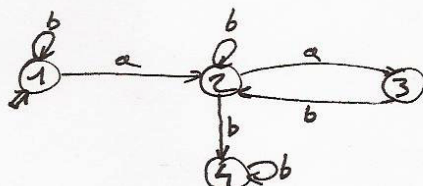


1.1.1

$$\mathcal{L}(A) = b^* a (b^* a b)^* b^\omega$$

formule LTL: $b \vee [a \wedge X((b \vee (a \wedge X b)) \vee G b)]$

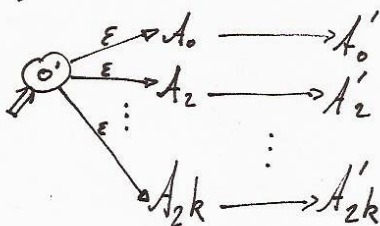
1.1.2



$$F = \{4\}$$

1.2.(a) On colorie les états de \mathcal{Q} . Si $q \notin F$,
 $X(q) = 1$; si $q \in F$, $X(q) = 2$.

(b) Soit A un automate de parité à $2k$ états.
 $\{0, 1, \dots, 2k\}$
Soit A' l'automate :



On devine
la couleur max
et on compute
l'entrée dans
la copie de A
correspondante

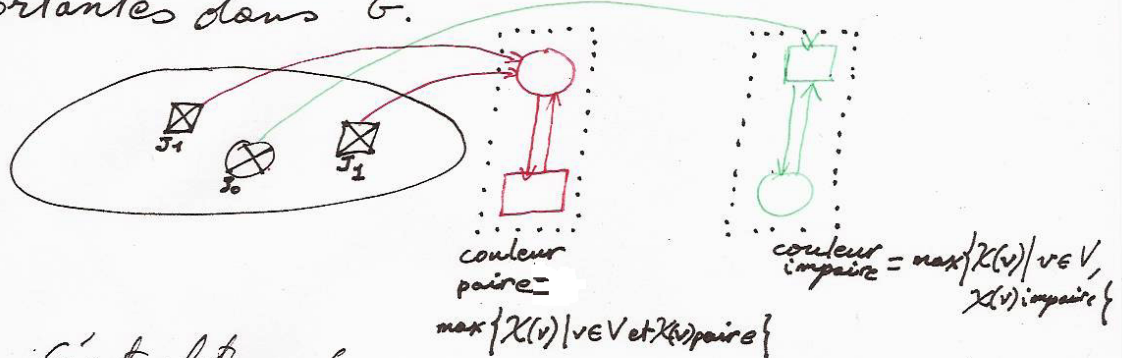
A'_i est la restriction de A à tous les états $\leq i$
toutes les transitions $p \xrightarrow{a} q$ où q est
inférieur à i sont reproduites de A_i vers i
 $p \xrightarrow{a} q$.
Cette étape devine à partir de
quelle étape on n'aura plus
de passage vers des états $> i$.

$$F = \bigcup \{q' \mid \forall i \in \{0, \dots, 2k\}, q' \in A'_i \text{ et } X_A(q) = i\}$$

De toute évidence, $|Q'| \leq \sum_{i=0}^{2k} |Q| + 1$.

On a bien obtenu A' un automate de
Büchi équivalent à A dont la taille est
polynomiale en $|Q|$.

2.1 Soit \mathcal{G} une arène (\mathcal{J}_0 rond, \mathcal{J}_1 carré)
on considère les états puits (\boxtimes ou \otimes) sans arêtes sortantes dans \mathcal{G} .



C'est obtenu (comme sur la figure) on faisant sortir des états puits des transitions vers une situation d'alternance entre deux états dont le couleur garantit de conserver le même gagnant.

2.2 $d \in \text{Win}_1$

$$\text{Attr}_1(d) = \{d, b\}$$

$V' = V \setminus \{d, b\}$ est un piège pour \mathcal{J}_1 et donc un sous-jeu pour \mathcal{J}_0 .

$$g \in \text{Win}_0|_{V'}$$

$$\text{Attr}_0(g) = \{g, c, f\}$$

$V'' = V' \setminus \{g, c, f\}$ piège pour \mathcal{J}_0 donc sous jeu pour \mathcal{J}_1 .

$$\text{Win}_1|_{V''} = \emptyset ; \text{Win}_0|_{V''} = \{a, e\}$$

donc

$$V_{\text{Win}_0} = \{a, c, e, f, g\} \quad V_{\text{Win}_1} = \{b, d\}$$

\mathcal{J}_0 doit appliquer une stratégie positionnelle d'attracteur:

$$\begin{array}{l} a \rightarrow c \\ f \rightarrow g \\ g \rightarrow f \\ d \rightarrow a \end{array}$$

2.3

Un jeu de parité faible admet une stratégie positionnelle donc l'ensemble des sommets depuis lesquels \exists une stratégie gagnante est un p.f. de $t_0(X) = (p_0 \wedge \Diamond X) \vee (p_1 \wedge \Box X)$

- si tous les sommets sont de couleur 0 ou 1
 \exists gagne si \exists est bloqué ou si toute la partie (infinie) est dans C_0

On a donc

$$Win_0 = \forall X. C_0 \wedge t_0(X)$$

- Si 0, 1 et 2

\exists gagne si la partie reste dans C_0 ou si elle passe par C_2 et est infinie ou si \exists est bloqué.

$$Win_0 = \forall X. C_0 \wedge t_0(X) \vee \mu X. C_2 \vee \Box t_0(X)$$

- On généralise à k couleurs

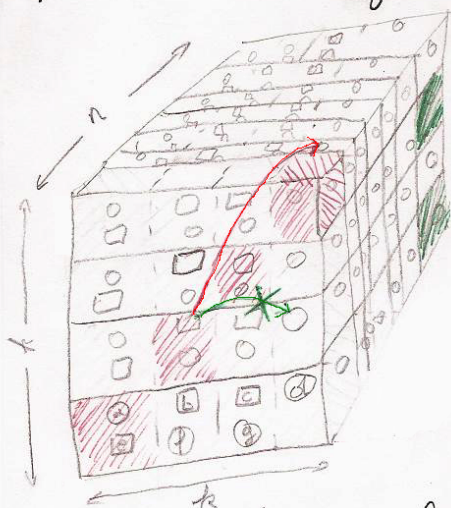
$$Win_0 = \bigvee_{i \in \{1, 2, \dots, \lfloor \frac{k}{2} \rfloor\}} \left(\forall X. \left(\bigvee_{j \leq i-1} C_j \right) \wedge t_0(X) \wedge \bigwedge_{j \geq i} \mu X. C_j \vee \Box t_0(X) \right)$$

2.4

Soit n = le nombre d'états de G .

Soit k = le nombre de couleurs de l'arène ($k \leq n$, v.B.)

Remarque les stratégies étant positionnelles, la plus grande couleur atteinte au cours d'une partie l'est déjà à l'issue du $(n-1)$ -ème coup.



On construit un graphe G' à partir de $n \times k$ copies des états de G .

Chaque transition est remplacée par une nouvelle qui fait "monter" au niveau si la

couleur cible est supérieure strictement à "l'étage" en cours,

et qui fait "reculer" d'un niveau de profondeur (sauf dernier niveau).

On déclare terminaux les états des niveaux paires de la dernière tranche (en vert sur le dessin).

• la hauteur représente la couleur maximale atteinte.

• la profondeur compte les coups avant de décider qu'on atteindra pas de couleur supérieure.

Si on considère le sous ensemble de V marqué en rouge, isomorphe à V , la condition d'équivalence est respectée en conservant les stratégies de départ (à remplacement de transition près).

Les parties finies restent finies et le vainqueur est inchangé.

La construction est, de toute évidence, polynomiale.

3. On utilise une pile p de sommets, variable globale initialement vide.

```

Algo VERIF ( $A, \sigma, v$ ) booléen
début
  si ( $v \in p$ )
    alors si ( $\max \{X(u) \mid u \text{ entre } v \text{ et le sommet de } p \text{ est paire}\}$ )
      /* on a détecté un cycle dans le graphe */
      alors dépiler();
      retourner VRAI;
      sinon retourner FAUX;
    fin si
    sinon empiler ( $v$ );
    /*  $\sigma$  est gagnante en  $v$  si elle l'est dans les sommets visités antérieurement */
    si ( $v \in V_0$ )
      alors si ( $\text{VERIF}(A, \sigma, \sigma(v))$ )
        alors dépiler();
        retourner VRAI;
        sinon retourner FAUX;
      fin si
      sinon pour  $w$  successeur de  $v$  faire
        si ( $\neg \text{VERIF}(A, \sigma, w)$ )
          alors retourner faux;
        fin si
      fin pour
      dépiler()
      renvoyer VRAI;
    fin si
  fin si
fin
  
```

Le nombre d'appels récursifs à VERIF est le nombre de sommet d'un arbre d'au plus n sommets et de degré $\leq n$. Le parcours de la pile se fait en temps $O(n)$. L'algorithme est en $O(n^3)$.

3.1

1: $Win_0 := \emptyset$; $Win_1 := \emptyset$

2: Pour tout $v \in V$ faire /* on remplit Win_0 et Win_1 avec les v */

si [$(X(v)$ est paire $\wedge (v \in V_0$ et v a une boucle) $\vee (v \in V_1$ et v n'a qu'une boucle)]
alors $Win_0 := Win_0 \cup \{v\}$ $\vee (v \in V_1 \wedge v$ sans arêtes sortantes)]

fin si

si [$(X(v)$ est impaire $\wedge (v \in V_1$ et v a une boucle) $\vee (v \in V_0$ et v n'a qu'une boucle)]
alors $Win_1 := Win_1 \cup \{v\}$ $\vee (v \in V_0 \wedge v$ sans arêtes sortantes)]

fin si

fin pour

3: Soit $F := V - (Win_0 \cup Win_1)$ une file.

4: tant que (F non vide) faire /* on calcule simultanément nos deux attracteurs */

$v :=$ supprimer tête (F);

si ($v \in V_0$ et a une arête vers un sommet de Win_0)

alors $Win_0 := Win_0 \cup \{v\}$; /* J_0 peut jouer vers Win_0 */

sinon si ($v \in V_1$ et toutes ses arêtes vont dans Win_0)

alors $Win_0 := Win_0 \cup \{v\}$; /* J_1 est forcé à jouer vers Win_0 */

sinon si ($v \in V_0$ et toutes ses arêtes vont dans Win_1)

alors $Win_1 := Win_1 \cup \{v\}$; /* J_0 forcé */

sinon si ($v \in V_1$ et a une arête vers Win_1)

alors $Win_1 := Win_1 \cup \{v\}$; /* J_1 peut jouer vers Win_1 */

sinon ajouter en queue (F, v); /* le sommet est à une distance > 1 des attracteurs */

fin si;

fin tant que

5: retourner Win_0 et Win_1 ;

Remarque Il est impossible de boucler infiniment dans la boucle tant que car F est vide ou contient un sommet à distance 1 des attracteurs (par construction).

La complexité est donc en $O(n^2)$.

