Minimisation de machines de Turing, application au problème 2Color

Elowan Harnisch

August 28, 2024

Machine de Turing

Définition Une machine de Turing est un sixtuplet $(Q, \Sigma, S, q_0, F, \delta)$ où:

- Q est un ensemble fini d'états
- Σ est un ensemble fini de symboles
- $S \notin \Sigma$ est le symbole blanc
- $q_0 \in Q$ est l'état initial
- $F \subseteq Q$ est l'ensemble des états acceptants
- $\delta : Q \times \Sigma \to Q \times \Sigma \times \{L, R\}$ est la fonction de transition

Elle possède de plus une bande infinie sur laquelle elle peut lire et écrire les symboles de $\Sigma \cup S$.

Machine de Turing

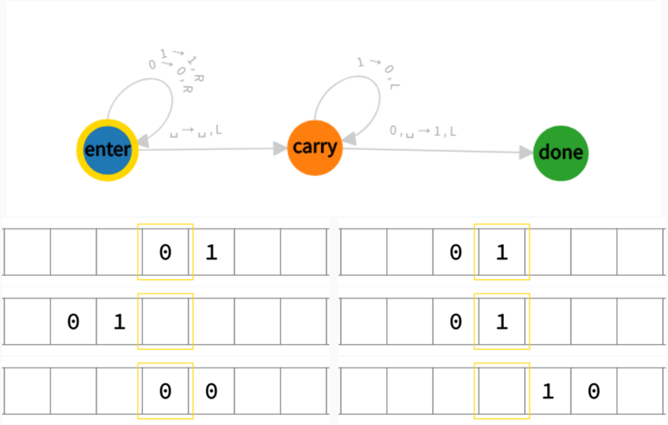

Figure 1: Schema d'execution de machine de Turing qui ajoute 1 à un nombre binaire

Automate fini déterministe (DFA)

Définition Un automate fini déterministe (DFA) est un quintuplet $(Q, \Sigma, q_0, F, \delta)$ où:

- Q est un ensemble fini d'états
- Σ est un ensemble fini de symboles
- $q_0 \in Q$ est l'état initial
- $F \subseteq Q$ est l'ensemble des états finaux
- $\delta : Q \times \Sigma \to Q$ est la fonction de transition

Automate fini déterministe (DFA)

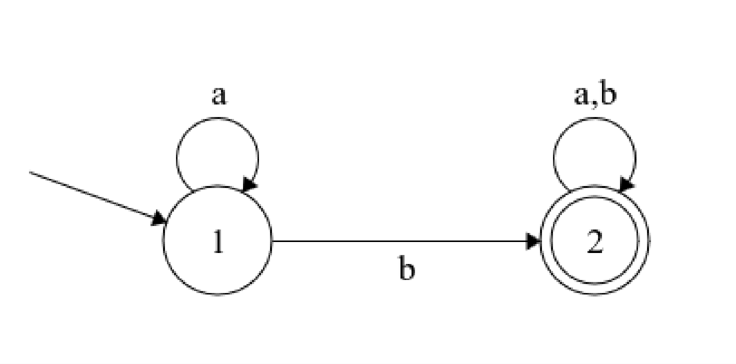

Figure 2: Schema d'un automate reconnaissant le langage des mots ayant au moins un b

Minimisation d’un DFA

Définition La minimisation d’un automate fini déterministe est le processus qui consiste à réduire le nombre d’états de l’automate tout en conservant le langage reconnu.

Théorème
L’algorithme de Hopcroft permet depuis un automate fini déterministe d’obtenir un automate minimal reconnaissant le même langage.

Minimisation d'un automate fini déterministe

Algorithm 1 Algorithme de Hopcroft, $A = (Q, \Sigma, q_0, F, \delta)$ un automate fini déterministe

$P \leftarrow \{F, Q \setminus F\}$
$W \leftarrow \{\}$
for for $a \in \Sigma$ do **do**
    ajouter $(min(F, Q \setminus F), a)$ à W
end for
while $W \neq \{\}$ do **do**
    retirer (Z, a) de W
    for $X \in P$ tel que X est coupé par (Z, a) en X' et X'' do **do**
        remplacer X par X' et X'' dans P
        for $b \in \Sigma$ do **do**
            if (X, b) est dans W **then**
                remplacer (X, b) par (X', b) et (X'', b) dans W
            else
                ajouter $min((X', b), (X'', b))$ à W
            end if
        end for
    end for
end while

Minimisation d'une machine de Turing

Transformation d'une machine de Turing en DFA Soit une machine de Turing $M = (Q, \Sigma, S, q_0, F, \delta)$, on peut construire un DFA $A = (Q, \Sigma', q_0, F, \delta')$ tel que:

- $\Sigma' = \Sigma \cup \{aKb | ((a, b) \in \Sigma^2, K \in \{R, L\})\} \cup \{S\}$
- $\delta'(q, aKb) = q'$, avec $(q', b, K) = \delta(q, a)$ et $a \in \Sigma, q \in Q$

On appelle f la fonction qui réalise cette transformation.

On en déduit son inverse en décomposant chaque lettre de A

Théorème
La composition de f, de l'algorithme de minimisation de Hopcroft et de f^{-1} permet d'obtenir une machine de Turing de taille au plus égale à celle de départ.

Problème n-2Color

Définition Le problème 2Color est un problème de décision qui consiste à déterminer si un graphe est coloriable avec 2 couleurs.

Théorème
Le problème 2Color est décidable.

Définition Le problème n-2Color est un problème de décision qui consiste à déterminer si un graphe est coloriable avec 2 couleurs pour un graphe de maximum n sommets.

Théorème
Le problème n-2Color est décidable.

Problème n-2Color

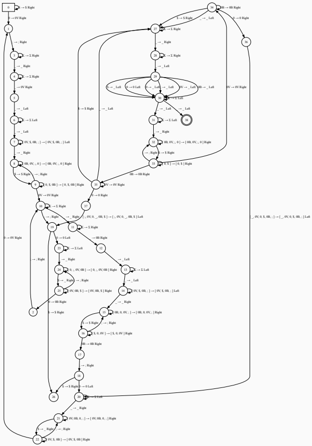

Figure 3: Machine de Turing créée pour résoudre le problème 2Color, pour un graphe de 2 sommets maximum

Conclusion

Pour la suite j'aimerais :

- Bien comprendre la preuve de l'algorithme de Hopcroft et mieux l'adapter aux machines de Turing plutôt que mon bricolage actuel (même si je pense que ce n'est pas possible, dans ce cas démontrer que ce problème n'est pas décidable mais qu'on peut essayer de s'en approcher par cette méthode)
- Finir de débugger la machine de Turing pour le problème n-2Color