Amélioration de la procédure de déterminisation des automates (max,+)

Sébastien Lahaye[1], Jan Komenda[2]*, and Jean-Louis Boimond[1]

[1] Université d'Angers, France
sebastien.lahaye@univ-angers.fr, jean-louis.boimond@univ-angers.fr
[2] Institute of Mathematics - Brno Branch, Czech Academy of Sciences, Czech Republic
komenda@ipm.cz

Abstract

Une classe importante de systèmes à événements discrets peut être modélisée à l'aide d'automates (max,+) et les études utilisant ce formalisme ont notamment contribué à l'évaluation de performances et la commande supervisée de ces systèmes. Pour ces résultats, la propriété de déterminisme des automates manipulés est souvent prépondérante. Or, contrairement aux automates logiques, tous les automates (max,+) ne peuvent pas être déterminisés, c'est-à-dire transformés en un automate (max,+) déterministe ayant le même comportement. Une généralisation aux automates (max,+) de la procédure classique de séquentialisation a tout de même été intensivement étudiée et celle-ci se termine avec succès pour des classes importantes d'automates. Cette procédure utilise une condition sur la normalisation des vecteurs d'état pour détecter et fusionner les états engendrant un identique comportement ultérieur. Dans cette contribution, on identifie une nouvelle condition garantissant cette propriété. Cela nous permet d'enrichir la procédure de déterminisation de sorte qu'elle aboutisse pour une classe plus large d'automates (max,+).

1 Introduction

Les automates à multiplicités dans l'algèbre (max,+), usuellement appelés automates (max,+), permettent d'appréhender une large classe de systèmes à événements discrets. Il a en effet été mis en évidence que les automates (max,+) permettent de modéliser certains systèmes temporisés mettant en jeu à la fois des phénomènes de synchronisation et de partage de ressources (voir par exemple [6, 8, 13]). Des études basées sur ce formalisme ont en particulier contribué significativement à l'évaluation de performances et la commande supervisée [6, 12, 4]. Pour ces résultats, la propriété de déterminisme des automates manipulés est souvent décisive. Cette propriété est par exemple requise pour pouvoir évaluer avec une complexité raisonnable le temps d'exécution minimum de séquences [17], ou encore pour être sûr d'obtenir des superviseurs réalisables [2]. Malheureusement, il a été montré que, contrairement aux automates logiques, tous les automates (max,+) ne peuvent pas être déterminisés, c'est-à-dire transformés en un automate (max,+) déterministe ayant le même comportement (voir par exemple [14]). De nombreux travaux ont quand même contribué à généraliser aux automates (max,+) la procédure classique de séquentialisation des automates (sans être exhaustif, citons [6, 15, 11, 14, 16]). À notre connaissance, toutes les études existantes considèrent une procédure de déterminisation employant une condition sur la normalisation du vecteur d'état pour identifier et fusionner des états engendrant d'identiques comportements ultérieurs. Les automates (max,+) satisfaisant la propriété de clonage [10] constituent, sauf erreur de notre part, la classe la plus

*Ce travail a été soutenu par GAČR projet no. 15-02532S et par RVO 67985840

large d'automates identifiés comme étant déterminisables *via* cette procédure. Dans cette contribution, on présente une nouvelle condition permettant de reconnaître des états dont les comportements postérieurs sont similaires. On montre ensuite comment cette condition peut être utilisée pour enrichir la procédure de déterminisation de sorte qu'elle aboutisse pour un plus grand nombre d'automates.

L'article est organisé comme suit. La deuxième section contient des rappels sur les notions utilisées. La procédure existante de déterminisation des automates (max,+) est présentée dans la troisième section. La quatrième section est dédiée à notre contribution pour améliorer cette procédure. Il s'ensuit une conclusion où sont évoquées les perspectives ouvertes par ce travail.

2 Rappels préliminaires

Dans cette section, on rappelle brièvement les notations, concepts et résultats utiles pour la suite du papier.

2.1 Algèbre (max,+)

On renvoie le lecteur aux livres [1, 9] ou à la thèse [5] pour une présentation exhaustive sur l'algèbre (max,+).

Définition 1 (Dioïde)**.** *Un* dioïde $\mathcal{D}$ *est un* semi-anneau *dans lequel l'addition* $\oplus$ *est idempotente. L'addition (resp., la multiplication* $\otimes$*) admet un élément nul* ε *(resp., un élément unité* e*).*

S'il existe, on note x^{-1} l'inverse de x, c.-à-d. l'élement de $\mathcal{D}$ tel que $x^{-1} \otimes x = x \otimes x^{-1} = e$.

L'ensemble $\mathbb{R} \cup \{-\infty\}$ avec l'opération maximum jouant le rôle d'addition et l'addition usuelle jouant le rôle de multiplication est un dioïde, noté $\mathbb{R}_{\max}$ et souvent appelé *algèbre (max,+)* (avec $e = 0$ et $\varepsilon = -\infty$). Pour $x \in \mathbb{R}_{\max}$ et $x \neq \varepsilon$, x^{-1} est égal à $-x$.
L'ensemble des matrices $n \times n$ à coefficients dans $\mathbb{R}_{\max}$, muni de l'addition et de la multiplication matricielles définies de façon conventionnelle à partir de $\oplus$ et $\otimes$, est également un dioïde, noté $\mathbb{R}_{\max}^{n\times n}$. L'élément nul est la matrice composée exclusivement de ε $(= -\infty)$ et également notée ε. On note I_n l'élément identité qui est la matrice avec des e $(= 0)$ sur la diagonale et ε $(= -\infty)$ partout ailleurs. La puissance k-ième d'une matrice (max,+) carrée est notée A^k. En toute rigueur, les opérations entre matrices de tailles compatibles requièrent de plonger ces matrices dans un dioïde de matrices carrées. Par exemple le produit $A \otimes B$ avec $A \in \mathbb{R}_{\max}^{m\times n}$, $B \in \mathbb{R}_{\max}^{n\times p}$ et $n > m$, $n > p$ nécessite de projeter A et B dans $\mathbb{R}_{\max}^{n\times n}$ en ajoutant $n - m$ lignes (resp. $n - p$ colonnes) remplies de ε dans A (resp. dans B). Pour alléger la présentation, cette manipulation sera souvent omise (sans affecter les résultats), en outre les coefficients égaux à ε dans les matrices seront remplacés par '$\cdot$'.

2.2 Mots et langages

Si Σ est un ensemble fini (*alphabet*), le *monoïde libre* sur Σ est défini comme l'ensemble Σ^* de mots finis formés à partir de lettres dans Σ. Un *mot* $w \in \Sigma^*$ s'écrit comme une séquence $w = a_1 a_2 \dots a_p$ avec a_1, a_2, …, $a_p \in \Sigma$ et p un entier naturel (w est la concaténation de a_1, a_2, …, a_p). Le mot vide est noté ϵ.

Un mot u est dit être un *préfixe* de w s'il existe un mot v tel que $w = uv$. On note Prefix(w) l'ensemble des préfixes de w. Les *langages formels* sont des sous-ensembles du monoïde libre Σ^*.

2.3 Automates (max,+)

On introduit maintenant les automates (max,+), ainsi que plusieurs résultats bien connus sur ces objets. Les notations sont autant que possible cohérentes avec celles des références [6] et [10].

Définition 2 (Automate (max,+))**.** *Un* automate (max,+) *sur l'alphabet* Σ *est un triplet* $G = (Q, \alpha, \mu)$ *où*

- Q *est un ensemble fini non vide d'états ;*
- $\alpha \in \{e, \varepsilon\}^{1\times|Q|}$*;*
- $\mu : \Sigma^* \to \mathbb{R}_{\max}^{|Q|\times|Q|}$ *est un morphisme spécifié par la familles de matrices* $\mu(a) \in \mathbb{R}_{\max}^{|Q|\times|Q|}$*,* $a \in \Sigma$*, et pour un mot* $w = a_1 a_2 \dots a_n$*, nous avons*

$$\mu(w) = \mu(a_1 a_2 \dots a_n) = \mu(a_1) \otimes \mu(a_2) \otimes \dots \otimes \mu(a_n).$$

De façon équivalente G *peut être défini par le triplet* (Q, Q_i, t)*, dans lequel* Q_i *est l'ensemble des états* initiaux *et* $t : Q \times \Sigma \times Q \to \mathbb{R}_{\max}$ *est la* fonction de transition *:*

$$Q_i \triangleq \{q \in Q : \alpha_q = e\}, \quad t(q, a, q') \triangleq \mu(a)_{qq'}.$$

Remarque 1 À la différence de la définition dans [6], on considère des automates (max,+) avec des poids initiaux nuls (α_q est égale à $e = 0$ si q est un état initial). Cette hypothèse est sans perte de généralité car un automate (max,+) avec des poids initiaux non nuls peut aisément être transformé en un automate conforme à la définition 2. De plus, tous les états sont ici considérés comme finaux. ◇

On peut donner une représentation graphique à un automate (max,+) (alors vu comme un multigraphe valué) :

- à chaque état $q \in Q$ correspond un sommet ;
- un arc de l'état q à l'état q' existe s'il existe une lettre $a \in \Sigma$ telle que $\mu(a)_{qq'} \neq \varepsilon$ et il est alors étiqueté par $a/\mu(a)_{qq'}$;
- un arc entrant (sans sommet de départ, et qui, en toute rigueur, devrait être étiqueté par ϵ/e) est utilisé pour indiquer un état initial.

Un coefficient $[\mu(a)]_{qq'} \neq \varepsilon$ signifie que depuis l'état q l'occurrence de la lettre a cause une transition d'état vers l'état q'. Dans un automate (max,+), le poids $[\mu(a)]_{qq'}$ peut être interprété comme une temporisation associée à l'événement symbolisé par la lettre a (dans ce cas, le temps d'activation de l'événement a avant qu'il puisse se produire).

Si $[\mu(a)]_{qq'} \neq \varepsilon$, on note alors (q, a, q') la *transition* dans G. Soit $m \geq 1$ et $\pi = (q_0, a_1, q_1)$ (q_1, a_2, q_2) $\dots$ (q_{m-1}, a_m, q_m) une séquence de transitions. On appelle π un *chemin* depuis q_0 jusqu'à q_m. On note $\sigma(\pi)$ le produit $\otimes$ des poids sur π, c'est-à-dire

$$\sigma(\pi) = \bigotimes_{i=1,\dots,m} t(q_{i-1}, a_i, q_i) = \bigotimes_{i=1,\dots,m} \mu(a_i)_{q_{i-1},q_i}.$$

Soient $p, q \in Q$ et $w \in \Sigma^*$. On note $p \overset{w}{\rightsquigarrow} q$ l'ensemble des chemins depuis p jusqu'à q et étiquetés par le mot w. On a

$$\mu(a_1 a_2 \dots a_m)_{q_0 q_m} = \bigoplus_{\pi \in q_0 \overset{a_1 \dots a_m}{\rightsquigarrow} q_m} \sigma(\pi) \ . \tag{1}$$

On utilise souvent par la suite la notation x (pour faire référence à l'*état* de l'automate) défini par

$$\begin{array}{rcl} x & : & \Sigma^* \to \mathbb{R}_{\max}^{1\times Q} \\ & & w \mapsto x(w) = \alpha\mu(w). \end{array} \tag{2}$$

Il est connu que x est solution du système d'équations suivant :

$$\left\{ \begin{array}{rcl} x(\epsilon) & = & \alpha \\ x(wa) & = & x(w)\mu(a) \end{array} \right. \tag{3}$$

Définition 3 (Comportement d'un automate (max,+))**.** *À un automate (max,+) G, on associe l'application y :*

$$\begin{array}{rcl} y & : & \Sigma^* \to \mathbb{R}_{\max} \\ & & w \mapsto y(w) = \bigoplus_{q\in Q} x(w)_q \ . \end{array} \tag{4}$$

L'application y est souvent appelée le comportement *de G.*

Il est facile de vérifier que

$$y(w) = \bigoplus_{\pi\in Q_i \overset{w}{\rightsquigarrow} Q} \sigma(\pi).$$

On note $S(w)$ l'ensemble *des états gagnants pour* w, c'est-à-dire l'ensemble des états pour lesquels la valeur d'état est égale à celle du comportement pour w:

$$S(w) \triangleq \{q \in Q | x(w)_q = \bigoplus_{p\in Q} x(w)_p = y(w)\}. \tag{5}$$

On note $R(w)$ l'ensemble des états atteignables pour w :

$$R(w) \triangleq \{q \in Q | x(w)_q \neq \varepsilon\}. \tag{6}$$

Soit p un état atteignable pour w, on note $R(wa, p)$ le sous-ensemble de $R(wa)$ composés des états atteignables depuis p *via* une transition étiquetée par a:

$$R(wa, p) \triangleq \{q \in Q | \mu(a)_{pq} \neq \varepsilon \text{ et } x(w)_p \neq \varepsilon\}. \tag{7}$$

On dit que $q \in Q$ est *accessible* s'il existe $w \in \Sigma^*$ tel que $R(w) \neq \emptyset$. Puisque tous les états sont supposés finaux, tout état est co-accessible, et si tout état est accessible alors G est dit *émondé.*

Définition 4 (Automate (max,+) équivalent)**.** *Deux automates (max,+) sont* équivalents *s'ils admettent le même comportement.*

Définition 5 (Automate (max,+) déterministe)**.** *Un automate (max,+) est dit* déterministe *si*

- *il a un unique état initial (Q_i est un singleton, ou de façon équivalente, il existe un unique $q \in Q$ tel que $\alpha_q \neq \varepsilon$) ;*
- *depuis chaque état, il n'existe pas deux transitions étiquetées par la même lettre (pour tout $a \in A$ chaque ligne de $\mu(a)$ contient au plus un coefficient différent de ε).*

Pour un automate (max,+) déterministe, $\forall q, q' \in Q, w \in \Sigma^*, q \overset{w}{\rightsquigarrow} q'$ est l'ensemble vide ou un singleton. Pour $(q_0, a_1, q_1)(q_1, a_2, q_2)\dots(q_{m-1}, a_m, q_m)$ l'unique chemin reconnaissant $a_1a_2\dots a_m$ depuis q_0 jusqu'à q_m, l'équation (1) se réduit à

$$\mu(a_1a_2\dots a_m)_{q_0,q_m} = \bigotimes_{i=1\dots m} \mu(a_i)_{q_{i-1},q_i} = \bigotimes_{i=1\dots m} t(q_{i-1}, a_i, q_i).$$

On a alors avec q_0 l'état initial

$$y(a_1a_2\dots a_m) = \bigotimes_{i=1\dots m} \mu(a_i)_{q_{i-1},q_i} = \bigotimes_{i=1\dots m} t(q_{i-1},a_i,q_i). \tag{8}$$

La notion d'ambiguité permet de distinguer certaines situations d'indéterminisme.

Définition 6 (Automate (max,+) ambigu)**.** *Pour $R,R' \subseteq Q$, on note $R \overset{w}{\rightsquigarrow} R'$ l'union des $r \overset{w}{\rightsquigarrow} r'$ pour tous les $r \in R$, $r' \in R'$.*
Soit $k \geq 1$.

a) Si pour tout $w \in \Sigma^*$ il y a au plus k chemins dans $Q_i \overset{w}{\rightsquigarrow} Q$, alors G est dit k-ambigu.

b) Si G est 1-ambigu, alors G est dit non ambigu.

c) Soit $P : \mathbb{N} \to \mathbb{N}$ une fonction polynomiale. Si pour tout $w \in \Sigma^*$, il y a au plus $P(|w|)$ chemins dans $Q_i \overset{w}{\rightsquigarrow} Q$, alors G est dit polynomialement ambigu.

Remarque 2

- Un automate déterministe est non ambigu.
- Des définitions différentes peuvent être rencontrées dans la littérature pour la notion d'ambiguité. De façon courante, un automate est dit non ambigu s'il y a au plus un chemin pour un état d'origine, un état d'arrivée et un mot donnés (comme par exemple dans [3]), c'est-à-dire $\forall p,q \in Q, \forall w \in \Sigma^*; |p \overset{w}{\rightsquigarrow} q| \leq 1$. Alors que dans [11, 10] un automate est qualifié de non ambigu s'il y a au plus un chemin reconnaissant tout mot, c'est-à-dire $\forall w \in \Sigma^*; |Q_i \overset{w}{\rightsquigarrow} Q| \leq 1$. Cette dernière définition est plus restrictive mais c'est celle que nous adoptons pour pouvoir faire référence aux résultats de [11, 10].

◇

Le théorème suivant (tiré de [10]) permet de caractériser les automates (max,+) polynomialement ambigus.

Théorème 3. *Un automate (max,+) G est polynomialement ambigu* ssi *pour tout $q \in Q$ et tout $w \in \Sigma^*$, il y a au plus un chemin dans $q \overset{w}{\rightsquigarrow} q$.*

3 Déterminisation des automates (max,+) reposant sur la normalisation

On s'intéresse à la déterminisation d'automates (max,+), à savoir, une procédure qui transforme un automate non déterministe G en un automate déterministe G' équivalent, si celui-ci existe. Dans cette section purement bibliographique, on rappelle en premier lieu une telle procédure, parfois appelée *algorithme de Mohri* (par exemple dans [10]) pour faire référence à l'algorithme présenté dans [15], sachant qu'une procédure similaire, dite de *déterminisation via normalisation*, est introduite dans [6, §VIII] et affinée dans [7]. Cette procédure est également présentée dans [14, §2.2.2] comme une spécialisation d'une procédure générale de séquentialisation. On rappelle ensuite plusieurs résultats liés à cette procédure.

On définit la relation binaire $\simeq$ de $\mathbb{R}_{\max}^n$ vers $\mathbb{R}_{\max}^n$ par :

$$x_1 \simeq x_2 \iff \exists \lambda \in \mathbb{R}_{\max} \setminus \{\varepsilon\}, x_1 = \lambda x_2. \tag{9}$$

Considérons $w \in \Sigma^*$ reconnu par l'automate (max,+) G et $v \in \text{Prefix}(w)$ tels que

$$x(w) \simeq x(v), \tag{10}$$

on a alors $\forall u \in \Sigma^*$

$$\begin{aligned} y(wu) &= \bigoplus_{p\in Q} x(wu)_p \\ &= \bigoplus_{p\in Q} [x(w) \otimes \mu(u)]_p \\ &= \bigoplus_{p\in Q} [\lambda \otimes x(v) \otimes \mu(u)]_p \\ &= \lambda \otimes \bigoplus_{p\in Q} [x(v) \otimes \mu(u)]_p \\ &= \lambda \otimes y(vu). \end{aligned} \tag{11}$$

En d'autres termes, si $x(w) \simeq x(v)$, alors le comportement pour toute évolution possible après w peut être déduite du comportement pour la même évolution après v et d'un décalage de λ. La procédure de déterminisation ci-dessous construit un automate déterministe G' dont les états sont étiquetés par les mots reconnus dans G. Les deux mots w et v correspondent alors à des états qui peuvent être fusionnés (en définissant un circuit sur l'état atteint après v reconnaissant $a_1 \dots a_p$ tel que $va_1 \dots a_p = w$ et avec un poids égal à λ) tout en conduisant à un même comportement, c'est-à-dire à un automate (max,+) équivalent.

La procédure de determinisation *via* normalisation, qui, si elle se termine, construit un automate $G' = (Q', q'_i, t')$ équivalent à $G = (Q, \alpha, \mu)$ possiblement non déterministe, peut être décrite au travers de la procédure 1.

Procédure 1 Déterminisation *via* normalisation

```
1:
2: Q_j ← {ε}, Q_{j+1} ← ∅, q'_i ← {ε}, Q' ← {ε}                ▷ Initialisation
3: while Q_j ≠ ∅ do
4:     for all w ∈ Q_j, a ∈ Σ tels que y(wa) ≠ ε do
5:         if ∃v ∈ Prefix(wa) tel que x(wa) ≃ x(v) then   ▷ Les comportements après wa et v
   sont identiques selon (11)
6:             t'(w, a, v) = y(w)^{-1} ⊗ y(wa)            ▷ Définit une transition vers l'état v
7:         else
8:             Q' ← Q' ⋃ {wa}                              ▷ Ajoute l'état wa dans Q'
9:             t'(w, a, wa) = y(w)^{-1} ⊗ y(wa)            ▷ Définit une transition vers l'état wa
10:            Q_{j+1} ← Q_{j+1} ⋃ {wa}
11:        end if
12:    end for
13:    Q_j ← Q_{j+1}
14:    Q_{j+1} ← ∅
15: end while
```

Exemple 4

Considérons l'automate (max,+) non déterministe G_2 de la figure 1 et détaillons comment la procédure opère pour construire l'automate déterministe G'_2 équivalent à G_2 (et représenté dans la même figure).

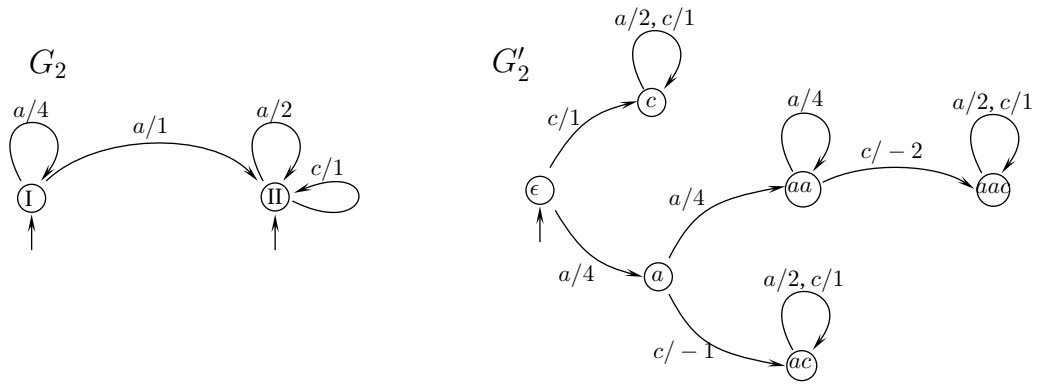

Figure 1: L'automate (max,+) non déterministe G_2 et G'_2 qui est déterministe et équivalent.

- L'initialisation conduit à $Q' = \{\epsilon\}$ et $q'_i = \{\epsilon\}$.
- À la première évaluation de la condition du **while** l.3, on a $\mathcal{Q}_j = \{\epsilon\}$.
  Puisque $x(\epsilon) = \begin{pmatrix} e & e \end{pmatrix}$ et $x(a) = \begin{pmatrix} 4 & 2 \end{pmatrix}$, $x(c) = \begin{pmatrix} \cdot & 1 \end{pmatrix}$, pour la variable '$wa$' égale à a et c la condition du **if** l.5 est fausse, et alors : les états a et c sont ajoutés dans Q' ; les transitions $t'(\epsilon, a, a) = 4$ et $t'(\epsilon, c, c) = 1$ sont définies.
- À la seconde évaluation de la condition du **while** l.3, on a $\mathcal{Q}_j = \{a, c\}$.
  Puisque $x(aa) = \begin{pmatrix} 8 & 5 \end{pmatrix}$ et $x(ac) = \begin{pmatrix} \cdot & 3 \end{pmatrix}$, la condition du **if** l.5 est fausse, et alors : les états aa et ac sont ajoutés dans Q' ; les transitions $t'(a, a, aa) = 4$ et $t'(a, c, ac) = -1$ sont définies.
  Puisque $x(ca) = \begin{pmatrix} \cdot & 3 \end{pmatrix}$ et $x(cc) = \begin{pmatrix} \cdot & 2 \end{pmatrix}$, on a $x(cc) \simeq x(c)$ et $x(ca) \simeq x(c)$ (la condition du **if** l.5 est vraie dans les deux cas), et alors les transitions $t'(c, a, c) = 2$ et $t'(c, c, c) = 1$ sont définies.
- À la troisième évaluation de la condition du **while** l.3, on a $\mathcal{Q}_j = \{aa, ac\}$.
  Puisque $x(aaa) = \begin{pmatrix} 12 & 9 \end{pmatrix}$ et $x(aac) = \begin{pmatrix} \cdot & 6 \end{pmatrix}$, $x(aaa) \simeq x(aa)$ mais la condition du **if** l.5 est fausse pour aac, et alors : la transition $t'(aa, a, aa) = 4$ est définie ; l'état aac est ajouté dans Q' et la transition $t'(aa, c, aac) = -2$ est définie.
  Puisque $x(aca) = \begin{pmatrix} \cdot & 5 \end{pmatrix}$ et $x(acc) = \begin{pmatrix} \cdot & 4 \end{pmatrix}$, on a $x(aca) \simeq x(ac)$ et $x(acc) \simeq x(ac)$, et alors les transitions $t'(ac, a, ac) = 2$ et $t'(ac, c, ac) = 1$ sont définies.
- À la quatrième évaluation de la condition du **while** l.3, on a $\mathcal{Q}_j = \{aac\}$.
  Puisque $x(aaca) = \begin{pmatrix} \cdot & 8 \end{pmatrix}$ et $x(aacc) = \begin{pmatrix} \cdot & 7 \end{pmatrix}$, on a $x(aaca) \simeq x(aac)$ et $x(aacc) \simeq x(aac)$, et alors les transitions $t'(aac, a, aac) = 2$ et $t'(aac, c, aac) = 1$ sont définies.
- À la cinquième évaluation de la condition du **while** l.3, $\mathcal{Q}_j$ est vide et la procédure se termine.

Considérons l'automate (max,+) G_3 de la figure 2. On a pour $k \geq 1$, $x(a^k) = \begin{pmatrix} 2 \times k & 4 \times k \end{pmatrix}$. Dans ce cas, la procédure ne se termine pas puisque pour tout $k \geq 1$ il n'existe pas $i \leq k$ tel que $x(a^k) \simeq x(a^i)$. ◇

Remarque 5 Pour faciliter la compréhension de notre contribution, nous avons présenté la procédure de déterminisation *via* normalisation :

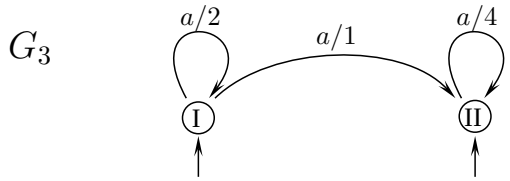

Figure 2: L'automate (max,+) non déterministe G_3.

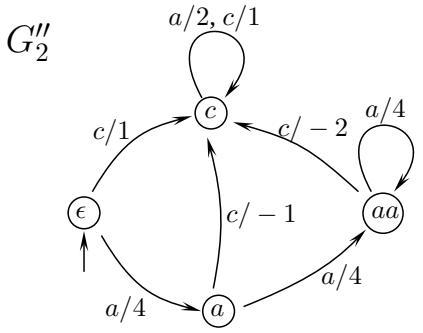

Figure 3: L'automate (max,+) déterministe G_2'' équivalent à G_2 et G_2'.

- avec des états dans l'automate résultant G' étiquetés avec les mots comme dans [6], alors que des labels correspondant aux valeurs de l'état sont utilisés dans [7, 15, 10] ;
- en utilisant l'ordre préfixiel dans la condition du **if** l.5, alors que l'ordre hiérarchique (aussi appelé militaire et basé sur la longueur ainsi que l'ordre lexicographique à longueur égale) est utilisé dans [6, 7, 15, 10]. En utilisant l'ordre hiérarchique, la procédure teste parmi tous les états précédemment ajoutés dans Q' (au lieu de seulement les préfixes) si l'un conduit à un comportement identique, et permet ainsi de minimiser le nombre d'états dans Q'. Par exemple, nous obtenons G_2'' de la figure 3 en utilisant l'ordre hiérarchique dans la procédure. Quelle que soit la relation d'ordre (préfixielle ou hiérarchique), notons que les automates résultants sont équivalent et les conditions pour lesquelles la procédure se termine sont identiques.

◇

A notre connaissance la condition de clonage introduite dans [10] est la condition la plus générale permettant de caractériser les automates pouvant être déterminisés à l'aide de la procédure *via* normalisation. Soient $u,\ v \in \Sigma^*$ tels que $R(uv) = R(u)$, on note $M(u,v)$ le sous-ensemble de $R(u)$ contenant les états dans $R(u)$ à partir desquels les circuits reconnaissant v ont le poids maximum, à savoir

$$M(u,v) \triangleq \{p \in R(u) \mid \mu(v)_{pp} = \bigoplus_{q \in R(u)} \mu(v)_{qq}\}. \tag{12}$$

Définition 7 (Propriété de clonage)**.** *Soient $u, v \in \Sigma^*$. Un état $q \in R(u)$ est appelé un* clone *pour v si $\mu(v)_{qq} \neq \varepsilon$ implique qu'il existe $p \in M(u,v)$ tel que $\mu(v)_{pq} \neq \varepsilon$. Un automate (max,+)*

G a la propriété de clonage *si* $\forall u \in \Sigma^*$, $\forall p, q \in R(u)$, *p et q sont des clones pour tout* $v \in \Sigma^*$ *tel que* $R(uv) = R(u)$.

Théorème 6 ([10, th. 3.4])**.** *Soit G un automate (max,+) émondé et polynomialement ambigu. La procédure* via *normalisation termine sur G* ssi *G satisfait la propriété de clonage.*

Exemple 7 L'automate (max,+) G_2 de la figure 1 est polynomialement ambigu (*cf.* théorème 3) et satisfait la propriété de clonage. L'automate G_3 représenté sur la figure 2 est polynomialement ambigu mais ne satisfait pas la propriété de clonage puisque $\mu(a)_{I,I} \neq \bigoplus_{r \in R(a)} \mu(a)_{rr} = 4 = \mu(a)_{II,II}$ et $\mu(a)_{II,I} = \varepsilon$. ◇

4 Amélioration de la procédure de déterminisation

Dans cette section, on propose une amélioration de la procédure de déterminisation qui permet d'élargir la classe d'automates (max,+) pouvant être traités avec succès. L'idée directrice est d'affaiblir la condition permettant de détecter des états admettant un identique comportement futur. Plus précisément, on propose une relation différente de $\simeq$ (définie par (10)) pour laquelle on prouve que les états en relation admettent un même comportement ultérieur. En combinant cette relation avec $\simeq$ (sous la forme d'une disjonction dans le test de la ligne 5 de la procédure 1), un plus grand nombre d'états peuvent être fusionnés lors des itérations de la procédure 1 et celle-ci peut ainsi terminer pour une classe plus large d'automates (max,+).

4.1 Nouvelle relation pour des états ayant d'identiques comportements ultérieurs

Considérons $w \in \Sigma^*$ reconnu par un automate (max,+) G et $v \in \text{Prefix}(w)$, on définit la relation $\sim$ par

$$x(w) \sim x(v)$$
$$\iff$$
$$S(w) = S(v), \tag{13}$$
$$\forall u \in \Sigma^*, \forall q \in S(vu), \exists p \in S(v) : x(vu)_q = x(v)_p \otimes \mu(u)_{pq}, \tag{14}$$
$$\forall u \in \Sigma^*, \forall q \in S(wu), \exists p \in S(w) : x(wu)_q = x(w)_p \otimes \mu(u)_{pq}. \tag{15}$$

Lemme 1. *Si $x(w) \sim x(v)$, alors $S(wu) = S(vu)$ pour tout $u \in \Sigma^*$.*

Proof. Supposons que $x(w) \sim x(v)$. Notons tout d'abord que pour tout $q_w \in S(wu)$, la condition (15) implique que

$$\exists p_w \in S(w), \quad x(wu)_{q_w} = x(w)_{p_w} \otimes \mu(u)_{p_w q_w} \tag{16}$$
$$\text{avec} \quad \mu(u)_{p_w q_w} \geq \mu(u)_{pq}, \forall p \in S(w) = S(v), \forall q \in Q. \tag{17}$$

En effet pour (17), $\mu(u)_{p_w q_w} < \mu(u)_{pq}$ pour un $p \in S(w)$ et un $q \in Q$ impliquerait que $x(wu)_{q_w} < x(wu)_q$ ce qui contredirait $q_w \in S(wu)$.

Notons aussi que pour tout $q_v \in S(vu)$, la condition (14) implique que

$$\exists p_v \in S(v), \quad x(vu)_{q_v} = x(v)_{p_v} \otimes \mu(u)_{p_v q_v} \tag{18}$$
$$\text{avec} \quad \mu(u)_{p_v q_v} \geq \mu(u)_{pq}, \forall p \in S(v) = S(w), \forall q \in Q. \tag{19}$$

On montre d'abord que tout $q_w \in S(wu)$ appartient aussi à $S(vu)$, et donc que $S(wu) \subseteq S(vu)$. Puisque $p_w \in S(w)$ et $S(w) = S(v)$, on a

$$x(v)_{p_w} = y(v) = x(v)_p, \forall p \in S(v), \tag{20}$$

et

$$\begin{aligned}
x(vu)_{q_w} &= \bigoplus_{p\in Q} x(v)_p \otimes \mu(u)_{pq_w} \\
&\geq x(v)_{p_w} \otimes \mu(u)_{p_w q_w} \\
&= x(v)_p \otimes \mu(u)_{p_w q_w}, \forall p \in S(v) && \text{(selon (20))} \\
&\geq x(v)_p \otimes \mu(u)_{pq}, \forall p \in S(v), \forall q \in Q && \text{(selon (17))} \\
\Rightarrow x(vu)_{q_w} &\geq x(v)_{p_v} \otimes \mu(u)_{p_v q_v} \\
&= x(vu)_{q_v}. && \text{(selon (18))}
\end{aligned}$$

De façon similaire, on montre que tout $q_v \in S(vu)$ appartient aussi à $S(wu)$, et donc que $S(vu) \subseteq S(wu)$. □

Lemme 2. *Soit $\lambda \in \mathbb{R}_{\max}$ défini par $y(w) = \lambda \otimes y(v)$ (i.e. $\lambda = y(v)^{-1} \otimes y(w)$ si $y(v) \neq \varepsilon$). On a*

$$x(w) \sim x(v) \quad \Rightarrow \quad y(wu) = \lambda \otimes y(vu), \forall u \in \Sigma^*. \tag{21}$$

Proof. Supposons que $x(w) \sim x(v)$ et considérons $u \in \Sigma^*$. Selon le lemme 1, $q \in S(wu)$ appartient aussi à $S(vu)$, et

$$\exists p_v \in S(v) : \; x(vu)_q = x(v)_{p_v} \otimes \mu(u)_{p_v q} \qquad \text{(selon (14))} \tag{22}$$

$$\exists p_w \in S(w) : \; x(wu)_q = x(w)_{p_w} \otimes \mu(u)_{p_w q}. \qquad \text{(selon (15))} \tag{23}$$

On montre d'abord que

$$\mu(u)_{p_v q} = \mu(u)_{p_w q}. \tag{24}$$

Supposons que $\mu(u)_{p_v q} > \mu(u)_{p_w q}$, alors

$$\begin{aligned}
y(wu) = x(wu)_q &\qquad \text{(puisque } q \in S(wu)\text{)} \\
= x(w)_{p_w} \otimes \mu(u)_{p_w q} &\qquad \text{(selon (23))} \\
< x(w)_{p_w} \otimes \mu(u)_{p_v q} &\qquad \text{(par hypothèse)} \\
= x(w)_{p_v} \otimes \mu(u)_{p_v q} &\qquad \text{(puisque } p_v \text{ et } p_w \text{ sont dans } S(v) = S(w)\text{)}
\end{aligned}$$

ce qui contredit

$$y(wu) = x(wu)_q = \bigoplus_{p\in Q} x(w)_p \otimes \mu(u)_{pq} \geq x(w)_p \otimes \mu(u)_{pq}, \forall p \in Q.$$

De façon symétrique, on montre que le cas $\mu(u)_{p_v q} < \mu(u)_{p_w q}$ n'est pas possible. Finalement,

on a

$$\begin{aligned}
y(wu) &= x(wu)_q \\
&= x(w)_{p_w} \otimes \mu(u)_{p_w q} && \text{(selon (23))} \\
&= y(w) \otimes \mu(u)_{p_w q} && \text{(puisque } p_w \in S(w)\text{)} \\
&= \lambda \otimes y(v) \otimes \mu(u)_{p_w q} \\
&= \lambda \otimes x(v)_{p_v} \otimes \mu(u)_{p_w q} && \text{(puisque } p_v \in S(v)\text{)} \\
&= \lambda \otimes x(v)_{p_v} \otimes \mu(u)_{p_v q} && \text{(selon 24)} \\
&= \lambda \otimes x(vu)_q && \text{(selon (22))} \\
&= \lambda \otimes y(vu).
\end{aligned}$$

□

Le premier exemple ci-dessous met en évidence qu'il existe des cas pour lesquels $x(w) \sim x(w)$ est vérifiée mais pas $x(w) \simeq x(v)$. Dans la suite, on propose de détecter les états conduisant à d'identiques comportements ultérieurs en testant s'ils sont en relation selon $\sim$ ou selon $\simeq$. En procédant ainsi, il est clair que cette détection va aboutir plus souvent (et la procédure de déterminisation va se terminer plus souvent) que si seule la relation $\simeq$ est utilisée (comme dans la procédure 1). Le deuxième exemple ci-dessous montre qu'il existe des cas pour lesquels $x(w) \simeq x(v)$ est vérifiée mais pas $x(w) \sim x(w)$. Cela signifie que les conditions définissant $\sim$ ne sont pas toujours moins restrictives que celles définissant $\simeq$ et cela justifie de tester les deux relations. On souligne également dans cet exemple qu'il existe des cas pour lesquels les deux relations sont vérifiées.

Exemple 8 Intéressons-nous à l'automate (max,+) G_3 de la figure 2 qui reconnaît seulement les mots a^n, $n \geq 1$. On se convainc facilement que l'ensemble des états gagnants pour tout ces mots se restreint à l'état de droite, à savoir $S(a^n) = \{II\}$ pour tout n. En particulier, on alors $x(a^2) \sim x(a)$. Mais $x(a^2) = \begin{pmatrix} 4 & 8 \end{pmatrix}$ et $x(a) = \begin{pmatrix} 2 & 4 \end{pmatrix}$, ce qui montre que $x(a^2) \simeq x(a)$ n'est pas vérifiée. ◇

Exemple 9 Considérons l'automate (max,+) G_4 représenté sur la figure 4. On a :

$$x(a) = \begin{pmatrix} 2 & 1 & \cdot & \cdot \end{pmatrix}, \quad x(a^2) = \begin{pmatrix} 3 & 2 & \cdot & \cdot \end{pmatrix},$$

et donc $x(a) \simeq x(a^2)$. De plus, on a $S(a) = S(a^2) = \{I\}$ et

$$x(ab) = \begin{pmatrix} \cdot & \cdot & 3 & 4 \end{pmatrix}, \quad x(a^2b) = \begin{pmatrix} \cdot & \cdot & 4 & 5 \end{pmatrix}.$$

En observant qu'il n'existe pas de chemin depuis I jusqu'à IV, on peut constater que pour l'état $IV \in S(ab)$, il n'existe pas $q \in S(a)$ tel que $x(ab)_{IV} = x(a)_q \otimes \mu(b)_{q,IV}$ et donc $x(a) \sim x(a^2)$ est faux. Soulignons enfin que pour G_2 sur la figure 1, tous les cas tels que $x(w) \simeq x(v)$ sont aussi tels que $x(w) \sim x(v)$. ◇

4.2 Utilisation de la nouvelle relation pour améliorer la procédure de déterminisation

Le lemme 2 montre que la relation $\sim$ définie par (13-15) permet d'identifier des états dont le comportement ultérieur est identique. En pratique, la relation $\sim$ ne peut pas être testée puisque

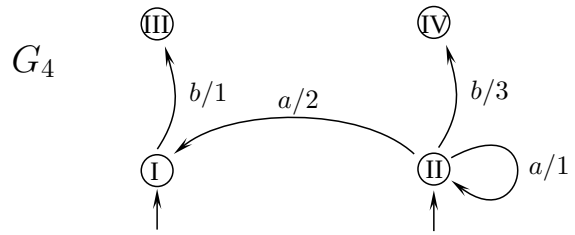

Figure 4: L'automate (max,+) non déterministe G_4.

cela requiert de vérifier les conditions (14) et (15) pour un nombre infini de mots $u \in \Sigma^*$. On spécifie dans ce qui suit une autre relation qui peut être testée et qui est suffisante pour la relation $\sim$. Cette nouvelle relation peut être combinée avec la relation $\simeq$ dans la procédure de déterminisation afin d'élargir la classe d'automates (max,+) pour lesquels celle-ci aboutit.

Soulignons qu'il s'agit là d'une première approche et que la relation proposée est une solution assez naïve pour obtenir une condition testable et suffisante pour la relation $\simeq$. Celle-ci est assez restrictive et les travaux futurs devront prospecter des relations testables et plus faibles.

Définissons la relation binaire $\approx$ de $\mathbb{R}^n_{\max}$ vers $\mathbb{R}^n_{\max}$ par

$$x(w) \approx x(v)$$
$$\Longleftrightarrow$$
$$R(w) = R(v), S(w) = S(v), \tag{25}$$
$$\forall p \in R(w) \setminus S(w), \quad \exists q \in S(w) \text{ tel que } \forall a \in \Sigma$$
$$R(wa, p) \subseteq \{p, q\}, \tag{26}$$
$$\mu(a)_{qq} \geq \mu(a)_{pr}, \quad \forall r \in R(wa, p). \tag{27}$$

Lemme 3. *Si $x(w) \approx x(v)$, alors $x(w) \sim x(v)$.*

Proof. La condition (25) implique de façon évidente (13). Montrons maintenant que (26-27) implique (15) et remarquons que l'égalité $R(w) = R(v)$ dans (25) induit que l'implication (26-27) $\Rightarrow$ (14) peut se déduire de la même manière.

Pour tout $u = a_1 a_2 \dots a_n \in \Sigma^*$, considérons un chemin de poids maximal reconnaissant wu et notons $p_1, p_2, \dots, p_n$ et $q_u \in S(wu)$ les états tels que

$$y(wu) = x(wu)_{q_u} = x(w)_{p_1} \otimes \mu(a_1)_{p_1 p_2} \otimes \dots \otimes \mu(a_{n-1})_{p_{n-1} p_n} \otimes \mu(a_n)_{p_n q_u}. \tag{28}$$

Si $p_1 \in S(w)$, il n'y a rien à montrer puisque (15) est alors vérifiée. Si $p_1 \in R(w) \setminus S(w)$, on montre que (26-27) impliquent qu'il existe $q_1 \in S(w)$ tel que

$$x(wu)_{q_u} = x(w)_{q_1} \otimes \mu(a_1 \dots a_n)_{q_1 q_u} \tag{29}$$

et la condition (15) est alors vérifiée.

Observons que la condition (26) induit que $p_i \in \{q_1, p_1\}$ pour $i = 2, \dots, n$ et $q_u = q_1$. De plus, la condition (27) impose que pour tout a_i, $i = 1, \dots n$

$$\mu(a_i)_{q_1 q_1} \geq \mu(a_i)_{p_1 p_1} \text{ et } \mu(a_i)_{q_1 q_1} \geq \mu(a_i)_{p_1 q_1}. \tag{30}$$

On a

$$\begin{aligned} x(w)_{q_1} \otimes \mu(a_1 \dots a_n)_{q_1 q_u} &\geq x(w)_{q_1} \otimes \mu(a_1)_{q_1 q_1} \otimes \dots \otimes \mu(a_{n-1})_{q_1 q_1} \otimes \mu(a_n)_{q_1 q_u} \\ &\geq x(w)_{q_1} \otimes \mu(a_1)_{p_1 p_2} \otimes \dots \otimes \mu(a_{n-1})_{p_{n-1} p_n} \otimes \mu(a_n)_{p_n q_u} \\ &\quad (\text{ à partir de } p_i \in \{q_1, p_1\} \text{ et (30) pour tout } i) \\ &\geq x(w)_{p_1} \otimes \mu(a_1)_{p_1 p_2} \otimes \dots \otimes \mu(a_{n-1})_{p_{n-1} p_n} \otimes \mu(a_n)_{p_n q_u} \\ &= x(wu)_{q_u} \qquad (\text{ selon (28)}) \end{aligned}$$

ce qui prouve (29) puisque par définition $x(wu)_{q_u} \geq x(w)_q \otimes \mu(a_1 \dots a_n)_{q q_u}$ pour tout q. □

Ensemble les lemmes 3 et 2 montrent que la relation $\approx$ est suffisante pour identifier des états admettant d'identiques comportements ultérieurs. De plus, cette relation peut être testée. Il est alors naturel d'adapter la procédure 1 en remplaçant la ligne 5 comme suit :

$$5 : \qquad \textbf{if} \quad \exists v \in \text{Prefix}(wa) \text{ tel que } x(wa) \simeq x(v) \text{ ou } x(wa) \approx x(v) \quad \textbf{then}$$

La procédure ainsi modifiée aboutit identiquement pour les automates (max,+) qui peuvent être traités avec succès à l'aide de la procédure 1. De plus, on montre ci-dessous qu'il existe des automates (max,+) pour lesquels la procédure modifiée aboutit alors que la procédure initiale échoue à construire un automate déterministe équivalent.

Exemple 10 Considérons à nouveau l'automate (max,+) non déterministe G_3. Dans l'exemple 4 on a mis en évidence que la procédure existante (utilisant seulement la relation $\simeq$) ne se termine pas et échoue donc à déterminiser G_3.

En utilisant à la fois les relation $\simeq$ et $\approx$, la procédure modifiée opère comme suit avec G_3 en entrée :

- L'initialisation pose $Q' = \{\epsilon\}$ et $q'_i = \{\epsilon\}$.
- À la première évaluation de la condition du **while** à la ligne 3, on a $\mathcal{Q}_j = \{\epsilon\}$.
  a est la seule lettre énumérée dans le for all à la ligne 4 ($y(a) \neq \varepsilon$). Puisque $S(\epsilon) = \{I, II\}$ et $S(a) = \{II\}$, $x(a) \simeq x(\epsilon)$ $x(a) \approx x(\epsilon)$ sont faux et la condition du **if** à la ligne 5 est fausse, et alors, l'état a est ajouté dans Q' ; la transition $t'(\epsilon, a, a) = 4$ est définie.
- À la deuxième évaluation de la condition du **while** à la ligne 3, on a $\mathcal{Q}_j = \{a\}$.
  a est la seule lettre énumérée dans le for all à la ligne 4 ($y(aa) \neq \varepsilon$). Puisque $x(a) = \begin{pmatrix} 2 & 4 \end{pmatrix}$ et $x(aa) = \begin{pmatrix} 4 & 8 \end{pmatrix}$, $x(a^2) \simeq x(a)$ est faux. Mais on a $S(aa) = S(a)$, $R(aa) = R(a)$, et les conditions (26-27) sont satisfaites. On a alors $x(a^2) \approx x(a)$ et la condition du **if** à la ligne 5 est vraie, et alors la transition $t'(a, a, a) = 4$ est définie.
- À la troisième évaluation de la condition du **while** à la ligne 3, $\mathcal{Q}_j$ est vide et la procédure termine.

La procédure construit ainsi G'_3 (déterministe et équivalent à G_3) représenté sur la figure 5. ◇

5 Conclusion et perspectives

On a proposé une nouvelle condition permettant d'identifier les états d'un automate (max,+) engendrant un identique comportement ultérieur. Cela nous permet d'enrichir la procédure de déterminisation de sorte qu'elle aboutisse pour un plus grand nombre d'automates (max,+). Nos travaux futurs viseront :

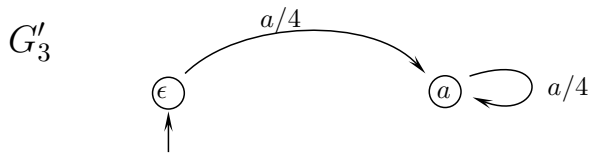

Figure 5: L'automate (max,+) déterministe G_3' équivalent à G_3.

- à généraliser notre approche en utilisant l'ordre hiérarchique dans la procédure de déterminisation et en considérant des automates pour lesquels tous les états ne sont pas finaux,
- à trouver des relations moins restrictives que $\approx$ et suffisantes pour $\sim$ afin d'étendre encore la classe d'automates pouvant être déterminisés à l'aide de la procédure enrichie, et à identifier les classes d'automates (max,+) pour lesquelles notre procédure enrichie aboutit.

References

[1] F. Baccelli, G. Cohen, G.-J. Olsder, and J.-P. Quadrat. *Synchronization and Linearity*. Wiley, 1992.

[2] E. Badouel, A. Bouillard, P. Darondeau, and J. Komenda. Residuation of tropical series: rationality issues. In *joint 50th IEEE CDC-ECC*, 2011.

[3] M.-P. Béal, E. Czeizler, J. Kari, and D. Perrin. Unambiguous automata. *Mathematics in Computer Science*, 1(4):625–638, 2008.

[4] R. Boukra, S. Lahaye, and J.-L. Boimond. New representations for (max,+) automata with applications to performance evaluation and control of discrete event systems. *Disc. Event Dyn. Syst.*, 25:295–322, 2015.

[5] S. Gaubert. *Théorie des systèmes linéaires dans les dioïdes*. Thèse de doctorat, Ecole des Mines de Paris, July 1992.

[6] S. Gaubert. Performance Evaluation of (max,+) Automata. *IEEE TAC*, 40(12):2014–2025, 1995.

[7] S. Gaubert and J. Mairesse. Asymptotic analysis of heaps of pieces and application to timed Petri nets. In *Proceedings of Petri Nets and Performance Models*, pages 158 – 169, 1999.

[8] S. Gaubert and J. Mairesse. Modeling and Analysis of Timed Petri Nets using Heaps of Pieces. *IEEE TAC*, 44(4):683–698, 1999.

[9] B. Heidergott, G. J. Olsder, and J. V. D. Woude. *Max Plus at work*. Princeton Press, 2006.

[10] D. Kirsten. A burnside approach to the termination of Mohri's algorithm for polynomially ambiguous min-plus-automata. *RAIRO - TIA*, 42(3):553–581, 2008.

[11] I. Klimann, S. Lombardy, J. Mairesse, and C. Prieur. Deciding unambiguity and sequentiality from a finitely ambiguous max-plus automaton. *TCS*, 2004.

[12] J. Komenda, S. Lahaye, and J.-L. Boimond. Supervisory Control of (max,+) Automata: A Behavioral Approach. *Disc. Event Dyn. Syst.*, 19(4):525–549, 2009.

[13] S. Lahaye, J. Komenda, and J.-L. Boimond. Compositions of (max, +) automata. *Discrete Event Dynamic Systems*, 25:323–344, June 2015 2015.

[14] S. Lombardy and J. Sakarovitch. Sequential ? *TCS*, 359(1-2):224–244, 2006.

[15] M. Mohri. Finite-state transducers in language and speech processing. *Comp. Lingu.*, 23:269–311, 1997.

[16] M. Mohri. Weighted automata algorithms. In *Handbook of weighted automata*. Springer, 2011.

[17] Rong Su and Gerhard J. Woeginger. String execution time for finite languages: Max is easy, min is hard. *Automatica*, 47(10):2326–2329, 2011.