Jeux et Langages Formels

Matthieu Boyer

22 décembre 2023

Table des matières

Résumé

Dans ce rapport, on s'intéresse à des manières de représenter des jeux par la théorie des langages formels, en particulier par les automates et par les automates à pile. On présente alors une nouvelle caractérisation de la régularité d'un langage comme sa reconnaissance par une certaine représentation des jeux. Puis, en modifiant cette représentation, on trouve une caractérisation des langages algébriques par les jeux. Enfin, on s'intéresse à une manière de représenter le *Game Design* par des automates en introduisant un délai sur les transitions, et on regarde quelques exemples d'applications.

Introduction

Dans la suite on ne s'intéresse qu'à des jeux à plusieurs joueurs i.e. plus de deux.
Un jeu à plusieurs joueurs est, de manière informelle, une abstraction d'un jeu ressemblant par exemple à la belote : chaque joueur, à son tour, va choisir, parmi un éventail de coups possibles, celui qu'il souhaite effectuer, modifiant alors l'état du jeu. En reprenant l'exemple de la belote, un coup est une carte de la main du joueur, qui ne sait pas exactement quelles cartes ont ses adversaires, et qui ne peut la jouer que sous certaines conditions, à résoudre dans l'ordre :

1. Soit il est le premier à jouer
2. Soit la carte est de la *bonne* couleur
3. Soit la carte est un atout plus grand que le dernier joué
4. Soit la carte est une coupe (atout lorsqu'une autre couleur est jouée)
5. Soit il ne peut rien jouer d'autre remplissant les conditions précédentes

Il paraît alors raisonnable qu'en limitant les transitions possibles de l'état de jeu selon l'état et selon les règles du jeu, on puisse modéliser un jeu par un automate. Toutefois, le manque d'informations d'un joueur sur les mains de ses adversaires, semble aussi limiter la description directe du jeu avec un caractère non-déterministe.
On s'intéresse donc ici à des manières de décrire les jeux par les automates et les grammaires formelles.

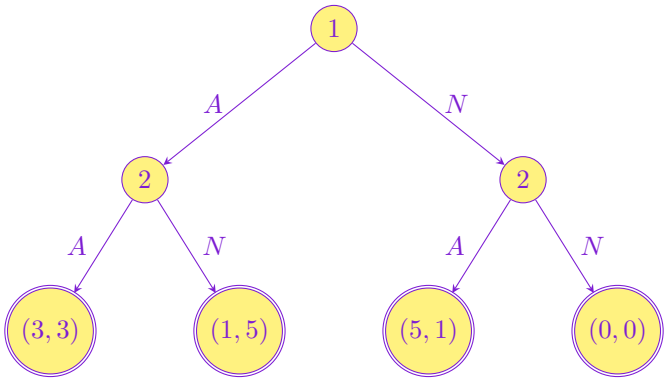

FIGURE 1 – Représentation Extensive du Jeu du DILEMME DU PRISONNIER à information totale.

1 Jeux et Automates

1.1 Première définition de Jeu

On introduit d'abord une première définition de jeu suivant [1] :

Définition 1.1: Première définition de Jeu

Un jeu est un triplet $(P, A_i, \succeq_i)$ où P est un ensemble de joueurs, A_i est un ensemble d'actions pour le joueur $i \in P$ et $\succeq_i$ est une relation de préférence pour le joueur i.

Pour le DILEMME DU PRISONNIER, on a par exemple : $P = \{1, 2\}$, $\forall i, A_i = \{A, N\}$ et $\succeq_i$ peut se lire sur l'arbre de jeu 1 de manière numérique.

Ici, chaque noeud interne de l'arbre correspond à un joueur, chaque arête à un coup qu'il peut jouer, c'est-à-dire dans notre cas, choisir d'avouer (A) son crime, ou bien de le nier (N) et les feuilles de l'arbre représentent les résultats, cf. Table 1.

TABLE 1 – Résultats du DILEMME DU PRISONNIER

		Prisonnier 1	
		Aveu	Négation
Prisonnier 2	Aveu	(3, 3)	(5, 1)
	Négation	(1, 5)	(0, 0)

Définition 1.2: Description Extensive

On appelle la description d'un jeu à information totale sous forme d'arbre (cf. figure 1) précédente la description extensive de ce jeu.

Toutefois, d'autre descriptions peuvent être préférables, notamment lorsque l'un des joueurs n'a pas d'information sur les coups des autres joueurs, comme c'est le cas dans certaines variantes du DILEMME DU PRISONNIER. En effet, on considérait ici que le prisonnier numéro [1] 2 est informé du choix du prisonnier numéro 1, ce qui n'est pas un très bon choix de la part des geôliers.

1. NDA : *I am not a number ! I am a free man !*

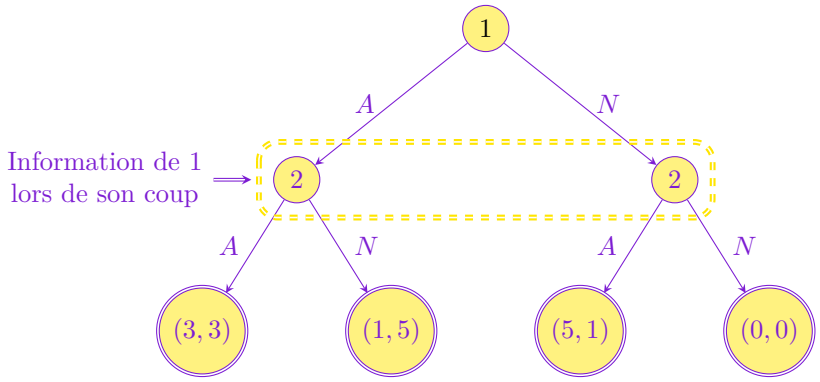

FIGURE 2 – Représentation Extensive du Jeu du DILEMME DU PRISONNIER à information imparfaite.

Définition 1.3: Information d'un Jeu

On parle de jeu à information imparfaite (par opposition aux jeux à information totale), lorsque le joueur actuel n'a pas d'informations sur les coups des joueurs précédents. On représente ceci en rajoutant sur l'arbre de jeu l'information possédée par un joueur lors de son coup en entourant l'ensemble des positions possibles.

Dans le cas du DILEMME DU PRISONNIER, on modifie l'arbre 1 comme suit :

Définition 1.4: Partie sur un Jeu

Une partie sur un jeu est une suite d'états de ce jeu, ou, de manière équivalente, une suite de coups $s_0 s_1 \dots s_k$ tels que :

- $\forall i, s_{2i} \in A_0$ et $s_{2i+1} \in A_1$.
- $\nexists s_{k+1} \in A_i$ où $i = 1$ si $k \equiv 0 \mod 2$ et $i = 0$ sinon.

1.2 Les Automates vus comme des Jeux

Définition 1.5: Jeu d'un Automate

Si $A = (Q, \Sigma, \delta, q_0, F)$ est un automate, on se donne G_A un jeu sous représentation extensive à deux joueurs TATIANA et PIERRE. Dans ce jeu, TATIANA joue des états de Q et PIERRE joue des lettres de Σ.

On joue comme suit :

- TATIANA joue q_0 l'état initial.
- PIERRE doit jouer une lettre $s \in \Sigma$
- La partie se termine lorsque TATIANA joue un état de F et que PIERRE n'a plus de coup valable.

Définition 1.6: Règles du Jeu

Les règles du jeu pour chaque joueur sont les suivantes : si TATIANA joue $q \in \Sigma$ alors PIERRE doit jouer s tel que $\delta(q, s) \neq \varnothing$ sinon il perd.

On peut par ailleurs faire un lien entre le caractère déterministe de l'automate et l'information donnée au joueur PIERRE, ce qui correspond intuitivement au fait qu'on puisse aller d'un état à plusieurs.

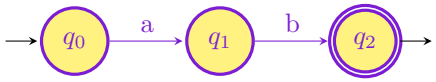

FIGURE 3 – Un Automate Déterministe

- Si A est déterministe, G_A est équivalent à un jeu à information parfaite, puisqu'en effet, après le coup de l'adversaire, on peut être certain de l'état du jeu.
- A l'inverse, si A n'est pas déterministe, on peut d'une meilleure manière définir des ensembles d'informations pour chacun des joueurs récursivement : On a en effet : $\delta(q, s) = B \subseteq Q$ et on a alors un ensemble d'informations autour du prochain coup de PIERRE lorsque TATIANA joue $q' \in B$. En effet, PIERRE doit alors jouer $s' \in \Sigma$ tel que $\delta(q, s')$ est défini et $q \in B$.

Dans la suite, on se place d'abord dans le cas déterministe.
Pour l'automate de la figure 3, on obtient la représentation extensive de la figure 4, en utilisant λ comme symbole pour signifier que PIERRE n'a plus de lettres jouables :

1.3 Langage et Stratégies Gagnantes

On se place ici dans le cas déterministe :
On définit formellement la notion de stratégie d'un joueur sur un tel jeu. La stratégie d'un joueur assigne une action choisie par le joueur pour chaque historique des coups dans lequel c'est à son tour de jouer. L'historique d'un joueur est l'ensemble des actions qu'il peut choisir de faire durant la partie. Pour TATIANA, c'est l'ensemble des combinaisons d'états qu'il peut utiliser pour affronter PIERRE et pour PIERRE, c'est n'importe quel mot $w \in \Sigma^\star$:

Définition 1.7: Stratégie

Une stratégie pour PIERRE définie par $w \in \Sigma^\star$ est telle que PIERRE joue les lettres de w indépendamment de ce que TATIANA joue.

On a besoin de se restreindre à des mots de longueur finie n pour les stratégies pour pouvoir vérifier l'équivalence entre l'acception par A et le fait d'être une stratégie gagnante.

Définition 1.8: Stratégie Gagnante

Une stratégie gagnante pour PIERRE est une stratégie w de longueur n où TATIANA n'a pas de coup valide (i.e. la partie est arrivée à son terme) et où son dernier coup est un état final de A. On définit alors : $S\left(G_A\right)^n$ l'ensemble des parties gagnantes pour PIERRE sur G_A où PIERRE joue selon un mot de longueur n, et $L(A)^n$ l'ensemble des mots de longueur n reconnus par A.

Ainsi, on doit montrer qu'étant donnée une stratégie gagnante $s \in S\left(G_A\right)^n$, il existe un mot w de longueur n accepté par A. On a alors :

Théorème 1.9: Reconnaissance des Stratégies

$S(G_A)^n = L(A)^n$

Démonstration. En effet, il est clair qu'on peut trouver une équivalence entre le calcul d'un mot w sur A et le déroulement d'une partie où PIERRE joue selon w. Ceci se lit sur l'arbre de jeu. ■

1.4 Non-Déterminisme

On considère désormais le cas des automates finis non-déterministes (NFA). S'ils n'apportent pas d'expressivité, ils permettent tout de même un point de vue intéressant.

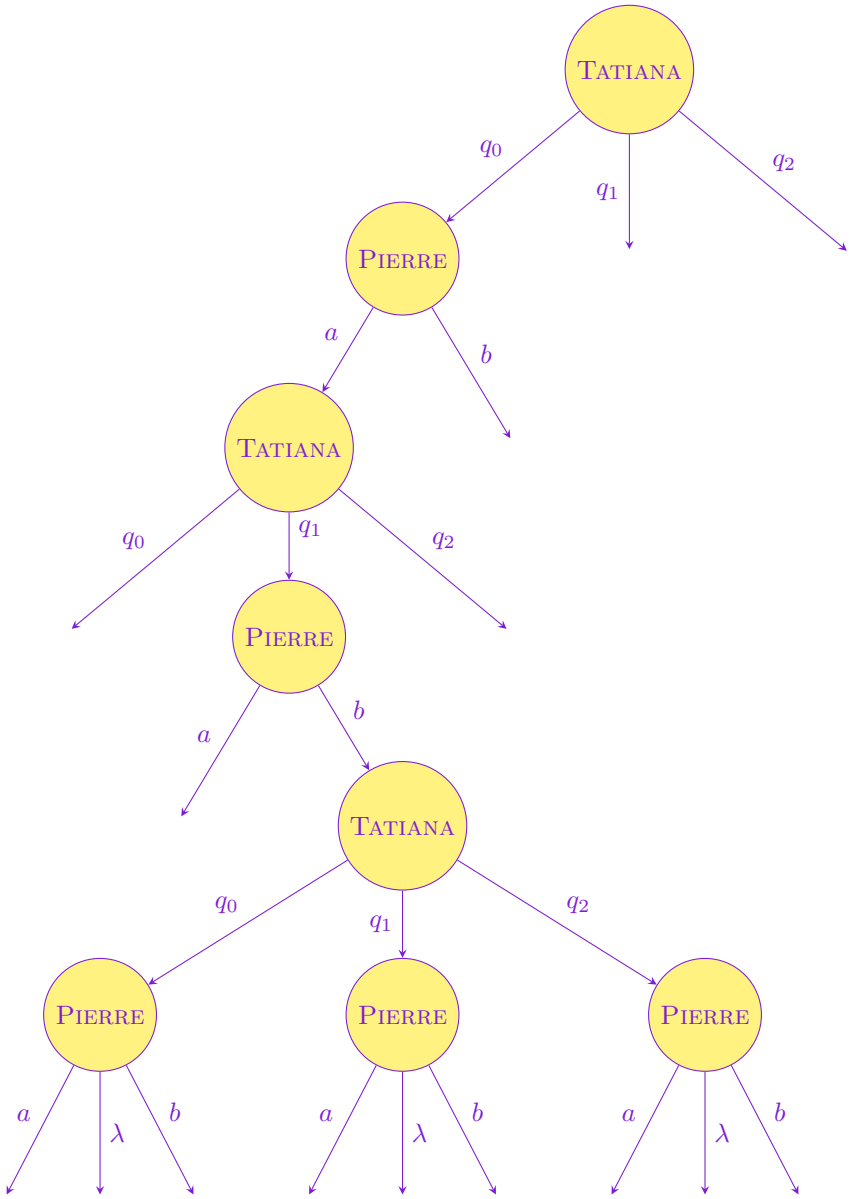

Figure 4 – Représentation Extensive du Jeu à Information Parfaite de l'automate figure 3

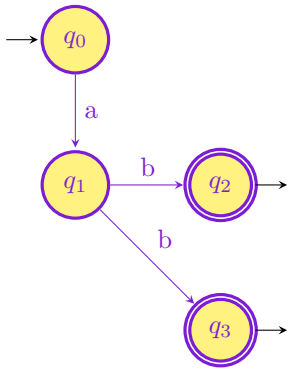

FIGURE 5 – Un Automate Non-Déterministe

Définition 1.10: Jeu d'un NFA

Etant donné un automate non-déterministe A, on définit :

- P un ensemble de deux joueurs où TATIANA joue des états et PIERRE des lettres.
- Un ensemble I d'ensembles d'informations B, où on définit $\delta(q, s) = B$ lorsqu'il y a du non-déterminisme dans la transition.
- Des relations de préférence $\succeq_i$ entre deux transitions δ_1, δ_2 où $\delta_1 \succeq \delta_2$ si et seulement si $d(\delta_1) \leq d(\delta_2)$ où $d(\tau)$ renvoie la distance dans l'automate vu comme un graphe orienté entre l'état de fin de τ et son état de début.

Pour l'automate de la figure 5 on obtient l'arbre de la figure 6. On vérifie bien sur la figure 6 qu'il n'y a pas besoin d'indiquer les ensembles d'informations sur les autres étapes du jeu.
On remarque par ailleurs, qu'en reprenant les définitions précédentes, on obtient bien le même théorème d'équivalence entre les stratégies gagnantes et les mots acceptés. Ceci se comprend quitte à déterminiser l'automate. On obtient donc en particulier le théorème suivant :

Théorème 1.11: Information et Reconnaissance

Tout jeu à information imparfaite est équivalent, au sens de la reconnaissance des langages, à un jeu à information totale.

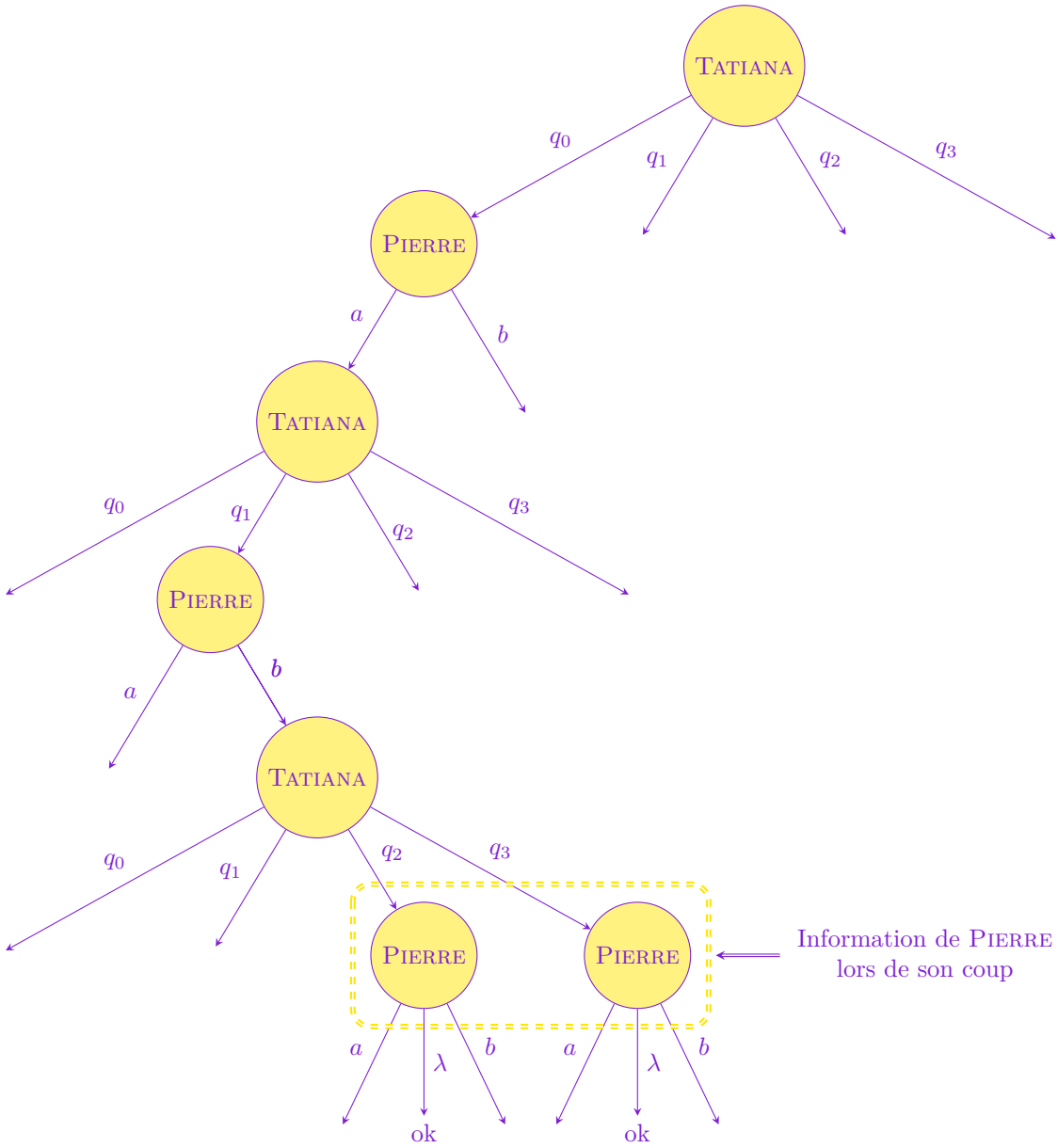

Figure 6 – Représentation Extensive du Jeu à Information Imparfaite de l'Automate figure 5

2 Jeux et Grammaires

On s'intéresse désormais à une nouvelle définition de jeu, les jeux hors-contexte, qu'on va écrire avec des automates à pile (et donc des grammaires), et on verra ensuite une application à deux jeux de cartes particuliers : le UNO et le TEXAS HOLD'EM POKER.

2.1 Jeux Hors-Contexte

Définition 2.1: Jeu Hors-Contexte

Un Jeu Hors-Contexte est un triplet $G = \langle \Sigma, R, T \rangle$ où Σ est un alphabet fini, $R \subseteq \Sigma \times \Sigma^+$ est un ensemble fini de règles et T est un automate représentant un langage régulier cible.

- Une partie du jeu G est jouée par deux joueurs TATIANA et PIERRE. Une partie consiste, à chaque étape, à appliquer une des règles de R à un mot donné sur Σ. A chaque étape, TATIANA choisit une position dans le mot et PIERRE choisit une règle de R associée.
- Un état du jeu C est une paire (w, i) où w est un mot et $i \leq |w|$ est la position courante. Un choix de position de la configuration $(a_1 \dots a_n, i)$ est un entier $j \leq n$, un choix de règle consiste à remplacer a_j par un mot u tel qu'on a $a_j \to u$ dans R. On appelle configuration résultante la configuration $(a_1 \dots a_{j-1} u a_{j+1} \dots a_n, j)$ obtenue.
- Une partie sur w commence dans la configuration initiale $C_0 = (w, 1)$ et s'arrête ou bien lorsque le mot résultat est dans $L(T)$ auquel cas TATIANA gagne ou bien lorsque la position choisie par TATIANA est un terminal auquel cas PIERRE gagne.

On peut alors de même définir la notion de stratégie gagnante :

Définition 2.2: Stratégie Gagnante

On dit que TATIANA a une stratégie gagnante en (w, i) lorsque quelque soient les coups de PIERRE, une configuration de $L(T)$ est atteinte en un nombre fini de coups. On dit que TATIANA gagne (G, w) si TATIANA a une stratégie gagnante dans G sur w.

2.2 Jeux Hors-Contexte, Machines de Turing et Automates à Pile

On commence déjà par démontrer un résultat sur l'expressivité des jeux hors-contexte.

Théorème 2.3: Reconnaissance de la Victoire

Soit M une machine de Turing bornée en espace par $s(n)$. On peut construire un jeu de sorte que pour tout mot $w = a_1 \dots a_n$, on ait :

$$M \text{ accepte } w \Leftrightarrow \text{TATIANA gagne } (G, \$q_0 a_1 \dots a_n \sqcup^{s(n)-n} \#)$$

Démonstration. On ne donne ici que le squelette de la preuve qui peut être retrouvée dans une forme plus forte dans [3] :

- On introduit d'abord des extensions des règles autorisées dans les jeux.
- On montre ensuite que tout jeu étendu est équivalent à un certain jeu non-étendu.
- On construit alors un jeu simulant une machine alternante en représentant chaque étape de calcul par plusieurs étapes de jeu.
- On conclut alors puisque les machines alternantes permettent d'exprimer les mêmes langages que toute machine de Turing.

■

On définit maintenant, de la même manière que les machines de Turing alternantes, les systèmes à pile alternants :

Définition 2.4: Système à Pile Alternant

Un système à pile alternant est un quadruplet $\mathcal{P} = \langle S = S_T \cup S_P, \Gamma, \delta, F\rangle$ i.e. un automate à pile sans entrée avec des états existentiels et universels. On note ici les configurations de cet automate entre crochets $[q, u]$ pour les distinguer des configurations d'un jeu. On dit qu'une configuration est gagnante si Tatiana peut toujours atteindre une configuration finale sur le jeu, quels que soient les choix de Pierre.

On obtient alors le théorème suivant, qui donne l'équivalence d'expressivité entre les automates à pile (ceux-ci étant équivalents à des automates à pile alternants) et les jeux hors-contexte au sens de la victoire :

Théorème 2.5: Equivalence Hors-Contexte

1. Étant donné un système à pile $\mathcal{P}$, on peut construire un jeu hors-contexte $G = \langle \Sigma, R, T\rangle$ où T est déterministe de sorte qu'une configuration $[q, a_1 \dots a_n]$ est gagnante dans $\mathcal{P}$ si et seulement si Tatiana gagne G sur $w = q, a_1 \dots a_n$.
2. Étant donné un jeu hors-contexte $G = \langle \Sigma, R, T\rangle$ où T est déterministe d'état initial q_0, on peut construire un système à pile $\mathcal{P}$ tel que Tatiana gagne G sur w si et seulement si $[q_0, w\$]$ est gagnante dans $\mathcal{P}$.

Démonstration. 1. On ne donne qu'un squelette de la (longue) preuve de ce résultat qui peut être retrouvée dans [3].

- On suppose sans perte de généralité que $\mathcal{P} = \langle S = S_T \cup S_P, \Gamma, \delta, F\rangle$ modifie la pile d'au plus un caractère.
- On définit un jeu pour lequel les configurations $[q, a_1 \dots a_n]$ de $\mathcal{P}$ correspondent aux configurations $\left(u, \boxed{q, a_1} \dots a_n\right)$ du jeu, où u est un mot sur les états de $\mathcal{P}$ auxquels on ajoute des symboles vides.
- On simule alors les étapes d'ajout/de retrait de la pile en ajoutant des règles.

2. On pose $T = (Q, \Sigma, \delta_T, q_0, F_T)$. On définit alors $\mathcal{P} = (S = S_T \cup S_P, \Sigma \cup \{\$\}, \delta, \{f\})$ où :
   - $S_T = Q \cup \{f\}$ et $S_P = \bar{Q}$ est une copie de Q disjointe de S_T.
   - Pour chaque paire $(q, a) \in Q \times \Sigma$, il y a des transitions $\delta(q, a) = \{(\delta_T(q, a), \varepsilon), (\bar{q}, a)\}$. Ici, les transitions $(\delta_T(q, a), \varepsilon)$ correspondent au cas où Tatiana ignore la position actuelle, tandis que les transitions $(\bar{q}, a)$ correspondent au cas où Tatiana choisit cette position. En effet, dans le second cas Tatiana *rend* la main à Pierre.
   - Pour chaque paire $(\bar{q}, a) \in \bar{Q} \times \Sigma$, et pour chaque règle $a \to u$ de R il y a une transition (q, u) dans $\delta(\bar{q}, a)$. Ces transitions correspondent au choix de la règle de R par Pierre.
   - Il y a une transition, pour chaque état q acceptant de T, $(f, \$) \in \delta(q, \$)$

   On vérifie alors que Tatiana gagne (G, w) si et seulement si $[q_0, w\$]$ est gagnant dans $\mathcal{P}$. En effet, pour chaque configuration $[q, u\$]$, l'état q est atteint par T sur le préfixe déjà joué dans G et u est le suffixe restant dans G.

■

On peut par ailleurs montrer (voir [3]), que toutes ces réductions se font en temps polynomial.

2.3 Application aux Jeux de Cartes

On va maintenant proposer une application de la définition par des grammaires des jeux à deux jeux de cartes particuliers : le Uno [2] et le Texas Hold'Em Poker. Suivant [2], commençons par une série de définitions d'un langage de description de jeux de cartes :

2. Le Mao n'ayant pas un nombre fixe de règles

TABLE 2 – Antécédents Conditionnels et leurs Conditions Associées

Antécédent	Condition	Exemple
`jetons`, KA, `comparaison`, KB	Compare les montants de jetons en KA et en KB selon la règle `comparaison`.	`jetons`, $K01$, $\geq$, $K11$
`jeu`, LA, `comparaison`, LB	Compare la carte en jeu en LA et LB selon `comparaison`.	`jeu`, $H0 + T0$, $<$, $H1 + T0$
`somme`, LA, `comparaison`, LB	Compare la somme des cartes en LA et la somme des cartes en LB selon `comparaison`.	`somme`, $H0$, $=$, $H1$
`a`, `combinaison`	Vérifie si le joueur courant possède la `combinaison` donnée dans sa main.	`a`, 2_3
`pioche`	Enlève une carte du haut de la pioche et la place dans la main du joueur courant	`pioche`
`montrer`, `restriction`, LA	Montre un ensemble de cartes de la main du joueur courant qui vérifie, comparé à une carte de LA, une `restriction` dans $(<, >, =, \leq, \geq, \textit{same suit}, \textit{same number})$.	`montrer`, `carte de la même couleur`, $T0$.
λ	Pas de condition	λ

Définition 2.6: Langage de Description

- On note P l'ensemble des joueurs.
- On appelle *emplacement de carte* une position notée LA dans le jeu où un nombre de cartes peut être posé. Le UNO possède les emplacements suivants :
  - P mains H, une par joueur, notées HI, et on notera HX la main du joueur courant, et HA les mains de tous les joueurs.
  - Un jeu ordonné de cartes $D = \{X_Y \mid (X, Y) \in [\![1, 13]\!] \times [\![1, 4]\!]\}$, qu'on peut séparer en sous-ensembles si besoin pour représenter les couleurs.
  - Un certain nombre d'emplacements TK sur la table.
- On définit de même des emplacements notés KA de jetons pour les jeux à mise. Chaque joueur i en possède deux notés $KI0$ et $KI1$.
- On décompose le jeu en plusieurs phases, qui se déroulent, par joueur, en rebouclant (après la dernière phase, on reprend la première). À chaque phase, durant son tour, un joueur peut jouer un nombre variable de règles. Son tour s'arrête lorsque :
  1. Il a `fini`, il revient alors en jeu à la prochaine phase.
  2. Il passe à la `suite`, il revient alors en jeu, dans la même phase, une fois que les tours des autres joueurs sont finis.
  3. Il est `sorti` du jeu, il a perdu la partie et ne peut plus jouer [a].
  4. Il a `gagné` la partie, auquel cas la partie s'arrête.

  Une phase se termine lorsque tous les joueurs ont `fini` ou sont `sortis`. Tant qu'il reste des joueurs à la `suite`, on reboucle sur la liste des joueurs.
- Un jeu s'arrête lorsque toutes les phases ont été jouées ou que tous les joueurs sont `sortis`.

a. Dommage pour lui...

On peut alors plus aisément représenter un jeu de cartes par une grammaire. On va toutefois se donner des règles *conditionnelles* (en réalité des terminaux), qu'il suffirait de dupliquer pour toute paire de cartes/de localisations dans la table 2

TABLE 3 – Résultats Possibles

Conséquent	Action	Exemple
`prendre`, $LA \times n$, f	Pioche n cartes face f depuis LA	`prendre`, $D \times 2$, `cachée`
`poser`, $LA \times n$, `restriction`, f	Pose n cartes face f sur LA si ces cartes vérifient une `restriction`.	`poser`, $T0 \times 1$, $>$, `visible`
`parier`, `restriction`, KX	Parie un nombre de jetons qui vérifie `restriction` quand comparé avec KX	`parier`, $>$, $K01$
`gagner`, KX	Remporte les jetons de KX	`gagner`, $K11$
`jouer`	Place les cartes spécifiées dans l'antécédent au lieu spécifié par l'antécédent	-
`suite`	Le joueur passe en statut `suite`	-
`fini`	Le joueur passe en statut `fini`	-
`sorti`	Le joueur passe en statut `sorti`	-
`gagne`	Le joueur gagne et la partie s'arrête	-
`fin`	La partie s'arrête	-

On ajoutera, lorsqu'une règle ne peut être jouée qu'une fois le marqueur `uniq`, et lorsqu'une règle doit être joué une et une unique fois le marqueur `oblig` au début de la règle. On définit alors les résultats possibles pour les règles dans la table 3

On a ainsi défini dans 2 et 3 des outils permettant de définir aisément une grammaire de certains jeux, puisqu'il est clair que chacune de ces règles peut, à duplication pour ajuster les valeurs des comparaisons près, se ramener à une règle de production :
Pour la règle « `montrer`, `carte de la même couleur`, $T0$, `jouer` », on peut l'écrire comme 636 règles de production différentes, puisqu'on peut toujours réordonner dans une grammaire les lettres d'un mot (on peut aisément générer les transpositions entre voisins) :

$$\begin{aligned} &X\#H0\#H1\#\ldots\#H(X-1)\#V1_{i_1}\ldots VK_i\#H(X+1)\#HN\#V_i \\ &\quad\longrightarrow (X+1)\#H0\#H1\#\ldots\#H(X-1)\#V1_{i_1}\ldots V(K-1)_{i_{k-1}}\#H(X+1)\#HN\#VK_i \end{aligned} \quad (1)$$

On encode ici l'état de la partie avec l'état des mains et la dernière carte posée sur le tas $T0$ ainsi que le numéro du joueur en cours, ce qui permet de se repérer dans la chaîne de caractères. Ici, il ne s'agit par réellement d'une règle de production, mais de plusieurs combinées :

- Une *récupérant* la main HX
- Une supprimant la dernière carte d'une main.
- Une remplaçant la carte V_i par cette dernière carte.
- Une modifiant la valeur du prochain joueur à jouer.

Il est clair qu'on peut modifier ces règles avec des non-terminaux pour que les règles s'emboîtent correctement.
Il suffit alors de dupliquer pour les 52 valeurs de V_i possibles et les, à chaque fois, 12 valeurs de VK_i possibles. On donne dans la suite des notes d'applications à deux jeux particuliers, mais le travail

2.3.1 Uno ![3]

Voici, le moment que vous attendiez avec impatience depuis le début : une grammaire du UNO ! On pourrait séparer le jeu en deux phases, une phase de préparation et une phase de jeu, mais on

3. Je n'ai plus qu'une sous-partie dans ma main !

TABLE 4 – Règles de la Grammaire du UNO

`montrer`, `même couleur`, $T0$, `jouer`
`montrer`, `même valeur`, $T0$, `jouer`
`piocher`, `suite`
`oblig_a`, λ, `gagne`

ne va se concentrer que sur la seconde.
On rappelle que tous les joueurs commencent avec 7 cartes et que l'unique emplacement de carte sur la table $T0$ en contient à l'origine une. On considère les règles suivantes. A son tour, chaque joueur peut :

- Jouer une carte de la même couleur que la carte en $T0$
- Jouer une carte de la même valeur que la carte en $T0$
- Piocher une carte et passer son tour

Une fois qu'un joueur n'a plus de carte (ce qu'on peut repérer lorsque deux symboles # sont côte à côte), il gagne la partie. Evidemment, la partie où les joueurs crient UNO et Contre-UNO est hors d'atteinte de notre système. On va ici se limiter à des cartes sans effets, pour simplifier l'écriture, bien qu'il serait possible de les prendre en compte. C'est ce qu'on a écrit dans la table 4.

2.3.2 Texas Hold'Em Poker

Dans ce jeu, on va mettre en place 12 phases du jeu qui vont couvrir les étapes principales du jeu (pré-flop, flop, turn, river et showdown), ainsi que des phases pour gérer les mises, ce qu'on détaille dans la table 5

Pour finir voici une réminiscence de la supposée troisième partie sur le game design et l'utilisations d'automates pour observer le déroulé d'une partie.
[4] propose une méthode de ce style, tandis que [7] propose d'utiliser des délais pour décrire, en temps réel, le déroulé d'une partie.

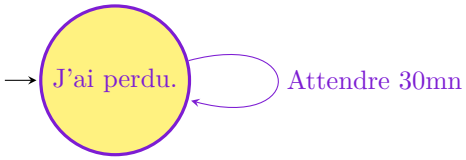

FIGURE 7 – LE JEU

Références

[1] A-Games : using game-like representation for representing finite automatas *Cleyton Slaviero, Edward Hermann Haeusler*

[2] A Card Game Description Language *Jose M. Font, Tobias Mahlmann, Daniel Manrique, and Julian Togelius*

[3] Active Context-Free Games *Anca Muscholl, Thomas Schwentick, and Luc Segoufin*

[4] Computing Game Design with Automata Theory *Noman Sohaib Qureshi, Hassan Mushtaq, Muhammad Shehzad Aslam, Muhammad Ahsan, Mohsin Ali and Muhammad Aqib Atta*

[5] Summary for Context Free Games *Lukáš Holík, Roland Meyer and Sebastian Muskalla*

TABLE 5 – Règles de la Grammaire du TEXAS HOLD'EM POKER

Nom	Règle

Phase 1
(Premier pari)
	`oblig_nocond_parier`, λ, λ
	`nodonc_parier`, $=$, $KA1$
	`uniq_nocond`, `next`
	`uniq_nocond_fini`

Phase 2 (Vérifier paris)	`oblig_jetons`, $KX1$, $<$, $KA1$ `sorti`
Phrase 3 (Flop)	`piocher`, $D \times 1$ `poser`, $D \times 3$, $T0$

Phase 4
(Deuxième pari)
	`nocond_parier`, $\geq$, $KA1$
	`uniq_nocond_suite`
	`uniq_nocond_fini`

Phase 5 (Vérifier paris)	`oblig_jetons`, $KX1$, $<$, K `sorti`
Phase 6 (Turn)	`piocher`, $D \times 1$ `poser`, $D \times 1$, $T0$

Phase 7
(Troisième pari)
	`nodonc_parier`, $\geq$, $KA1$
	`uniq_nocond_suite`
	`uniq_nocond_fini`

Phase 8 (Vérifier paris)	`oblig_jetons`, $KX1$, $<$, $KA1$ `sorti`
Phase 9 (River)	`piocher`, $D \times 1$ `poser`, $D \times 1$, $T0$

Phase 10
(Dernier Pari)
	`nodonc_parier`, $\geq$, $KA1$
	`uniq_nocond_suite`
	`uniq_nocond_fini`

Phase 11 (Vérifier paris)	`oblig_jeton`, $KX1$, $<$, $KA1$ `sorti`
Phase 12 (Showdown)	`oblig_jouer`, $HX + T0$, $>$, $HA + T0$`_gain`, $KA1$

[6] Permutational Grammar for free word order languages, *Mats Eeg-Olofsson and Bengt Sigurd*

[7] Timed Automata for Video Games and Interaction, *Jaime Arias, Raphael Marczak, Myriam Desainte-Catherine*