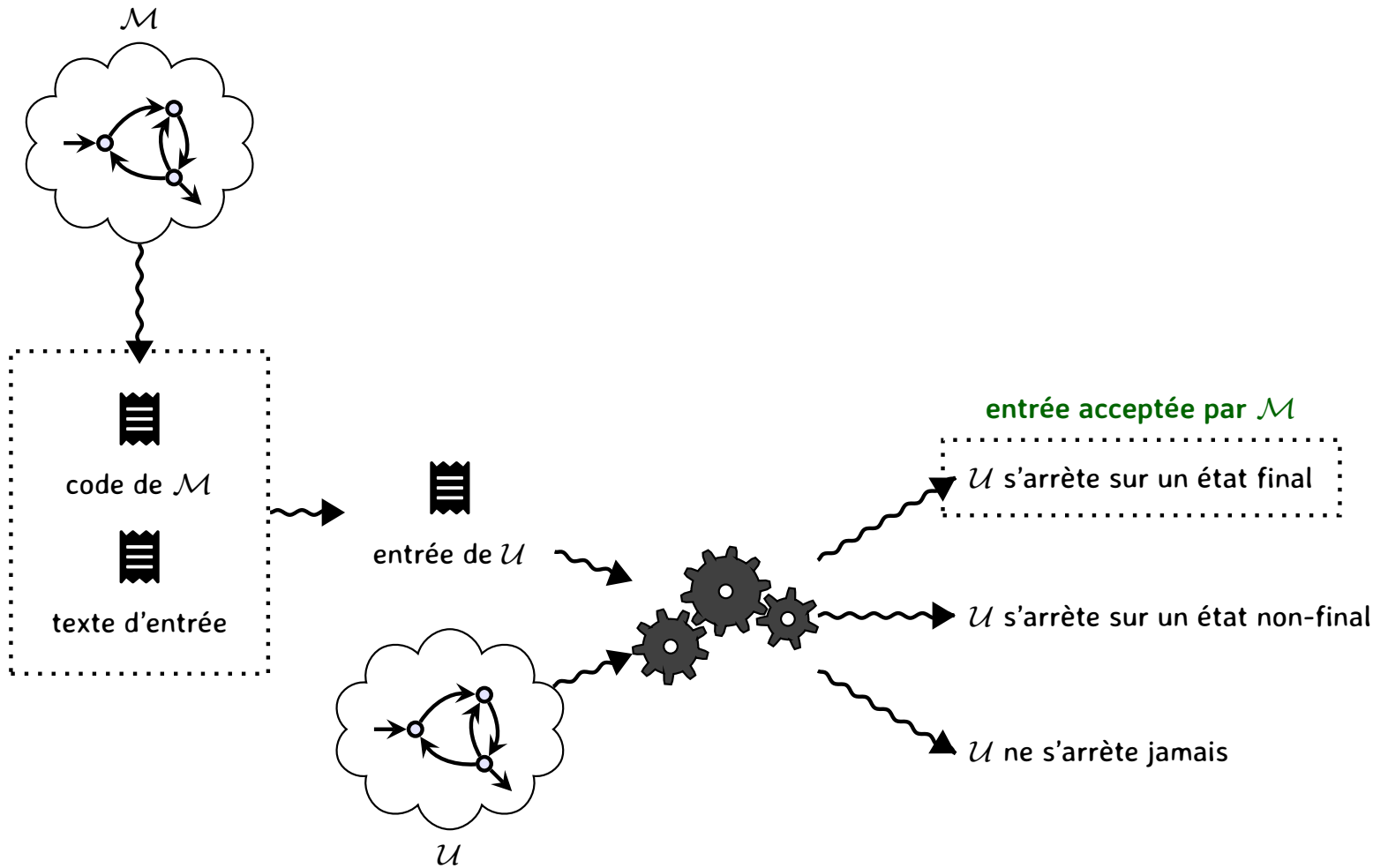
Utilisation d'une machine de Turing via la machine universelle



La machine universelle
Utilisation d'une machine de Turing via la machine universelle



La machine universelle
Utilisation d'une machine de Turing via la machine universelle



La machine universelle
Utilisation d'une machine de Turing via la machine universelle