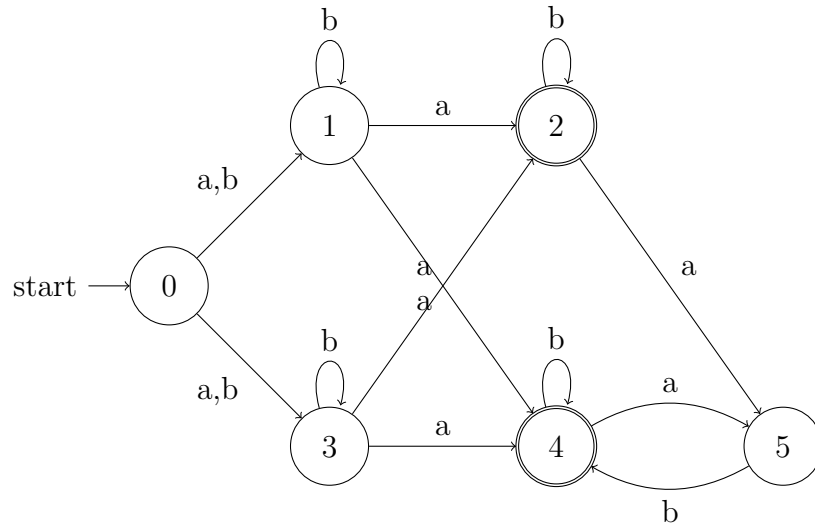


# 1 Automates à parité et automates de Buchi

## 1.1 Exemple

1. Le langage reconnu par cet automate est  $b^*a(b^*ab)^*b^\omega$ .  
Une formule LTL représentant ce langage est :  $b\mathbf{U}a\wedge\mathbf{X}[b\mathbf{U}(a\wedge\mathbf{X}b)]\mathbf{U}Gb$
2. Automate de Büchi correspondant :



## 1.2 Parité et Büchi

- (a) Pour transformer un automate de Büchi à un automate de parité en conservant le nombre d'états et les transitions, il suffit de transformer les labels des états où l'on doit passer infiniment souvent en "2" et les autres en "1".
- (b) Pour passer d'un automate de parité à un automate de Büchi, on utilise l'algorithme suivant
  - On crée un automate  $\mathcal{A}_{2i}$  qui vérifie que la couleur  $\max(\chi(\text{Inf}(p))) = 2i$ .

- A partir de  $\mathcal{A}_{2i}$ , on construit  $\mathcal{A}'_{2i}$ , qui est l'automate initial sans les états dont la couleur est supérieure à  $2i$ . Les états de couleur  $2i$  sont finaux.
- On relie  $\mathcal{A}$  et  $\mathcal{A}'_{2i}$  par des transitions  $= p \rightarrow^a q$  qui existaient dans  $\mathcal{A}$ .
- On effectue cette manoeuvre pour toutes les couleurs inférieures à  $2i$ .

On a donc au maximum  $n^2$  états dans l'automate de Büchi équivalent à l'automate de parité,  $n$  étant le nombre d'états dans l'automate d'origine.

## 2 Jeux de parité faible

### 2.1 Transformation en jeu à parties infinies

Sur l'arène  $G$ , le jeu s'arrête lorsque l'on tombe dans un sommet-puits. Si la couleur de ce sommet-puits est paire, le jeu est gagné par  $J_0$ , sinon il est gagné par  $J_1$ .

On note  $P$  l'ensemble des sommets-puits :  $P = P_0 \cup P_1$ , avec  $P_0 \subseteq V_0$  et  $P_1 \subseteq V_1$  ; et  $k$  la couleur maximale de  $G$ .

Pour transformer notre jeu en un jeu à parties infinies, on remarque qu'il faudra passer un nombre infini de fois dans un des sommets-puits. Pour ce faire, on rajoute à chaque sommet  $v \in P$  une arête  $\{v \rightarrow v\}$ .

On ajoute aussi 2 nouvelles couleurs,  $i$  et  $j$ , ( $i$  paire et  $j$  impaire), n'existant pas dans  $G$  et maximales ( $i > k, j > k$ ). Les sommets de  $P_0$  auront alors la couleur  $i$  et les sommets de  $P_1$  auront la couleur  $j$ .

On aura bien sur  $G'$  un jeu à parties infinies, dont les régions gagnantes déterminent automatiquement celles de  $G$  (ce sont les mêmes sommets).

### 2.2 Exemple

- On calcule en premier lieu les attracteurs de  $d \in \text{Win}_1$  :  $\text{Attr}_1(d) = b$ .
- On peut alors calculer l'ensemble  $V' = V \setminus \{b, d\}$ , qui est un piège pour  $J_1$ , donc un sous jeu pour  $J_0$ .
- Dans ce sous jeu,  $\text{Win}_0 = \{g\}$  et  $\text{Attr}_0(g) = \{c, f\}$ .
- Dans l'ensemble  $V = V \setminus \{b, c, d, f, g\}$ , les sommets  $a$  et  $e$  appartiennent à  $J_1$ .

- On a alors  $W_0 = \{a, e, c, f, g\}$
- La stratégie positionnelle :

$$\begin{array}{lcl}
a & \rightarrow & a \\
c & \rightarrow & g \\
e & \rightarrow & a \\
f & \rightarrow & g \\
g & \rightarrow & f
\end{array}$$

est gagnante pour  $J_0$ .

### 2.3 Positions gagnantes et $\mu$ -calcul

- Jeu à 2 couleurs , 0 et 1 :  $Win_0 = \nu X \cdot C_0 \wedge ((\diamond X \wedge p_0) \vee (\Box W \wedge p_1))$
- Jeu à 3 couleurs, 0, 1, et 2 :  $Win_0 = (\mu X \cdot C_2 \vee (\diamond X \wedge p_0) \vee (\Box X \wedge p_1))) \vee (\nu X \cdot C_0 \wedge (\diamond X \wedge p_0) \vee (\Box X \wedge p_1)))$
- Jeu à  $k$  couleurs :
  - On définit pour  $k \geq i > 0$  :  $A_i = \mu X \cdot \neg A_{i-1} \wedge (C_i \vee (\diamond X \wedge p_0) \vee (\Box W \wedge p_1))$
  - $A_0 = \nu X \cdot C_0 \wedge ((\diamond X \wedge p_0) \vee (\Box W \wedge p_1))$
  - Si  $k$  est pair :  $Win_0 = A_k \vee A_{k-2} \vee \dots \vee A_{2i} \vee \dots \vee A_2 \vee A_0$
  - Si  $k$  est impair :  $Win_0 = A_{k-1} \vee A_{k-3} \vee \dots \vee A_{2i-1} \vee \dots \vee A_3 \vee A_1$

### 2.4 Jeux de parité et jeux de Büchi

- Transformation d'un jeu de parité faible en jeu de Büchi. on pose  $n = |V|$  et  $k$  le nombre de couleurs de  $G$ .
  - Le graphe  $G'$  sera construit à partir de copies des sommets de  $G$ , copiés  $k$  fois. On numérote les copies  $G_0 \dots G_k$ . On a donc bien  $V \subseteq V'$
  - Les états terminaux seront ceux de la plus grande couleur de  $G_i$ ,  $i$  pair. Ce sont aussi ceux de  $G$ .
  - Les transitions seront remplacées par des transitions de  $G_i$  à  $G_{i+1}$  : Il y a une transition de  $u_i$  à  $v_{i+1}$  s'il existait une transition de  $u_i$  à  $v_i$  dans le graphe d'origine et si la couleur de  $v_{i+1}$  est supérieure à celle de  $u_i$ .
- Une stratégie positionnelle dans  $G$  le sera aussi dans  $G'$ , elle ne change pas, il faut simplement prendre en compte les nouvelles transitions.

- Un vainqueur dans  $G'$  le sera aussi dans  $G$ , étant donné qu'on peut appliquer les mêmes stratégies.

## 3 Jeux de parité et algorithmes

### 3.1 Jeux de parité sur des arènes sans grands cycles

Les sommets gagnants dans une arène finie où les seuls cycles sont de longueur 1 sont :

- Les sommets-puits de couleur paire.
- Les sommets à partir desquels il n'est pas possible d'atteindre autre chose qu'un sommet-puits de couleur paire.
- Les sommets cycliques dont la seule arête n'étant pas  $(v, v)$  ne permet pas d'atteindre autre chose qu'un sommet-puits de couleur paire.

Pour calculer l'ensemble de ces sommets, on part de l'ensemble des sommets-puits de couleur paire, noté  $P$ , et on remonte pour parcourir tous leurs prédécesseurs. Pour calculer  $P$ , on utilise l'algorithme suivant :

```
Pour tout  $v$  dans  $V$ , faire :
  Si  $\text{Coul}(v) \% 2 == 0$  &&
     $[ (|\text{Succ}(v)| == 1) \ \&\& \ (\text{Succ}(v) == v))$ 
     $|| \ (|\text{Succ}(v)| == 0) ]$  :
     $P = P \cup \{v\}$  ;
```

Cet algorithme est en  $O(n)$  (avec  $n$  le nombre de sommets), étant donné que l'on parcourt une seule fois la liste des sommets. De plus, on peut espérer qu'une structure de donnée adaptée nous permette de connaître la couleur d'un sommet, d'avoir accès à sa liste de successeurs en  $O(1)$ . Étant donné qu'on ne parcourt la liste des successeurs que si celle-ci contient un unique élément, cela ne fait pas augmenter la complexité de l'algorithme.

Pour calculer  $\text{Attr}_0^{|V|}(P)$ , on utilise l'algorithme suivant :

```
Attr0 =  $\emptyset$  ; // Région gagnante dont les sommets ont déjà été traités
B =  $\emptyset$  ; // sommets traités n'appartenant pas à Attr0
V =  $V \setminus P$  ; // sommets non traités
TMP =  $P$  ; // sommets à traiter appartenant à la région gagnante
TMP2 =  $\emptyset$  ;
int card ;
```

```

Tant que  $TMP \neq \emptyset$  :
   $TMP2 = \emptyset$ ;
  Pour tout  $u$  dans  $TMP$  faire :
    Pour tout  $v$  dans  $Pred(u)$  &&  $v$  dans  $V$  faire :
       $card = 0$ ;
      Pour tout  $w$  dans  $Succ(v)$  faire :
        Si ( $w$  dans  $B$ ) :
           $B = B \cup \{v\}$ ;
           $V = V \setminus \{v\}$ ;
          Break;
        Si ( $w$  dans  $TMP$ ) || ( $w$  dans  $Attr0$ ) || ( $w == v$ ) :
           $card++$ ;
      Si  $card == |Succ(v)|$  // Tous les successeurs de  $v$ 
                          // sont dans la région gagnante
         $TMP2 = TMP2 \cup \{v\}$ ;
         $V = V \setminus \{v\}$ ;
       $Attr0 = Attr0 \cup \{u\}$ ;
       $TMP = TMP \setminus \{u\}$ ;
   $TMP = TMP2$ ;
retourner  $Attr0$ ;

```

Le principe de cet algorithme est d'utiliser  $TMP$  pour stocker les sommets qui appartiennent à la région gagnante mais dont les prédécesseurs n'ont pas encore été traités. Pour tous les sommets de  $TMP$ , on traite les prédécesseurs qui sont dans  $V$  (si les prédécesseurs sont dans  $TMP$  ou  $Attr_0$ , ce n'est pas la peine). On laisse lesdits sommets dans  $V$  si on n'arrive pas à statuer sur eux (ses successeurs appartiennent à  $V$  et la région gagnante), on les accepte dans la région gagnante si tous leurs successeurs y sont déjà et on les envoie dans  $B$ , région gagnante de  $J_1$ , si un de ses successeurs est dans  $B$ . Cela permet de traiter les sommets "indécis" une fois que l'on a pu classer ses successeurs.

Un sommet peut être traité au maximum  $d^-$  fois,  $d^-$  étant le nombre d'arêtes sortantes. Si l'on nomme  $D$  la moyenne des degrés sortants du graphe, on peut dire que l'algorithme est en  $O(Dn) \approx O(n)$ , en utilisant une structure de données adaptée qui permet d'avoir accès aux Successeurs et Prédécesseurs en  $O(1)$ .

L'algorithme s'arrête lorsque  $TMP$  est vide, car c'est le moment où tous les sommets font soit partie de  $Attr_0$  soit n'en feront jamais partie.