

Devoir maison Informatique classique

Luc Lapointe
luc.lapointe@ens-paris-saclay.fr
home.lmf.cnrs.fr/LucLapointe/

Exercice : Automate des parties

Le modèle des automates non déterministes est aussi puissant que celui des automates déterministes. En d'autres mot, tout langage reconnu par un automate non déterministe peut aussi être reconnu par un automate déterministe. Cet exercice vise à présenter un algorithme qui prend en entrée un automate non déterministe et renvoie un automate déterministe reconnaissant le même langage.

L'*automate des parties* d'un automate non déterministe est une construction qui reconnaît le même langage. C'est la construction sur laquelle cet exercice s'appuie.

Definition: Automate des parties

Soit $\mathcal{A} = (Q, \Sigma, I, F, \delta)$ un automate non déterministe. Son **automate des parties** est l'automate déterministe $\mathcal{A}' = (Q', \Sigma, q_i, F', \delta')$ défini de la manière suivante:

- Q' est l'ensemble des ensembles d'éléments de Q :

$$Q' = \{E \mid E \subseteq Q\}$$

- $q_i = I$
- F' est l'ensemble des ensembles d'éléments de Q contenant au moins un état final :

$$F' = \{E \mid E \subseteq Q \wedge E \cap F \neq \emptyset\}$$

- $\delta'(E, l)$ est l'image de E par $\delta(., x)$:

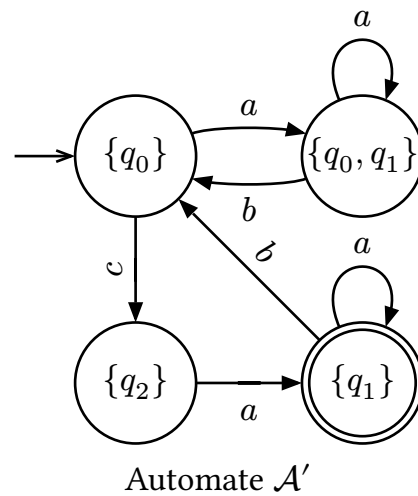
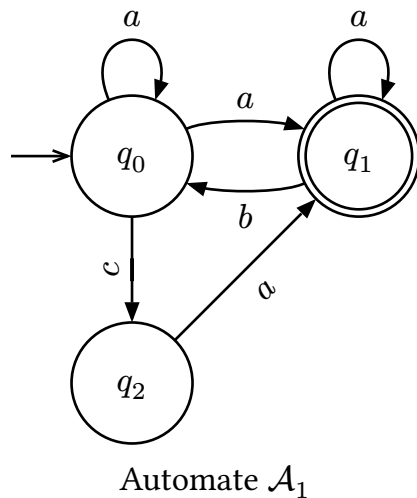
$$\delta'(E, l) = \{y \mid \exists x \in E, \delta(x, l) = y\}$$

On notera $\text{Part}(\mathcal{A})$ l'automate des parties de $\mathcal{A}$.

Théorème

Soit $m \in \Sigma^*$. Lire m avec $\text{Part}(\mathcal{A})$ amène à un état E si et seulement si lire m avec $\mathcal{A}$ peut amener à n'importe quel état de E , et aucun autre.

Question 1 Soit $\mathcal{A}_1$ l'automate ci-dessous. Prouver que l'automate $\mathcal{A}'$ ne reconnaît pas le même langage. Proposer une correction.



Question 2 En admettant le théorème ci-dessus, montrer par l'absurde que $\text{Part}(\mathcal{A})$ est un automate déterministe.

Question 3 Exprimer le nombre d'états de $\text{Part}(\mathcal{A})$ en fonction du nombre d'états de $\mathcal{A}$.

Comme un automate des parties compte beaucoup d'états, il est intéressant de réduire sa taille. Nous allons aborder ici un algorithme qui construit un automate des parties duquel ont été enlevés des états considérés comme inutile.

Definition: États accessibles, co-accessibles, automate émondé

Soit $\mathcal{A}$ un automate. Un état q de l'automate est **accessible** s'il existe un mot qui, lu par l'automate depuis un état initial, atteint q . Un état q est **co-accessible** s'il existe un mot qui, lu par l'automate depuis q , atteint un état final. Un automate est **émondé** si tous ses états sont à la fois accessibles et co-accessibles.

Exemple L'automate $\mathcal{A}'$ est *presque* une version émondée de l'automate $\text{Part}(\mathcal{A}_1)$.

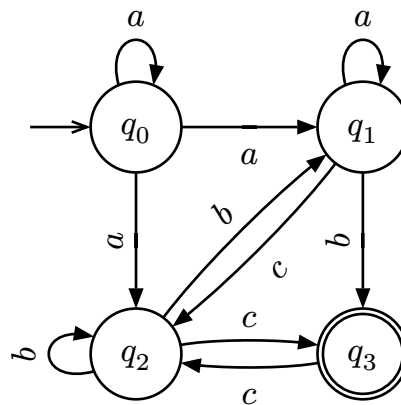
Cet algorithme prend en entrée un automate $\mathcal{A}$ et construit petit à petit la version émondée de $\text{Part}(\mathcal{A})$, qu'il finit par renvoyer.

Algorithme

Entrée : $\mathcal{A} = (Q, \Sigma, I, F, \delta)$

- Initier F' avec $\emptyset$
- Initier Q' avec l'état I
- Si $I \cap F \neq \emptyset$, rajouter I à F'
- Initier δ' comme une fonction vide
- Créer une liste Visités, initialement vide
- Tant que Q' contient un état qui n'est pas dans Visités :
 - Sélectionner un état quelconque S de Q' qui n'est pas dans Visités
 - Ajouter S à Visités
 - Pour tout $l \in \Sigma$:
 - Initier T à $\emptyset$
 - Pour tout $x \in S$, ajouter $\delta(x, l)$ à T
 - Ajouter à δ' une transition $S \rightarrow T$
 - Ajouter T à Q' s'il n'y est pas déjà
 - Si $T \cap F \neq \emptyset$, rajouter T à F'
- Renvoyer $(Q', \Sigma, I, F', \delta')$

Question 4 Soit $\mathcal{A}_2$ l'automate ci-dessous. Lui appliquer l'algorithme et dessiner l'automate $\mathcal{A}' = (Q', \Sigma, I, F', \delta')$ qui en résulte.



Automate $\mathcal{A}_2$

Automates à observation partielle

Abstract

Les **automates à observation partielle** sont un outil qui a pour but de modéliser des systèmes dont un observateur connaît le fonctionnement mais n'a qu'une vue partielle; par exemple, la DSI d'un parc informatique, ou alors des pirates attaquant ou espionnant un système. Ils présentent l'avantage d'être un modèle suffisamment proche des automates vus en cours, ce qui permet de réutiliser les outils et méthodes développés pour ceux-ci.

Les automates à observation partielle présentés ici ont pour but de modéliser des systèmes en fonctionnement, comme par exemple une machine, ou un intranet. Le fonctionnement normal du système se traduit dans le modèle par un déplacement d'état en état selon les arêtes.

Definition: Chemin

Un **chemin** est une suite, finie ou infinie, d'arêtes consécutives dans un automate.

On suppose qu'un observateur extérieur ne peut jamais voir quelles arêtes sont empruntées. Par contre, les étiquettes jouent le rôle d'informations observables.

Definition: Trace

La suite des étiquettes d'un chemin est appelée une **trace**.

Ainsi, une longue exécution de l'automate se traduit par un long chemin, dont les observateurs ne peuvent distinguer qu'une longue trace.

Les automates à observation partielle peuvent servir à modéliser des machines qui ont vocation à être faciles à utiliser, auquel cas les traces peuvent correspondre aux messages que la machine affiche sur son écran, qui gagnent à être aussi peu ambigus que possible. Mais ils peuvent aussi servir à modéliser des systèmes fermés aux observations extérieurs, où les étiquettes des arêtes représentent les fuites d'information; par exemple, l'augmentation de la consommation d'électricité ou de la température d'un serveur.

Les deux sections qui suivent présentent des variations du modèle des automates à observation partielles adaptées à ces deux cas d'usage.

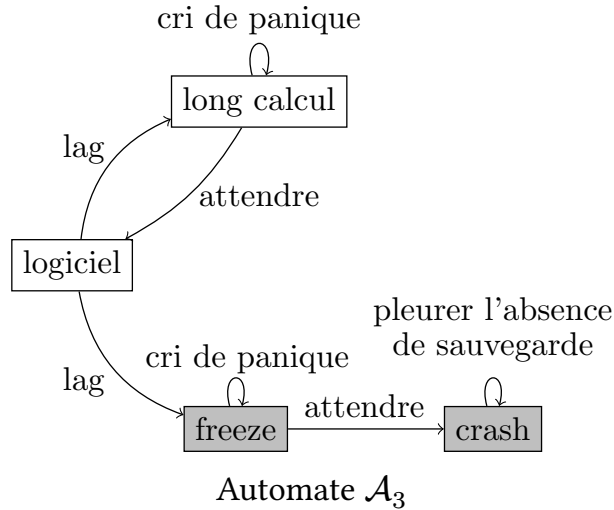
1. Automates à diagnostiquer

J'ai découvert cette notion dans une présentation du papier [1].

Definition: Automate à diagnostiquer

Un **automate à diagnostiquer** est un automate non déterministe $\mathcal{A}$ muni d'un ensemble $Q_b \subseteq Q$ d'**états buggés**.

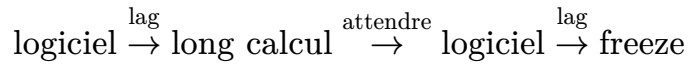
Exemple L'automate ci-dessous est un automate à diagnostiquer, dont les états buggés sont grisés.



Definition: Chemin buggé

Un **chemin buggé** est un chemin qui passe par un état buggé.

Exemple Le chemin suivant est un chemin buggé:



Definition: Automate diagnosticable

Un automate à diagnostiquer est **diagnosticable** si aucune trace infinie dans cet automate ne peut correspondre à la fois à un chemin buggé et à un chemin non buggé.

Question 5 Est-ce que l'automate $\mathcal{A}_3$ est diagnosticable ? Si oui, le prouver, si non, exhiber une trace infinie qui peut correspondre à la fois à un chemin buggé ou un chemin non buggé.

Question 6 Montrer qu'il existe au moins un automate diagnosticable et au moins un automate non diagnosticable.

Question 7 Exhiber un automate diagnosticable dans lequel quelle que soit la trace, l'observateur extérieur ne peut jamais être sûr du chemin emprunté.

L'automate $\mathcal{A}_3$ peut modéliser une situation où on observe le comportement d'un utilisateur qu'on observe face à sa machine; mais il pourrait aussi modéliser notre

propre comportement face à une machine. Dans ce cas, on aimerait pouvoir décider, à certains endroits, quelles transitions on décide d'emprunter. L'extension suivante le permet.

Definition: Automate à diagnostiquer activement

Un **automate à diagnostiquer activement** est un automate à diagnostiquer $\mathcal{A}$ tel que:

- l'alphabet Σ est partitionné entre un **alphabet de contrôle** Σ_c et un **alphabet d'environnement** Σ_e ,
- chaque état ne peut avoir soit que des arêtes sortantes étiquetées par Σ_e , soit que des arêtes sortantes étiquetées par Σ_c .

Les éléments de l'alphabet de contrôle sont appelés **action**.

Exemple L'automate $\mathcal{A}_3$ peut être vu comme un automate à diagnostiquer activement avec $\Sigma_c = \{\text{cri de panique, attendre, pleurer l'absence de sauvegarde}\}$ et $\Sigma_e = \{\text{lag}\}$.

Definition: Stratégie de contrôle

Une **stratégie de contrôle** σ est une fonction de Σ^* vers Σ_c qui à toute trace associe une action, qui dicte la chose à faire lorsqu'un choix est proposé.

Une trace t est **cohérente avec une stratégie de contrôle** σ si chaque action dans la trace est le résultat de la stratégie par les éléments qui précèdent.

Autrement, dit, si $t = t_1 t_2 t_3 \dots$, alors

$$\forall i \in \mathbb{N}, t_i \in \Sigma_c \Rightarrow t_i = \sigma(t_1 t_2 \dots t_{i-1})$$

Exemple Dans l'automate $\mathcal{A}_3$, la fonction qui à tout mot associe « cri de panique » est une stratégie de contrôle.

Definition: Automate activement diagnosticable

Un automate à diagnostiquer activement est **activement diagnosticable** s'il existe une stratégie de contrôle σ telle que aucune trace infinie cohérente avec σ ne peut correspondre à la fois à un chemin buggé et à un chemin non buggé.

Question 8 Est-ce que l'automate $\mathcal{A}_3$ est activement diagnosticable ? Quelle que soit la réponse, le prouver.

Question 9 Montrer qu'il existe au moins un automate activement diagnosticable et au moins un automate non activement diagnosticable.

Question 10 Est-ce qu'il existe un automate à diagnostiquer activement qui est à la fois activement diagnosticable et non diagnosticable ? Si oui, en dessiner un, si non, le prouver.

Question 11 Est-ce qu'il existe un automate à diagnostiquer activement qui est à la fois diagnosticable et non activement diagnosticable ? Si oui, en dessiner un, si non, le prouver.

2. Automates à secret

J'ai découvert cette notion dans une présentation du papier [2].

Definition: Automate à secret

Un **automate à secret** est un automate non déterministe $\mathcal{A}$ muni d'un ensemble $Q_s \subseteq Q$ d'**états secrets**.

Definition: Automate opaque

Un automate à secret est **opaque** si, pour tout chemin passant par un état secret, il existe un autre chemin ne passant pas par un état secret avec la même trace.

Un automate opaque est donc un automate à secret dont on ne peut pas, simplement en observant ses traces, décider s'il passe par un état secret ou non.

Exemple L'automate $\mathcal{A}_3$ peut être vu comme un automate à secret, en considérant que les états grisés ne sont plus des états buggés mais des états secrets.

Question 12 Est-ce que l'automate $\mathcal{A}_3$ est opaque ? Justifier.

Question 13 Montrer qu'il existe un automate à secret opaque et un automate à secret pas opaque.

Question 14 Écrire un algorithme qui prend en entrée un automate à secret $\mathcal{A}$, et répond « Oui » si et seulement si $\mathcal{A}$ est opaque.

Bibliographie

- [1] S. Haddad, E. Lefauchaux, et S. Schwoon, « Active Diagnosis with Costs and Rewards », in *37th International Conference on Concurrency Theory (CONCUR 2026)*, A. Sokolova et P. Totzke, Éd., in Leibniz International Proceedings in Informatics (LIPIcs), vol. 391. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2026, p. 37:1–37:16. doi: 10.4230/LIPIcs.CONCUR.2026.37.
- [2] É. André, S. Dépernet, et E. Lefauchaux, « Buffered Control for Opacity in Timed Automata », in *37th International Conference on Concurrency Theory (CONCUR 2026)*, A. Sokolova et P. Totzke, Éd., in Leibniz International Proceedings in Informatics (LIPIcs), vol. 391. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2026, p. 11:1–11:21. doi: 10.4230/LIPIcs.CONCUR.2026.11.