Uniformisation des relations automatiques par transducteur subséquentiel

Nicolas Mazzocchi
Stage court 2015 encadré par Christof Löding
Logique et Théorie de Systèmes Discrets/Lehrstuhl für Informatik 7
RWTH Aachen University

23 août 2015

Le contexte général

De quoi s'agit-il ? Le contexte est celui de la synthèse d'un transducteur (subséquentiel) dénotant une fonction à partir d'un automate qui dénote une relation (automatique) tel que la fonction soit l'uniformisation de la relation pour des mots finis. Il s'avère que la procédure de décision utilise la résolution d'un jeux de sureté (avec anticipation).

D'où vient la question ? Cette question s'inscrit directement dans la catégorie des problèmes de synthèse. Depuis quelques années, ces derniers sont de plus en plus étudié dans le cadre des systèmes réactifs car ils deviennent un enjeux pour la vérification.

Quels sont les travaux déjà accomplis dans ce domaine dans le monde ? Des travaux existent déjà sur la résolution de jeux avec anticipation. J'ai d'ailleurs eu l'occasion dans lire un certain nombre me permettant de m'appuyer sur des résultats sur des mots infinis et des conditions d'acception de parité. En fait, le cas des mots finis fait survenir de nombreux phénomène contraignant qui n'apparaisse pas avec les mots infinis.

Le problème étudié

Quelle est la question que vous avez résolue ? Comme le descriptif du stage l'annonçait ma tache consistait à analyser la complexité de la construction présenté dans le papier "Uniformization in Automata Theory" de Arnaud Carayol & Christof Löding dans l'objectif d'identifier les bornes supérieure et inférieure de la procédure de décision. J'ai aussi apporté de la clarté sur un point imprécis de la procédure (le cas où le joueur **IN** s'arrête).

Pourquoi est-elle importante, à quoi cela sert-il d'y répondre ? Dans le papier de mon encadrant, la question de complexité n'est pas du tout abordée. Pourtant, les constructions qui sont présentées ne sont pas triviales. Cette réponse est donc une certaine finalisation à la procédure de décision présentée tout comme le détail de l'arrêt de l'entrée.

Pourquoi êtes-vous le premier chercheur de l'univers à l'avoir posée ? Exhiber cette complexité est une question bien moins capitale que celle de la procédure de décision elle-même. De plus les complexités des procédures de synthèses sont souvent inacceptable en pratique. Ceci qui rend le résultat moins enthousiasmant. Ce problème était donc à la fois une belle occasion de se plonger dans le problème d'uniformisation et un très bon sujet de stage qui m'a beaucoup plus d'ailleurs.

La contribution proposée

La première phase était de comprendre parfaitement le papier proposé. Pour ce faire, j'ai en parallèle lu plusieurs autres papiers et j'ai suivi le cours "Infinite Games" présenté sur place par

de mon encadrant. J'ai ensuite effectuer un découpage de mon travail. Sachez que la procédure de décision dépend d'un certain paramètre K. J'ai commencé par identifier les borne de complexité supérieure et inférieure en fonction de ce paramètre, puis je suis penché sur les bornes de la valeur de ce paramètre lui-même.

sup procédure : Pour déterminer la complexité de l'algorithme du papier, j'ai procéder par un découpage de complexité connu ou trivial.

inf procédure : Pour réduire le problème de résolution d'un jeux de sécurité (avec anticipation), je me suis appuyer sur un résultat avec des mots infinis.

min K : Pour identifier la valeur minimale du paramètre K, il s'est avéré que ce résultat existait déjà. J'ai trouvé la preuve après lecture de quelques papier.

max K : Pour exhiber la valeur maximale du paramètre K, j'ai étendu le raisonnement du papier en relation avec le théorème de Ramsey.

Les arguments en faveur de sa validité

Comment la validité de la solution dépend-elle des hypothèses de travail ? La robustesse des résultats que je propose est qu'elles s'appuient sur des preuves de domaines comparable que j'aie énoncé plutôt. Ainsi, plusieurs résultats auto-justifient.

Qu'est-ce qui montre que cette solution est une bonne solution ? Il aurait été instructif d'implémenter la procédure de décision afin de tester si la borne supérieur est cohérente mais ça n'a pas été fait d'une part par manque de temps (c'était un stage court) et d'autre part car ça aurait été légèrement un programme ad-hoc. Néanmoins, mes résultats sont en accord avec les conjectures qui ont été faite à partir des autres travaux et s'inscrivent comme la continuation de l'existant.

Le bilan et les perspectives

Qu'est-ce que votre contribution a apporté au domaine ? Ma contribution à permis d'identifier la complexité de la procédure présenté dans le papier tout en recoupant plusieurs travaux existant. Mes résultats attestent que le problème de synthèse d'un transducteur reste un problème difficile malgré le fait qu'on ce place dans un cadre simple (celui de relation automatique sur des mots finis). J'ai aussi apporté de la précision à la procédure.

Que faudrait-il faire maintenant ?, Quelle est la bonne *prochaine* question ?

- La borne supérieur pour K que je propose semble pas optimale car elle s'appuie sur le théorème de Ramsey, et ce parce que la construction présenté s'appuie sur ce théorème. Si le Lemme 19 du papier utilisait un argument d'itération plutôt qu'un argument d'idempotence, la borne serait plus petite (mais dans la même classe de complexité). C'est une discutions que j'ai eu avec mon encadrant mais je n'ai pas trouvé d'alternative au Lemme 19 dans le temps imparti.
- Il serait intéressant d'étudier la procédure de décision sur les arbres finis (pour une relation tree-automatique). Cette question n'est pas triviale puisque les modèles de transducteur qui définissent des relations sur les arbres finis nécessites des adaptations non canonique (contrairement au relation ω-automatique).

Un mot sur le stage

La période de mon stage officielle (est strictement respectée) de mon stages est du 8/6 au 12/7 et du 20/7 au 30/8 (soit 55 jours). Elle inclut la rédaction de la présentation et du mémoire. J'ai préféré rédiger un document clair et qui ne référence pas à outrance (et détaché de cette fiche). Bonne lecture !

Uniformisation des relations automatiques par transducteurs subséquentiels

Nicolas Mazzocchi*
Stage sous la direction de Christof Löding†

26 août 2015

Résumé

L'abstraction de la synthèse de programme réactif par la théorie des jeux est un domaine très étudié, ayant notamment des applications en vérification. Ce rapport de stage propose un exemple complet en détaillant la procédure d'uniformisation des relations automatiques par transducteurs subséquentiels présenté dans "*Uniformization in Automata Theory*" [3] écrit par C. Löding et A. Carayol. L'analyse de la complexité démontrera que la procédure de décision est dans 2ExpTime et que ce type de synthèse s'avère être ExpTime-difficile malgré les restrictions considérées.

Table des matières

*Élève à l'École Normale Supérieur de Cachan
†Lehrstuhl für Informatik 7, RWTH Aachen University

1 Introduction

1.1 Problème de synthèse

Le concept de générer automatiquement un programme à partir de sa spécification est une ambition très ancienne. Plus exactement, c'est en 1957, qu'Alonzo Church a posé le problème de la manière suivante :

"*Given a requirement which a circuit is to satisfy, we may suppose the requirement expressed in some suitable logistic system which is an extension of restricted recursive arithmetic. The synthesis problem is then to find recursion equivalences representing a circuit that satisfies the given requirement (or alternatively, to determine that there is no such circuit)*" [5].

Une instance du *problème de synthèse* est une spécification $\varphi(X, Y)$ qui exprime une relation binaire R_φ entre l'ensemble des entrées et celui des sorties. La synthèse consiste donc à décider s'il existe une *machine de Mealy* $\mathcal{M}$ (dans le contexte de Church) qui dénote une fonction $f_\mathcal{M}$ telle que $\forall X \cdot R_\varphi(X, f_\mathcal{M}(X))$. Et, au cas échéant, propose une constuction automatiquement $\mathcal{M}$.

Douze ans plus tard, Büchi et Landweber ont résolu ce problème [2]. Ainsi, pour les spécifications MSO-définissables, la réalisabilité de telles spécifications est décidable. Dans la positive, il est possible de construire algorithmiquement un programme (plus précisément, un automate) qui respect les contraintes données.

De nos jours et depuis quelques années, la syntèse des *systèmes réactifs* est un enjeux pour la recherche. Les programmes réactifs sont abstraits (en général par des *transducteurs séquentiels* déterministes) et interagissent avec leur environnement (usuellement représenté par un mot infini). En particulier, la communauté vérification s'intéresse aux problèmes de synthèse des systèmes réactifs pour la complexité dans le cas de restrictions sur les ressources, ou bien le cas où la spécification est donnée par une formule LTL, des variantes avec aspects temporisés, distribués ...

1.2 Objectif

Le stage Ce rapport a été rédigé dans le cadre de mon stage court de première année du Master Parisien de Recherche en Informatique (MPRI). Suite à la rédaction d'un mémoire sur l'expressivité des transducteurs encadré par B. Bollig[1] et S. Schwoon[2] à l'occasion du cours "*initiation à la recherche*", l'étude approfondi de l'emploi des transducteurs dans la théorie des automates ainsi que leurs mises en relation avec la vérification est apparu comme la continuation exacte de mon cursus vers ma spécialisation. En d'autres termes, le sujet qui m'a été proposé me fait progresser dans la direction de mon choix, sans être redondant avec l'enseignement que j'ai reçu jusqu'à présent. Par ailleurs, les deux événements cités m'ont mené à m'interroger et m'entretenir avec des enseignants au sujet de ma future thèse (avec B. Bollig puis E.Filiot[3]). Autant d'arguments pour exprimer la motivation, l'intérêt et la satisfaction de ce stage à mon égard.

[1] Chargé de recherche au LSV, CNRS & ENS Cachan
[2] Maître de conférence au LSV, ENS Cachan
[3] Chercheur qualifié à l'Université Libre de Bruxelles, FMRS

Je tiens par ailleurs à adresser mes remerciements les plus sincère aux membres de l'équipe "*Logik und Theorie diskreter Systeme*" pour leur gentillesse et leur convivialité, et plus particulièrement à C. Löding pour sa pédagogie.

Le rapport L'objectif de ce document est de détailler la procédure présenté dans "*Uniformization in Automata Theory*" [3] ainsi que sa complexité. Cette procédure permet de décider si une *relation automatique* peut être *uniformisée* par un *transducteur subséquentiel*. Le fait que celle-ci travail seulement sur des relations automatiques et synthétise seulement des transducteurs subséquentiels est une double restriction. Toutefois, ce document va mettre en évidence les liens étroits entre la synthèse de Church (introduit plutôt), le problème d'uniformisation et la résolution de jeux avec anticipation [13].

Tout d'abord, il sera nécessaire d'introduire plusieurs définitions et constructions auxiliaires qui seront utilisées par la suite. Puis, la procédure de décision précédemment cité sera présentée, ainsi que la génération automatique. Enfin, la complexité sera détaille.

2 Préliminaire

On commence par introduire la syntaxe, les définitions et constructions de bases nécessaires à la présentation de la procédure de décision. À travers ces notions, nous comprendront d'abord en quoi la synthèse de Church est une forme particulière d'uniformisation et ensuite en quoi les restrictions appliquées ramène notre problème vers celui de la synthèse de Church.

2.1 Notions de base

Alphabet Un alphabet Σ est un ensemble non vide. Les éléments de Σ sont appelés lettres, caractères ou symboles. Dans ce document, on considère que les alphabets sont finis, et on les notera Σ, Γ. Les lettres seront noté $\sigma, a, \alpha, b, \beta$. Usuellement, on ordonne un alphabet avec un ordre total lexicographique nommé lex. Pour tout alphabet Σ on aura parfois besoin d'ajouter un demi-symbole pour des constructions spécifiques $\Sigma_\diamond = \Sigma \cup \{\diamond\}$ et $\Sigma_\# = \Sigma \cup \{\#\}$.

Langage Un ensemble de mot sur un alphabet est appelé langage. En particulier l'ensemble de tout les mots sur un alphabet Σ est noté Σ^*.

Mot Un mot sur un alphabet Σ est une suite, éventuellement vide, de symboles appartenant à Σ. Le mot vide (l'absence de lettres) est noté ε. La longueur d'un mot $w \in \Sigma^*$ est le nombre de symboles qui le composent, noté $|w|$. Ainsi, $|\varepsilon| = 0$. Le domaine d'un mot w est défini par $dom(w) = \{1, \ldots, |w|\}$. En particulier, $dom(\varepsilon) = \varnothing$. Pour tout $i \in dom(w)$, i est appelé une position de w et $w(i)$ représente la i-ème lettre du mot w. Dans ce document, on considère que les mots sont finis, et on les notera u, v, w.

Concaténation Le concaténé de deux mots u et v de Σ^* est le mot obtenu en juxtaposant le mot v à la suite du mot u. On note $u \cdot v$ ou bien simplement uv le mot obtenu par concaténation. En particulier on a $\forall w \cdot w\varepsilon = \varepsilon w = w$.

Automate [9] Un automate fini $\mathcal{A} = (\Sigma, Q, i, F, \delta)$ est une forme de graphe. L'ensemble des états Q contient notamment, i l'état initial et F l'ensemble des états acceptants. Dans ce document, les automates seront déterministe. Ainsi, δ sera la fonction de transition (et non une relation) telle que $\delta : Q \times \Sigma \mapsto Q$ où Σ est l'alphabet de l'automate. On notera $q \xrightarrow{\sigma}_{\mathcal{A}} \delta(q, \sigma)$ la transition de $\mathcal{A}$ depuis un état q labellisé par σ. La sémantique d'un automate est une application de Σ^* dans $\{0, 1\}$. On utilisera les symboles $\mathcal{A}$ et $\mathcal{B}$ pour nommé des automates. L'ensemble des antécédents de 1 de la fonction décrite par l'automate $\mathcal{A}$ est le langage qu'il dénote, on le nomme usuellement $\mathcal{L}(\mathcal{A})$. Toutefois, lorsque le langage dénoté est une relation, on préfèrera utiliser $[\![\mathcal{A}]\!]$. Dans la suite du document $\mathcal{A}_q$ décrira l'automate construit à partir de $\mathcal{A}$, pour lequel on a placé q comme état initial et $\mathcal{A}^u$ sera l'intersection entre $\mathcal{A}$ et un automate auxiliaire qui contraint le mot consommé d'être préfixé par u. Notez que si $\mathcal{L}(\mathcal{A})$ est une relation, on s'assurera que l'alphabet du préfixe u permet d'identifier sans ambiguïté sur qu'elle(s) composante(s) il s'applique. Ainsi, $\mathcal{L}(\mathcal{A}) \subseteq \Sigma^* \times \Gamma^*$, pour tout $u, v \in \Sigma^*$, si u n'est pas préfixe de v alors $[\![\mathcal{A}^u]\!](v) = \varnothing$ sinon $[\![\mathcal{A}^u]\!](v) \subseteq \Gamma^*$.

Jeu [7] Un jeu $\eth = (\Sigma, \Gamma, Q_1, Q_2, i, \delta_1, \delta_2, F)$ est un automate où l'ensemble Q des états est partitionné en, Q_1 l'ensemble des nœuds contrôlés par le joueur **1** et Q_2 l'ensemble des nœuds contrôlés par le joueur **2**. La fonction de transition δ est l'union entre la fonction δ_1 du telle que $\delta_1 : Q_1 \times \Sigma \mapsto Q$ avec Σ l'alphabet du joueur **1** et la fonction δ_2 telle que $\delta_2 : Q_2 \times \Gamma \mapsto Q$ avec Γ l'alphabet du joueur **2**.

Transformation [12] Une transformation de mots finis sur deux alphabets Σ et Γ est une relation binaire $R \subseteq \Sigma^* \times \Gamma^*$ et peut-être dénotée par un transducteur. Par conséquent, les transformations sont une généralisation du concept de langages. Une transformation est dite *fonctionnelle* si R est une fonction. Dans ce document, les transformations dénoteront seulement des fonctions de Σ dans Γ. On notera $[\![\mathcal{T}]\!]$ pour désigner la fonction de transformation que défini le transducteur $\mathcal{T}$. Ainsi $[\![\mathcal{T}]\!](w)$ représente le résultat de la transformation du mot w par la fonction $[\![T]\!]$.

Machine alternante [4] Une machine alternante $\mathcal{M} = (\Sigma, Q_\exists Q_\forall, i, F, \Delta)$ est une machine de Turing où l'ensemble Q des états est partitionné en $Q_\exists$ l'ensemble des états existentiels et $Q_\forall$ l'ensemble des états universels. $\Delta \subseteq Q \times \Sigma \times Q \times \Sigma \times \{\leftarrow, \downarrow, \rightarrow\}$ est la relation de transition de la machine. Une configuration de la machine est un mot dans $(\Sigma \cup Q)^*$ qui contiennent exactement une lettre de Q. Un calcul pour une machine alternante est un arbre dont les nœuds sont étiquetés par les configurations de celle-ci et qui vérifie les conditions suivantes. Si un nœud s est étiqueté par une configuration $C = uqv$ où $q \in Q_\exists$ et $u, v \in (\Sigma^*)$ alors s doit avoir un seul fils qui est étiqueté par une des configurations C' tel que $C' \in \Delta(C)$. Si un nœud s est étiqueté par une configuration $C = uqv$ où $q \in Q_\forall$ et $u, v \in (\Sigma^*)$ alors s doit avoir un fils étiqueté pour chaque configuration C' tel que $C' \in \Delta(C)$. Le nœud s a donc autant de fils que de transitions applicables à la configuration C. Une branche du calcul est dite acceptante si une des configurations le long de la branche est acceptante. Le calcul sur une entrée $x \in \Sigma^*$ est acceptant si la configuration

à la racine est la configuration initiale $C_0 = ix$ et si toutes ses branches sont acceptantes.

2.2 Relations automatiques

Les relations automatiques vont être le point de départ de notre problème d'uniformisation. Il est important de comprendre la restriction qui est appliquée aux relations considérées. Alors que dans le problème classique de synthèse, il est question d'une relation MSO-définissable, dans notre cas, la procédure de décision prendra en entrée une relation automatique binaire exprimée par un automate fini.

Le principe des automates dénotant une relation automatique est de lire simultanément plusieurs mots synchrones (ici, deux) en travaillant sur des tuples de lettres. En d'autres termes, les mots sont lu lettre à lettre sans la possibilité de créer de décalage entre eux. Ainsi la k-ième lettre de chaque mots seront consommées par la même transition. Puisque l'on travaille avec des mots finis, pour deux mots en relation le plus cours des deux sera comblé par le demi-symbole '$\diamond$' afin que l'automate reçoive toujours un couple de lettre.

Définition 2.1. Plus formellement, on défini la relation $\otimes$ de complétion comme suit :

$$\forall u \in \Sigma^*, \forall v \in \Gamma^*, u \otimes v \subseteq (\Sigma_\diamond \times \Gamma_\diamond)^*$$

$$u \otimes v := \begin{cases} \begin{pmatrix} u(1) \\ v(1) \end{pmatrix} \cdots \begin{pmatrix} u(|u|) \\ v(|u|) \end{pmatrix} \begin{pmatrix} \diamond \\ v(|u|+1) \end{pmatrix} \cdots \begin{pmatrix} \diamond \\ v(|v|) \end{pmatrix} & \text{si } |u| < |v| \\ \begin{pmatrix} u(1) \\ v(1) \end{pmatrix} \cdots \begin{pmatrix} u(|v|) \\ v(|v|) \end{pmatrix} \begin{pmatrix} u(|v|+1) \\ \diamond \end{pmatrix} \cdots \begin{pmatrix} u(|u|) \\ \diamond \end{pmatrix} & \text{si } |u| > |v| \\ \begin{pmatrix} u(1) \\ v(1) \end{pmatrix} \cdots \begin{pmatrix} u(|u|) \\ v(|v|) \end{pmatrix} & \text{si } |u| = |v| \end{cases}$$

Pour tout langage $L \subseteq (\Sigma_\diamond \times \Gamma_\diamond)^*$, la relation $R \subseteq \Sigma^* \times \Gamma^*$ tel que $(u, v) \in R$ si est seulement si $u \otimes v \in L$, est dite automatique.

Exemple 2.1. L'automate suivant accepte les couples de mots de R_1 avec $\Sigma = \{1\}$ et $\Gamma = \{0, 1\}$. Notez que le premier mot peut être plus court que de second.

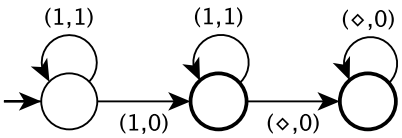

FIGURE 1 - $R_1 := \{(1^n, 1^m 0 1^k 0^*) \mid n = m + k + 1\}$

Remarque 2.1. La relation $\otimes$ est peut être dénoté par un automate fini comme le montre l'image suivante.

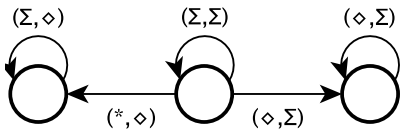

FIGURE 2 - Relation de complétion $\otimes$

2.3 Problème d'uniformisation

À présent, il est temps de formaliser le problème considéré. L'uniformisation est l'extraction d'une fonction à partir d'une relation. Le problème de Church est donc, en effet, une variante d'uniformisation car il ajoute le fait que cette dernière soit reconnaissable par une machine de Mealy. Dans notre cas, on souhaite que l'uniformisation soit reconnaissable par un transducteur subséquentiel dont la définition sera faite prochainement. Mais auparavant, on va expliquer comment uniformiser sans restriction.

Définition 2.2. L'uniformisation est une fonction qui, étant donné une relation R, retourne une fonction f_R telle que :

- $R \subseteq \Sigma^* \times \Gamma^*$
  $\Downarrow$
  $f_R : \Sigma^* \mapsto \Gamma^*$
- $\forall u \in \Sigma^* \cdot f_R(u) \in R(u)$

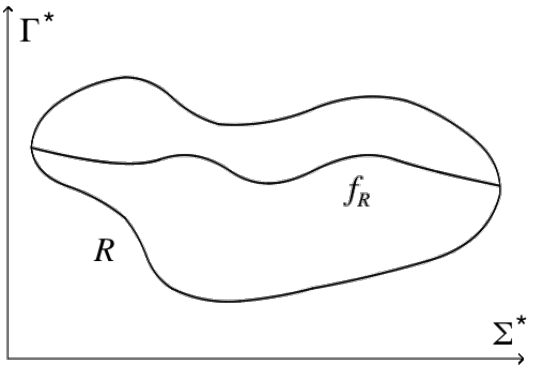

FIGURE 3 - Uniformisation

Une technique simple pour uniformiser, est de définir un ordre total bien fondé (sans suite infini décroissante) sur les éléments du domaine. Ainsi, uniformiser revient à choisir le plus petit v tel que $(u, v) \in R$, pour tout u.

Définition 2.3. L'ordre lenght-lexicographic (en abrégé $llex$) est un ordre total bien fondé qui ordonne les mots :

1. En fonction de leurs taille.

2. Lexicographiquement (si leur taille est identique).

Remarque 2.2. L'ordre length-lexicographic est une relation automatique, comme le montre l'automate $\mathcal{B}$ ci-dessous. Notez que $\Sigma = \Gamma$.

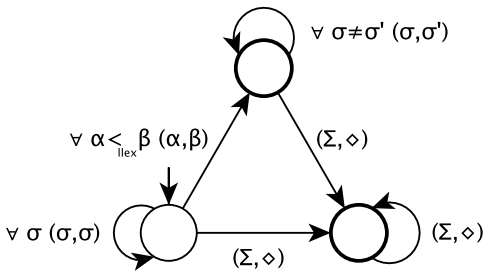

FIGURE 4 - $\mathcal{L}(\mathcal{B}) := \{(u, v) \mid u <_{llex} v\}$

Définition 2.4. On défini l'uniformisation length-lexicographic d'une relation R comme suit :

$$\forall u \in dom(R) \qquad f_R(u) := \min_{<_{llex}} \{v \in \Gamma^* \mid (u,v) \in R\}$$

Ainsi, l'uniformisation length-lexicographic de la relation dénoté par un automate $\mathcal{A}$ est la fonction automatique $f_{[\![\mathcal{A}]\!]}$ construite comme suit :

$$\begin{array}{ll}
\mathcal{L}(\mathcal{A}) = [\![\mathcal{A}]\!] \quad \& \quad \mathcal{L}(\mathcal{B}) = \{(u,v) \mid u <_{llex} v\} & \\
\mathcal{A} \wedge \mathcal{B} = \{(u,w,v) \mid w <_{llex} v \wedge (u,w) \in [\![\mathcal{A}]\!]\} & \text{Intersection} \\
\Pi_2(\mathcal{A} \wedge \mathcal{B}) = \{(u,v) \mid \exists w \cdot w <_{llex} v \wedge (u,w) \in [\![\mathcal{A}]\!]\} & \text{Projection} \\
\overline{\Pi_2(\mathcal{A} \wedge \mathcal{B})} = \{(u,v) \mid \forall w \cdot w <_{llex} v \Rightarrow (u,w) \notin [\![\mathcal{A}]\!]\} & \text{Négation} \\
\overline{\Pi_2(\mathcal{A} \wedge \mathcal{B})} \wedge \mathcal{A} = \{(u,v) \in [\![\mathcal{A}]\!] \mid \forall w \cdot w <_{llex} v \Rightarrow (u,w) \notin [\![\mathcal{A}]\!]\} & \text{Intersection}
\end{array}$$

$$f_{[\![\mathcal{A}]\!]} = \{(u,v) \in [\![\mathcal{A}]\!] \mid \forall w \cdot w <_{llex} v \Rightarrow (u,w) \notin [\![\mathcal{A}]\!]\}$$

Remarque 2.3. Notez qu'une projection suivit d'une négation impute à l'uniformisation un coût exponentiel par rapport au nombre d'état de l'automate initial, car la construction nécessite d'une déterminisation.

Exemple 2.2. La fonction automatique ci-dessous dénote l'uniformisation automatique length-lexicographic de la relation automatique R_1 présenté par la figure 1.

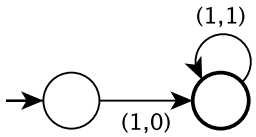

FIGURE 5 - $f_{R_1} = \{(1^n, 01^{n-1})\}$

2.4 Transducteurs subséquentiels

On a exprimé quelles sont les relations traitées et comment les uniformiser dans leurs propre classe. Cette partie, introduit la restriction sur l'uniformisation, autrement dit, le modèle de l'objet synthétisé. Les transducteurs subséquentiels ont été définie en 1977 par M. P. Schützenberger, comme un cas particulier de fonctions rationnelles. Leur expressivité y est détaillé dans le livre "*Transductions and context-free langages*" [1].

Les transducteurs subséquentiels, étendent les automates finis avec un mécanisme de sortie. Chaque fois qu'un transducteur subséquentiel lit un symbole d'entrée, il met à jour son état interne, écrit un mot de sortie partielle et passe au symbole d'entrée suivant. Si le transducteur termine dans un état acceptant, le résultat de son calcul est alors la concaténation, prise dans l'ordre, de tous les mots de sorties partielles produit lors du traitement du mot d'entrée, suffixé par la sortie final propre à l'état d'acceptation.

Définition 2.5. Un transducteur subséquentiel déterministe peut-être défini par un 8-uplet $\mathcal{T} = (\Sigma, \Gamma, Q, i, F, \delta, \lambda, f)$ tel que :

- Σ est l'alphabet d'entrée

- Γ est l'alphabet de sortie
- Q est l'ensemble fini des états
- i est l'état initial tel que $i \in Q$
- F est l'ensemble des états finaux tel que $F \subseteq Q$
- δ est la fonction de transition telle que $\delta : Q \times \Sigma \mapsto Q$
- λ est la fonction de sorti partielle telle que $\lambda : Q \times \Sigma \mapsto \Gamma^*$
- f est la fonction de sortie final telle que $f : F \mapsto \Gamma^*$

On notera $q \xrightarrow{\sigma|\lambda(\sigma)}_{\mathcal{T}} \delta(q, \sigma)$ pour dénoter la transition d'un transducteur dans l'état q en lisant la lettre σ.

Par induction sur la taille du mot, on peut étendre la fonction δ à la fonction $\overline{\delta} : Q \times \Sigma^* \mapsto Q$.

$$\forall w \in \Sigma^*, \forall q \in Q \qquad \overline{\delta}(q, w) := \begin{cases} q & \text{si } w = \varepsilon \\ \overline{\delta}(\delta(q, \sigma), w') & \text{avec } w = \sigma w' \end{cases}$$

De même que pour la fonction λ à la fonction $\overline{\lambda} : Q \times \Sigma^* \mapsto \Gamma^*$.

$$\forall w \in \Sigma^*, \forall q \in Q \qquad \overline{\lambda}(q, w) := \begin{cases} \varepsilon & \text{si } w = \varepsilon \\ \lambda(q, \sigma) \cdot \overline{\lambda}(\delta(q, \sigma), w') & \text{avec } w = \sigma w' \end{cases}$$

Ainsi la sémantique d'un transducteur subséquentiel $\mathcal{T}$ est la fonction partielle $[\![\mathcal{T}]\!]$ de Σ^* dans Γ^* suivante :

$$\forall w \in \Sigma^* \quad [\![\mathcal{T}]\!](w) := \overline{\lambda}(i, w) \cdot f(\overline{\delta}(i, w)) \in \Gamma^*$$

Exemple 2.3. La figure ci-près, présente la décomposition du mot retourné par le transducteur subséquentiel en fonction des transitions appliquées lors de la lecture du mot d'entrée.

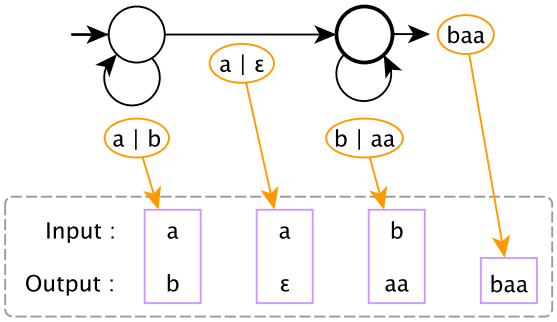

FIGURE 6 - Transducteur subséquentiel

2.5 Fonctions reconnaissables

Précédemment, on a décrit comment à partir d'un automate dénotant une relation, il est possible d'uniformiser la relation qu'il dénote. Maintenant, supposons que la fonction obtenu soit une *fonction reconnaissable*. Dans ce cas, la synthèse d'un transducteur subséquentiel est grandement simplifiée.

Définition 2.6. Une fonction dite reconnaissable, est une fonction de la forme :

$$\bigcup_{i=1}^{n}(U_i, v_i)$$

où $U_i \subseteq \Sigma^*$ sont des langages réguliers disjoint, $v_i \in \Gamma^*$ sont des mots et n est un entier.

Proposition 1. *Toute fonction reconnaissable peut être dénotée par un transducteur subséquentiel.*

Démonstration. Soit g une fonction reconnaissable, la synthèse d'un transducteur subséquentiel $\mathcal{T}_g$ tel que $[\![\mathcal{T}_g]\!] = g$ peut être décrite par récurrence sur n :

n = 0 Dans ce cas le langage est vide et la construction est triviale.

n > 0 Soit $g := g' \cup (U, v)$ avec $g' := \bigcup_{i=1}^{n-1}(U_i, v_i)$. On suppose que l'on a un transducteur subséquentiel $\mathcal{T}_{g'}$ qui dénote g' (i.e $[\![\mathcal{T}_{g'}]\!] = g'$). On commence par construire un automate $\mathcal{A}$ qui reconnait le langage régulier U. Puis à partir de $\mathcal{A}$ ont construit $\mathcal{T} = (\Sigma, \Gamma, Q_{\mathcal{A}}, i_{\mathcal{A}}, F_{\mathcal{A}}, \delta_{\mathcal{A}}, \lambda, f)$ un transducteur subséquentiel simplement en définissant λ et f par les fonctions constantes retournant respectivement ε et v. Enfin, ont construit $\mathcal{T}_g = \mathcal{T} \cup \mathcal{T}_{g'}$. Notez que $\mathcal{T}_g$ peut être déterminisé (puisque sa sortie est de taille bornée) et que sa fonction de sortie final n'est pas ambigüe (puisque g est une fonction).

□

Exemple 2.4. Soit la fonction reconnaissable g qui retourne la dernière lettre de son input sur l'alphabet $\Sigma = \Gamma = \{a, b\}$ est définie ainsi :

$$\forall w \in \Sigma^* \qquad g(w) := \begin{cases} \varepsilon & \text{si } w = \varepsilon \\ \sigma & \text{si } w = w'\sigma \end{cases}$$

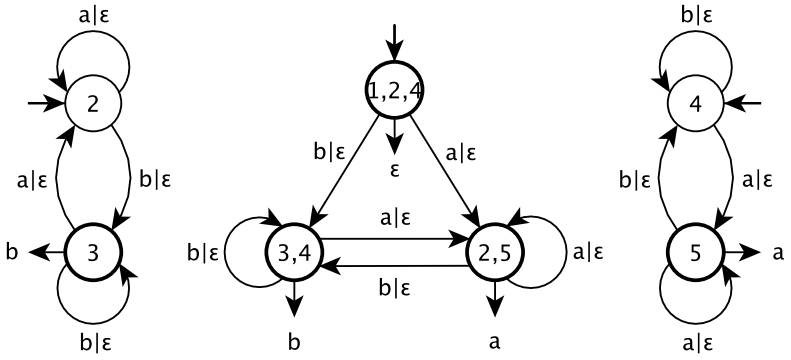

FIGURE 7 - $g = ((a+b)^*a, a) \cup ((a+b)^*b, b) \cup (\varepsilon, \varepsilon)$

Ainsi on construit trois transducteur subséquentiel, un pour chaque sortie possible. Cette image présente, à gauche le cas où la sortie vaut "b", à droite le cas où elle vaut "a" et au centre l'union des trois. Le transducteur subsquentiel du cas où l'entrée est vide ne figure pas car il est trivial mais on sous entend qu'il est décrit par le unique état 1.

Proposition 2. *Savoir si une relation automatique est uniformisable par une fonction reconnaissable est décidable.*

Démonstration. On montre que la relation automatique R est uniformisable par une fonction reconnaissable f si est seulement si l'uniformisation length-lexicographic automatique f_R a un codomaine V fini :

- Montrons que si le codomaine V de f_R est finis, alors R est uniformisable par f : On suppose que V est fini. Pour tout $v \in V$, on pose U_v l'ensemble des antécédents de v par la fonction f_R. Ainsi $f = \bigcup_{v \in V}(U_v, v)$.
- Montrons que si R est uniformisable par f, alors le codomaine de f_R est fini : On suppose $f = \bigcup_{i=1}^{n}(U_i, v_i)$. Soit $W := \{w \in \Gamma^* \mid \forall i \in \{1 \dots n\} \cdot w \leq_{llex} v_i\}$. Puisque l'ordre length-lexicographic est un ordre bien fondé, W est fini. Or, V est nécessairement inclus dans W car il représente le cas où les v_i sont minimaux.

□

Remarque 2.4. Notez que décider si une relation automatique est uniformisable par une fonction reconnaissable est exponentielle du fait de l'uniformisation length-lexicographic.

2.6 Du délai à l'anticipation

Cette dernière notion est essentielle à la compréhension du lien entre relation automatique et relation reconnaissable établie dans la procédure de décision. Forcé de constater que depuis un automate synchrone, on souhaite synthétiser un transducteur asynchrone, on en déduit qu'il va falloir permettre un certain décalage entre les lettres du premier mot et celles du second (qui sera maintenant perçu comme une réponse). La question serait donc de savoir comment choisir la valeur de ce délai.

À partir d'un certain seuil sur le délai, le Théorème de Ramsey [6] assure l'existence d'un facteur dit *idempotent* dans le mot formé par les lettres mises en attentes (les lettres anticipées). Le lemme 19 de l'article étudié démontre qu'en présence d'un tel facteur, soit l'uniformisation par transducteur subséquentiel est impossible, soit elle est très simple (à savoir, réduite à celle d'une relation reconnaissable). Ainsi, dès lors que cette "*borne sur le délai*" est atteinte on pourra, soit utiliser une boite noire (créé à partir de la construction décrite plus tôt), soit conclure que l'uniformisation par transducteur subséquentiel est impossible. Les terminologies délai et anticipation sont distincte mais décrivent la même intuition : L'anticipation est l'exercice du délai.

Définition 2.7. On commence par définir notre monoïde par la notion de profile. Le profile d'un mot u pour un automate $\mathcal{A}$ est $P_u = (P_{u,=}, P_{u,<}, P_{u,\varepsilon})$ avec :

- $P_{u,=} := \{(p,q) \in Q^2 \mid \exists v \in \Gamma^* \cdot |v| = |u| \wedge p \xrightarrow{u \otimes v}_{\mathcal{A}} q\}$
- $P_{u,<} := \{(p,q) \in Q^2 \mid \exists v \in \Gamma^* \cdot |v| < |u| \wedge p \xrightarrow{u \otimes v}_{\mathcal{A}} q\}$
- $P_{u,\varepsilon} := \{(p,q) \in Q^2 \mid p \xrightarrow{u \otimes \varepsilon}_{\mathcal{A}} q\}$

L'ensemble des profils est naturellement équipé de l'opération de concaténation, comme suit :

$$\forall u, u' \in \Sigma^* \qquad P_{uu'} := \{(s,t) \in Q^2 \mid \exists q \in Q \cdot (s,q) \in P_u \wedge (q,t) \in P_{u'}\}$$

Ainsi que d'un élément neutre (le profile du mot vide). Un mot $u \in \Sigma^*$ est appelé idempotent si $P_u = P_{uu}$.

Remarque 2.5. Une conséquence du Théorème de Ramsey est qu'il existe un certain $K \in \mathbb{N}$ tel que $\forall u \in \Sigma^* \wedge |u| \geq K$ implique que u contient un facteur idempotent. D'avantage de précision sera donné au moment de l'élicitation de la borne supérieure de K (qui sera donc notre délai).

Lemme 3. *Soit $q \in Q$ et $u, v \in \Sigma^+$ avec v idempotent. Si $[\![\mathcal{A}_q^{uv}]\!]$ est uniformisable par un transducteur subséquentiel $\mathcal{T}$ tel que $|\mathcal{T}(uv^n)| \leq |u|$ pour tout $n \in \mathbb{N}$, alors $[\![\mathcal{A}_q^{uv}]\!]$ peut être uniformisable par une fonction reconnaissable.*

Démonstration. Lemme 19 de la référence principale. □

3 Procédure de décision

On poursuit la progression en présentant la procédure tant cité, permettant de décider si une relation automatique peut être uniformisée par un transducteur subséquentiel. On commencera par comprendre comment la procédure exploite la théorie de jeux et en quoi ce type résolution permet d'abstraire le problème de façon élégante. Puis on intéressera à la génération du transducteur subséquentiel dans le cas où une telle uniformisation est possible.

3.1 Jeux de sureté avec anticipation

Le problème de Church fût introduit dans le cadre des jeux infinis pour première fois en 1965 par R. McNaughton [11]. Le jeux met en interaction deux parties **IN** et **OUT**. Les joueurs choisissent tour à tour des lettres et **IN** commence. Dans notre cas de figure, l'alternance ne sera pas symétrique car l'objectif est précisément d'introduire un délai. En effet, **OUT** pourra jouer aucune ou plusieurs lettres (mais au plus autant que **IN** en ait déjà joué) alors que **IN** jouera une unique lettre de (sauf si la borne du délai est atteinte auquel cas le jeu s'arrête). Ainsi, **OUT** peut anticiper un nombre fini de coup du joueur **IN**. Notez que puisque l'on considère des mots finis, il est nécessaire de munir l'alphabet de **IN** du demi-symbole '#' afin de décrire la terminaison de l'entrée.

À la fin du jeu, si le mot u produit par **IN** et le mot v répondu par **OUT** sont tels que $u \otimes v \in \mathcal{L}(\mathcal{A})$ alors **OUT** gagne, autrement la victoire revient à **IN**. À terme, la procédure doit trancher s'il la synthèse est possible et auquel cas générer un transducteur subséquentiel qui puisse répondre à n'importe quelle entrée. Ce sera le cas si et seulement si le joueur **OUT** à une stratégie lui permettant de toujours gagner. La résolution du jeux se présente donc sous la forme d'une stratégie de sureté pour **OUT**.

Définition 3.1. Formellement, le jeu $\eth_{\mathcal{A}}^{K}$ dépend d'un automate $\mathcal{A}$ (qui dénote une relation automatique $[\![\mathcal{A}]\!] \subseteq \Sigma^* \times \Gamma^*$) et K la taille minimale pour la présence inévitable d'un facteur idempotent (tel qu'il l'a été présenté en préliminaire). Les nœuds du jeu ont deux composante, la première étant l'état courant du calcul de l'automate et la seconde décrit la file de lettres jouée par **IN**, mises en attentes. Chaque nœud étant un couple, tel que $(q, u) \in Q$ avec $q \in Q_{\mathcal{A}}$ et $u \in \Sigma^*$, représente la relation $[\![\mathcal{A}_q^u]\!]$. Le jeu a donc :

- $Q_{in} := \{(q, u)_{in} \in Q_{\mathcal{A}} \times \Sigma^* \mid 2K \geq |u|\}$
- $Q_{out} := \{(q, u)_{out} \in Q_{\mathcal{A}} \times \Sigma_{\#}^* \mid 2K \geq |u|\}$
- $\delta_{in} := \{(q, u)_{in} \xrightarrow{\sigma}_{\eth} (q, u \cdot \sigma)_{out} \mid u \in \Sigma^*, |u| < 2K \wedge \forall \sigma \in \Sigma_{\#}\}$
- $\delta_{out} := \{(q, \sigma \cdot u)_{out} \xrightarrow{\gamma}_{\eth} (\delta(q, (\sigma, \gamma), u)_{out} \mid u \in \Sigma_{\#}^* \wedge \forall \sigma \in \Sigma \wedge \forall \gamma \in \Gamma\}$
  $\cup \{(q, u)_{out} \xrightarrow{\varepsilon}_{\eth} (q, u)_{in} \mid u \in \Sigma^*\}$
- $i = (i_{\mathcal{A}}, \varepsilon)_{in}$

Les nœuds qui ne sont pas sûr pour **OUT** sont ceux pour lesquels il ne puisse répondre sans sortir du langage dénoté par l'automate, ou bien ceux pour lesquels l'anticipation est saturée et la relation représentée n'est pas uniformisable par une fonction reconnaissable (d'après le lemme 3). Ainsi on a les trois cas suivant, $\forall (q, u) \in Q_{out}$:

- $|u| \leq 2K \wedge u \in \Sigma^* \wedge \mathcal{L}(\mathcal{A}_q^u) = \varnothing$
- $|u| \leq 2K \wedge \forall u' \in \Sigma^*, u = u' \cdot \# \wedge [\![\mathcal{A}_q]\!](u') = \varnothing$
- $|u| = 2K \wedge [\![\mathcal{A}_q^u]\!]$ ne peut être uniformisée par une fonction reconnaissable.

Afin de gagner synthétiquement, l'objectif pour **OUT** est de trouver une stratégie gagnante lui permettant de toujours rester sur des nœuds sain.

Remarque 3.1. Notez que tester chaque nœud pour décider s'il peut être uniformisé par une fonction reconnaissable est le coût le plus important de cette procédure. En effet, une uniformisation est faite sur chaque $[\![\mathcal{A}_q^u]\!]$ qui ont $|Q_{\mathcal{A}}| \times 2K$ états car $|u| = 2K$: Le paramètre K est donc en exposant.

Lemme 4. *La relation automatique $[\![\mathcal{A}]\!]$ peut être uniformisée par un transducteur subséquentiel si et seulement si,* ***OUT*** *a une stratégie gagnante pour le jeu $\eth_{\mathcal{A}}^{K}$.*

Démonstration. Lemme 21 de la référence principale. □

3.2 Synthèse du transducteur subséquentiel

On en vient au moment tant attendu, la synthèse du transducteur subséquentiel qui uniformise la relation automatique initiale. Ce dernier va bien évidemment être construit à partir du jeux qui à permis la décision, tout en exploitant la stratégie du joueur **OUT** comme sortie partielle ou finale. Plus exactement, cette stratégie va créer un chemin vers un nœud de V_{in} dans le cas où **IN** n'a pas joué $\#$, et autrement, elle va créer un chemin en direction d'une impasse. On va donc voir disparaitre les nœuds de V_{out}, laissant place à des transitions de sorties partielles et à des sorties finales.

On se place donc dans le cas où le joueur **OUT** a une stratégie gagnante pour le jeu $\eth_{\mathcal{A}}^{K}$. On peut supposer, sans perte de généralité [7], que cette stratégie dépend seulement du nœud courant. Soit $\pi : Q_{out} \mapsto \Gamma \cup \{\varepsilon\}$ la stratégie gagnante de **OUT**. Il y a trois cas de figure :

L'anticipation est saturée : Comme il a été dit plus tôt, lorsque la borne sur le délai est atteinte (et que la synthèse est possible) cela signifie que l'uniformisation est réduite à celle d'une relation reconnaissable. Grâce aux constructions décrites en préliminaire, il est simple d'uniformiser la relation $[\![\mathcal{A}_q^u]\!]$ par une fonction reconnaissable, puis de synthétiser un transducteur subséquentiel à partir de cette dernière. Ainsi, tout nœud de **IN** dont l'anticipation est saturée, vont être remplacé par le point d'entré d'une boite noire au moment de la génération. Toutes les explications qui suivent, concerne plus ou moins implicitement les nœuds autres que ceux des boites noire (cet ensemble sera noté $Q_{in}^{<2K}$). En résumé, les boites noires sont irrémédiablement inviolables.

IN joue une lettre : Si le joueur **IN** joue une lettre, l'itéré de la stratégie π de **OUT** va ramener le jeu vers un nœud de Q_{in}.

$$\forall \sigma \in \Sigma, \forall q, q' \in Q_{\mathcal{A}}, \forall u, u' \in \Sigma^* \qquad (q,u)_{in} \xrightarrow{\sigma}_{\eth} (q, u\cdot\sigma)_{out} \overset{\pi^*}{\rightsquigarrow}_{\eth} (q', u')_{in}$$

En effet, chaque transition de Q_{out} vers Q_{out} fait strictement décroitre la file de lettres anticipées en attentes. À terme, le joueur **OUT** est obligé de rendre le trait à **IN**.

IN joue $\#$: Si le joueur **IN** joue $\#$, l'itéré de la stratégie π de **OUT** va mener le jeu vers une impasse.

$$\forall \sigma \in \Sigma, \forall q, q' \in Q_{\mathcal{A}}, \forall u, u' \in \Sigma^* \qquad (q,u)_{in} \xrightarrow{\#}_{\eth} (q, u\cdot\sigma)_{out} \overset{\pi^*}{\rightsquigarrow}_{\eth} (q', \#)_{out}$$

Le joueur **OUT** ne peut pas rendre la main à **IN** si ce dernier à joué le demi-symbole $\#$, il est donc obligé d'épuiser la file de lettres en attentes.

Enfin, on construit $\mathcal{T} = (\Sigma, \Gamma, Q_{\mathcal{T}}, i_{\mathcal{T}}, F_{\mathcal{T}}, \delta_{\mathcal{T}}, \lambda_{\mathcal{T}}, f_{\mathcal{T}})$ le transducteur subséquentiel tel que $[\![\mathcal{T}]\!]$ soit l'uniformisation de $[\![\mathcal{A}]\!]$ comme suit :

- Σ et Γ depuis le départ restent inchangés. Ce sont les alphabets de la relation automatique et du jeu.
- L'ensemble $Q_{\mathcal{T}}$ va être composé des états du joueur **IN** (accessible) dans le jeux et des états des boites noires.

- Sans surprise, l'état initial de $\mathcal{T}$ sera le nœud initial du jeu, à savoir $i_{\mathcal{T}} = i = (i_{\mathcal{A}}, \varepsilon)$.
- Tout nœud de $Q_{in}^{<2K}$ a une transition sortante labellisée par $\#$. Et donc, tout nœud de $\mathcal{T}$ représentant un élément de $Q_{in}^{<2K}$ est acceptant. Bien sur certain états des boites noires peuvent aussi être dans $F_{\mathcal{T}}$.
- La fonction de transition $\delta_{\mathcal{T}}$ représente simplement la fonction qui renvoi la destination du chemin créé par l'itéré de la stratégie π. On commence par définir la fonction $\pi_\delta : Q \mapsto Q_{in}$ qui se contente de dérouler la stratégie jusqu'à atteindre un nœud de Q_{in}.

$$\forall q \in Q \quad \pi_\delta(q) = \begin{cases} q & \text{si } q \in Q_{in} \\ \pi_\delta(\delta(q, \pi(q))) & \text{si } q \in Q_{out} \end{cases}$$

$$\forall \sigma \in \Sigma, \forall q \in Q_{in}^{<2K} \subset Q_{\mathcal{T}} \qquad \delta_{\mathcal{T}}(q, \sigma) = \pi_\delta(\delta(q, \sigma))$$

- La fonction de sortie partielle $\lambda_{\mathcal{T}}$ représente simplement la fonction qui renvoi la concaténation des labels des transitions du chemin créé par l'itéré de la stratégie π. On commence par définir la fonction $\pi_\lambda : Q \mapsto \Gamma^*$ qui se contente de dérouler la stratégie jusqu'à atteindre un nœud de Q_{in}.

$$\forall q \in Q \quad \pi_\lambda(q) = \begin{cases} \varepsilon & \text{si } q \in Q_{in} \\ \pi(q) \cdot \pi_\lambda(\delta(q, \pi(q))) & \text{si } q \in Q_{out} \end{cases}$$

$$\forall \sigma \in \Sigma, \forall q \in Q_{in}^{<2K} \subset Q_{\mathcal{T}} \qquad \lambda_{\mathcal{T}}(q, \sigma) = \pi_\lambda(\delta(q, \sigma))$$

- La fonction de sortie final $f_{\mathcal{T}}$ est construite comme $\lambda_{\mathcal{T}}$ mais pour le cas particulier du demi-symbole de fin d'entré $\#$. On commence par définir la fonction $\pi_f : Q_{out} \mapsto \Gamma^*$ qui se contente de dérouler la stratégie jusqu'à atteindre une impasse.

$$\forall q \in Q_{out} \quad \pi_f(q) = \begin{cases} \varepsilon & \text{si } q = (q, \#)_{out} \\ \pi(q) \cdot \pi_f(\delta(q, \pi(q))) & \text{sinon} \end{cases}$$

$$\forall q \in F_{\mathcal{T}} \cap Q_{in}^{<2K} \qquad f_{\mathcal{T}}(q) = \pi_f(\delta(q, \#))$$

4 Complexité computationnelle

Jusqu'à lors, on a présenté la procédure de décision et compris quelle était exponentielle par rapport à K du fait des décisions d'uniformisation par fonction reconnaissable pour les nœuds où l'anticipation est saturée. Il est maintenant temps d'éliciter les bornes inférieure et supérieure de K, ainsi que de trouver la complexité minimale que pourrai avoir la procédure de décision.

4.1 Complexité du problème

On montre que la résolution d'un jeu de sureté avec délai est ExpTime-difficile par une réduction du problème d'acceptation d'une machine de Turing alternante polynômiale en espace. La preuve est en fait l'adaptation de résultats existants pour le cas des mots infinis [8], [10].

Théorème 5. *Soit un jeu de sureté $\eth$ et un délai fixé, décider si le joueur* ***OUT*** *à une stratégie gagnante est* ExpTime*-difficile.*

Démonstration. Soit $\mathcal{M}$ une machine de Turing alternante polynômiale en espace et p un polynôme qui délimite sa consommation en espace. La machine $\mathcal{M}$ est équipée d'un compteur de transition avec une représentation binaire. Lorsque le compteur de la machine atteint le nombre maximal de configuration différente possible (qui dépend de p), la machine s'arrête (ou plus exactement ne continue pas cette branche du calcul). Ainsi, la machine $\mathcal{M}$ s'arrête sur toute entrée $x \in \Sigma^*$.

On souhaite construire un jeu de sureté $\eth$ de taille polynômiale en $|\Delta_{\mathcal{M}}|$ et $p(|x|)$ tel que $\mathcal{M}$ rejette son entrée x si et seulement si le joueur **OUT** gagne avec une certaine anticipation. Dans ce jeu, **IN** contrôlera les positions existentielles, et **OUT** les positions universelles (l'inversion des rôles est dû au fait que l'on souhaite que x soit rejeté). Le joueur **IN** sera chargé de représenter les configurations de la machine $\mathcal{M}$ en les précédant d'un demi-symbole annonçant s'il s'agit d'une nouvelle configuration '$\ominus$' ou bien d'une copie '$\otimes$' (les demi-symboles étant ajoutés à son alphabet). Le principe des copies est simplement de combler le délai. Le joueur **OUT** quand à lui, sera chargé de fournir des transitions. Pour les positions universelles, **IN** devra appliquer la transition choisi par **OUT**, autrement, pour les positions existentielles **IN** jouera la transition qu'il souhaite (et celle de **OUT** sera ignorée). Toutefois, le joueur **OUT** aura aussi le rôle d'arbitre (dans un certain sens). Il aura les deux demi-symboles '$\top$' et '$\bot$' (ajoutés à son alphabet) qui lui permettront de revendiquer (ou pas) une erreur de la part de son adversaire.

Formellement, le jeu $\eth$ est construit comme le produit des sous-jeux suivant (un par règle), pour lesquels tout état qui ne soit pas un puits sera acceptant :

- Le mot u produit par le joueur **IN** doit être tel que $u \in (\{\ominus, \otimes\} \cdot C)^*$, où C est l'ensemble des codages de configurations de la machine $\mathcal{M}$ de longueur $p(|x|)$. Si le mot de **IN** n'est pas de cette forme, alors le jeu $\eth$ passe dans un puits acceptant, ce qui cause sa défaite.
- Le mot v produit par le joueur **OUT** doit être tel que $v \in (\Delta \cdot \{\bot, \top\}^{p(|x|+1)})^*$, où Δ est la relation de transition de la machine $\mathcal{M}$. Si le mot de **OUT** n'est pas de cette forme, alors le jeu $\eth$ passe dans un puits non-acceptant, ce qui cause sa défaite.
- Évidemment, il est important que la première configuration choisie par le joueur **IN** (puisqu'il commence), soit bien le configuration initiale de la machine $\mathcal{M}$ sur x. Si ce ne est pas le cas, le jeu passe dans un puits acceptant.
- Lorsque le joueur **OUT** utilise le symbole $\bot$, un sous-jeu va contrôler s'il y a réellement une erreur ou pas. Ce sous-jeu doit stocker, le type de position ($\forall$, $\exists$), le type de configuration ($\ominus$, $\otimes$) et la configuration précédente. Ceci est faisable en utilisant un ensemble d'états dont la taille est polynomiale en $|\Sigma \times Q \times \Sigma|$. Le contrôleur travail sur $p(|x|) + 1$ lettres (soit la taille de la configuration), et contrôle si la lettre litigieuse est correctement mise à jour ou non :

- Si le type de la configuration est $\ominus$, alors la lettre courante est correctement mise à jour si elle est égal à celle de la précédente configuration.
- Si le type de la configuration est $\otimes$, la mise à jour dépend du type de position :
  * Si la configuration dans laquelle l'erreur a été marquée est existentielle, alors la lettre est correctement mise à jour si elle est compatible avec une transition de la machine $\mathcal{M}$ depuis la configuration précédente.
  * Si la configuration dans laquelle l'erreur a été marquée est universelle, alors la lettre est correctement mise à jour si elle compatible avec la transition que le joueur **OUT** à dernièrement choisi. Dans le cas où, **OUT** à donné une mauvaise transition la mise à jour de **IN** est considéré comme correcte.

  Si la mise à jour est pas correct, alors **OUT** à revendiqué une erreur à juste titre et ainsi le jeu $\eth$ passe dans un puits rejetant (i.e **OUT** gagne). Autrement, **OUT** à menti, ainsi le jeu $\eth$ passe alors dans un puits non-acceptant (i.e **IN** gagne). Notez que si une erreur est revendiquée, l'issu du jeu est scellée et le reste des mots des deux joueurs seront sans effet.

- Enfin, dès lors que la machine $\mathcal{M}$ passe dans un état acceptant, le mot produit par le joueur **IN** contient la représentation d'un état acceptant. Dans ce cas, le jeu $\eth$ passe dans un puits non-acceptant car on souhaite que x soit rejeté par la machine $\mathcal{M}$.

Remarque 4.1. Notez que tous ces sous-jeux ont une taille polynomiale par rapport à $p(|x|)$ et Δ_M.

Montrons que si $\mathcal{M}$ rejette son entrée x, alors le joueur **OUT** a une stratégie gagnante avec une anticipation de taille $p(|x|) + 2$: On suppose que $\mathcal{M}$ rejette x. Avec une telle anticipation, le joueur **OUT** peut prévoir une configuration complète, et donc bloquer n'importe quelle éventuelle erreur. Le joueur **IN** ne peut donc pas tricher, autrement, **OUT** s'en rendrait compte et le dénoncerait. En outre, puisque la machine $\mathcal{M}$ rejette x alors, d'une part toutes les transitions sur des positions existentielle conduisent au rejet de x et d'autre part, il existe une transition sur chaque position universelle qui conduise au rejet de x. Ainsi, la stratégie pour **OUT** est de toujours choisir la transition qui rejette x, quelque soit les choix du joueur **IN**. Le joueur **OUT** a donc bien une stratégie gagnante pour le jeu $\eth$ avec un délai de taille $p(|x|) + 2$.

Montrons que si $\mathcal{M}$ accepte son entrée x, alors le joueur **IN** a une stratégie gagnante quelque soit la taille de l'anticipation : On suppose que $\mathcal{M}$ accepte x. Le joueur **IN** n'a pas besoin de tricher pour gagner puisque la machine $\mathcal{M}$ accepte x alors, d'une part toutes les transitions sur des positions universelle conduisent à l'acceptation de x et d'autre part, il existe une transition sur chaque position existentielle qui conduise à l'acceptation de x. Ainsi, la stratégie pour **IN** est de toujours choisir la transition qui accepte x, quelque soit les choix du joueur **OUT**. Le joueur **IN** a donc bien une stratégie gagnante pour le jeu $\eth$ avec un délai de taille quelconque.

On démontre ainsi que $\mathcal{M}$ rejette son entrée x si et seulement si le joueur **OUT** a une stratégie gagnante pour un certain délai fixé. Or, ASPACE = EXPTIME sont clos par complémentaire, ce qui permet de conclure. □

4.2 Borne inférieure sur le délai

Dans cette partie, on présente un résultat existant sur la borne inférieur du délai d'un jeu de sureté [10]. Cette borne s'appuie sur un jeu pour lequel **OUT** n'aura une stratégie gagnante que si son anticipation est exponentielle par rapport à l'alphabet alors que le nombre de nœud est linéaire par rapport à l'alphabet.

Définition 4.1. On commence par définir la notion de mauvaise paire $\sigma \in \Sigma$ dans un mot $u \in \Sigma^*$ comme suit :

$$\exists i, j \in dom(u) \cdot i < j \wedge u(i) = u(j) = \sigma \wedge \forall k \in dom(u) \cdot i < k < j \Rightarrow \sigma <_{lex} u(k)$$

Proposition 6. *Soit $w_{|\Sigma|} \in \Sigma^*$ le plus grand mot sans mauvaise paire, $|w_{|\Sigma|}| = 2^{|\Sigma|} - 1$.*

Démonstration. Par récurrence sur la taille de l'alphabet Σ :

$|\Sigma| = \mathbf{1}$ Deux lettres identiques accolées créent trivialement une mauvaise paire.

$|\Sigma| > \mathbf{1}$ On suppose que la propriété est vrai pour un alphabet de taille $|\Sigma|-1$. Il ne peut y avoir qu'une occurrence de la lettre $m \in \Sigma$ maximale (au sens lexicographique), l'ajout de cette lettre crée trivialement une mauvaise paire. On décompose $w_{|\Sigma|} = w_{|\Sigma|-1} \cdot m \cdot w_{|\Sigma|-1}$ avec $w_{|\Sigma|-1}$ sans occurrence de m. On peut alors appliquer l'hypothèse de récurrence sur $w_{|\Sigma|-1}$, ce qui implique que $|w_{|\Sigma|}| = 2^{|\Sigma|} - 1$. De plus, $w_{|\Sigma|}$ est le plus grand mot sans mauvaise paire, car l'ajout d'une lettre autre que m créerait une mauvaise avec $w_{|\Sigma|-1}$ par hypothèse de récurrence.

□

Théorème 7. *Il existe un jeu pour lequel la victoire du joueur **OUT** nécessite un délai exponentielle par rapport à la taille de ce jeu.*

Démonstration. On cherche un jeu tel que si une mauvaise paire apparait dans l'anticipation, l'état courant soit totalement sûre pour le joueur **OUT** (un puits acceptant). En revanche, si aucune mauvaise paire n'est apparut dans l'anticipation, l'état courant doit être perdant ou au pire instable (un état perdant est accessible), afin que le joueur **IN** puisse piéger **OUT** indépendamment de sa réponse. C'est ce que réalise l'automate ci-dessous en executant l'anticipation.

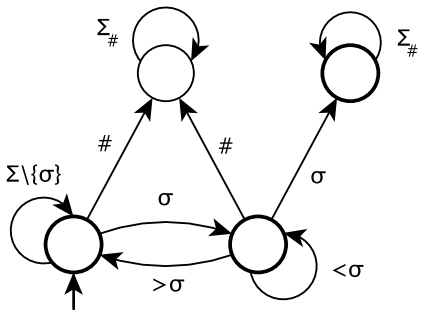

FIGURE 8 - Sous-automate sûr en présence d'une mauvaise paire $\sigma \in \Sigma$

Si une mauvaise paire σ apparait, l'état courant est sûr pour **OUT**. Autrement, **IN** n'a qu'à jouer le demi-symbole de piège # pour gagner. En construisant une grande union de tous ces types de sous-automates (pour toutes lettres $\sigma \in \Sigma$) tel que **OUT** en choisisse un (après avoir consulté son anticipation) en donnant quelle sera la mauvaise paire qui apparaitra. Si l'anticipation est au moins de $2^{|\Sigma|}$, alors une mauvaise paire apparait forcément (d'après la proposition 6) et **OUT** à une stratégie gagnante. Cependant, un détail reste à régler : Le cas où le joueur **IN** tente de piéger **OUT** au moment où celui-ci anticipe.

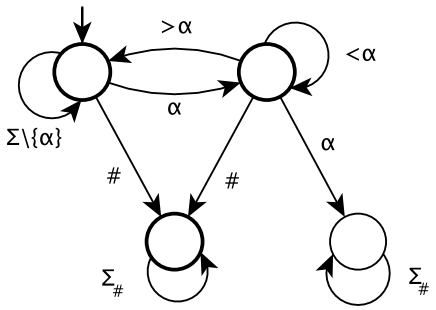

FIGURE 9 - Sous-automate sûr en présence de # avant une mauvaise paire α

Si **IN** utilise le demi-symbole de piège # en l'absence de toute mauvaise paire, et donc en particulier avant l'apparition d'une mauvaise paire $\alpha \in \Sigma$, alors en jouant le demi-symbole #, le joueur **OUT** choisira le sous-automate ci-dessus qui est totalement sûr dans ce cas particulier. Autrement la situation est inchangé pour **OUT**. Enfin, voici l'union qu'on l'on devrait construire pour obtenir le jeu complet :

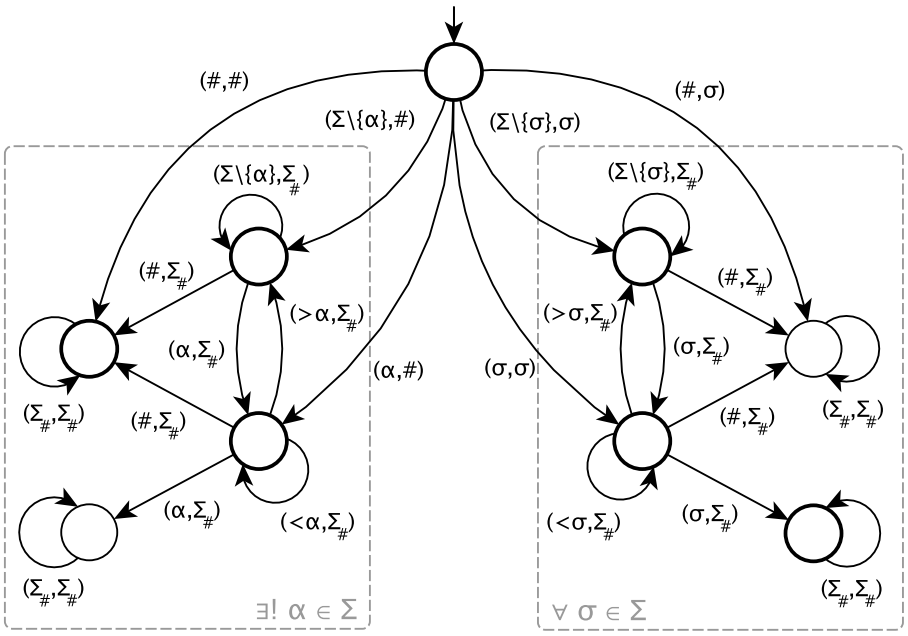

FIGURE 10 - Jeu construit à partir des sous-automates

Ainsi, **OUT** à une stratégie gagnante si son anticipation est au moins de taille $2^{|\Sigma|}$. En revanche, si l'anticipation a une taille strictement inférieur à $2^{|\Sigma|}$, le joueur **IN** peut la saturer avec le mot $w_{|\Sigma|}$. Dans le cas où **OUT** parie sur une mauvaise paire, **IN** termine par le symbole de piège #. Dans le cas où **OUT** répond par #, **IN** n'a cas terminer par α, ce qui créera une mauvaise paire (d'après la proposition 6) avant l'apparition d'un symbole #.

Remarque 4.2. Notez que le jeu décrit a une taille linéaire par rapport à l'alphabet Σ, alors que pour la victoire du joueur **OUT**, l'anticipation doit être exponentielle par rapport à Σ.

□

4.3 Borne supérieure sur le délai

Pour clore l'analyse de la complexité, on souhaiterait connaitre la taille d'un mot, pour qu'un facteur idempotent apparaisse inévitablement, à savoir, la valeur du paramètre K de la procédure de décision. Pour ce faire, on commence par représenter un mot quelconque $w \in \Sigma^*$ par un graphe, simplement en créant des nœuds avant et après chaque lettre (soit $|w|+1$ nœuds). Afin de représenter tous les facteurs du mot w, on lie chaque nœuds entre eux, obtenant ainsi un graphe complet.

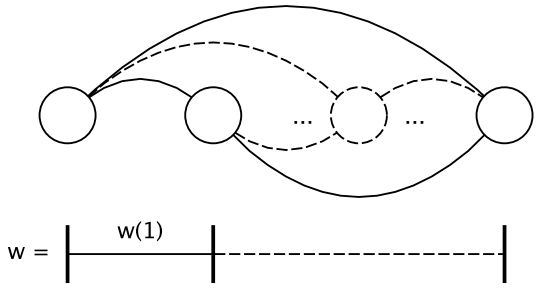

FIGURE 11 - Graphe complet à partir d'un mot

À présent, on cherche donc deux arc représentant deux facteurs u et v tel que $P_u = P_v = P_{uv}$. On va coloré tous les arcs en fonction du profile du facteur qu'il représente (soit $c = 2^{3\times|Q|^2}$ couleurs différentes). Ainsi, la présence d'une clique uni-coloré de taille trois sera synonyme à celle d'un facteur idempotent.

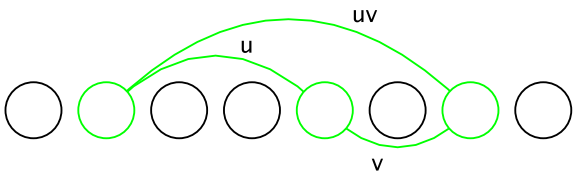

FIGURE 12 - Clique uni-colorée de taille trois ≡ facteur idempotent

Remarque 4.3. Le théorème de Ramsey assure la présence d'une borne sur le nombre de nœud nécessaire à l'apparition d'une clique uni-colorée de taille trois [6].

Théorème 8. *Un mot de taille* $\left(1 + \frac{1}{c-1}\right) \times c^c + \frac{1}{c-1}$ *à nécessairement un facteur idempotent (avec* $c = 2^{3\times|Q|^2}$*).*

Démonstration. On commence par définir la notion de couleur pour un nœud. Lorsque qu'un nœud prend une coloration, tous les nœuds suivant qui ne sont pas lié à lui par cette même couleur sont supprimé. Afin de minimiser la perte, un nœud choisira la couleur qui maximise sont nombre d'arcs sortants, comme le montre l'image suivante.

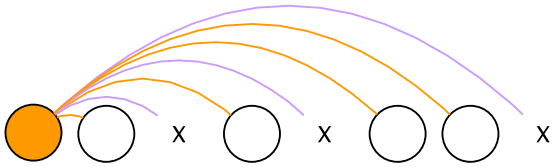

FIGURE 13 - Un nœud orange car il y a plus d'arc orange

Soit le motif, ci-dessous, oblige la présence d'une clique uni-colorée de taille trois. Le fait que le dernier nœud puisse être indifféremment coloré, et qu'un nœud de chaque couleur apparait plutôt, quelque soit la couleur du pénultième une clique uni-colorée de taille trois apparaitra.

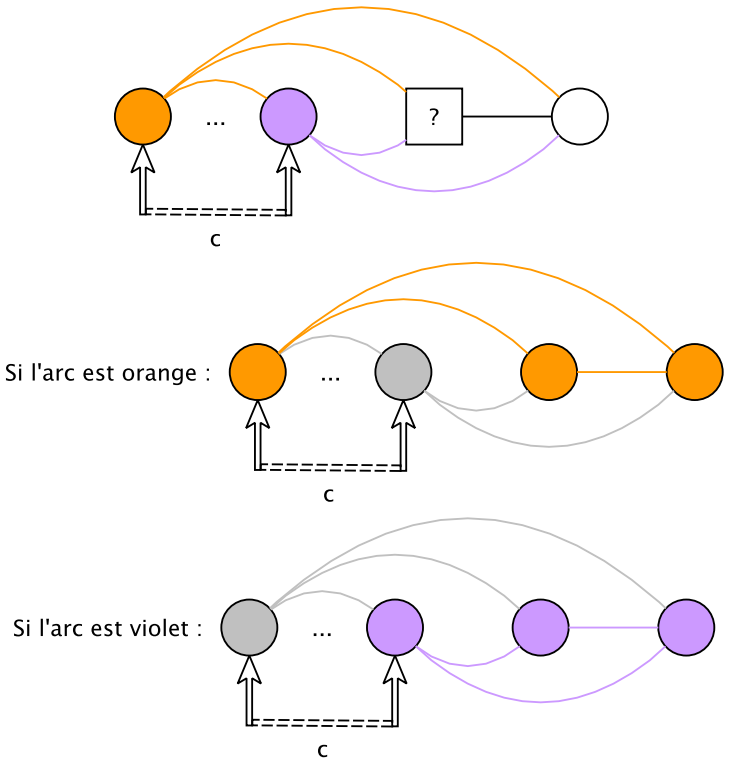

FIGURE 14 - Motif obligeant la présence d'une clique uni-colorée de taille trois

En remontant les étapes de coloration des c nœuds, on sera en mesure de connaitre le nombre nécessaire de nœuds pour l'apparition du motif présenté

et donc pour celui d'une clique uni-colorée de taille trois. Notez que le cas où toutes les couleurs apparaissent est le pire qui puisse arriver. En effet, si un nœud prend une coloration déjà apparue alors une clique uni-coloré de taille trois se produit instantanément (dans la mesure où il reste au moins un nœud supplémentaire non encore coloré).

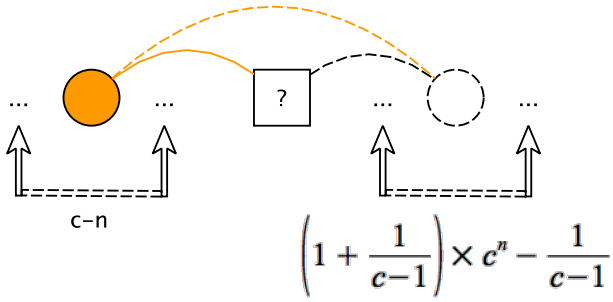

FIGURE 15 - n itération antérieures

Soit n étape plutôt, il n'y a que $c-n$ nœuds colorés, un nœud en attente de coloration et un reste. Parmi ce reste, il y a notamment les nœuds qui formeront à terme le motif mais pas seulement. Il y a aussi tous les nœuds qui seront supprimé lors de la coloration des n nœuds colorés manquant. Il est intéressant de savoir combien de nœuds représente ce reste, car lorsque $n = c$ on pourra immédiatement conclure.

Le reste peut en fait être perçu comme une suite récurrente U. Initialement (dans le motif de la figure 14), on a $U_0 = 1$. Étant donné, un nœud coloré, un nœud en attente de coloration et le reste U_n, on retrouve U_{n+1} comme suit. Le nœud coloré a gardé un maximum de nœud en sélectionnant la couleur qui minimisait la perte. Mais dans le pire des cas, un seul nœud permet de trancher (toutes les couleur ont autant de nœud sauf une qui en a un de plus). Notez que si aucun nœud ne permet de trancher, soit on peut choisir une couleur déjà à apparu (parmi les $c-n$), soit on introduit un nœud inutile puisqu'un choix arbitraire le supprimera (au moment de la coloration). On obtient donc le reste $U_{n+1} = c \times U_n + 1$. Le nœud en attente de coloration étant celui qui a permis de trancher. La dé-récursion de cette suite (en annexe) produit la formule de l'image ci-dessus.

Si maintenant $n = c$, il n'y a plus qu'un noeud en attente de coloration et le reste U_c. Toutefois, il y a un nœud de plus dans le graphe complet que de lettre dans le mot d'origine. Ainsi, la taille d'un mot pour qu'un facteur idempotent apparaisse inévitablement est de :

$$\left(1+\frac{1}{c-1}\right)\times c^c+\frac{1}{c-1}$$

□

5 Conclusion

Il est donc démontré que la procédure d'uniformisation des relations automatiques par transducteur subséquentiel est doublement exponentielle. Toutefois,

la borne supérieur pourrait peut-être être meilleur que $\mathcal{O}\left((|Q|^2)^{(|Q|^2)}\right)$ si la construction ne dépendait pas des facteurs idempotent. En effet, si le lemme 19 de l'article de référence (lemme 3 ici) utilisait un argument d'itération (de pompage) alors la borne serait $2^{|Q|}$. Ceci correspond en fait à la taille minimale pour la partition d'une boucle dans la déterminisation de la projection (sur la première composante) de l'automate de base. Ainsi, si le joueur **IN** à une stratégie gagnante pour un délai de taille $2^{|Q|}$, la stratégie pour un délai plus grand consiste à itérer cette boucle suffisamment pour ne pas donner à **OUT** plus d'information qu'il n'en aurait avec un délai de taille $2^{|Q|}$. Malheureusement, je n'ai pas trouver une telle alternative au lemme 19.

Une direction pour de future travaux serait l'uniformisation des relations non automatique sur des arbres finis. En effet, l'uniformisation pour des arbres infinis peut échoué (comme il est démontré dans la référence principale). Toutefois, la question d'uniformisation des relations rationnelles sur les arbres finis mériterait d'être étudiés car il existe différents modèles de transducteur qui les définissent.

Références

[1] J. Berstel. *Transductions and context-free languages.* Teubner Verlag, 1979.

[2] J. R. Büchi and L. Landweber. Solving sequential conditions by finite-state strategies. *Transactions of the American Mathematical Society, 138, 295–311*, 1969.

[3] A. Carayol and C. Löding. Uniformisation in automata theory. In *Logic, Methodology and Philosophy of Science - Proceedings of the 14th Internatinal Congress, Nancy, 2011.* College Publications, 2015.

[4] A. K. Chandra, D. Kozen, and L. J. Stockmeyer. Alternation. volume 28, pages 114–133. 1981.

[5] A. Church. Applications of recursive arithmetic to the problem of circuit synthesis. In *Summaries of the Summer Institute of Symbolic Logic*, volume 1, pages 3–50. Ithaca, N.Y., 1957.

[6] R. Diestel. *Graph Theory.* Springer, 2000.

[7] E. Grädel, W. Thomas, and T. Wilke. Automata, logic and infinite games. In *Lecture Notes in Computer Science*, volume 2500. Springer, 2002.

[8] M. Holtmann, Ł. Kaiser, and W. Thomas. Degrees of lookahead in regular infinite games. 2010.

[9] J. E. Hopcroft and J. D. Ullman. *Introduction to automata theory, langage and computation.* Addison-Wesley, 1979.

[10] F. Klein and M. Zimmermann. How much lookahead is needed to win infinite games ? In *Automata, Languages, and Programming*, volume 9135 of *Lecture Notes in Computer Science*, pages 452–463. Springer Berlin Heidelberg, 2015.

[11] R. McNaughton. Finite-state infinite games. *Project MAC Rep*, 1965.

[12] J. Sakarovitch. *Élements de théorie des automates*. Vuibert, 2003.

[13] W. Thomas. Facets of synthesis : Revisiting church's problem. In *Proceedings of the 12th international conference on foundations of software science and computational structures*, volume 5504 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2009.

Annexe

On veux dé-récursiver la suite suivante :

$$\begin{aligned} U_{n+1} &= c \times U_n + 1 \\ U_0 &= 1 \end{aligned}$$

Soit la suite auxiliaire :

$$V_n = U_n + \frac{1}{c-1}$$

v_n est une suite géométrique car :

$$\frac{V_{n+1}}{V_n} = \frac{U_{n+1} + \frac{1}{c-1}}{U_n + \frac{1}{c-1}} = \frac{c \times U_n + \frac{c}{c-1}}{U_n + \frac{1}{c-1}} = \frac{c \times \left(U_n + \frac{1}{c-1}\right)}{U_n + \frac{c}{c-1}}$$

Ainsi :

$$\begin{aligned} V_n &= V_0 \times c^n \\ U_n &= \left(1 + \frac{1}{c-1}\right) \times c^n - \frac{1}{c-1} \end{aligned}$$