Minimisation d'automates sur les mots infinis

Jean Abou Samra

Résumé. Ce rapport de stage aborde le problème de la minimisation des automates de divers types sur les mots infinis : automates de Büchi ou co-Büchi, déterministes ou GFG. Le problème est relativement bien compris avec la définition classique de l'acceptation, en termes des états visités infiniment souvent ou non par une exécution. Nous explorons le cas de l'acceptation par transitions, où ce sont les transitions visitées infiniment souvent, et non les états, qui définissent les exécutions acceptantes.

Table des matières

I Introduction

Les automates font partie des modèles de calcul les plus étudiés et les plus utiles en informatique. Le cadre le plus classique est celui des automates déterministes sur des mots finis, qui reconnaissent exactement les langages décrits par une expression régulière (théorème de Kleene), et sur lesquels toutes les opérations usuelles (notamment l'intersection, l'union et la complémentation) sont réalisables en temps polynomial, ainsi que le test d'universalité (« le langage de l'automate contient-il tous les mots ? »).

Le cadre des automates finis non-déterministes est également classique. Ces automates ont le même pouvoir expressif que les automates finis déterministes, puisque l'on peut convertir un automate non-déterministe en automate déterministe grâce à la « construction des sous-ensembles » (*subset construction*). Toutefois, grâce au pouvoir du non-déterminisme, un automate non-déterministe peut être exponentiellement plus succinct (c'est-à-dire de taille exponentiellement plus petite) qu'un automate déterministe minimal qui reconnaît le même langage. L'intersection et l'union restent polynomiales ; en revanche, un automate non-déterministe reconnaissant le complémentaire du langage d'un automate non-déterministe peut demander exponentiellement plus d'états, si bien que la complémentation d'un automate non-déterministe demande un temps exponentiel (mais un espace polynomial). Le test d'universalité sur les automates non-déterministes est PSPACE-complet [1].

Les automates sur les mots finis sont donc un modèle bien compris et agréable à manipuler, qui possède de plus d'innombrables applications, parmi lesquelles on peut citer : l'implémentation efficace de la recherche d'une expression régulière dans un texte, en convertissant l'expression régulière en automate ; la recherche d'une sous-chaîne de caractères par l'algorithme de Knut-Morris-Pratt ; etc.

Toutefois, le formalisme des automates sur les mots finis n'est pas adéquat pour les applications issues de la vérification (*model checking*). Dans ce domaine, les propriétés à vérifier sont exprimées en termes de logiques qui s'appliquent à des objets infinis, comme LTL (la logique linéaire temporelle), dont la sémantique est définie sur les mots infinis (« un mot infini $a_0a_1...$ satisfait la formule φ si et seulement si … »)[1]. Il est très utile en pratique de pouvoir exprimer une formule LTL sous la forme d'un automate équivalent. Pour cela, il faut d'abord définir une extension des automates usuels aux mots infinis. La notion de chemin parcouru par l'automate sur un mot a toujours du sens, mais un chemin infini n'a pas d'état final, donc la condition d'acceptation classique des automates sur les mots finis n'a plus de sens.

[1] Il existe d'autres logiques intéressantes en vérification, comme CTL (*computation tree logic*), dont la sémantique s'applique à des arbres infinis. Toutefois, ce rapport se limite au cas des automates sur des mots, finis ou infinis, et ne traite pas le cas des arbres, qui est plus complexe.

Au contraire des automates sur les mots finis, où la condition d'acceptation par état final est presque toujours utilisée, il existe de nombreux types de condition d'acceptation pour les mots infinis, parmi lesquels on peut citer les conditions de Büchi, de co-Büchi, de Büchi faible, de Büchi généralisé, de parité, de Rabin, de Street ou encore de Muller. (On peut trouver dans [2] un aperçu très complet des différents types.) Toutes ces conditions sont définies en termes des états qui sont visités infiniment souvent par un chemin. Par exemple, dans un automate de Büchi, la condition d'acceptation est un ensemble d'états F, et une exécution est acceptée si et seulement si F contient un état visité infiniment souvent.

Le grand nombre de types d'automates différents est l'une des raisons pour lesquelles beaucoup de problèmes naturels sur les automates infinis ont une complexité encore mal comprise, malgré les décennies de recherche qui ont déjà été effectuées en vérification.

Dans ce rapport, nous nous penchons plus spécialement sur le problème de la minimisation, qui se pose ainsi : étant donné un automate d'un certain type, trouver un automate du même type, avec un nombre minimal d'états, qui reconnaisse le même langage. Pour tous les types d'automates cités ci-dessus, l'égalité des langages de deux automates est décidable, donc une simple recherche exhaustive montre que le problème de minimisation est décidable. Toute la question est de connaître sa complexité.

Comme il est usuel en complexité, on peut définir à partir de ce problème de calcul un problème de décision, à savoir : étant donné un automate d'un certain type, et un entier k, existe-t-il un automate du même type, dont le nombre d'états est au plus k, qui reconnaît le même langage ?

Non seulement ce problème est naturel du point de vue théorique, mais il est d'un intérêt pratique évident. Si un algorithme sur les automates a une haute complexité (par exemple exponentielle), et que la minimisation est efficace (par exemple polynomiale), on peut minimiser l'automate avant d'exécuter l'algorithme, et espérer un gain de temps significatif.

Avant de donner les résultats existants, notons quelques limites de notre étude :

- La mesure retenue de taille d'un automate est le nombre d'états. Cette mesure est raisonnable car la taille totale d'une représentation de l'automate est au plus quadratique en le nombre d'états, au moins pour les conditions de Büchi, co-Büchi, Büchi faible et parité. Toutefois, il pourrait être intéressant d'étudier la minimisation d'une mesure différente, par exemple la somme du nombre d'états et du nombre de transitions (dans un automate non complet).
- Nous n'abordons que la minimisation parfaite, qui produit un automate le plus petit possible. On pourrait aussi étudier le problème d'approximation associé à ce problème de minimisation (ce qui serait surtout intéressant dans les cas où la minimisation est difficile). Une autre direction possible est de développer des techniques qui réduisent l'automate sans pour autant forcément produire un automate minimal, ce qui est fait par exemple dans [3] (section 4), ou encore [4].
- Dans notre problème, l'automate en sortie doit obligatoirement reconnaître le même langage que l'automate d'entrée. Au vu de la mauvaise complexité du problème pour certains types d'automates, on peut se demander s'il est possible d'obtenir une meilleure complexité en permettant un écart entre le langage de départ et le langage d'arrivée, par exemple en exigeant seulement que leur différence symétrique soit finie. Dans une direction différente, on peut restreindre le problème à des classes de langages spécifiques, ce qui est fait dans [5] en considérant les langages associés à des propriétés LTL de type « liveness » (qui sont, de manière très schématique, des propriétés exprimant que toute action d'un certain type est suivie d'une réaction d'un certain type).

Comme nombre d'autres problèmes, la minimisation est bien plus simple à étudier sur les automates de mots finis que sur les automates de mots infinis. Dans le cas des automates déterministes sur les mots finis, on peut définir une relation d'équivalence entre états en posant que q et s sont équivalents si et seulement si les mots acceptés à partir de q sont exactement les mots acceptés à partir de s. On montre alors que cette relation peut se calculer en temps polynomial, et qu'en quotientant l'automate de départ par la relation, on obtient un automate minimal, qui est de plus unique à isomorphisme près. Pour les automates non-déterministes sur les mots finis, le problème de minimisation est PSPACE-complet [6].

Dans ce rapport, nous nous concentrerons principalement sur les automates de Büchi et co-Büchi, déterministes ou GFG (voir plus bas), mais mentionnerons des résultats existants sur d'autres types d'automates, notamment non-déterministes.

En 2010, dans l'article [3], Sven Schewe a prouvé que la minimisation des automates déterministes de Büchi, de co-Büchi et de parité est NP-complète. La preuve utilise une réduction à partir du problème de la couverture par sommets d'un graphe.

Quant aux automates de Büchi et co-Büchi non-déterministes, il est connu depuis 1993 (voir [6]) que leur minimisation est PSPACE-complète, comme la minimisation des automates non-déterministes sur les mots finis.

Au vu de ces résultats, on pourrait penser que le problème de minimisation est résolu pour les automates de Büchi et co-Büchi. Toutefois, les dernières années ont vu se développer plusieurs variantes des automates sur les mots infinis qui obligent à revisiter ce problème.

La première de ces variantes est l'étude des automates dits « GFG », abréviation de « Good For Games », parfois appelés « history-deterministic » (automates déterministes à histoire). En termes intuitifs, un automate GFG est un automate non-déterministe dont le non-déterminisme peut être résolu d'une manière qui ne dépend que du passé. À chaque étape d'exécution, une alternative non-déterministe peut se poser, mais il existe une stratégie qui permet de choisir quelle transition prendre, en fonction de la partie du mot infini qui a déjà été lue. Cette stratégie doit être telle que tout mot accepté par l'automate en tant qu'automate non-déterministe général doit être accepté en suivant la stratégie.

En 2020, dans [7], Sven Schewe a étendu son résultat de 2010, à savoir la NP-complétude de la minimisation des automates déterministes de Büchi, co-Büchi et parité, au cas non-déterministe GFG, avec une preuve essentiellement identique.

La seconde variante est le passage d'une acceptation par les états (*state-based acceptance*) à une acceptation par les transitions (*transition-based acceptance*). Les types d'automates classiques cités (Büchi, co-Büchi, parité, etc.) sont tous définis en termes des états visités infiniment souvent par un chemin. Ce type d'automates a été introduit dans les années 2000 pour simplifier la traduction des formules LTL en automates (voir [8]). Pour tous les types classiques, le pouvoir expressif reste le même en passant de l'acceptation par états à l'acceptation par transitions, mais la taille d'un automate minimal est modifiée.

Dans un article de 2019 [9], Bader Abu Radi et Orna Kupferman ont prouvé que, de manière surprenante, la minimisation des automates GFG co-Büchi à acceptation par transitions, peut être réalisée en temps polynomial. Ce résultat, comparé à celui de Schewe en 2020 sur l'acceptation par états, mérite presque le nom de coup de théâtre. Il montre que les automates à acceptation par transitions n'ont pas qu'un avantage de simplification, mais aussi, dans le cas GFG co-Büchi, un avantage de complexité pour la minimisation, qui devient polynomiale au lieu d'être NP-complète. Cela suggère que l'acceptation par transitions n'est pas qu'un changement de commodité mais peut avoir des avantages pratiques par rapport à l'acceptation par états, qui reste plus commune.

En revanche, pour les automates déterministes de Büchi et co-Büchi à acceptation par transitions, la complexité n'est pas connue actuellement, tout comme le cas des automates Büchi GFG à acceptation par transitions. L'étude du cas des automates déterministes de Büchi et co-Büchi à acceptation par transitions était l'objectif de ce stage.

Nous n'avons pas réussi à déterminer la complexité du problème. Ce rapport présente donc deux pistes de recherche qui n'ont pas abouti. L'une est une variante de la preuve de Schewe sur les automates déterministes de Büchi et co-Büchi à acceptations par états (que l'on aurait pu espérer adapter à l'acceptation par transitions pour montrer un résultat de NP-complétude). Par ailleurs, nous analysons l'algorithme polynomial d'Abu Radi pour les automates GFG co-Büchi à acceptation par transitions, afin de comprendre pourquoi il ne s'étend pas au cas déterministe, et faisons quelques remarques expérimentales.

II Automates sur les mots infinis : définitions et propriétés

Cette partie introduit les définitions des automates considérés, leurs propriétés élémentaires, et quelques résultats moins élémentaires mais importants.

Notation On fixe un ensemble fini non-vide Σ, appelé **alphabet**, dont les éléments sont appelés **lettres**. On note Σ^* l'ensemble des mots finis sur Σ, ainsi que $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$ l'ensemble des mots finis non-vides, et Σ^ω l'ensemble des mots infinis. Le mot vide est noté ε.

Pour $z \in \Sigma^+$ (mot fini *non-vide*), on note z^ω le mot $zzzz... \in \Sigma^\omega$.

On note w_1w_2 la concaténation de w_1 et w_2 : $w_1w_2 \in \Sigma^*$ pour $w_1, w_2 \in \Sigma^*$, et $w_1w_2 \in \Sigma^\omega$ pour $w_1 \in \Sigma^*$ et $w_2 \in \Sigma^\omega$.

On étend ces notations aux langages :

Pour $L \subset \Sigma^+$ (langage de mots finis *ne contenant pas le mot vide*), on pose $L^\omega = \{z_1z_2z_3..., (z_k) \in L^{\mathbb{N}}\}$.

Pour $L \subset \Sigma^*$ et $L' \subset \Sigma^*$, on pose $LL' = \{z_1z_2, (z_1, z_2) \in L \times L'\} \subset \Sigma^*$. De même, pour $L \subset \Sigma^*$ et $L' \subset \Sigma^\omega$, on pose $LL' = \{z_1z_2, (z_1, z_2) \in L \times L'\} \subset \Sigma^\omega$.

Enfin, pour $L \subset \Sigma^*$, on note $L^* = \{z_1z_2...z_n, n \in \mathbb{N}, (z_1, ..., z_n) \in L^n\}$ et $L^+ = \{z_1z_2...z_n, n \in \mathbb{N}^*, (z_1, ..., z_n) \in L^n\}$ (ce qui justifie *a posteriori* les notations Σ^* et Σ^+).

Définition Si $z \in \Sigma^*$ et $L \subset \Sigma^*$, le **langage quotient** de L par z, noté $u^{-1}L$, est défini par $u^{-1} = \{z' \in \Sigma^* \mid zz' \in L\}$.

Si $z \in \Sigma^*$ et $L \subset \Sigma^\omega$, $u^{-1}L$ est défini par $u^{-1}L = \{z' \in \Sigma^\omega \mid zz' \in L\}$.

II 1 Automates

Définition Une **structure d'automate** est un quadruplet $< \Sigma, Q, q_0, \delta >$, où

- Σ est l'alphabet,
- Q est un ensemble fini d'états,
- $q_0 \in Q$ est l'état initial,
- $\delta : Q \times \Sigma \to \mathcal{P}(Q)$ est la table de transitions.

Un automate sur la structure est dit **complet** si $\forall q \in Q, \forall a \in \Sigma, |\delta(q, a)| \geq 1$. **Sauf indication contraire, tous les automates considérés sont complets.**

Un automate sur la structure est **déterministe** si $\forall q \in Q, \forall a \in \Sigma$, $|\delta(q, a)| = 1$.

Comme d'usage, on étend la fonction de transition $\delta : Q \times \Sigma \to \mathcal{P}(Q)$ en $\delta : Q \times \Sigma^* \to \mathcal{P}(Q)$ en posant $\delta(q, \varepsilon) = \{q\}$ et $\delta(q, za) = \cup_{q' \in \delta(q,z)} \delta(q', a)$.

Une **exécution** de l'automate sur le mot $a_1a_2a_3...$ (fini ou infini) est une suite $q_0, q_1, q_2, ...$ (commençant donc par l'état initial q_0) telle que $q_1 \in \delta(q_0, a_1)$, $q_2 \in \delta(q_1, a_1)$, etc.

Remarque On montrera dans le théorème III 3.6 que le fait de considérer des automates complets dans le problème de minimisation ne fait pas perdre en généralité.

Remarque Cette définition des automates non-déterministes est différente de la définition usuelle, car on autorise la plupart du temps plusieurs états initiaux. Dans le cadre de ce rapport, nous pouvons nous contenter de cette définition simplifiée (qui est légèrement plus pratique), car nous travaillerons presque toujours avec des automates soit déterministes, soit non-déterministes GFG (notion définie plus bas), et un automates non-déterministe GFG à plusieurs états initiaux est toujours équivalent à un automate non-déterministe GFG à un seul état initial (pour les conditions d'acceptation usuelles, notamment Büchi et co-Büchi). En revanche, dans le cas général des automates non-déterministes (sans l'hypothèse GFG), la conversion d'un automate à plusieurs états initiaux en un automate à un seul état initial peut nécessiter d'ajouter un état (cet état q_0 devient l'unique état initial, et on rajoute des transitions $q_0 \xrightarrow{a} q'$ pour toute transition $q \xrightarrow{a} q'$ telle que q était au départ un état initial). Le problème de minimisation est donc différent *a priori*.

Définition Les divers types d'automates sont obtenus en adjoignant à une structure d'automate une **condition d'acceptation** qui définit si une exécution est acceptée.

- Dans un **automate de mots finis**, la condition est un ensemble d'états $F \subset Q$, et une exécution finie est acceptée ssi son dernier état appartient à F.
- Dans un **automate de Büchi**, la condition est un ensemble d'états $F \subset Q$, et une exécution infinie est acceptée ssi il existe un état de F visité infiniment souvent. Les états de F seront dits **acceptants**, les autres **non-acceptants**.
- Dans un **automate de co-Büchi**, la condition est $F \subset Q$, et une exécution infinie est acceptée ssi tous les états de F sont visités un nombre fini de fois seulement. Les états de F seront dits **rejetants**, les autres **sûrs**.
- Dans un **automate de parité**, la condition est $p : Q \to \mathbb{N}$ (à chaque état est associé un « indice de parité »), et une exécution accepte si et seulement si l'indice de parité minimum d'un état visité infiniment souvent est pair.
- Dans un **automate de Rabin**, la condition est une famille de couples $(R_i, A_i)_{i \in [\![1,n]\!]} \in (Q \times Q)^n$. Une exécution est acceptée ssi il existe i tel qu'un état de A_i soit visité infiniment souvent, mais tous les états de R_i ne soient visités qu'un nombre fini de fois.
- Dans un **automate de Street**, la condition est une famille de couples $(R_i, A_i)_{i \in [\![1,n]\!]} \in (Q \times Q)^n$. Une exécution est acceptée ssi *pour tout* i, un état de A_i est visité infiniment souvent *ou* tous les états de R_i sont visités finiment souvent.
- Dans un **automate de Muller**, la condition est $\mathcal{F} \subset \mathcal{P}(Q)$. Une exécution est acceptée ssi l'ensemble des états visités infiniment souvent est un élément de $\mathcal{F}$.

Remarque Les conditions de Büchi et co-Büchi sont des cas particuliers de la condition de parité. Pour voir un automate de Büchi comme un automate de parité, on peut associer l'indice de parité 0 aux états acceptants et 1 aux non-acceptants ; pour co-Büchi, prendre 1 pour les états rejetants et 2 pour les états sûrs. La condition de parité est elle-même un cas particulier des conditions de Rabin et de Streett (mettre une paire d'ensembles dans la condition d'acceptation pour deux entiers consécutifs). Enfin, toutes les conditions sont des cas particuliers de la condition de Muller, qui est très générale (mais difficile à manier algorithmiquement).

Remarque Dans les automates de Büchi, co-Büchi et parité, la condition d'acceptation est de taille linéaire en le nombre d'états. En revanche, dans les automates de Rabin, de Streett et de Muller, elle est potentiellement de taille exponentielle en le nombre d'états, si bien qu'il devient plus intéressant d'étudier la minimisation avec une mesure de taille qui prenne en compte la condition d'acceptation.

Définition Un mot est **accepté** par un automate $\mathcal{A}$ ssi il existe une exécution de $\mathcal{A}$ sur ce mot qui est acceptée par $\mathcal{A}$. Le langage de $\mathcal{A}$, c'est-à-dire l'ensemble des mots acceptés par $\mathcal{A}$, est noté $\mathcal{L}(\mathcal{A})$.

Dans les parties qui suivent, nous nous restreignons principalement au cas des automates de Büchi à acceptation par états. Toutes les propositions s'adaptent sans difficulté au cas des automates de co-Büchi.

II 2 Opérations booléennes sur les automates

Les opérations booléennes sont fondamentales car elles sont la base de nombreuses autres constructions.

Proposition II 2.1 Étant donné deux automates de Büchi non-déterministes (resp. déterministes) $\mathcal{A}$ et $\mathcal{B}$, on peut construire en temps polynomial un automate de Büchi non-déterministe (resp. déterministe) $\mathcal{C}$ tel que $\mathcal{L}(\mathcal{C}) = \mathcal{L}(\mathcal{A}) \cup \mathcal{L}(\mathcal{B})$.

Preuve On construit $\mathcal{C}$ comme l'automate produit de $\mathcal{A}$ et $\mathcal{B}$, dont les états sont les couples (q, s) où q est un état de $\mathcal{A}$ et s un état de $\mathcal{B}$, et dont les transitions sont $(q, s) \xrightarrow{a} (q', s')$ pour $q \xrightarrow{a} q'$ dans $\mathcal{A}$ et $s \xrightarrow{a} s'$ dans $\mathcal{B}$. Un état (q, s) est dans la condition d'acceptation de $\mathcal{C}$ ssi q est dans celle de $\mathcal{A}$ ou s est dans celle de $\mathcal{B}$. Il est facile de voir que $\mathcal{L}(\mathcal{A}) \subset \mathcal{L}(\mathcal{C})$ et $\mathcal{L}(\mathcal{B}) \subset \mathcal{L}(\mathcal{C})$ (rappel : les automates sont supposés complets). Réciproquement, si $z \in \mathcal{L}(\mathcal{C})$, en notant $(q_0, s_0), (q_1, s_1), (q_2, s_2), \ldots$ une exécution acceptante sur z, il existe une infinité de i tels que q_i est acceptant dans $\mathcal{A}$ ou s_i est acceptant dans $\mathcal{B}$. Par principe des tiroirs infinis, il existe une infinité de i tels que q_i est acceptant dans $\mathcal{A}$, ou une infinité de i tels que s_i est acceptant dans $\mathcal{B}$, ce qui montre que $z \in \mathcal{L}(\mathcal{A}) \cup \mathcal{L}(\mathcal{B})$. □

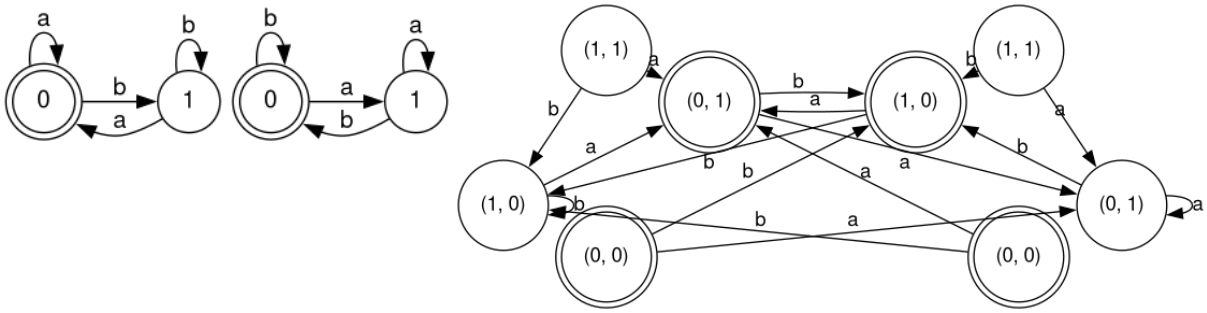
Proposition II 2.2 Étant donné deux automates de Büchi non-déterministes (resp. déterministes) $\mathcal{A}$ et $\mathcal{B}$, on peut construire en temps polynomial un automate de Büchi non-déterministe (resp. déterministe) $\mathcal{C}$ tel que $\mathcal{L}(\mathcal{C}) = \mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{B})$.

Preuve La construction est légèrement plus subtile que l'union et nous la donnerons de manière un peu plus informelle. On fait deux copies de l'automate produit. Les états acceptants de la première copie sont les (q, s) tels que q est acceptant dans $\mathcal{A}$, et les états acceptants de la seconde copie sont les (q, s) tels que s est acceptant dans $\mathcal{B}$. De plus, on « relie » les deux copies en redirigeant dans chaque copie les transitions $(q, s) \xrightarrow{a} (q', s')$ partant d'un état acceptant (q, s) vers l'état correspondant à (q', s') dans l'autre copie.

Si $z \in \mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{B})$, en considérant $q_0, q_1, q_2, \ldots$ et s_0, s_1, s_2 des exécutions acceptantes de $\mathcal{A}$ et $\mathcal{B}$ sur z, la suite $(q_0, s_0), (q_1, s_1), (q_2, s_2), \ldots$ correspond à une exécution acceptante dans $\mathcal{C}$ (qui passe dans la deuxième copie en trouvant un état q_i acceptant dans $\mathcal{A}$, puis dans la première en trouvant s_k acceptant dans $\mathcal{B}$, etc.). Inversement, si une exécution est acceptante, elle atteint une infinité de fois un état acceptant de $\mathcal{A}$ dans la première copie, et une infinité de fois une état acceptant de $\mathcal{B}$ dans la seconde copie, en altenant entre les deux.

Schématiquement, cette construction repose sur le fait que pour une suite infinie de paires de booléens $\left(\text{acc}_i^{\mathcal{A}}, \text{acc}_i^{\mathcal{B}}\right)$, il y a équivalence entre « $\text{acc}_i^{\mathcal{A}}$ est vrai pour une infinité de i et $\text{acc}_i^{\mathcal{B}}$ est vrai pour une infinité de i », et « il existe i tel que $\text{acc}_i^{\mathcal{A}}$ est vrai, et pour tout i tel que $\text{acc}_i^{\mathcal{A}}$, il existe $k > i$ tel que $\text{acc}_k^{\mathcal{B}}$, et pour tout k tel que $\text{acc}_k^{\mathcal{B}}$, il existe $l > k$ tel que $\text{acc}_l^{\mathcal{A}}$ ». □

Illustration de la construction pour l'intersection : À gauche, un automate de Büchi $\mathcal{A}$, au milieu, au milieu automate $\mathcal{B}$, à droite, l'automate $\mathcal{C}$ qui reconnaît $\mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{B})$, obtenu par la construction ci-dessus.



Proposition II 2.3 Étant donné un automate de Büchi non-déterministe $\mathcal{A}$, on peut construire en temps *exponentiel* (borne inférieure atteinte) un automate de Büchi non-déterministe $\mathcal{B}$ tel que $\mathcal{L}(\mathcal{B}) = \mathcal{L}(\mathcal{A})^c$.

Preuve Cette construction est bien plus longue et difficile, et ne sera pas donnée ici. La construction la plus simple n'est pas exponentielle mais doublement exponentielle. Elle est expliquée assez clairement sur la page Wikipédia en anglais « *Complementation of Büchi automaton* ». □

La proposition précédente est en un sens négative : la complémentation d'un automate de Büchi non-déterministe demande un temps exponentiel. Néanmoins, on peut largement améliorer la complexité si l'automate d'entrée est déterministe. Ce lemme est fondamental car il permettra de tester efficacement l'équivalence entre deux automates déterministes.

Proposition II 2.4 Étant donné un automate de Büchi *déterministe* $\mathcal{A}$, on peut construire en temps polynomial un automate de Büchi *non-déterministe* $\mathcal{B}$ tel que $\mathcal{L}(\mathcal{B}) = \mathcal{L}(\mathcal{A})^c$.

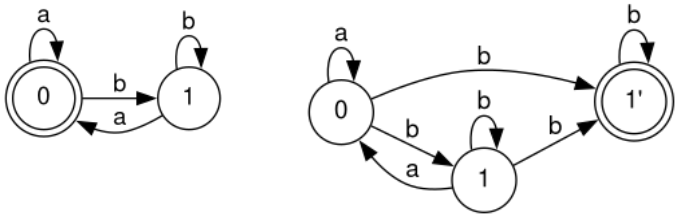
Preuve Pour construire $\mathcal{B}$, on fait deux copies de $\mathcal{A}$, on supprime les états acceptants dans la deuxième copie, et on ajoute, pour chaque transition $q \xrightarrow{a} q'$ dans la première copie avec q' non-acceptant, une transition $q \xrightarrow{a} q''$, où q'' est l'état qui correspond à q' dans la deuxième copie. Pour finir, on modifie l'acceptation de manière à ce les états acceptants soient exactement les états de la deuxième copie.

(L'automate résultant peut être non-complet. Pour le rendre complet, il suffit d'ajouter un état puits si nécessaire.)

Si un mot est accepté par $\mathcal{B}$, son exécution dans $\mathcal{B}$ passe dans la deuxième copie à un certain rang i (car elle doit visiter un état acceptant une infinité de fois, en particulier au moins une fois), et puisque l'exécution est infinie (elle ne s'arrête pas par incomplétude de l'automate), l'exécution correspondante dans $\mathcal{A}$ ne visite plus d'état acceptant de $\mathcal{A}$ à partir de i, donc le mot n'est pas accepté par $\mathcal{A}$. Réciproquement, si un mot n'est pas accepté par $\mathcal{A}$, l'exécution de $\mathcal{A}$ sur ce mot ne visite plus d'état acceptant à partir d'un certain rang i. Il existe alors une exécution acceptante dans $\mathcal{B}$ qui reste dans la première copie jusqu'à i, et passe dans la deuxième copie à i.

On peut résumer cette construction intuitivement en disant que l'automate $\mathcal{B}$ « devine » par non-déterminisme le rang i, s'il existe, à partir duquel l'exécution de $\mathcal{A}$ sur un mot ne visite plus d'état acceptant. □

Illustration de la construction du complémentaire: À gauche, un automate de Büchi déterministe $\mathcal{A}$, à droite, un automate de Büchi non-déterministe $\mathcal{B}$ (non-complet) qui reconnaît $\mathcal{L}(\mathcal{A})^c$, obtenu par la construction ci-dessus.



Proposition II 2.5 Étant donné un automate de mots finis non-déterministe $\mathcal{A}$ et un automate de Büchi non-déterministe $\mathcal{B}$, on peut construire en temps polynomial un automate de Büchi non-déterministe $\mathcal{C}$ tel que $\mathcal{L}(\mathcal{C}) = \mathcal{L}(\mathcal{A})\mathcal{L}(\mathcal{B})$.

Preuve On construit $\mathcal{C}$ en mettant $\mathcal{A}$ et $\mathcal{B}$ côte à côte, et en ajoutant, pour toute transition $q \xrightarrow{a} q'$ dans $\mathcal{A}$ telle que q' est un état acceptant de $\mathcal{A}$, on rajoute une transition $q \xrightarrow{a} s_0$, où s_0 est l'état initial de $\mathcal{B}$. Les états acceptants de $\mathcal{C}$ sont choisis comme étant les états acceptants de $\mathcal{B}$ (donc les états acceptants de $\mathcal{A}$ sont rendus rejetants). L'état initial de $\mathcal{C}$ est celui de $\mathcal{A}$. Ainsi, les chemins acceptants sont exactement les chemins qui commencent comme des chemins dans $\mathcal{A}$, et à un moment où ils auraient visité un état acceptant de $\mathcal{A}$, continuent avec un chemin dans $\mathcal{B}$ qui est acceptant. Intuitivement, lorsqu'il est exécuté sur un mot z de $\mathcal{L}(\mathcal{A})\mathcal{L}(\mathcal{B})$, l'automate $\mathcal{C}$ « devine » par non-déterminisme la longueur d'un mot a tel que z s'écrive $z = ab$ où $a \in \mathcal{L}(\mathcal{A})$ et $b \in \mathcal{L}(\mathcal{B})$. □

Proposition II 2.6 Étant donné un automate de mots finis non-déterministe $\mathcal{A}$ qui ne reconnaît pas le mot vide, on peut construire en temps polynomial un automate de Büchi non-déterministe $\mathcal{B}$ tel que $\mathcal{L}(\mathcal{B}) = \mathcal{L}(\mathcal{A})^\omega$.

Preuve Cette construction est très similaire à la précédente. Soit q_0 l'état initial de $\mathcal{A}$. $\mathcal{B}$ réutilise la structure de $\mathcal{A}$. Son état initial est q_0. Il a, pour toute transition $q \xrightarrow{a} q'$ où q' est acceptant dans $\mathcal{A}$, une transition supplémentaire $q \xrightarrow{a} q_0$ (où q_0 est l'état initial de $\mathcal{A}$). Le seul état acceptant est q_0.

TOOD : illustration □

II 3 Tests sur les automates

On cherche ici à tester algorithmiquement si le langage d'un automate est vide, ou s'il est le même que celui d'un autre automate.

Proposition II 3.1 Étant donné un automate de Büchi non-déterministe $\mathcal{A}$, on peut tester en temps polynomial (et même linéaire) si $\mathcal{L}(\mathcal{A}) = \emptyset$.

Preuve Il suffit de tester l'existence d'un état q acceptant qui soit accessible à partir de l'état initial, et tel qu'il existe une boucle de q à q (qui n'ait pas zéro transition). Une manière efficace de procéder est de calculer les SCC de l'automate (vu comme un graphe orienté en « effaçant » les lettres qui étiquettent les transitions), en temps linéaire, par l'algorithme de Tarjan par exemple. Puis on calcule les SCC accessibles à partir de l'état initial (avec l'algorithme standard pour l'accessibilité, en s'aidant d'un marquage), et on vérifie s'il existe parmi les SCC accessibles une SCC qui soit comporte au moins deux états dont au moins un état acceptant, soit ne comporte qu'un état, acceptant, qui possède une boucle (transition de lui-même vers lui-même). □

Proposition II 3.2 Le problème de déterminer, étant donnés deux automates de Büchi *non-déterministes* $\mathcal{A}$ et $\mathcal{B}$, si $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{B})$, est dans EXPTIME. Restreint aux automates de Büchi déterministes, il est dans P. La même affirmation s'applique au problème de déterminer si $\mathcal{L}(\mathcal{A}) \subset \mathcal{L}(\mathcal{B})$.

Preuve L'idée est de remarquer que

$$\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{B}) \iff (\mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{B})^c = \emptyset) \wedge (\mathcal{L}(\mathcal{A})^c \cap \mathcal{L}(\mathcal{B}) = \emptyset)$$

et

$$\mathcal{L}(\mathcal{A}) \subset \mathcal{L}(\mathcal{B}) \iff \mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{B})^c = \emptyset$$

Le fait que les problèmes soient dans PSPACE est alors une conséquence immédiate des propositions II 2.2, II 2.3 et II 3.1. Pour les automates déterministes, la complémentation (qui produit un automate non-détermniste) peut être réalisée en temps polynomial (proposition II 2.4), ce qui rend les algorithmes polynomiaux. □

Remarque Le problème d'équivalence de deux automates de Büchi non-déterministes est en fait PSPACE-complet [10].

II 4 Automates GFG

La notion d'automate GFG a été développée en lien avec la théorie des jeux. La formalisation précise en termes de théorie des jeux n'était pas centrale pour ce stage et ne sera pas abordée ici. Mentionnons simplement le terme de « stratégie gagnante » dans la définition suivante n'est pas fortuit : en reformulant la définition en termes d'un jeu, la notion de stratégie gagnante définie ici correspond à la notion de stratégie gagnante au sens des jeux.

Intuitivement, un automate non-déterministe est GFG si son non-déterministe peut être « résolu d'une manière qui ne dépend que du passé ».

Définition Soit un automate de Büchi non-déterministe $\mathcal{A}$ sur la structure d'automate $< \Sigma, Q, q_0, \delta >$.

On appelle **stratégie sur $\mathcal{A}$** une fonction $f : \Sigma^* \to Q$ telle que $f(\varepsilon) = q_0$, et pour tout mot $z \in \Sigma^*$, pour toute lettre $a \in \Sigma$, il existe une transition $f(z) \xrightarrow{a} f(za)$ dans $\mathcal{A}$.

Si f est une stratégie et $z = a_0a_1a_2...$ est un mot infini, on définit l'**exécution de $\mathcal{A}$ sur z selon la stratégie f** comme l'exécution $q_0, q_1, q_2, q_3, ...$ où pour tout i, $q_i = f(a_0a_1a_2...a_{i-1})$. La définition d'une stratégie assure que l'exécution de $\mathcal{A}$ sur z selon la stratégie f est bien une exécution valable de $\mathcal{A}$ sur z.

Une stratégie f sur $\mathcal{A}$ est dite **stratégie gagnante** si pour tout $z \in \mathcal{L}(\mathcal{A})$, l'exécution de $\mathcal{A}$ sur z selon la stratégie f est acceptante.

On dit que $\mathcal{A}$ est **GFG** (« Good for Games ») s'il admet une stratégie gagnante.

Remarque Insistons sur le fait qu'un automate GFG n'est pas fourni avec une stratégie gagnante. Seule l'existence d'une stratégie gagnante est garantie. Par exemple, lorsque l'on dit qu'une machine de Turing prend en entrée un automate GFG, elle reçoit simplement la structure d'automate non-déterministe, qui est supposée représenter un automate sur lequel il existe une stratégie gagnante (pour être complètement formel, les problèmes sur les automates GFG sont donc des problèmes à promesse). Nous allons voir que pour certains types d'automates, une stratégie gagnante peut être calculée efficacement, mais par pour d'autres.

Un automate non-déterministe est associé à un jeu sur un graphe, qui admet une stratégie gagnante si et seulement si l'automate est GFG. L'objectif de ce jeu est ω-régulier. Un théorème usuel en théorie des jeux permet alors d'affirmer que si $\mathcal{A}$ admet une stratégie gagnante f, alors f peut être calculée par une machine de Moore, c'est à dire un automate fini déterministe (complet) muni d'une fonction qui à chaque état associe une « lettre de sortie ». L'exécution d'une machine de Moore sur un mot produit comme résultat la suite des lettres associées aux états visités, et f est calculée par la machine de Moore au sens où $f(z)$ est la dernière lettre de la sortie de la machine de Moore sur z. Si $\mathcal{A}$ est GFG, f est une stratégie gagnante sur $\mathcal{A}$, et M est une machine de Moore qui calcule f, on peut construire un automate sur $Q_\mathcal{A} \times Q_M$ (où $Q_\mathcal{A}$ et Q_M sont les ensembles d'états de $\mathcal{A}$ et M), qui comporte une transition $(q_a, q_m) \xrightarrow{a} (q_{a'}, q_{m'})$ s'il existe une transition $q_m \xrightarrow{a} q_{m'}$ dans M et si $q_{a'}$ est la lettre de sortie de l'état q_m. Intuitivement, cette construction produit permet à l'automate de se « rappeler » à chaque étape de l'état courant de la machine M, ce qui permet de suivre la stratégie f. On montre ainsi que tout automate de Büchi GFG peut être transformé en un automate de Büchi déterministe équivalent. De plus, si la stratégie est calculée par une machine de Moore de taille polynomiale, la déterminisation augmente polynomialement le nombre d'états.

Le paragraphe précédent justifie le principe général qu'*un automate GFG est une présentation plus petite d'un automate déterministe*. En particulier, le pouvoir expressif des automates de Büchi GFG est le même que celui des automates de Büchi déterministes (langages limite d'un langage régulier).

Il est prouvé dans [11] que la déterminisation d'un automate de Büchi GFG peut se faire en n'augmentant que quadratiquement le nombre d'états, mais la déterminisation d'un automate de co-Büchi GFG peut nécessiter une augmentation exponentielle. Le même article prouve que l'on peut déterminer en temps polynomial si un automate de Büchi ou co-Büchi non-déterministe est GFG. Notons que pour les automates de co-Büchi, ce résultat, fondamental, est assez contre-intuitif. Il signifie que l'on peut tester en temps polynomial l'existence d'une stratégie, même s'il faudrait un temps exponentiel pour écrire la stratégie si elle existe.

II 5 Acceptation par transitions

On définit les automates à acceptation par transitions de la même manière que les automates à acceptation par états, mais en considérant les transitions visitées infiniment souvent plutôt que les états. Par exemple, si $\mathcal{A}$ est un automate de Büchi à acceptation par transitions, $\mathcal{A}$ accepte un mot infini z ssi il existe une exécution infinie de $\mathcal{A}$ qui passe par une transition acceptante une infinité de fois.

Du moins pour les types classiques, l'acceptation par transitions ne change que linéairement le nombre d'états.

Proposition II 5.1 Si $\mathcal{A}$ est un automate de Büchi non-déterministe à acceptation par états, dont le nombre d'états est n, il existe un automate de Büchi non-déterministe à acceptation par transitions équivalent à $\mathcal{A}$, avec n états.

Réciproquement, si $\mathcal{A}$ est un automate de Büchi non-déterministe à acceptation par transitions dont le nombre d'états est n, il existe un automate de Büchi non-déterministe à acceptation par états équivalent à $\mathcal{A}$, avec au plus $2n$ états.

Preuve Dans la conversion de l'acceptation par états à l'acceptation par transitions, on conserve la structure de $\mathcal{A}$ pour $\mathcal{A}'$, et on marque simplement comme acceptantes les transitions $q \xrightarrow{a} q'$ dont le but q' est un état acceptant. Si une exécution dans $\mathcal{A}'$ est acceptante, elle passe infiniment souvent par une transition acceptante $q \xrightarrow{a} q'$, donc l'exécution correspondante dans $\mathcal{A}$ visite infiniment souvent l'état acceptant q'. Réciproquement, si un état acceptant q' est visité infiniment souvent par une exécution dans $\mathcal{A}$, par principe des tiroirs infinis, l'exécution passe infiniment souvent par l'une des transitions qui ont q pour but.

La conversion inverse se fait en dédoublant chaque état q de $\mathcal{A}$ en un état q_a acceptant et un état q_r non-acceptant. Pour chaque transition $q \xrightarrow{a} q'$ acceptante dans $\mathcal{A}$, $\mathcal{A}'$ possède des transitions $q_a \xrightarrow{a} q'_a$ et $q_r \xrightarrow{a} q'_a$, et pour chaque transition $q \xrightarrow{a} q'$ non-acceptante dans $\mathcal{A}'$, $\mathcal{A}'$ possède des transitions $q_a \xrightarrow{a} q'_r$ et $q_r \xrightarrow{a} q'_r$. Ainsi, étant donné des chemins sur z dans $\mathcal{A}$ et $\mathcal{A}'$, lorsqu'une transition acceptante (resp. non-acceptante) est prise dans $\mathcal{A}$, l'état but de la transition prise dans $\mathcal{A}'$ est acceptant (resp. non-acceptant). L'équivalence des langages de $\mathcal{A}$ et $\mathcal{A}'$ est à nouveau une application du principe des tiroirs infinis. □

Remarque Les résultats des sections précédentes peuvent s'étendre facilement aux automates avec acceptation par transitions. Par exemple, on a le théorème suivant : si $\mathcal{A}$ et $\mathcal{B}$ sont deux automates de Büchi non-déterministes à acceptation par transitions, on peut construire en temps polynomial un automate de Büchi non-déterministe à acceptation par transitions $\mathcal{C}$ tel que $\mathcal{L}(\mathcal{C}) = \mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{B})$. Une manière de le prouver est d'adapter la construction donnée dans le cas de l'acceptation par états, mais on peut aussi tout simplement convertir $\mathcal{A}$ et $\mathcal{B}$ en deux automates à acceptation par états $\mathcal{A}'$ et $\mathcal{B}'$, construire $\mathcal{C}'$ à acceptation par états tel que $\mathcal{L}(\mathcal{C}') = \mathcal{L}(\mathcal{A}') \cap \mathcal{L}(\mathcal{B}')$, puis convertir $\mathcal{C}'$ en un automate à acceptation par transitions $\mathcal{C}$.

III Analyse du problème de minimisation

Après la présentation du contexte scientifique, passons au problème central du stage, à savoir la minimisation. Cette partie donne une première analyse. Le cadre est celui des automates de Büchi ou co-Büchi, complets, déterministes ou GFG, à acceptation par états ou par transitions.

III 1 Appartenance à NP

Dans le cas des automates déterministes, par la proposition II 3.2, on peut vérifier en temps polynomial si les langages de deux automates déterministes (Büchi ou co-Büchi) sont équivalents. Le problème de décision associé à la minimisation peut donc être résolu en « devinant » par non-déterminisme un automate de taille inférieure à l'entier k pris en paramètre, et en vérifiant en temps polynomial s'il est équivalent à l'automate $\mathcal{A}$ pris également en paramètre.

Il est montré dans [7] que l'équivalence de deux automates GFG Büchi ou co-Büchi est dans NP, ce qui montre le résultat dans le cas GFG.

III 2 Lien entre Büchi et co-Büchi

Si $\mathcal{A}$ est un automate de Büchi déterministe (complet), un mot z est accepté si et seulement si l'unique exécution de $\mathcal{A}$ sur z visite un état acceptant infiniment souvent. En réinterprétant $\mathcal{A}$ comme un automate de co-Büchi $\mathcal{A}'$, z est accepté par $\mathcal{A}'$ si et seulement si l'unique exécution visite tous les états rejetants de $\mathcal{A}$ (c'est-à-dire les états acceptants de $\mathcal{A}$) un nombre fini de fois. Ainsi, on a $\mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{A})^c$.

Par conséquent, on peut minimiser un automate de Büchi déterministe $\mathcal{A}$ en minimisant $\mathcal{A}'$ en tant qu'automate de co-Büchi déterministe, et en réinterprétant le résultat comme un automate de Büchi déterministe.

En revanche, les automates de co-Büchi non-déterministes (même GFG) ne sont plus du tout symétriques. En effet, si $\mathcal{A}$ est non-déterministe, $\mathcal{A}$ accepte z ssi il existe une exécution de $\mathcal{A}$ sur z qui visite un état acceptant de $\mathcal{A}$ infiniment souvent, ce dont la négation est « *pour toute* exécution de $\mathcal{A}$ sur z, tout état acceptant de $\mathcal{A}$ est visité un nombre fini de fois », alors que $\mathcal{A}'$ accepte z si et seulement si *il existe* une exécution de $\mathcal{A}'$ sur z qui visite tous les états rejetants de $\mathcal{A}'$ un nombre fini de fois.

III 3 Minimisation des automates non-complets

Dans notre cadre d'étude, les automates sont considérés comme complets. Par exemple, nous nous demandons quelle est la complexité de la minimisation d'un automate de Büchi déterministe complet, en tant qu'automate de Büchi déterministe complet (c'est-à-dire que non seulement l'automate d'entrée est complet, mais on recherche un automate minimal qui soit complet).

En général, un automate non complet peut être rendu complet en ajoutant un état puits (qu'il soit de Büchi ou co-Büchi, déterministe ou non-déterministe). Toutefois, cet ajout modifie le nombre d'états. Ne devrions-nous donc pas considérer, pour chaque type d'automates (comme « Büchi déterministe à acceptation par états »), deux problèmes de minimisation séparés, celui sur les automates de ce type complets et celui sur les automates de ce type non-complets ?

La proposition suivante montre que cela n'est pas nécessaire car les deux problèmes sont équivalents (plus précisément, logspace-interréductibles), du moins pour les quatre classes {Büchi, co-Büchi} × {déterministe, GFG} (nous n'avons pas étudié le cas des automates non-déterministes non-GFG, car le stage se concentrait plutôt sur les automates déterministes, et par extension GFG). Dans la suite, on fixe $\mathcal{T}$ l'une de ces quatre classes.

La preuve est rédigée pour le cas de l'acceptation par états, mais on peut vérifier que toutes les étapes s'adaptent mutatis mutandis au cas de l'acceptation par transitions.

Définition Soit $\mathcal{A}$ un automate complet de la classe $\mathcal{T}$. On note $< \Sigma, Q, q_0, \delta >$ la structure de $\mathcal{A}$.

- On dit qu'un état $q \in Q$ est un **puits** s'il est non-acceptant (pour Büchi) ou rejetant (pour co-Büchi), et si toutes les transitions partant de q sont des boucles (c-à-d. qu'elles vont de q à q lui-même).
- On dit qu'un état q est un **pseudo-puits** si aucun mot n'est accepté à partir de q, c-à-d. $\mathcal{L}(\mathcal{A}^q) = \emptyset$.

Lemme III 3.1 Soit $\mathcal{A}$ un automate de la classe $\mathcal{T}$. Si q est un pseudo-puits de $\mathcal{A}$, alors q' est un pseudo-puits pour toute transition $q \xrightarrow{\#} q'$ de $\mathcal{A}$.

Preuve Si q' n'était pas un pseudo-puits, il existerait $z \in \mathcal{L}\left(\mathcal{A}^{q'}\right)$. Alors, on aurait $az \in \mathcal{L}(\mathcal{A}^q)$, contredisant le fait que q est un pseudo-puits. □

Lemme III 3.2 Soit $\mathcal{A}$ un automate complet de la classe T. Si $\mathcal{A}$ est minimal en tant qu'automate complet de la classe T, alors tout pseudo-puits de $\mathcal{A}$ est un puits.

Preuve Supposons qu'un état q soit un pseudo-puits mais pas un puits. Soit H l'ensemble des états accessibles depuis q. Comme q est un pseudo-puits, H n'est pas réduit à $\{q\}$. Par le lemme précédent appliqué itérativement, tout état de H est un pseudo-puits. Ainsi, aucune exécution qui passe dans un état de H ne peut être acceptante. Donc, en fusionnant tous les états de H en un seul puits, on ne modifie pas le langage de l'automate (ni son caractère complet), ce qui contredit sa minimalité car $|H| > 1$.□

Définition Soit $\mathcal{A}$ un automate de $\mathcal{T}$, de structure $< \Sigma, Q, q_0, \delta >$. Soit un état $q \in Q$ et soit un mot $z \in \Sigma^*$.

Si $\mathcal{T}$ est une classe d'automates déterministes, on dit que z est un **témoin d'utilisation** de q de q si $\delta(q_0, z) = \{q\}$ (en d'autres termes, si l'unique exécution de $\mathcal{A}$ sur z se termine par l'état q).

Si $\mathcal{T}$ est une classe d'automates GFG, on dit que z est un témoin d'utilisation de q si $f(z) = q$ (en d'autres termes, si l'unique exécution de $\mathcal{A}$ sur z en suivant la stratége f se termine par l'état q).

Dans les deux cas, on dit que q est **utilisé** s'il existe un témoin d'utilisation de q.

Lemme III 3.3 Soit $\mathcal{A}$ un automate de $\mathcal{T}$. Si $\mathcal{T}$ est une classe d'automates GFG, soit f une stratégie gagnante sur $\mathcal{A}$. Si un mot z est un témoin d'utilisation d'un état q, alors tout mot infini z' est accepté à partir de q si et seulement si zz' est accepté. En d'autres termes, on a $\mathcal{L}(A^q) = {z'}^{-1}\mathcal{L}(\mathcal{A})$.

Preuve Le cas déterministe est évident. Dans le cas GFG, si $z' \in \mathcal{L}(\mathcal{A}^q)$, en prolongeant l'exécution de $\mathcal{A}$ sur z en suivant f (qui se termine par q, car z est un témoin d'utilisation de q) par une exécution acceptante de $\mathcal{A}$ sur z' à partir de q, on obtient une exécution acceptante de $\mathcal{A}$ sur zz'. Réciproquement, si zz' est accepté par $\mathcal{A}$, l'exécution de $\mathcal{A}$ sur zz' en suivant f est acceptante, et en retirant le préfixe qui correspond à z, on une exécution acceptante de $\mathcal{A}$ sur z' à partir de q. □

Lemme III 3.4 Si $\mathcal{A}$ est un automate de $\mathcal{T}$ minimal en tant qu'automate de $\mathcal{T}$, ou un automate complet de $\mathcal{T}$ minimal en tant qu'automate complet de $\mathcal{T}$, alors tous les états de $\mathcal{A}$ sont utilisés.

Preuve On raisonne par l'absurde en supposant qu'il existe des états non-utilisés.

- Cas déterministe :

  En supprimant tous les états non utilisés, on obtient un automate qui est évidemment équivalent à $\mathcal{A}$, et qui est encore complet si $\mathcal{A}$ est complet (il ne peut pas exister de transition $q \xrightarrow{a} q'$ d'un état utilisé q à un état non utilisé q', car en prenant un témoin d'utilisation z de q, za serait un témoin d'utilisation de q').

- Cas GFG :

  Soit f une stratégie gagnante sur $\mathcal{A}$. Soit $\mathcal{A}'$ l'automate obtenu à partir de $\mathcal{A}$ en supprimant les états non-utilisés. Il est évident que $\mathcal{L}(\mathcal{A}') \subset \mathcal{L}(A)$ (toute exécution sur $\mathcal{A}'$ est aussi une exécution sur $\mathcal{A}$). Réciproquement, si $x \in \mathcal{L}(\mathcal{A})$, l'exécution de $\mathcal{A}$ en suivant la stratégie f est acceptante, or cette exécution est encore une exécution dans $\mathcal{A}'$ car tous les états qu'elle traverse sont utilisés (les préfixes de x fournissent les témoins d'utilisation). Ceci montre que $\mathcal{L}(\mathcal{A}) \subset \mathcal{L}(\mathcal{A}')$. Finalement, $\mathcal{A}$ et $\mathcal{A}'$ sont équivalents. Si $\mathcal{A}$ est complet, alors $\mathcal{A}'$ est encore complet, car pour tout état utilisé q et pour toute lettre a, en prenant un témoin d'utilisation z de q, il existe une transition $q = f(z) \xrightarrow{a} f(az)$ (par définition d'une stratégie), et $f(az)$ est un état utilisé.

Dans les deux cas, on obtient une contradiction avec la minimalité de $\mathcal{A}$. □

Lemme III 3.5 Soit $\mathcal{A}$ un automate complet de la classe T qui est minimal en tant qu'automate complet dans la classe T. Alors, $\mathcal{A}$ contient un puits si et seulement si il existe $z \in \Sigma^*$ tel que pour tout $z' \in \Sigma^\omega$, $zz' \notin \mathcal{L}(\mathcal{A})$. Si c'est le cas, le puits est unique.

Preuve Commençons par l'implication directe. Supposons que $\mathcal{A}$ contient un puits q. Rappelons que tous les états de $\mathcal{A}$ sont utilisés par le lemme III 3.4. En particulier, q est utilisé. Soit z un témoin d'utilisation de q. Par le lemme III 3.4, pour tout $z' \in \Sigma^\omega$, on a $zz' \in \mathcal{L}(\mathcal{A})$ si et seulement si z' est accepté à partir de q, ce qui ne peut jamais être le cas car q est un puits.

Passons à présent à l'implication réciproque. Fixons $z \in \Sigma^*$ tel que pour tout $z' \in \Sigma^\omega$, $zz' \notin \mathcal{L}(\mathcal{A})$. Soit q un état quelconque de $\delta(q_0, z)$. Fixons $z' \in \Sigma^\omega$. Si z' était accepté à partir de q, alors zz' serait accepté à partir de q_0, donc on aurait $zz' \in \mathcal{L}(\mathcal{A})$, ce qui n'est pas le cas par hypothèse. Ainsi, z' n'est pas accepté à partir de q. Ceci étant vrai pour tout z', l'état q est un pseudo-puits. Par la proposition III 3.2, c'est un puits. Il existe donc un puits dans $\mathcal{A}$.

Enfin, l'unicité d'un puits est facile : s'il existait deux puits, ils pourraient être fusionnés en un seul puits sans changer le langage (et l'automate résultant serait encore complet), ce qui contredirait la minimalité. □

Théorème III 3.6 Soit $\mathcal{A}$ un automate complet de la classe $\mathcal{T}$. Notons C l'ensemble des automates complets de $\mathcal{T}$ équivalents à $\mathcal{A}$ qui sont minimaux en tant qu'automates complets de $\mathcal{T}$. Notons également N l'ensemble des automates (non nécessairement complets) de $\mathcal{T}$ équivalents à $\mathcal{A}$ qui sont minimaux en tant qu'automates de $\mathcal{T}$. Notons c la taille commune à tous les automates de C, et n la taille commune aux automates de N. Alors, il se produit l'un des deux cas suivants :

- Tous les automates de C contiennent un unique puits, et les automates de N sont exactement les automates obtenus en prenant un automate de C et en retirant son puits (en particulier, $n = c - 1$).
- Aucun automate de C ne contient de puits, et on a $N = C$ (en particulier, $n = c$).

Preuve On a $n \leq c$ car les automates de N sont minimaux en tant qu'automates de $\mathcal{T}$ équivalents à $\mathcal{A}$, et les automates de C sont en particulier des automates de $\mathcal{T}$ équivalents à $\mathcal{A}$. Inversement, en ajoutant éventuellement un puits à un automate de N, on obtient un automate complet de $\mathcal{T}$ équivalent à $\mathcal{A}$, par définition de C, on a $c \leq n + 1$. Ainsi, on a soit $n = c$, soit $n = c - 1$.

Une conséquence du lemme III 3.5 est que tous les états de C contiennent un unique puits ou tous n'en contiennent aucun, car pour un automate de C, contenir un puits est équivalent à une propriété qui porte uniquement sur $\mathcal{L}(A)$, indépendamment de la structure interne de l'automate.

- Supposons que tous les automates de C contiennent un unique puits. En prenant un automate $\mathcal{B}$ de C (donc de taille c) et en supprimant son puits, on obtient un automate $\mathcal{B}'$ de $\mathcal{T}$ de taille $c - 1$ équivalent à $\mathcal{A}$, d'où $n \leq c - 1$, donc nous sommes dans le cas $n = c - 1$, et puisque $\mathcal{B}'$ est de taille n, on a $\mathcal{B}' \in N$. Réciproquement, soit $\mathcal{B}'$ dans N. En rajoutant un puits pour compléter $\mathcal{B}'$, on obtient un automate $\mathcal{B}$ de taille $n + 1 = c$, et puisque $\mathcal{B}$ est de taille $\mathcal{C}$, complet, dans $\mathcal{T}$ et équivalent à $\mathcal{A}$, on a $\mathcal{B} \in C$, donc $\mathcal{B}'$ s'écrit bien comme un automate de C (à savoir $\mathcal{B}$) où le puits a été supprimé.
- Supposons qu'aucun automate de C ne contienne de puits. Cette fois, nous sommes dans le cas $n = c$. En effet, supposons au contraire que $n = c - 1$. En prenant un automate de N et en lui rajoutant un puits, on obtient un automate de C qui contient un puits, ce qui est absurde.

  On a $C \subset N$ car les automates de C ont la même taille (qui est c) que la taille optimale d'un automate de $\mathcal{T}$ équivalent à $\mathcal{A}$ (à savoir n).

  Réciproquement, montrons $N \subset C$. Il faut prouver que tout automate de N est complet. Fixons donc $\mathcal{B} \in N$ et montrons que $\mathcal{B}$ est complet. Raisonnons par l'absurde en supposant que $\mathcal{B}$ n'est pas complet. Soient un état q et une lettre a tels qu'il n'existe aucune transition sortant de q étiquetée par a.

□

Corollaire III 3.7 Le problème de minimisation des automates de $\mathcal{T}$ et le problème de minimisation des automates complets de $\mathcal{T}$ (en tant qu'automates complets) sont logspace-interréductibles. (Il s'agit ici des problèmes de calcul, pas des problèmes de décision.)

Preuve Si le problème sur les automates complets est résolu, il suffit de vérifier si la solution obtenue possède un état puits, et si oui de le retirer, pour obtenir un solution du problème sur les automates non-complets. Réciproquement, si on dispose d'une solution sur les automates non-complets, il suffit d'ajouter un puits si la solution n'est pas un automate complet, pour obtenir une solution du problème sur les automates complets. □

Remarque Les problèmes de décision ne sont pas logspace-interréductibles *a priori*.

III 4 Congruence de Myhill-Nérode

La congruence de Myhill-Nérode est à la base de la minimisation des automates déterministes sur les mots finis. Bien que la minimisation des automates sur les mots infinis soit moins simple, cet outil reste très utile.

Définition Si L est un langage de mots finis ($L \subset \Sigma^*$) ou infinis ($L \subset \Sigma^\omega$), on appelle **congruence de Myhill-Nérode** de L, et on note $\sim_L$ (ou simplement $\sim$ si L est implicite) la relation définie sur Σ^* par

$$u \sim_L v \iff u^{-1}L = v^{-1}L$$

Si $\mathcal{A}$ est un automate, on définit $\sim_{\mathcal{A}}$ comme une relation sur les états de $\mathcal{A}$ par

$$q \sim_{\mathcal{A}} s \iff \mathcal{L}(\mathcal{A}^q) = \mathcal{L}(\mathcal{A}^s)$$

Proposition III 4.1 Si $u \in \Sigma^*$, $L \subset \Sigma^*$ ou $L \subset \Sigma^\omega$, et $a \in \Sigma$, on a

$$(ua)^{-1}L = a^{-1}(u^{-1}L)$$

Preuve Pour tout z, $z \in (ua)^{-1}L \iff uaz \in L \iff az \in u^{-1}L \iff z \in a^{-1}(u^{-1}L)$. □

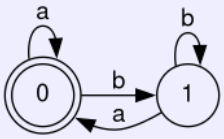
Proposition III 4.2 La congruence de Myhill-Nérode d'un langage L de mots finis ou infinis est une congruence : pour tous $u, v \in \Sigma^*$ tels que $u \sim_L v$, pour toute lettre $a \in \Sigma$, on a $ua \sim_L va$.

Preuve On a $(ua)^{-1}L = a^{-1}(u^{-1}L) = a^{-1}(v^{-1}L) = (va)^{-1}L$. □

Il est connu (et nous ne le prouverons pas ici, mais la preuve serait très semblable à celle de la proposition III 5.2) que si L est un langage régulier, $\sim_L$ est d'indice fini, et on peut construire un automate déterministe sur les mots finis qui reconnaît L en prenant pour états les classes d'équivalence, et en mettant les transitions $[z] \xrightarrow{a} [za]$ (en notant $[z]$ la classe de z), ce qui est valable car la proposition précédente permet, pour toute lettre a, de quotienter l'application $z \mapsto za$ par la relation $\sim_L$. Cet automate est de plus l'unique automate minimal (à isomorphisme près), en tant qu'automate déterministe complet sur les mots finis qui reconnaît L.

Dans le cas des automates de Büchi, la situation n'est pas aussi simple. Il n'est pas vrai qu'il existe toujours un automate minimal dont le nombre d'états est l'indice de la congruence de Myhill-Nérode.

Exemple III 4.3 Considérons l'automate de Büchi suivant :



Le langage reconnu est l'ensemble des mots qui contiennent une infinité de a. Tous les mots finis sont équivalents par la congruence de Myhill-Nérode, mais il n'existe pas d'automate de Büchi équivalent à un seul état.

En revanche, il reste vrai qu'il y a toujours au moins autant d'états que de classes dans la congruence de Myhill-Nérode.

On fixe une classe $\mathcal{T}$ d'automates parmi les classes {déterministe, GFG} × {Büchi, co-Büchi}. Encore une fois, les résultats de cette partie peuvent s'étendre sans difficulté particulière aux automates à acceptation par transitions.

Définition Soit $\mathcal{A}$ un automate de $\mathcal{T}$. On dit qu'un état q est un **représentant** d'un mot z si z est un témoin d'utilisation de q. Le mot z est dit **représenté** s'il admet un représentant.

Remarque Dans un automate complet de $\mathcal{T}$, tout mot fini admet un représentant (par définition d'un témoin d'utilisation).

Lemme III 4.4 Soit $\mathcal{A}$ un automate de $\mathcal{T}$. Soient $u, v \in \Sigma^*$ non équivalents par la congruence de Myhill-Nérode associée à $\mathcal{L}(\mathcal{A})$. Aucun représentant de u n'est un représentant de v et inversement.

Preuve Quitte à échanger u et v, supposons qu'il existe $z \in \Sigma^\omega$ tel que $uz \in \mathcal{L}(\mathcal{A})$ et $vz \notin \mathcal{L}(\mathcal{A})$. Si q est un représentant de u et de v, alors on a $z \in \mathcal{L}(\mathcal{A}^q)$ par le lemme III 3.3 appliqué à q, z et u, mais aussi $z \notin \mathcal{L}(\mathcal{A}^q)$ par le même lemme appliqué à q, z et v, d'où une contradiction. □

Théorème III 4.5 Soit $\mathcal{A}$ complet un automate de $\mathcal{T}$. Le nombre d'états de $\mathcal{A}$ est au moins égal à l'indice de la congruence de Myhill-Nérode associée à $\mathcal{L}(\mathcal{A})$.

Preuve Par la remarque et le lemme précédents, on peut associer à chaque classe d'équivalence un représentant d'un mot quelconque de cette classe, et ces états sont tous différents. □

III 5 Minimisation des automates sur les mots finis à acceptation par transitions

Puisque nous cherchons à étudier la minimisation des automates sur les mots finis à acceptation par transitions, et que la minimisation des automates sur les mots finis est beaucoup plus simple en règle générale que la minimisation des automates sur les mots infinis, il est naturel de se demander si la minimisation des automates sur les mots finis à acceptation par transitions est simple à réaliser. Cette partie montre qu'elle est très similaire à la minimisation des automates sur les mots finis à acceptation par états.

Les automates sur les mots finis à acceptation par transitions sont définis comme les automates sur les mots finis à acceptation par les états, sauf la condition d'acceptation, qui change de « l'état final est acceptant » à « la dernière transition visitée est acceptante ». Notons qu'un automate sur les mots finis à acceptation par transitions ne reconnaît pas une partie de Σ^* mais une partie de Σ^+ (car l'exécution vide n'a pas de dernière transition).

Définition Soit $L \subset \Sigma^+$ un langage de mots finis non-vides. On appelle **pseudo-congruence de Myhill-Nérode** de L la relation $\sim'_L$ définie sur Σ^* par $u \sim'_L v \iff (\forall x \in \Sigma^+, ux \in L \iff vx \in L)$.

La différence entre la congruence de Myhill-Nérode et la pseudo-congruence de Myhill-Nérode est le passage de Σ^* à Σ^+. En particulier, on a $u \sim_L v \Rightarrow (u \in L \iff v \in L)$ mais pas $u \sim'_L v \Rightarrow (u \in L \iff v \in L)$. Intuitivement, cela permet de modéliser le fait que dans un automate à acceptation par transitions $\mathcal{A}$, mettons déterministe, si deux mots u et v sont tels que les exécutions de $\mathcal{A}$ sur u et v aboutissent toutes les deux au même état q, ces deux exécutions ne se terminent pas forcément par la même transition, donc l'une peut être acceptante mais l'autre non ; en revanche, si on prolonge les deux exécutions avec un mot x, les exécutions prolongées se termineront par la même transition.

Remarque Pour tout langage de mots non-vides $L \subset \Sigma^+$, pour tous $u, v \in \Sigma^*$, on a $u \sim_L v \iff (u \sim'_L v \wedge (u \in L \iff v \in L))$. Ainsi, les classes de $\sim_L$ sont exactement les $L \cap A$ (si non-vide) et $L^c \cap A$ (si non-vide) pour A une classe de $\sim'_L$.

Par conséquent, si $\sim'_L$ est d'indice fini n, alors $\sim_L$ est d'indice fini compris entre n et $2n$, et si $\sim'_L$ est d'indice infini, $\sim_L$ est aussi d'indice infini. En particulier, $\sim'_L$ est d'indice fini si et seulement si $\sim_L$ est d'indice fini, donc si et seulement si L est régulier.

Proposition III 5.1 Pour tout langage $L \subset \Sigma^+$, la relation $\sim'_L$ est une congruence : $\forall u, v \in \Sigma^*, u \sim'_L v \Rightarrow \forall z \in \Sigma^*, uz \sim'_L vz$. On a même une propriété légèrement plus forte : $\forall u, v \in \Sigma^+, u \sim'_L v \Rightarrow \forall z \in \Sigma^+, uz \sim_L vz$ (remarquer le passage de $\sim'_L$ à $\sim_L$).

Preuve Similaire à la preuve de la proposition III 4.2. □

Proposition III 5.2 Soit L un langage supposé régulier (donc $\sim'_L$ est d'indice fini). On définit un automate de Büchi déterministe à acceptation par transitions en prenant pour états les classes de $\sim'_L$. Les transitions sont définies par le quotient de l'application $z \mapsto za$ pour toute lettre a, qui est bien défini par la proposition III 5.1. (En termes plus concrets, les transitions sont les $[z] \xrightarrow{a} [za]$, en notant $[x]$ la classe de x pour $\sim'_L$). On marque comme acceptantes les transitions $[z] \mapsto [za]$ où $za \in L$, ce qui ne dépend pas du choix d'un mot z dans une classe d'équivalence (c'est aussi un quotient, et il est encore bien défini par la deuxième partie de la proposition III 5.1). L'état initial est $[\varepsilon]$. Alors, l'automate construit (qui est complet) reconnaît L, et il est minimal, et unique (à isomorphisme près), en tant qu'automate de Büchi déterministe à acceptation par transitions et complet qui reconnaît L.

Preuve Notons $\mathcal{A}$ l'automate défini dans la proposition. Montrons d'abord que $\mathcal{A}$ reconnaît L. Soit un mot non-vide $x = a_1 a_2 ... a_n$. Par une récurrence directe, le chemin de $\mathcal{A}$ sur x se termine par une transition $[a_1 a_2 ... a_{n-1}] \xrightarrow{a_n} [a_1 a_2 ... a_n]$. Par définition de $\mathcal{A}$, cette transition est acceptante si et seulement si $a_1 a_2 ... a_n \in L$.

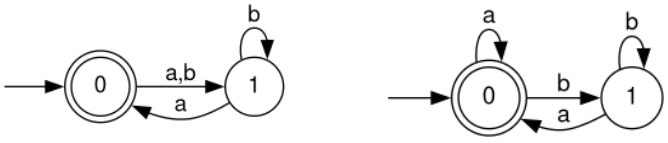
Montrons que $\mathcal{A}$ est minimal. Soit $\mathcal{A}'$ un automate (de Büchi déterministe complet à acceptation par transitions) qui reconnaît aussi L. Soient $z_1, ..., z_k$ des représentants de chaque classe d'équivalence de $\sim'_L$. Pour $i \neq j$, l'unique représentant de i est différent de celui de j, car sinon, on aurait $z_i \sim'_L z_j$, puisqu'en prolongeant des chemins qui débouchent sur le même état par une même suite non-vide de transitions, on obtient des chemins qui ont la même transition finale. Ainsi, $\mathcal{A}'$ a au moins autant d'états que $\mathcal{A}$, ce qui prouve la minimalité.

Enfin, montrons que $\mathcal{A}$ est unique à isomorphisme près. Soit $\mathcal{A}'$ est un automate de même taille que $\mathcal{A}$ qui reconnaît L. On va construire un isomorphisme entre $\mathcal{A}$ et $\mathcal{A}'$. Soit un état q de $\mathcal{A}'$. Rappelons que q est utilisé par minimalité (proposition III 3.4). Tous les témoins d'utilisation de q sont équivalents par $\sim'_L$, par le même raisonnement qu'au paragraphe précédent. Notons alors $\varphi(q)$ la classe commune aux témoins d'utilisation de q. On obtient ainsi une application φ de l'ensemble des états de $\mathcal{A}'$ dans l'ensemble des classes d'équivalence pour $\sim'_L$. Cette application est surjective car pour tout $z \in \Sigma^*$, en notant q l'unique représentant de z, on a $\varphi(q) = [z]$. De plus, il y a autant d'états de $\mathcal{A}'$ que de classes d'équivalence de $\sim'_L$, donc φ est bijective. Pour toute transition $q \xrightarrow{a} q'$, si z est un témoin d'utilisation de q, alors za est un témoin d'utilisation de q', donc il existe une transition $\varphi(q) \xrightarrow{a} \varphi(q')$ dans $\mathcal{A}$, par définition de $\mathcal{A}$. Ainsi, à toute transition de $\mathcal{A}'$ correspond une transition de $\mathcal{A}$. Mais il ne peut pas y avoir plus de transitions dans $\mathcal{A}'$ qu'il n'y en a dans $\mathcal{A}$, car tous les deux sont des automates complets. Finalement, φ est un isomorphisme. L'automate $\mathcal{A}$ est bien unique. □

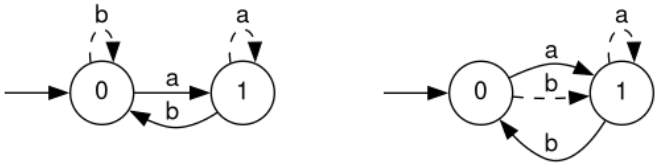
III 6 Non-unicité

Dans le cas des automates finis (à acceptation par états comme à acceptation par transitions), il y a unicité (à isomorphisme près, bien sûr) de l'automate minimal.

Le contre-exemple suivant montre que cela n'est plus vrai pour les automates de Büchi déterministes à acceptation par transitions. Les deux automates du dessin sont équivalents (ils reconnaissent le langage des mots qui contiennent une infinité de a), mais ils ne sont pas isomorphes. Ils ne diffèrent pas uniquement par leurs états acceptants ou leurs états initiaux respectifs, mais bien par leurs structures d'automate.



Voici également un contre-exemple pour les automates de Büchi à acceptation par transitions (les transitions en pointillés sont acceptantes) :



III 7 Saturation

On peut décomposer le problème de minimisation en plusieurs étapes. Prenons l'exemple des automates de Büchi déterministes à acceptation par états. Supposons que nous ayons à notre disposition une structure d'automate telle qu'il existe un automate sur

cette structure reconnaissant un langage L. Peut-on alors déterminer efficacement un ensemble d'états acceptants tel que l'automate reconnaisse L ? C'est l'objet des lemmes suivants.

Lemme III 7.1 Soit une structure d'automate $< \Sigma, Q, q_0, \delta >$. Pour tout ensemble d'états $F \subset Q$, notons $U(F)$ le langage de l'automate sur la structure où les états acceptants sont les états de F. Alors, pour tous $F_1, F_2 \subset Q$, on a $U(F_1 \cup F_2) = U(F_1) \cup U(F_2)$.

Preuve Les inclusions $U(F_1) \subset U(F_1 \cup F_2)$ et $U(F_2) \subset U(F_1 \cup F_2)$ sont triviales, et si un chemin visite une infinité de fois un état de $F_1 \cup F_2$, cet état peut être dans F_1 ou dans F_2, et dans un cas le chemin est accepté avec F_1 comme condition d'acceptation, dans l'autre avec F_2. □

Proposition III 7.2 Soit un langage $L \subset \Sigma^\omega$. Soit une structure d'automate $< \Sigma, Q, q_0, \delta >$ telle qu'il existe $F \subset Q$ tel que $U(F) = L$. Alors, l'ensemble $\{F \subset Q \mid U(F) = L\}$ admet un maximum pour l'inclusion, qui peut être calculé en temps polynomial.

Preuve On prouve sans difficulté que le maximum recherché est $\{q \in Q \mid U(\{q\}) \subset L\}$. On peut le calculer en temps polynomial par la proposition II 3.2. □

Pour l'acceptation par transitions, le même raisonnement s'applique. L'article [9] donne une description plus explicite dans ce cas, mais elle ne sera pas utile ici.

IV Une tentative (avortée) de preuve de NP-complétude

Nous nous intéressons ici au problème de la minimisation d'un automate de Büchi déterministe à acceptation par transitions. L'idée de départ était de montrer que ce problème est NP-complet en adaptant la preuve de Schewe dans le cas de l'acceptation par états. Cette preuve est présentée dans l'annexe Section VIII.

La preuve de Schewe est une réduction du problème de la couverture par sommet à la minimisation d'un automate qui reconnaît un langage défini à partir d'un graphe. La taille d'un automate minimal pour ce langage est liée à la taille d'une couverture par sommets minimale. Malheureusement, ce lien est perdu en passant à l'acceptation par transitions.

Nous espérions, en changeant le langage, retrouver une information intéressante sur le graphe dans un automate minimal (taille d'une couverture par sommets ? chemin hamiltonien ? nombre de couleurs d'une coloration minimale ?). Cette section est la tentative la plus aboutie. Hélas, non seulement nous n'avons pas complètement résolu la question de la taille de l'automate minimal, mais tout tend à montrer que cette taille ne donne pas d'informations intéressantes.

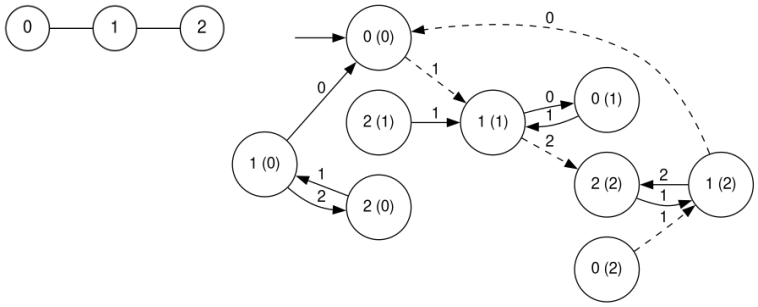
Dans cette partie, les automates seront non-complets.

Définition Soit $G = (V, E)$ un graphe non orienté muni d'un sommet distingué $v_0 \in V$. On note $L(G)$ le langage sur l'alphabet V des chemins infinis partant de v_0 dans G (vus comme des mots infinis sur V) qui visitent tous les sommets une infinité de fois.

Si G n'est pas connexe, $L(G)$ est trivialement vide (certains sommets ne sont pas accessibles depuis v_0, donc *a fortiori* il n'est pas possible de les visiter une infinité de fois). À partir de maintenant, nous fixons donc un graphe G connexe.

Commençons par donner une construction simple pour reconnaître $L(G)$ à l'aide d'un automate. On peut reconnaître les chemins infinis dans G à l'aide d'un automate qui a la même structure que G (chaque arête se transforme en deux transitions). Pour ne reconnaître que les chemins infinis qui visitent chaque sommet une infinité de fois, une possibilité est de faire $|V|$ copies de cet automate, chacune correspondant à un sommet v_i parmi les sommets de $V = \{v_1, ..., v_n\}$, et de passer de la copie i à la copie $i+1$ (ou à la copie 1 depuis la copie n) lorsque le sommet v_{i+1} est visité, ceci par une transition acceptante. Intuitivement, l'automate commence par « attendre » de voir v_1 dans le mot lu, tout en vérifiant à chaque étape que les transitions correspondent bien à des arêtes du graphe G, puis, une fois que v_1 est lu, visite une transition acceptante et « attend » v_2, etc.

Illustration : À gauche, un graphe, à droite, l'automate construit.



Sur l'exemple ci-dessus, on constate que cette construction n'est pas minimale. On remarque notamment des états inaccessibles. Par exemple, l'état « 2 (0) » (état 2 de la copie 0, où l'automate attend le sommet 1) est inaccessible car l'automate, dans cette copie, attend le sommet 1 pour changer de copie, et le sommet 2 ne peut pas être atteint depuis le sommet 0 sans être d'abord passé par le sommet 1. On note aussi que la copie 0, où l'automate attend le sommet 1, est en fait inutile, car lorsque l'automate a lu 2, il passe forcément par un 1 avant de voir un 0, donc il peut simplement attendre 0 au lieu d'attendre 1 puis 0.

Ces deux constatations sont liées au fait qu'un chemin entre 0 et 2 passe nécessairement par 1. Cette notion est formalisée en théorie des graphes par la notion de point d'articulation.

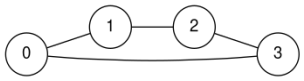
Définition Soit G un graphe connexe. On dit qu'un sommet v est un **point d'articulation** si en retirant le sommet v de G, le graphe obtenu n'est plus connexe. Un sommet qui n'est pas un point d'articulation est dit **sommet intérieur**.

Lemme IV.1 Soit G un graphe connexe. Si une boucle dans G visite tous les sommets intérieurs de G, alors elle visite tous les sommets de G.

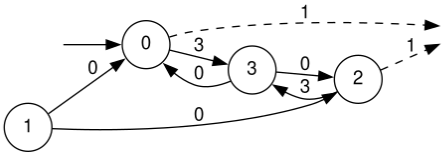
Preuve Récurrence forte facile sur le nombre de sommets de G. □

Par ce lemme, l'automate n'a besoin d'« attendre » que les sommets intérieurs de G.

La construction avec cette heuristique n'est toujours pas minimale. Considérons en effet le graphe suivant :



La première copie du graphe, qui attend le sommet 1, se présente ainsi :



Premièrement, il est évident que nous pouvons enlever l'état correspondant au sommet 1. En effet, cette copie attend le sommet 1 et passe dans la copie suivante dès qu'il est lu.

Mais on constate aussi que les états correspondant aux sommets 0 et 2 peuvent être fusionnés. Cela est lié au fait qu'ils ont exactement les mêmes voisins dans le graphe d'origine G.

Ainsi, nous obtenons une deuxième heuristique : dans chaque copie, on peut fusionner les états correspondant à des sommets qui ont les mêmes voisins dans G.

Nous conjecturons que la construction est alors minimale. La suite est un début de preuve non abouti.

Soit un automate $\mathcal{A}$ minimal qui reconnaît aussi L. On veut montrer $|\mathcal{A}| = |\mathcal{A}_0|$, où $\mathcal{A}_0$ est l'automate obtenu par la construction. Sans perte de généralité, supposons A saturé en transitions acceptantes.

Les automates $\mathcal{A}$ et $\mathcal{A}_0$ sont (a priori) non-complets. Dans $\mathcal{A}_0$, chaque copie de G a une structure très forte : s'il y a une transition entre deux états q et s, qui sont des copies de deux sommets v et w (éventuellement dans deux copies de G différentes), il y a toujours une arête entre v et w dans le graphe. Nous poursuivons le travail d'identification d'une structure similaire à $\mathcal{A}_0$ dans $\mathcal{A}$ par le lemme suivant, qui montre que le point précédent reste vrai dans $\mathcal{A}$.

Définition Dans un automate non-complet $\mathcal{A}$, un mot z est dit **faiblement accepté** s'il existe une exécution de $\mathcal{A}$ sur z (acceptante ou non). Autrement dit, $\mathcal{A}$ accepte z faiblement si (pour un certain choix non-déterministe) $\mathcal{A}$ ne s'arrête pas par manque de transition partant d'un état q pour une certaine lettre a. Dans le cas contraire, si toutes les exécutions de $\mathcal{A}$ sur z sont finies, on dit que z est **rejeté abruptement**.

Lemme IV.2 Les mots finis faiblement acceptés par $\mathcal{A}$ sont exactement les chemins finis dans G, et les mots infinis faiblement acceptés sont exactement les chemins infinis dans G.

Preuve Soit un chemin fini $v_0v_1...v_n$ dans G. Par connexité de G, ce mot admet une continuation acceptante (on peut parcourir le graphe depuis v_n pour atteindre v_0, puis v_1, ..., v_n, et à nouveau v_0, etc., à l'infini). Par conséquent, ce chemin doit être faiblement accepté, car sinon la continuation serait rejetée (abruptement).

Soit un mot $v_0v_1...v_n$ faiblement accepté. Soit q le représentant de $v_0v_1...v_n$ dans q (donc l'état auquel aboutit l'exécution de $\mathcal{A}$ sur ce mot). Par la proposition III 3.2 (qui peut s'étendre aux automates non-complets), q n'est pas un pseudo-puits. Ainsi, $v_0v_1...v_n$ est le préfixe d'un mot de L, donc en particulier, c'est un chemin dans G.

Étendons à présent la proposition aux chemins infinis. Soit un chemin infini $v_0v_1...$ dans G. Tout préfixe $v_0...v_n$ de ce chemin est faiblement accepté, donc le chemin infini entier est faiblement accepté. Réciproquement, soit un mot infini $v_0v_1...$ faiblement accepté. Tout préfixe est un chemin dans G, donc le mot infini entier est un chemin infini dans G. □

À présent, essayons de retrouver dans $\mathcal{A}$ une structure en plusieurs « copies » similaire à celle de $\mathcal{A}_0$.

Définition Considérons le graphe orienté formé par les états et les transitions de A. Les **composantes** sont les composantes connexes C_1, C_2, ..., C_n du sous-graphe où on a retiré les transitions acceptantes.

L'idée de cette définition est que dans la construction conjecturée optimale, les composantes sont les différentes copies du graphe de départ.

Lemme IV.3 Les composantes sont fortement connexes.

Preuve Supposons qu'une composante ne soit pas fortement connexe. Il existe une transition entre deux de ses composantes connexes, U et U'. Or, le graphe est saturé en transitions acceptantes. Le langage accepté si cette transition seule était acceptante est inclus dans L, car pour qu'une exécution visite cette transition une infinité de fois, elle doit sortir et rentrer une infinité de fois dans la composante (car elle ne peut pas revenir à U depuis U' tout en restant dans la composante), ce qui la fait passer une infinité de fois par des transitions acceptantes. Donc, cette transition pourrait être rendue acceptante sans changer le langage. Cela contredit le fait que A est saturé en transitions acceptantes. □

À présent, essayons de formaliser la structure que forment les composantes. Si on prend les composantes comme sommets et les transitions acceptantes comme arêtes (étiquetées par leurs lettres respectives), on obtient un graphe, que nous appellerons **graphe des composantes**. Dans $\mathcal{A}_0$, ce graphe est un simple cycle (dans chacune, une fois que la lettre « attendue » est lue, on passe à la suivante).

Définition On dira qu'une composante C *attend* une lettre v si aucune transition à l'intérieur de C n'est étiquetée par v.

Proposition IV.4 Toute composante attend au moins une lettre.

Preuve Supposons le contraire. On peut construire un mot infini qui n'est pas accepté alors qu'il est dans L, en trouvant un chemin dans l'automate vers la composante (ses états sont accessibles par minimalité) et en continuant à l'infini dans la composante, en allant vers la source d'une transition étiquetée par v_1, en prenant cette transition, puis en allant vers la source d'une transition étiquetée par v_2, et ainsi de suite, en recommençant avec v_1 après $v_{|V|}$. Les raccordements existent par connexité forte de la composante. □

Corollaire IV.5 Le graphe des composantes contient un cycle.

Preuve Lorsqu'une composante attend une lettre, il existe une transition sortant de la composante étiquetée par cette lettre (on le prouve en construisant un chemin vers la composante, puis un chemin à l'intérieur de la composante tel que le chemin correspondant dans le graphe se rende vers la lettre en question). Or, dans un graphe orienté où au moins une arête part de tout sommet, il existe un cycle (en partant d'un sommet quelconque et en prenant des arêtes sortantes successives, on construit un chemin, qui doit revenir vers un sommet déjà visité par principe des tiroirs infini). □

À ce stade, si nous savions montrer que le graphe des composantes est, tout entier, un cycle, il serait relativement simple de conclure, en analysant la structure de chaque composante à l'aide du lemme sur les mots faiblement acceptés.

Malheureusement, nous n'avons pas réussi à « maîtriser » la structure du graphe des composantes. Pour prouver que c'est un cycle, on aimerait utiliser la minimalité, montrer qu'un cycle est optimal. La difficulté principale qui se pose est que le graphe des composantes n'est pas indépendant des composantes elles-mêmes. Dans l'automate $\mathcal{A}_0$, chaque composante est une copie de G, sans les états qui ont les mêmes voisins, mais aussi sans le sommet correspondant à la lettre qui est « attendue », car il est inutile, comme vu dans l'exemple plus haut. Il est donc possible a priori qu'une structure plus complexe qu'un cycle sur le graphe des composantes permette à chaque composante d'être plus petite en « attendant » plus de lettres.

Néanmoins, nous avons vérifié expérimentalement la conjecture. Nous avons utilisé la bibliothèque de vérification Spot, qui implémente la minimisation des automates de Büchi à acceptation par transitions à l'aide d'une traduction vers le problème SAT et d'un SAT-solveur. En tirant des automates aléatoires de petite taille (quatre ou cinq états), nous n'avons pas trouvé de contre-exemple. Cette constatation nous a poussés à abandonner cette piste, car même si nous pouvions prouver que la construction est optimale, détecter les points d'articulation et les sommets ayant les mêmes voisins se fait en temps polynomial, et l'automate minimal ne permet donc pas d'obtenir une information intéressante (NP-difficile à calculer) sur le graphe.

V Minimisation polynomiale des automates de co-Büchi GFG à acceptation par transitions

Comme expliqué en introduction, il existe un cas où la minimisation des automates de Büchi ou co-Büchi est polynomiale. Ce cas est celui des automates co-Büchi GFG à acceptation par transitions, comme prouvé dans l'article [9]. Notons que pour les automates co-Büchi GFG à acceptation par états, [7] prouve que la minimisation est NP-complète. Mis ensemble, ces deux résultats peuvent surprendre car ils montrent que le changement syntaxique de la condition d'acceptation pour la placer sur les transitions, qui était au départ une manière de simplifier les détails techniques de certaines constructions, peut produire des classes d'automates qui se comportent de manière totalement différente des automates à acceptation par états. Un point particulièrement mystérieux est que cette preuve fonctionne dans le cas co-Büchi mais pas dans le cas Büchi (qui n'est pas symétrique pour les automates non-déterministes, même GFG, comme expliqué dans la section III 2). De plus, étant donné que les automates GFG sont a priori plus complexes, il est très probable que les auteurs aient d'abord cherché une preuve pour les automates déterministes et ne l'aient pas trouvée. Sans expliquer en entier la preuve, qui remplit plusieurs dizaines de pages denses, cette partie fait quelques remarques.

Définition Un automate de co-Büchi est dit **presque déterministe** (« *safe deterministic* » dans [9]) si pour tout état q, pour toute lettre a, il existe au plus une transition *sûre* sortant de q étiquetée par a.

Ainsi, dans un automate presque déterministe, il peut y avoir des choix non-déterministes entre plusieurs transitions rejetantes ou entre des transitions rejetantes et une transition sûre, mais jamais entre plusieurs transitions sûres.

La première étape de la minimisation est de transformer l'automate en un automate équivalent presque déterministe. Elle fait appel à un résultat prouvé dans [11] à l'aide de jeux. Notons que cette étape n'augmente pas le nombre d'états (mais peut éventuellement le diminuer).

Définition Un automate $\mathcal{A}$ est dit **sémantiquement déterministe** si, pour tout état q, pour toute lettre a, tous les états q' tels qu'il existe une transition $q \xrightarrow{a} q'$ sont équivalents par la relation de Myhill-Nérode sur les états, $\sim_{\mathcal{A}}$.

La deuxième étape est de rendre l'automate sémantiquement déterministe. Cette étape est beaucoup plus facile. La construction proposée supprime les états q tels que $\mathcal{A}^q$ n'est pas GFG, puis détecte certaines transitions à supprimer également. En particulier, le nombre d'états n'augmente toujours pas, et à l'issue de cette construction, pour tout état q, $\mathcal{A}^q$ est GFG.

Définition Un automate de co-Büchi $\mathcal{A}$ est dit **gentil** si
- $\mathcal{A}$ est presque déterministe,
- $\mathcal{A}$ est sémantiquement déterministe,
- Pour tout état q de $\mathcal{A}$, l'automate $\mathcal{A}^q$ est GFG,
- Tous les états de $\mathcal{A}$ sont accessibles,
- $\mathcal{A}$ est saturé en transitions acceptantes.

Proposition V.1 Tout automate de co-Büchi GFG à acceptation par transitions peut être transformé, en temps polynomial, en un automate de co-Büchi équivalent, sans augmenter le nombre d'états.

La preuve consiste à enchaîner la construction qui rend presque déterministe, puis la construction qui rend sémantiquement déterministe, qui supprime au passage les états q pour lesquels $\mathcal{A}^q$ n'est pas GFG, et les deux derniers points sont faciles à obtenir (cf. section III 7 pour le deuxième).

Corollaire V.2 Soit $\mathcal{A}$ est un automate de co-Büchi GFG à acceptation par transitions. Si $\mathcal{A}$ est minimal, alors il existe un automate $\mathcal{A}'$ avec le même nombre d'états que $\mathcal{A}$, équivalent à $\mathcal{A}$, qui est gentil.

Ainsi, il suffit de résoudre le problème de minimisation sur les automates gentils.

Définition Soit $\mathcal{A}$ un automate de co-Büchi GFG à acceptation par transitions. On note $A_{\text{sûr}}$ l'automate $\mathcal{A}$ où les transitions rejetantes sont retirées. Les **composantes** de $\mathcal{A}$ sont les composantes fortement connexes de $\mathcal{A}_{\text{sûr}}$.

On définit également deux relations sur les états de $\mathcal{A}$:

$$q \approx s \iff (\mathcal{L}(\mathcal{A}^q) = \mathcal{L}(\mathcal{A}^s) \wedge \mathcal{L}(\mathcal{A}^q_{\text{sûr}}) = \mathcal{L}(\mathcal{A}^s_{\text{sûr}}))$$

$$q \lesssim s \iff (\mathcal{L}(\mathcal{A}^q) = \mathcal{L}(\mathcal{A}^s) \wedge \mathcal{L}(\mathcal{A}^q_{\text{sûr}}) \subset \mathcal{L}(\mathcal{A}^s_{\text{sûr}}))$$

On dit que $\mathcal{A}$ est **centralisé** si pour tous états q et s tels que $q \lesssim s$, q et s sont dans la même composante. On dit que $\mathcal{A}$ est **presque minimal** si on n'a jamais $q \lesssim s$ pour $q \neq s$.

Le théorème fondamental est alors le suivant :

Théorème V.3 Soit $\mathcal{A}$ un automate de co-Büchi GFG à acceptation par transitions qui est gentil. Alors, $\mathcal{A}$ est minimal si et seulement si il est centralisé et presque minimal.

Les auteurs donnent alors : une preuve du théorème, une construction pour centraliser un automate, et une construction pour le rendre presque minimal. Sans rentrer dans des détails qui seraient longs, mentionnons les idées directrices. Pour la centralisation, l'idée de la construction est de trouver un ensemble de composantes $\mathcal{C}$ tel que pour toute composante $C \in \mathcal{C}$, tous les éléments de C soient maximaux pour $\lesssim$, et que pour tout état q d'une composante qui n'est pas dans $\mathcal{C}$, il existe un état q dans une composante de $\mathcal{C}$ tel que $q \lesssim s$. En choisissant bien les transitions entre composantes de $\mathcal{C}$, on peut parvenir à supprimer toutes les autres composantes. Intuitivement, un état q maximal pour $\lesssim$ est « intéressant » pour la minimisation car une exécution acceptante de $\mathcal{A}$ ne rencontre plus que des transitions sûres à partir d'un certain moment, et « beaucoup » de mots infinis sont acceptés avec des transitions sûres uniquement à partir de q. Pour la « presque minimisation », il suffit de fusionner tous les états équivalents par $\approx$.

À partir de cette preuve, nous avons cherché à comprendre pourquoi il n'était pas facile de transposer le résultat au cas déterministe.

D'abord, pour les automates déterministes, un automate « gentil » est simplement un automate normalisé dont tous les états sont accessibles (car les propriétés « presque déterministe », « sémantiquement déterministe » et « pour tout état q, $\mathcal{A}^q$ est GFG » sont automatiques).

Comme les automates déterministes sont un cas particulier des automates GFG, il reste vrai que tout automate (co-Büchi déterministe, à acceptation par transitions) gentil, centralisé et presque minimal est minimal… mais en tant qu'automate (co-Büchi *non-déterministe* à acceptation par transitions) *GFG*. Même si l'automate pris en entrée est déterministe, l'automate de sortie peut ne pas l'être.

Pour un automate déterministe, l'étape qui rend l'automate gentil consiste simplement à émonder l'automate (supprimer les états inatteignables) et à transformer certaines transitions sûres en transitions rejetantes. Cela n'introduit évidemment pas de non-déterministe.

En revanche, il s'avère que l'étape de centralisation et l'étape qui rend l'automate presque minimal peuvent toutes les deux, *a priori*, introduire du non-déterminisme. (Cependant, nous n'avons pas eu le temps de chercher, par exemple, des contre-exemples explicites où la centralisation préserve le déterminisme mais la presque-minimisation introduit un non-déterminisme.)

La centralisation préserve les transition sûres des composantes qu'elle retient, mais elle ajoute, pour tout état q dont ne sort aucune transition sûre pour a, une transition rejetante $q \xrightarrow{a} q'$ pour toute transition rejetante $q \xrightarrow{a} q''$ dans l'automate de départ où $q' \sim q''$. Même s'il n'y a qu'une transition $q \xrightarrow{a} q''$ dans l'automate de départ, il y a potentiellement plusieurs états q' retenus qui sont équivalents à q'' au sens de $\sim$.

Un phénomène similaire se produit pour la presque-minimisation. Cette étape quotiente l'automate par la relation $\approx$. Si on a $q \approx s$ et des transitions $q \xrightarrow{a} q'$ et $s \xrightarrow{a} s'$, alors il est vrai que $q' \sim s'$ mais il n'est pas vrai a priori que $q' \approx s'$. Ainsi, on peut avoir une transition $[q]_{\approx} \xrightarrow{a} [q']_{\approx}$ et une transition $[q]_{\approx} = [s]_{\approx} \xrightarrow{a} [s']_{\approx}$, où $[q']_{\approx} \neq [s']_{\approx}$ (la notation $[\cdot]_{\approx}$ représente la classe d'équivalence modulo $\approx$).

Pour conclure cette partie, les auteurs, après avoir proposé une manière d'ajouter des transitions rejetantes qui permet d'obtenir un automate canonique parmi les automates de co-Büchi GFG à acceptation par transitions minimaux pour un langage, posent la question suivante en la laissant ouverte : si $\mathcal{A}$ et $\mathcal{B}$ sont deux automates de co-Büchi déterministes à acceptation par transitions minimaux pour le même langage, il n'est pas toujours vrai que $\mathcal{A}$ et $\mathcal{B}$ sont isomorphes, comme le montre le contre-exemple vu dans la section III 6, mais est-il vrai que $\mathcal{A}_{\text{sûr}}$ et $\mathcal{B}_{\text{sûr}}$ sont isomorphes si on ne prend pas en compte l'état initial ? En effet, les auteurs prouvent un résultat analogue pour les automates co-Büchi GFG à acceptation par transitions.

Expérimentalement, nous avons testé cette conjecture sur tous les automates à deux ou trois états sur un alphabet de deux lettres (la minimisation est faite par recherche exhaustive, donc il est difficile d'aller plus loin que trois états à cause de l'explosion exponentielle du nombre d'automates). Le résultat est que la conjecture est vérifiée sur ces automates.

VI Conclusion, méta-informations et perspectives

Le problème de minimisation des automates de Büchi et co-Büchi déterministes à acceptation par transitions garde son mystère. La preuve de NP-complétude sur les automates de Büchi et co-Büchi, déterministes et GFG, à acceptation par états, semble difficile à adapter, car il faudrait trouver un bon langage sur un graphe qui permette d'obtenir des informations intéressantes, tandis que la preuve d'appartenance à P pour les automates de co-Büchi GFG à acceptation par transitions, qui est complexe, s'adapte elle aussi mal car elle introduit du non-déterminisme (GFG) dans l'automate que produit la construction.

VI 1 Méta-informations

Ce stage a été supervisé par Mickaël Randour à l'université de Mons. J'ai été largement suivi au jour le jour par Pierre Vandenhove, doctorant dans l'équipe dirigée par Mickaël Randour.

J'ai consacré les premières semaines du stage à me familiariser avec le sujet, notamment les divers types d'automates, leur pouvoir expressif, les constructions qui fonctionnent ou ne fonctionnent pas, comment convertir l'acceptation par états en acceptation par transitions, etc. J'ai réalisé l'essentiel de l'analyse « en surface » du problème, notamment comprendre pourquoi il est dans NP (section III 1), et surtout comprendre pourquoi on peut se restreindre aux automates complet. Sur ce dernier point, c'est Pierre Vandenhove qui m'a suggéré l'idée (qu'il connaissait) que les automates minimaux avaient tous ou n'avaient tous pas de puits, et j'ai formalisé la preuve (que j'ai plus tard adaptée aussi au cas GFG). J'ai aussi bénéficié des explications de Pierre Vandenhove sur la congruence de Myhill-Nérode, et trouvé le théorème sur la saturation (section III 7), dont je devais me rendre compte plus tard qu'il était en fait prouvé dans [9] (mais le résultat que je propose s'applique aussi à l'acceptation par états).

J'ai également compris la preuve de Schewe de NP-complétude de la minimisation des automates de Büchi et co-Büchi déterministes à acceptation par états.

J'ai alors essayé de la transposer, en recherchant divers gadgets pour réduire le problème de la couverture par sommets, en essayant d'autres langages définis sur un graphe et en espérant en tirer des informations, et en me renseignant sur les divers problèmes NP-complets qui existent sur les graphes. La tentative la plus aboutie est celle qui est décrite dans la partie IV.

Avec l'aide de Pierre Vandenhove pour prendre en main la bibliothèque Spot, j'ai également réalisé un programme qui recherche des contre-exemples à l'unicité d'un automate minimal, ce qui a fourni plusieurs contre-exemples dont le plus simple est donné section III 6.

Comme je n'arrivais pas à montrer le résultat, Mickaël Randour m'a conseillé de chercher à comprendre l'article [9]. J'ai donc pris à nouveau du temps pour me familiariser avec des concepts, en l'occurrence les automates GFG, et attaqué la preuve, qui m'a pris plus d'une semaine à comprendre complètement. (D'ailleurs, cet article parlait aussi de la non-unicité d'un automate minimal et donnait le même contre-exemple.) Cela a pris encore du temps de comprendre dans quelle mesure des parties de la preuve s'appliquaient ou ne s'appliquaient pas aux automates déterministes.

Dans le temps qui restait, j'ai écrit un nouveau programme qui teste la conjecture d'isomorphisme des automates minimaux restreints à leurs transitions sûres, et vérifié la conjecture sur des automates de petite taille, avant d'écrire ce rapport.

Au départ, j'étais convaincu que le problème serait NP-complet, et je cherchais une réduction. Après avoir lu [9], j'en suis beaucoup moins sûr, et plutôt persuadé que le problème est polynomial. Si je pouvais continuer ce stage, je reprendrais [9] de manière plus approfondie pour essayer de transposer les concepts, et j'essaierais également de prouver la conjecture d'isomorphisme. J'approcherais donc le problème différemment : au lieu de chercher des réductions, chercher un algorithme polynomial (quitte à ce que cela débouche sur une meilleure compréhension du problème qui donnerait une preuve de NP-complétude).

Il n'est pas non plus à exclure que le problème de calcul soit NP-complet alors que le problème de décision serait polynomial.

VI 2 Remerciements

Merci à Mickaël Randour pour avoir accepté la charge de superviser ce stage, pour m'avoir intégré à son équipe, et pour m'avoir donné de bons conseils de recherche, notamment celui de m'intéresser à [9]. Merci à Pierre Vandenhove qui a proposé le sujet du stage, m'a accompagné au jour le jour, m'a appris beaucoup de choses sur les automates, et m'a aidé par des discussions régulières. Merci également à tous les doctorants de l'équipe de vérification, qui m'ont réservé un très bon accueil, et par la bonne ambiance au sein de l'équipe, m'ont fait passer deux mois de stage très agréables.

Bibliographie

[1] Charlie Jacomme. https://www.lsv.fr/~jacomme/1718/ca/td4_sol.pdf (Un corrigé de TD du cours de complexité avancée en M1 à l'ENS Paris-Saclay, où il est montré que le problème d'universalité d'un automate non-déterministe sur les mots finis est PSPACE-complet. Remarque : ce document est cité sur Wikipédia !)

[2] Udi Boker, "Why these automata types?," 2018, doi: 10.29007/c3bj. (Un résumé assez complet sur les différents types d'automates et la complexité des conversions entre eux.)

[3] Sven Schewe, "Minimisation of Deterministic Parity and Büchi Automata and Relative Minimisation of Deterministic Finite Automata," 2010, doi: 10.48550/arXiv.1007.1333. (La preuve de la NP-complétude du problème de minimisation des automates de Büchi déterministes et des automates de parité déterministes.)

[4] Sudeep Juvekar, and Nir Piterman, "Minimizing Generalized Büchi Automata," doi: 10.1007/11817963_7.

[5] Bader Abu Radi, and Orna Kupferman, "Minimization of automata for liveness languages," 2022, doi: 10.1007/978-3-031-19992-9_12.

[6] T. Jiang, and B. Ravikumar, "Minimal NFA problems are hard.," *SIAM J. Comput.*, 1993.

[7] Sven Schewe, "Minimising good-for-games automata is NP-complete," 2020, doi: 10.48550/arXiv.2003.11979.

[8] Dimitra Giannakopoulou, and Flavio Lerda, "From States to transitions: Improving translation of LTL formulae to büchi automata," doi: 10.1007/3-540-36135-9_20.

[9] Bader Abu Radi, and Orna Kupferman, "Minimization and canonization of GFG transition-based automata," doi: 10.46298/LMCS-18(3:16)2022. (Donne un algorithme polynomial de minimisation des automates co-Büchi GFG à acceptation par transitions.)

[10] O. Kupferman, and M. Vardi, "Verification of fair transition systems," *Comput. Aided Verification*, vol. 4414, 1996, doi: 10.1007/11817963.

[11] Denis Kuperberg, and Michał Skrzypczak, "On determinisation of good-for-games automata." (Montre que les automates de Büchi GFG peuvent être déterminisés avec augmentation quadratique du nombre d'états, alors que les automates co-Büchi GFG peuvent nécessiter une augmentation exponentielle. Montre également que décider le caractère GFG d'un automate est NP pour les automates de Büchi et polynomial pour les automates co-Büchi.)

VII Annexe : Pouvoir expressif des automates sur les mots infinis

On répond ici à la question de savoir quels langages peuvent être reconnus par les automates des divers types. Cette partie n'est pas fondamentale pour la minimisation, mais elle permet de mieux comprendre l'intérêt des divers types d'automates. En particulier, le fait que les automates de Büchi non-déterministes soient strictement plus expressifs que les automates de Büchi déterministes indique que le problème de minimisation que nous considérons a un domaine d'application limité, car en pratique, en vérification, la plupart des langages considérés peuvent être reconnus par un automate de Büchi non-déterministe, mais tous ne peuvent pas être reconnus par un automate de Büchi déterministe.

VII 1 Pouvoir expressif des automates de Büchi non-déterministes

Définition L'ensemble Ω des langages **ω-réguliers** est défini inductivement par

- Pour tout langage régulier L *qui ne contient pas le mot vide*, L^ω est ω-régulier.
- Pour tout langage régulier L et tout langage ω-régulier L', LL' est ω-régulier.
- Pour tous langages ω-réguliers A et B, $A \cup B$ est ω-régulier.

Proposition VII 1.1 Les langages ω-réguliers sont exactement les langages de la forme $A_1 B_1^\omega \cup A_2 B_2^\omega \cup ... \cup A_n B_n^\omega$.

Preuve Induction facile. □

Théorème VII 1.2 Les langages reconnaissables par un automate de Büchi non-déterministe sont exactement les langages ω-réguliers.

Preuve Par la proposition précédente, un langage ω-régulier peut s'écrire $A_1 B_1^\omega \cup A_2 B_2^\omega \cup ... \cup A_n B_n^\omega$. On peut construire un automate de Büchi qui le reconnaît par les constructions de la partie précédente pour l'itération (opération $\cdot^\omega$), la concaténation et l'union.

Réciproquement, soit un automate $\mathcal{A}$. Notons $q_1, q_2, ..., q_n$ ses états acceptants. Pour tout i, notons L_i le langage de l'automate fini déterministe construit à partir de $\mathcal{A}$, en gardant le même état initial et en prenant seulement q_i pour état final. Notons aussi M_i le langage de l'automate fini déterministe construit en remplaçant l'état initial de $\mathcal{A}$ par q_i et en prenant q_i pour unique état final. On montre alors que $\mathcal{L}(\mathcal{A}) = L_1 M_1^\omega \cup L_2 M_2^\omega \cup ... \cup L_n M_n^\omega$. En effet, d'une part, s'il existe une exécution acceptante de $\mathcal{A}$ sur un mot infini z, il existe i tel que l'état q_i est visité infiniment souvent par cette exécution, et pour un tel i, on a $z \in L_i M_i^\omega$ par définition de L_i et M_i. D'autre part, l'inclusion $\forall i, L_i M_i^\omega \subset \mathcal{L}(\mathcal{A})$ est facile, d'où le résultat. □

Ainsi, les automates de Büchi non-déterministes jouissent d'une propriété analogue aux automates finis non-déterministes, en ce qu'ils reconnaissent une classe de langages définie de manière semblable aux langages réguliers.

Néanmoins, la partie suivante montre qu'ils diffèrent fondamentalement des automates sur les mots finis car le non-déterminisme ne fait pas que permettre des automates plus succincts, mais fait gagner strictement en pouvoir expressif.

Remarque Pour les automates co-Büchi, le théorème précédent ne s'applique pas. En fait, on pourrait montrer que les automates de co-Büchi non-déterministes ont un pouvoir expressif équivalent aux automates de co-Büchi déterministes.

VII 2 Pouvoir expressif des automates de Büchi déterministes

Définition Soit $L \subset \Sigma^*$. On définit $\lim L \subset \Sigma^\omega$, la **limite de L**, comme le langage des mots infinis dont une infinité de préfixes appartiennent à L.

Remarque On pourrait prouver (sans difficulté particulière) les propriétés suivantes pour tous $A, B \subset \Sigma^*$ (pour les inclusions, l'inclusion réciproque est fausse en général) :

- $A \subset B \Rightarrow \lim A \subset \lim B$
- $\lim(A \cup B) = \lim A \cup \lim B$
- $A \lim(B) \subset \lim(AB)$
- $\lim(A \cap B) \subset \lim A \cap \lim B$
- $(\lim A)^c \subset \lim(A^c)$
- $A^\omega \subset \lim(A^*)$

Théorème VII 2.1 Les langages reconnaissables par un automate de Büchi déterministe sont exactement les langages qui sont la limite d'un langage régulier. Plus précisément, étant donné un automate de Büchi déterministe $\mathcal{A}$, en réinterprétant $\mathcal{A}$ comme un automate déterministe sur les mots finis $\mathcal{B}$, on a $\mathcal{L}(\mathcal{A}) = \lim(\mathcal{L}(\mathcal{B}))$.

Preuve Si un mot $z = a_0 a_1 a_2 ...$ est accepté par $\mathcal{A}$, l'exécution $q_0, q_1, q_2, ...$ de $\mathcal{A}$ sur z visite un état acceptant pour une infinité de i, et pour tout i tel que q_i est acceptant dans $\mathcal{A}$, ce qui équivaut à être un état final dans $\mathcal{B}$, le préfixe $a_0 a_1 a_2 ... a_{i-1}$ de z est

accepté par $\mathcal{B}$. Réciproquement, si une infinité de préfixes de z sont acceptés par $\mathcal{B}$, une infinité des q_i sont des états finaux de $\mathcal{B}$, donc des états acceptants de $\mathcal{A}$. □

Théorème VII 2.2 Les langages limite d'un langage régulier, soit les langages reconnaissables par un automate de Büchi déterministe, sont une sous-classe *stricte* des langages ω-réguliers, soit les langages reconnaissables par un automate de Büchi non-déterministe.

Preuve L'inclusion est évidente (les automates déterministes sont un cas particuliers des automates non-déterministes). Un exemple de langage ω-régulier mais qui n'est pas reconnaissable par un automate de Büchi déterministe est le langage sur $\Sigma = \{a, b\}$ des mots qui ne contiennent qu'un nombre finis de a. En effet, supposons qu'il existe un automate de Büchi déterministe $\mathcal{A}$ qui reconnaisse ce langage. On va construire itérativement un mot infini qui est accepté alors qu'il contient un nombre infini de a. Pour cela, on part du mot fini a, et commence par ajouter des b à chaque étape, construisant ainsi le chemin de l'automate sur les mots a, puis ab, abb, etc. Étant donné que le mot infini ab^ω doit être accepté par l'automate (il contient un nombre fini de a, à savoir 1), il existe i tel que l'exécution de $\mathcal{A}$ sur ab^i se termine par un état acceptant. On recommence alors le processus avec ab^ia, ab^iab, ab^iabb, etc., jusqu'à trouver à nouveau un état acceptant, etc. On construit ainsi un mot infini contenant une infinité de fois la lettre a, et qui pourtant est accepté, ce qui est absurde. □

Remarque À première vue, on pourrait penser que le procédé du théorème VII 2.2 devrait marcher pour des automates de Büchi non-déterministes, et s'étonner de ce que ceux-ci reconnaissent une classe de langages strictement plus grande. Expliquons donc pourquoi la preuve ne fonctionnerait plus dans le cas non-déterministe. Soit $\mathcal{A}$ un automate de Büchi non-déterministe, et $\mathcal{B}$ l'automate $\mathcal{A}$ réinterprété comme automate non-déterministe sur les mots finis. On a toujours $\mathcal{L}(\mathcal{A}) \subset \lim(\mathcal{L}(\mathcal{B}))$, prouvé de la même manière, mais $\lim(\mathcal{L}(\mathcal{B})) \subset \mathcal{L}(\mathcal{A})$ devient faux en général. En effet, si $z \in \lim(\mathcal{L}(\mathcal{B}))$, z possède une infinité de préfixes qui sont acceptés par $\mathcal{B}$, mais comme $\mathcal{B}$ est non-déterministe, les chemins dans $\mathcal{B}$ qui acceptent ces préfixes font potentiellement des choix non-déterministes différents. Or, pour prouver $z \in \mathcal{L}(\mathcal{A})$, il faudrait exhiber un chemin unifié de $\mathcal{A}$ sur z qui contienne une infinité d'états acceptants, ce que l'on ne peut pas construire à partir de chemins qui divergent.

VIII Annexe : NP-complétude de la minimisation des automates de Büchi déterministes à acceptation par états

Cette partie présente les grandes lignes de la preuve de Schewe du du fait que la minimisation d'un automate de Büchi déterministe, avec l'acceptation classique par états, est NP-complète. Cette preuve provient de l'article [3]. Nous avons déjà noté dans la section III 1 que le problème est dans NP, il ne reste donc qu'à montrer qu'il est NP-difficile.

Le même auteur a généralisé ensuite cette preuve aux automates GFG dans [7,].

Dans cette partie, les automates seront non-complets.

La preuve se fait par réduction du problème de couverture minimale par sommets, dont nous commençons par rappeler la définition :

Définition Si $G = (V, E)$ est un graphe non-orienté, une **couverture par sommets** de G est un ensemble de sommets $C \subset V$ tel que pour toute arête $(s, t) \in E$, on ait $s \in C$ ou $t \in C$.

Le **problème de la couverture par sommets** (mis ici sous forme de problème de décision) est le problème de savoir, étant donné un graphe non-orienté G et un entier k, s'il existe une couverture par sommets de G avec k sommets au plus. Ce problème est connu comme étant NP-complet.

On montre d'abord (facilement) que le problème de couverture minimale par sommets d'un graphe quelconque se réduit à celui sur les graphes connexes.

Fixons un graphe $G = (V, E)$ connexe, simple et non-trivial. Fixons également un sommet distingué $v_0 \in V$ quelconque. Notons $C \subset V$ une couverture par sommets minimale de G. On définit un langage L limite d'un langage régulier, et on montre que la taille d'un automate de Büchi déterministe minimal qui reconnaît L est exactement $2|V| + |C|$. De plus, on montre qu'on peut construire en logspace un automate de Büchi qui reconnaît L. Ainsi, la réduction consiste à construire cet automate et à le minimiser, ce qui fournit $|C|$ en soustrayant $2|V|$ à la taille de l'automate obtenu.

Définissons d'abord le langage L. On forme l'alphabet $V_\natural$ en rajoutant une lettre fraîche notée $\natural$ à l'ensemble V des sommets de G, qui sera vu comme alphabet. Étant donné un chemin fini $v_0, v_1, v_2, ..., v_n$ (commençant par le sommet distingué v_0) dans G, on dit qu'un mot correspond à ce chemin s'il est de la forme $v_0^+v_1^+v_2^+...v_n^+$. De manière analogue, si $v_0, v_1, v_2, ...$ est un chemin infini, les mots infinis correspondant à ce chemin sont ceux de la forme $v_0^+v_1^+v_2^+...v_n^+$. On définit L comme le ω-langage sur $V_\natural$ formé des mots infinis qui correspondent à un chemin infini sur G, et des mots infinis dont un préfixe est de la forme $w\natural x$, où w correspond à un chemin fini sur G, et x est la dernière lettre de w.

Lemme VIII.1 On peut construire de manière effective, en logspace, un automate de Büchi déterministe complet qui reconnaît L, de taille $2|G| + |C|$.

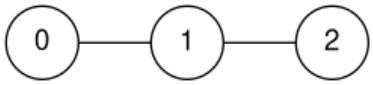
Avant de prouver le lemme, donnons l'intuition qui le sous-tend. Pour reconnaître L, l'idée la plus naturelle est d'utiliser trois copies de G dans l'automate. Les états de la copie 1 sont acceptants, et lorsqu'un état de la copie 1 est visité, cela correspond dans un chemin infini de la forme $v_0^+v_1^+...$ à un changement de lettre, où on passe de v_i à v_{i+1}. Les états de la copie 2 sont rejetants, et en lisant v_n^+, l'automate passe par l'état qui correspond au sommet v_n dans la copie 1 en lisant le premier v_n, puis passe de cet état à l'état correspondant dans la copie 2 pour lire les v_n suivants. Ainsi, l'automate visite un état acceptant à chaque fois qu'il

change de lettre v_i, ce qui permet de s'assurer qu'un mot qui se terminerait par v_n^ω n'est pas accepté. Enfin, les états de la copie 3 sont visités après un symbole $\natural$.

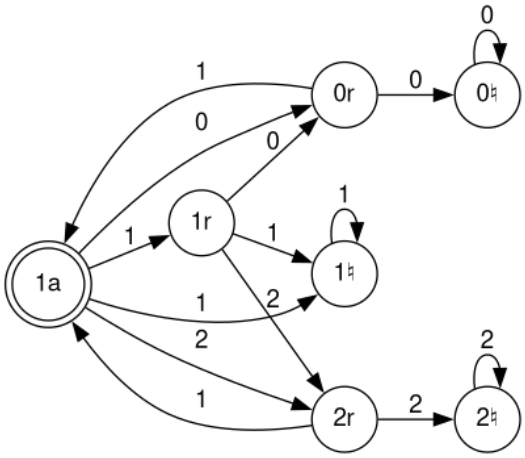
Ainsi, naïvement, on peut reconnaître L avec un automate de taille $3|G|$. L'idée centrale de la preuve est que la copie 1 peut être réduite à une taille de $|C|$ au lieu de $|G|$. En effet, considérons un mot de L de la forme $v_0^+ v_1^+ v_2^+ \ldots$. Il y a une infinité de changemens de lettre, de v_i à v_{i+1}. Pour chacun de ces changements, v_i et v_{i+1} sont reliés par une arête dans G, par définition de L, donc v_i ou v_{i+1} appartient à C. Ainsi, dans le cas où $v_i \notin C$, on a « oublié » de détecter le changement survenu en passant de v_{i-1} à v_i, ayant « économisé » un état dans la copie 1 en retirant v_i, mais le changement de v_i à v_{i+1} reste détecté, par le passage v_{i+1}. Finalement, au moins un « changement de sommet » sur deux est détecté, ce qui assure que le mot reste accepté.

Illustration de la construction :

Sur le graphe



dont une couverture par sommets minimale est $C = \{1\}$, l'automate construit est :



,

(la copie 1 est formée de l'état 1a, la copie 2 des états 0r, 1r et 2r, et la copie 3 des états 0♮, 1♮ et 2♮).

Preuve L'ensemble des états est formé de :

- Pour chaque $v \in V$, un état q_v^r rejetant,
- Pour chaque $v \in V$, un état $q_v^\natural$ acceptant,
- Pour chaque $v \in C$, un état q_v^a acceptant.

Les transitions sont :

- Pour chaque arête v—v' de G telle que $v' \in C$, une transition $q_v^r \xrightarrow{v'} q_{v'}^a$, et si $v \in C$, une transition $q_v^a \xrightarrow{v'} q_{v'}^a$,
- Pour chaque arête v—v' de G telle que $v' \notin C$, une transition $q_v^r \xrightarrow{v'} q_{v'}^r$,
- Pour chaque $v \in C$, une transition $q_v^a \xrightarrow{v} q_v^r$, et pour chaque $v \in V$, une transition $q_v^r \xrightarrow{v} q_v^r$,
- Pour chaque $v \in C$, une transition $q_v^a \xrightarrow{\natural} q_v^\natural$, et pour chaque $v \in V$, une transition $q_v^r \xrightarrow{\natural} q_v^\natural$,
- Pour chaque $v \in V$, une transition $q_v^\natural \xrightarrow{v} \top$.

Cet automate peut évidemment être construit en logspace. Si un mot est accepté, il est facile de voir qu'il doit être dans L. Réciproquement, s'il est dans L, on utilise le fait que C est une couverture par sommets de G pour montrer qu'il est accepté. □

À présent, le but est de prouver que ce nombre d'états, $2|V| + |C|$, est minimal. Fixons un automate de Büchi déterministe qui reconnaît L. On veut montrer qu'il est de taille au moins $2|V| + |C|$. Pour cela, on va chercher à montrer que cet automate contient un automate de structure similaire à celle de l'automate défini dans le lemme précédent.

Pour tout $v \in V$, on note S_v l'ensemble des états de l'automate qui sont l'aboutissement de l'exécution de l'automate sur un mot correspondant à un chemin fini de G. On pose $S_v = S_v^a \sqcup S_v^r$, où S_v^a et S_v^r contiennent respectivement les états acceptants et rejetants de Q_v. On note également $S_v^\natural$ l'ensemble des états accessibles depuis un état de S_v. On pose

$$S^a = \cup_{v \in V} S_v^a$$

$$S^r = \cup_{v \in V} S_v^r$$

$$S = \cup_{v \in V} S_v = S^a \sqcup S^r$$

$$S^\natural = \cup_{v \in V} S_v^\natural$$

Remarquons que dans l'automate de taille $2|V| + |C|$ défini précédemment, S^a est la copie 1, et S^r est la copie 2, $S^\natural$ est la copie 3. Pour tout $v \in V$, S^r_v et $S^\natural_v$ sont respectivement des singletons formés de l'état correspondant à v dans les copies 2 et 3, et si $v \in C$, S^a_v est le singleton formé de l'état correspondant à V dans la copie 1.

On peut alors prouver les lemmes suivants :

- S^a, S^r et $S^\natural$ sont disjoints. En particulier, l'automate est de taille au moins $|S^a| + |S^r| + |S^\natural|$.

  Idée de preuve : on peut arriver à tout état de $S^\natural$ après avoir lu un $\natural$, mais ce n'est le cas pour aucun état de $S = S^a \cup S^r$. Par ailleurs, les états de S^a sont acceptants et ceux de S^r rejetants.

- Pour tout $v \in V$, S^r_v est non-vide, et pour $v \neq v'$, S^r_v et $S^r_{v'}$ sont disjoints. En particulier, $|S^r| \geq |V|$.

  Idée de preuve : c'est l'ajout de la lettre $\natural$ et des mots de la forme $v_0^+ ... v_n^+ \natural v_n ...$ qui permet de prouver que S^r_v et $S^r_{v'}$ sont disjoints. Le mot $\natural v ...$ est accepté à partir d'un état de S^r_v, mais pas à partir d'un état de $S^r_{v'}$.

- Les $v \in V$ tels que S^a_v soit non-vide forment une couverture par sommets de G, et pour $v \neq v'$, S^a_v et $S^a_{v'}$ sont disjoints. En particulier, $|S^a_v| \geq |C|$.

  Idée de preuve : on montre comme au point précédent que S^a_v et $S^a_{v'}$ sont disjoints. Si S^a_v et $S^a_{v'}$ sont vides alors que v et v' sont reliés par une arête, un mot se terminant par $(vv')^\omega$ ne serait pas accepté alors qu'il devrait l'être.

- Pour tout $v \in V$, $S^\natural_v$ est non-vide, et pour $v \neq v'$, $S^\natural_v$ et $S^\natural_{v'}$ sont disjoints. En particulier, $|S^\natural| \geq |V|$.

  Idée de preuve : $S^\natural_v$ et $S^\natural_{v'}$ sont disjoints car un mot commençant par v est accepté à partir d'un état de $S^\natural_v$ mais pas à partir d'un état de $S^\natural_{v'}$.

Au total, on a prouvé que la taille de l'automate est au moins $|S^a| + |S^r| + |S^\natural|$, où $|S^a| \geq |C|$, $|S^r| \geq |G|$ et $|S^\natural| \geq |G|$, donc l'automate est bien de taille au moins $2|G| + |C|$, ce qui conclut la preuve.

Terminons en expliquant pourquoi cette preuve ne se transpose absolument pas au cas de l'acceptation par transitions : pour reconnaître le langage L, il suffit d'une copie « principale » de G, et une copie supplémentaire pour les $\natural$. Les transitions entre états de la copie principale sont toutes acceptantes, ce qui permet de reconnaître les changements de lettre. Malheureusement, l'automate minimal possède $2|G|$ états, ce qui ne permet plus d'extraire une information intéressante sur G, comme le nombre de sommets d'une couverture par sommets minimale.