La synthèse de contrôleurs pour les systèmes réactifs

Synthèse vs. Vérification

- Problème de vérification:
  - On a un modèle d'un système composé d'un environnement et d'un contrôleur, et on veut prouver que le système respecte une certaine propriété

$$\underbrace{\boxed{\text{Environnement}} \;||\; \boxed{\text{Contrôleur}}}_{=\ \text{système}} \overset{?}{\models} \varphi$$

Synthèse vs. Vérification

- Problème de Synthèse:
  - On a un modèle de l'environnement, et on voudrait obtenir automatiquement un (modèle de) contrôleur qui assure que la propriété donnée est vérifiée

$$\text{Environnement} \parallel \text{Contrôleur}\,? \models \varphi$$

Synthèse vs. Vérification

- Les techniques de synthèse évitent de concevoir un contrôleur par essai-erreurs
- Le contrôleur obtenu est correct par construction.

$\boxed{\text{Environnement}} \;||\; \boxed{\text{Contrôleur}} \models \varphi$

Quelques rappels de vérification

Vérification - rappels

- Un technique courante consiste à voir un système à vérifier comme un objet qui génère des séquences d'actions
- Les propriétés à vérifier = des ensembles de bonnes séquences d'actions
- On veut prouver que l'ensemble des séquences du système est inclus dans l'ensemble des bonnes séquences

$$\mathrm{Seq(Sys)} \subseteq \mathrm{BonnesSéquences}$$

Vérification - rappels

- Exemple:
  - un système reçoit une requête (action `req`), effectue trois actions internes (action `i`) puis satisfait la requête (action `ack`) et recommence (un nombre fini de fois).
  - Ce système génère l'ensemble d'actions suivant:

$$\{(\texttt{req i i i ack})^j \mid j \geq 1\}$$

Vérification - rappels

- Exemple:
  - un système reçoit une requête (action `req`), effectue trois actions internes (action `i`) puis satisfait la requête (action `ack`) et recommence (un … i de fois).
  - Ce système génère l'ens… suivant:

$$\{(\mathtt{req\ i\ i\ i\ ack})^j \mid j \geq 1\}$$

w^j = j concaténations du mot w

Vérification - rappels

- Exemple:
  - Ce système doit satisfaire la propriété suivante:
    - Chaque requête est suivie d'une réponse
    - Les bonnes exécutions constituent l'ensemble:

$$\{a_1 a_2 \cdots a_j \cdots a_n \mid \forall j \geq 1 : a_j = \mathtt{req} \Rightarrow (\exists i > j : a_i = \mathtt{ack})\}$$

Vérification - rappels

- Exemple:
  - Il est "facile " de voir que:

$$\{(\texttt{req i i i ack})^j \mid j \geq 1\}$$

$$\subseteq$$

$$\{a_1 a_2 \cdots a_j \cdots a_n \mid \forall j \geq 1 : a_j = \texttt{req} \Rightarrow (\exists i > j : a_i = \texttt{ack})\}$$

Vérification - rappels

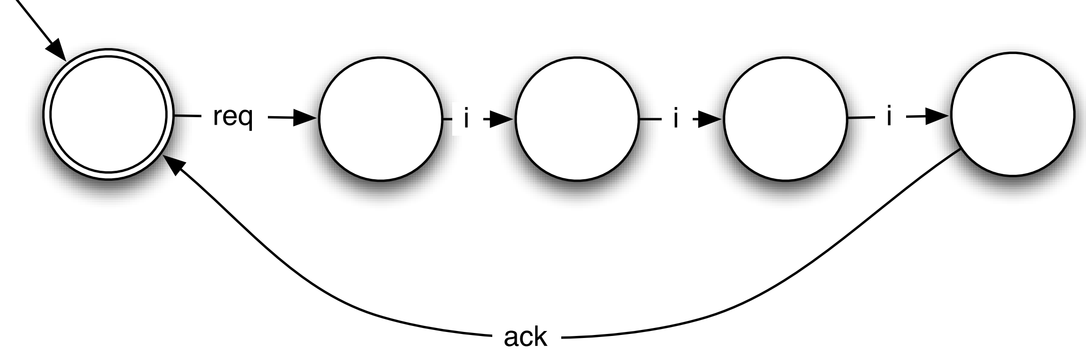
- Pour pouvoir effectuer cette vérification de façon algorithmique, on fait appel à la théorie des langages (cfr. cours de théorie des langages de T. Massart)
  - L'ensemble (fini) des actions du système est vu comme un alphabet Σ
  - L'ensemble des exécutions du système est un langage L_{sys} sur Σ
  - L'ensemble des bonne exécutions est un autre langage L_{prop} sur Σ
  - On veut déterminer si $L_{sys} \subseteq L_{prop}$

Vérification - rappels

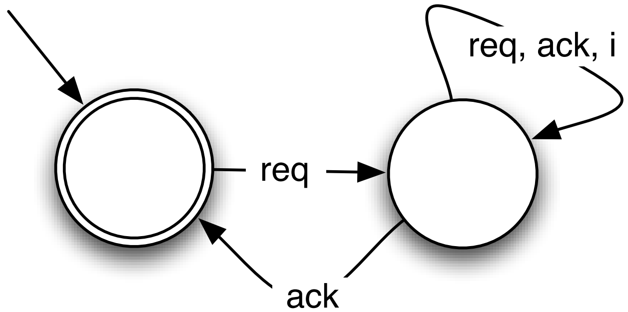
- On veut déterminer si $L_{sys} \subseteq L_{prop}$
    - Dans le cas où L_{sys} et L_{prop} sont des langages réguliers, ce problème est relativement simple:
    - on décrit les deux langages à l'aide d'automates finis A_{sys} et A_{prop}.
    - on utilise l'algorithme qui permet de déterminer si le langage d'un automate est inclus dans celui d'un autre: $L(A_{sys}) \subseteq L(A_{prop})$?

Vérification - rappels

- Exemple: $L(A_{sys}) \subseteq L(A_{prop})$? OUI
  - Système:



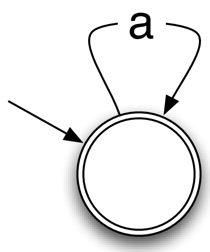
  - Propriété:



Vérification - rappels

- Les automates finis sont-ils suffisants dans le cadre des systèmes réactifs?
  - rappel: dans la définition “classique” (cfr. cours de théorie des langages), un automate fini définit:
    - Un ensemble potentiellement infini...
    - ...de mots finis

A

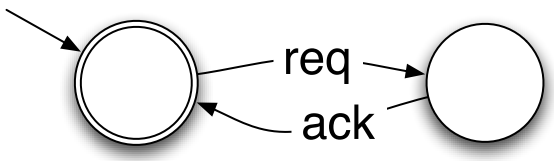


$$|L(A)| = \infty$$

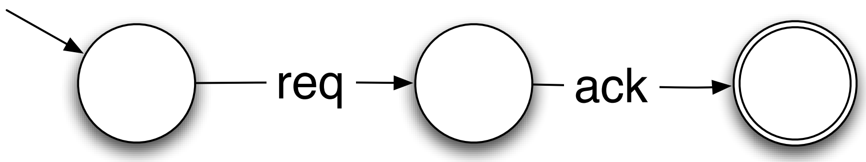
$$\forall\ w \in L(A): |w| \neq \infty$$

Vérification - rappels

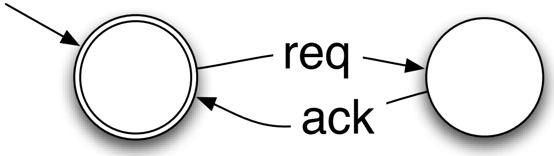
- Les automates finis sont-ils suffisants dans le cadre des systèmes réactifs ?
  - Considérons à nouveau la propriété "toute demande `req` est suivie d'une réponse `ack`"
  - Ce système simple respecte cette propriété:



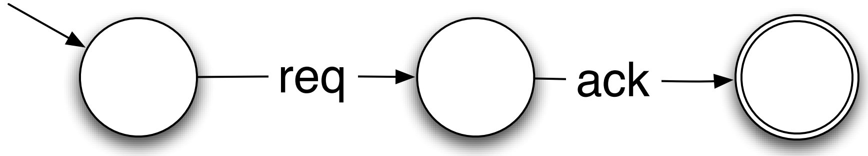
  - Celui-ci aussi:



Vérification - rappels



vs.



- Différence principale:
  - Le système de gauche permet un nombre quelconque de requêtes
    - Pour chaque i, l'automate accepte un mot de la forme (`req ack`)^{i}
  - Le système de droite limite le nombre de demandes.
    - Cela n'a pas de sens dans un système réactif !

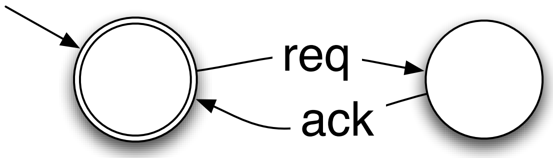
Vérification - rappels

- Dans un système réactif, cela n'a pas vraiment de sens de considérer des traces d'exécution finies
- Un système réactif peut être vu comme "ne s'arrêtant jamais"
  - e.g.: Après chaque requête traité, le système est à nouveau à même de recevoir une nouvelle requête, etc.

On va donc considérer des langages de mots infinis

Vérification - rappels

- Langages de mots infinis:
  - On conserve la structure de base d'automate
  - On lui donne une autre interprétation:
    - un mot est accepté ssi il étiquette un chemin infini qui traverse infiniment souvent un état accepteur.



$$L^{\omega}(A) = \{(\mathtt{req}\,\mathtt{ack})^{\omega}\}$$

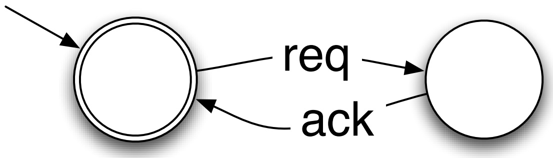
$$\neq$$

$$L(A) = \{(\mathtt{req}\,\mathtt{ack})^{i} \mid i \geq 0\}$$

Vérification - rappels

- Langages de mots infinis:
  - On conserve la structure de base d'automate
  - On lui donne une autre interprétation:
    - un mot est accepté ssi il étiquette u infini qui traverse infiniment souven accepteur.

w^{ω} = concaténation infinie de w



$$L^{\omega}(A) = \{(\mathtt{req}\,\mathtt{ack})^{\omega}\}$$

$$\neq$$

$$L(A) = \{(\mathtt{req}\,\mathtt{ack})^{i} \mid i \geq 0\}$$

Vérification - rappels

- Ces langages sont définis par des automates de Büchi:
  - Julius Richard Büchi (1924–1984) mathématicien et logicien Suisse.

J. Richard Büchi, 1983

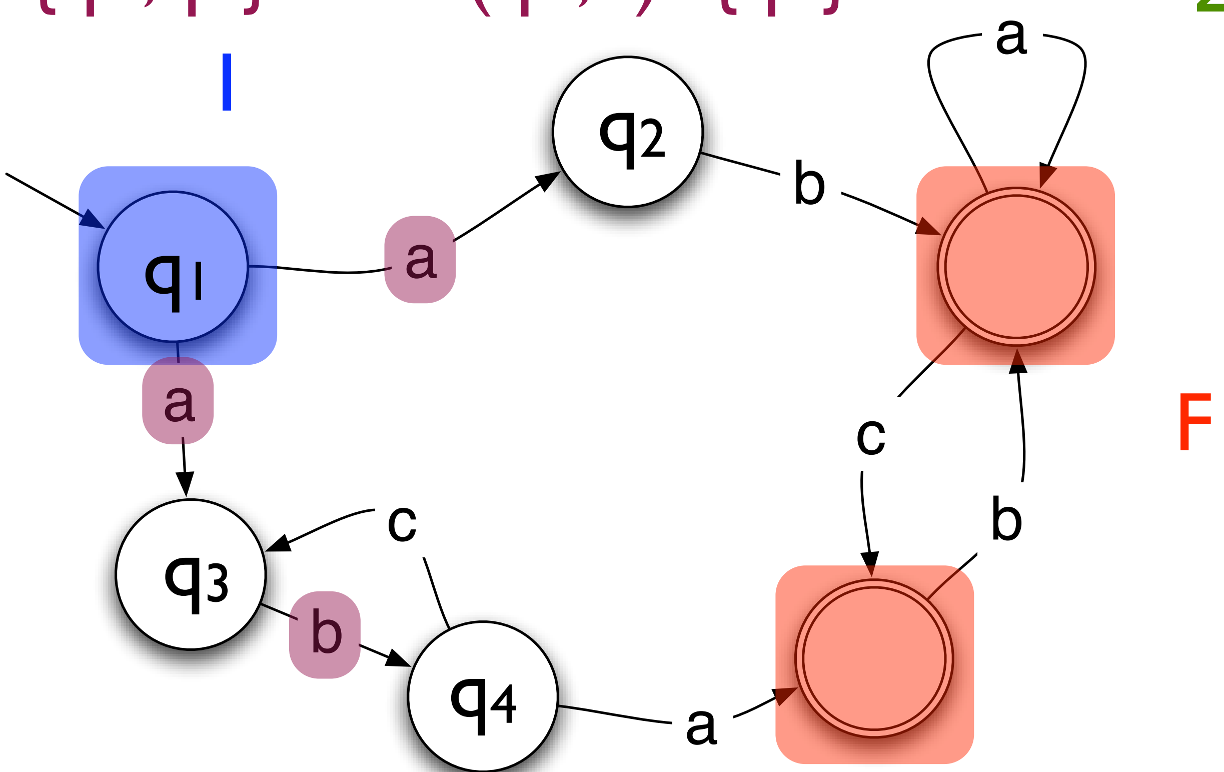
Vérification - rappels

- Définition: Un automate de Büchi est un tuple $A = \langle \Sigma, Q, I, R, F \rangle$ où:
  - Σ est un alphabet fini
  - Q est un ensemble fini d'états
  - $I \subseteq Q$ est un ensemble d'états initiaux
  - $R: Q \times \Sigma \rightarrow 2^Q$ est la relation de transition
  - $F \subseteq Q$ et un ensemble d'états finaux

Vérification - rappels

- Exemple: Un automate de Büchi est un tuple $A = \langle \Sigma, Q, I, R, F \rangle$

$R(q1,a)=\{q2,q3\}$ $R(q3,b)=\{q4\}$ $\Sigma=\{a,b,c\}$



Vérification - rappels

- Définition: Un chemin dans un automate de Büchi $A = \langle \Sigma, Q, I, R, F \rangle$ est une séquence infinie $q_1 q_2 \ldots q_j \ldots$

  d'états, telle que: pour tout $i \geq 1$, il existe une lettre a telle que $q_{i+1} \in R(q_i, a)$

- Définition: Etant donné un chemin p, on définit Inf(p) comme l'ensemble des états qui apparaissent infiniment souvent dans p

- Définition: Un chemin p est accepteur ssi $Inf(p) \cap F \neq \varnothing$

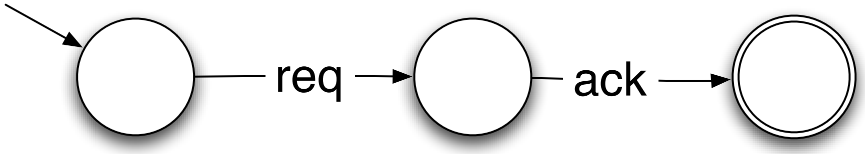
Vérification - rappels

- Définition: Un mot $w=w_1w_2...w_j...$ est accepté par un automate de Büchi A ssi il existe dans A un chemin $p=q_1q_2...q_j...$ tel que:
  - p est accepteur et
  - w étiquette p: pour tout $i \geq 1$: $q_{i+1} \in R(q_i,w_i)$
  - rem: il peut y avoir plusieurs chemins (dont certains non-accepteurs) par mot en raison du non-déterminisme.
- Définition: Le langage d'un automate de Büchi A est l'ensemble de tous les mots (infinis) acceptés par A.

Vérification - rappels

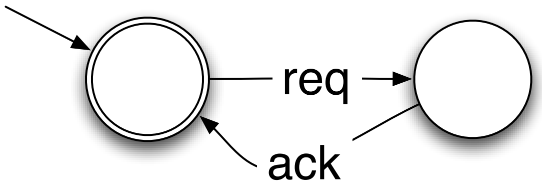
- Exemples d'automates de Büchi:

vide



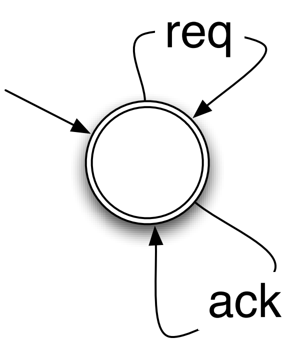
$$L^\omega(A) = \emptyset$$

1 mot ∞



$$L^\omega(A) = \{(\mathtt{req}\,\mathtt{ack})^\omega\}$$

une ∞té de mots ∞



$$L^\omega(A) = \{(\mathtt{req} + \mathtt{ack})^\omega\}$$

Vérification - rappels

- Les automates de Büchi acceptent une classe de langages appelée les langages ω-réguliers
- Que se passe-t-il si on considère des automates de Büchi déterministes ?
  - rappel: pour les automates sur mots finis, cela ne change pas l'expressivité.
- Définition: Un auto. de Büchi A= $\langle \Sigma, Q, I, R, F \rangle$ est

  déterministe ssi:
  - $|I| \leq 1$
  - $R: Q \times \Sigma \to Q$

Vérification - rappels

- Les automates de Büchi acceptent une classe de langages appelée les langages ω-réguliers
- Que se passe-t-il si on considère des automates de Büchi déterministes ?
  - rappel: pour les automates sur mots finis, cela ne change pas l'expressivité.
- Définition: Un auto. de Büchi A [illegible] est déterministe ssi:
  - $|I| \leq 1$
  - $R: Q \times \Sigma \to Q$

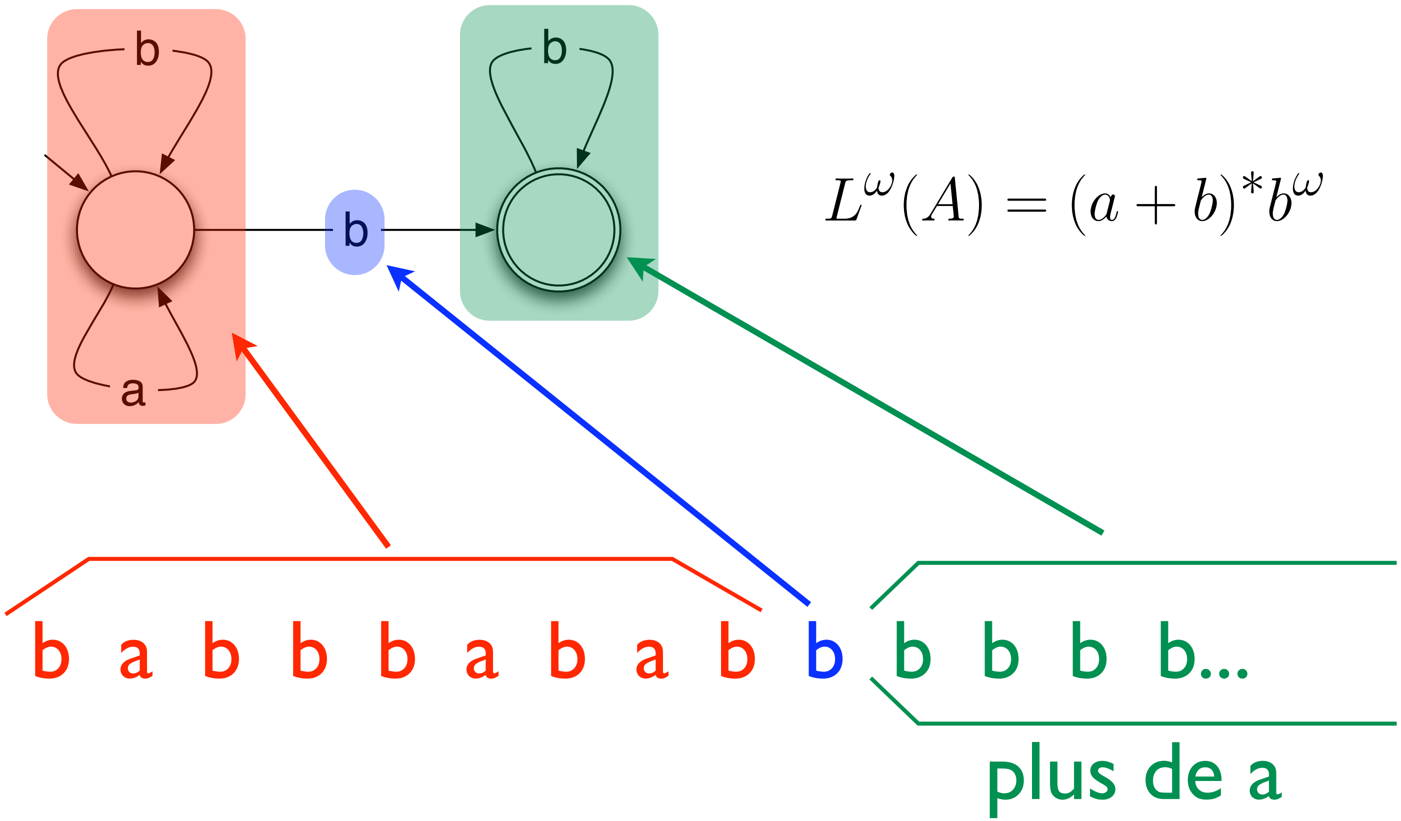
Pour chaque état et chaque lettre on a un et un seul successeur

Vérification - rappels

- Que se passe-t-il si on considère des automates de Büchi déterministes ?
  - Malheureusement, il existe des langages ω-réguliers qui ne sont reconnus par aucun automate de Büchi déterministe.
  - On ne peut donc "pas déterminiser" les automates de Büchi

Vérification - rappels

- Exemple d'automate de Büchi non-déterminisable:

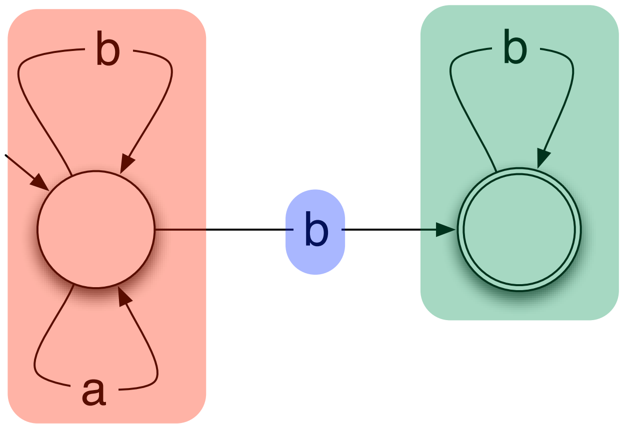


$$L^\omega(A) = (a+b)^* b^\omega$$

Langage = ensemble de tous les mots infinis sur {a,b} qui ne contiennent qu'un nombre fini de a

Vérification - rappels

- Exemple d'automate de Büchi non-déterminisable:



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

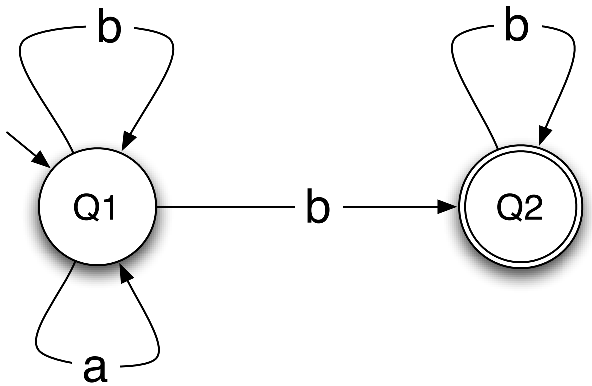
Intuition: Cet automate n'est pas déterminisable car il faut deviner, quand on voit un b, s'il y aura encore des a après ce b.

Si oui: on doit continuer à accepter les a, mais s'assurer qu'il n'y en a pas une ∞té

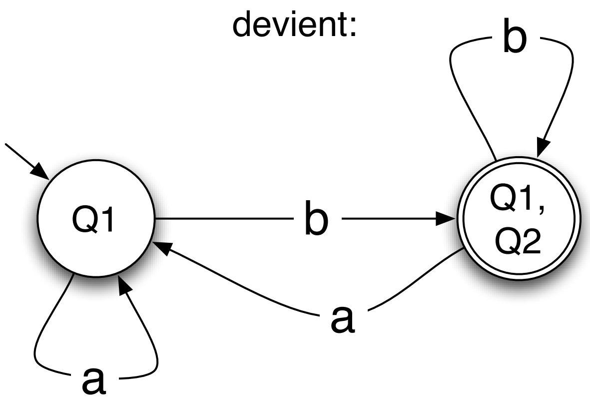
Si non: on doit ne plus accepter que des b

Vérification - rappels

- Exemple: que se passe-t-il si on applique la construction classique pour les mots finis ?



$$L^\omega(A) = (a+b)^* b^\omega$$

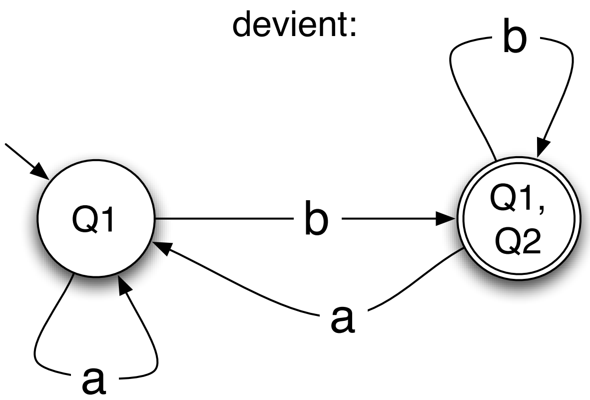


$$(a+b)^\omega \in L^\omega(A)$$

Vérification - rappels

- Exemple: que se passe-t-il si on applique la construction classique pour les mots finis ?

Problème: la condition d'acceptation oblige à passer ∞ment souvent par [Q1,Q2] mais n'interdit pas de passer ∞ment souvent par [Q1]



$$(a+b)^\omega \in L^\omega(A)$$

Vérification - rappels

- On va donc modifier la condition d'acceptation:
- Définition: un automate de Muller est un tuple $A = \langle \Sigma, Q, I, R, C \rangle$ où:
  - ...
  - $C \subseteq 2^Q$ est un ensemble d'ensembles d'états.
- Définition: Un chemin p est accepteur ssi $Inf(p) \in C$

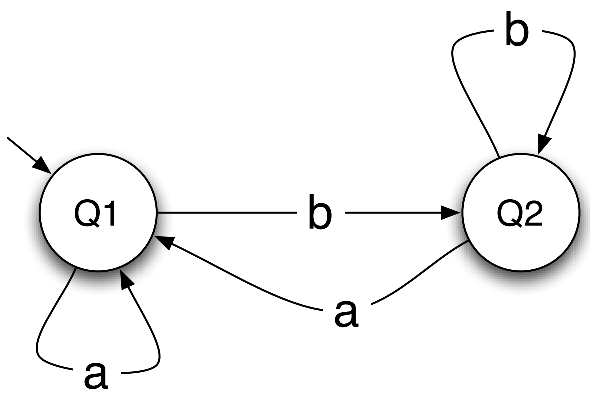
Vérification - rappels

- On va donc modifier la condition d'acceptation:
- Définition: un automate de Muller est un tuple $A = \langle \Sigma, Q, I, R, C \rangle$ où:
  - ...
  - $C \subseteq 2^Q$ est un ensemble d'ensembles d'états.
- Définition: Un chemin p est accepteur ssi $Inf(p) \in C$

Les états qui doivent être visités infiniment souvent sont maintenant définis précisément
Si on ne veut pas autoriser de visiter infiniment souvent Q, on ne met pas dans C d'ensemble S contenant Q

Vérification - rappels

- Exemple:



C={{Q2}}

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

Automate déterministe !

Les seuls chemins p tq Inf(p)={Q2} ne passent pas ∞ment souvent par Q1 et ne contiennent donc qu'un nombre fini de a.

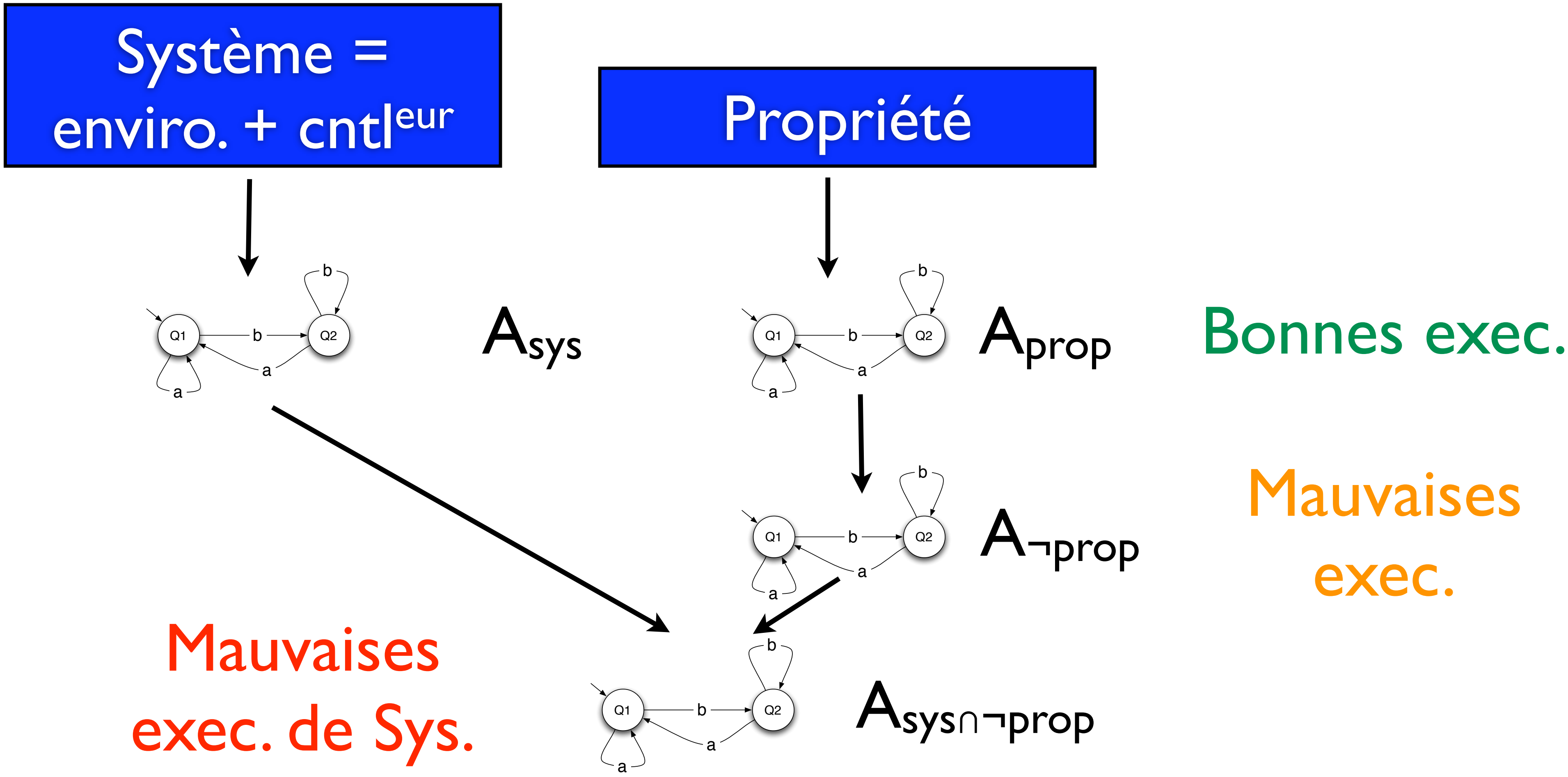
- Théorème (McNaughton): les automates de Muller sont déterminisables

Vérification - rappels

- Pour récapituler
  - Les automates de Muller forment une classe d'automates sur mots infinis
  - Ils supportent les opérations ensemblistes classiques (union, intersection, complément, test du vide)
  - Ils sont déterminisables

Vérification - rappels

- Le problème de vérification:



Sys respecte prop ssi $L(A_{sys \cap \neg prop})$ est vide

Problème de synthèse

- Dans le problème de synthèse on veut calculer automatiquement le contrôleur
- Pour la vérification, les automates sont bien adaptés, car ils décrivent toutes les actions possibles du système.
  - on peut donc dire si le système est OK pour toutes les actions

$$\underbrace{\boxed{\text{Environnement}} \parallel \boxed{\text{Contrôleur ?}}}_{= \text{ système}} \models \varphi$$

Problème de synthèse

- Dans la synthèse, par contre une partie du système n'est pas spécifiée...
- On doit donc introduire dans notre modèle le fait qu'il existe deux types d'actions:
  - celles de l'environnement: elles sont fixées, mais on ne les contrôle pas.
  - celles du contrôleur: on ne les connaît pas *a priori*, mais tout est autorisé dans les limites du contrôleur.

$$\underbrace{\boxed{\text{Environnement}} \;\|\; \boxed{\text{Cont?ôleur}}}_{=\ \text{système}} \models \varphi$$

Problème de synthèse

- On peut voir le problème de synthèse comme un affrontement entre deux joueurs:
  - l'environnement: fait tout ce qu'il peut pour que la propriété ne soit pas respectée
  - l'antagoniste: fait tout ce qu'il peut pour contrer l'environnement et assurer que la propriété est respectée
- Il s'agit d'un jeu...
- On espère que l'antagoniste peut gagner à tous les coups. Si c'est le cas, on dit qu'il a une stratégie, et cette stratégie forme un contrôleur.

Théorie des jeux
Définitions

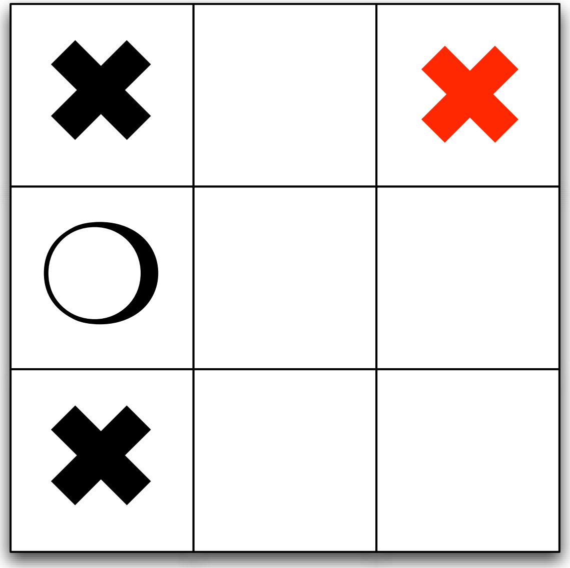
Jeux

- Exemple de Jeu:
  - Tic-tac-toe, avec un damier initial non-vide:

✖		
❍		
✖		

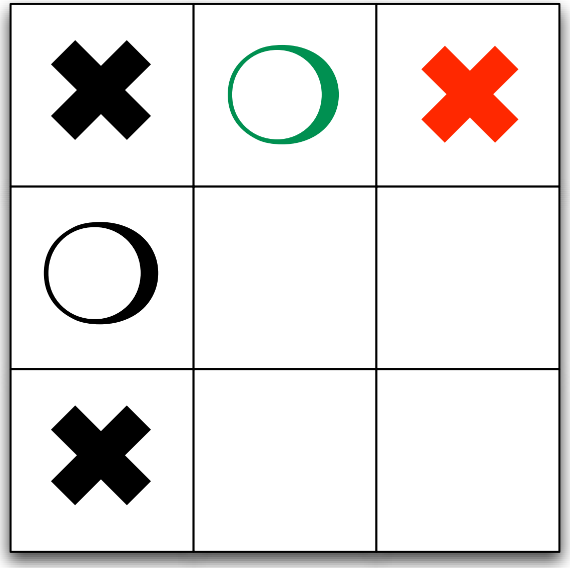
  - ❍ est l'environnement
  - ✖ est l'antagoniste: il joue en premier
  - propriété à respecter: ✖ gagne la partie

Jeux



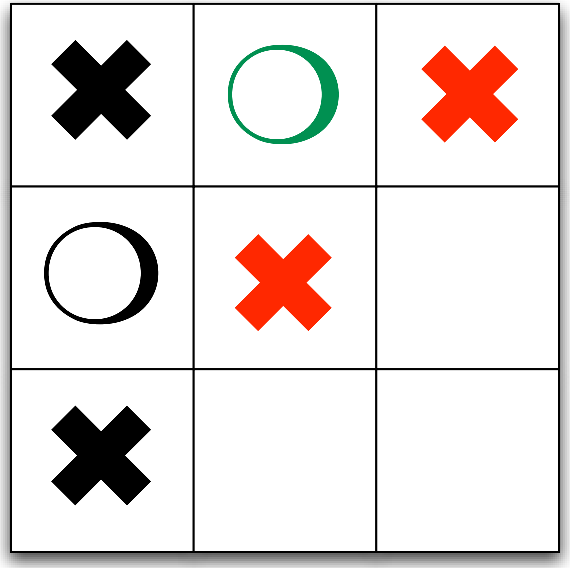
- Si ✖ joue correctement ses deux premiers coups, il est sûr de remporter la partie

Jeux



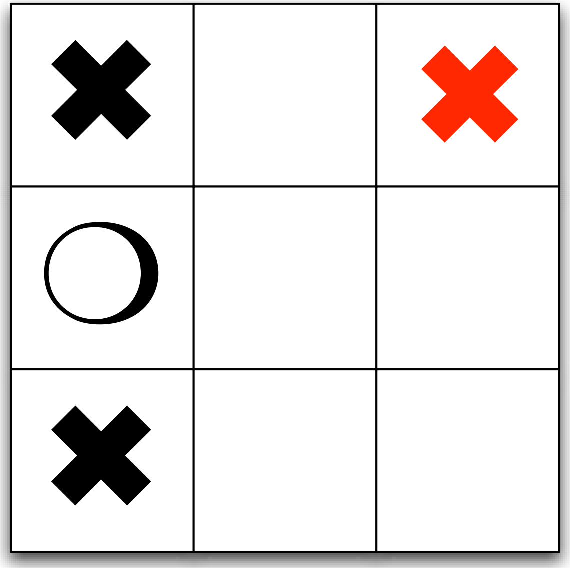
- Si ✖ joue correctement ses deux premiers coups, il est sûr de remporter la partie

Jeux



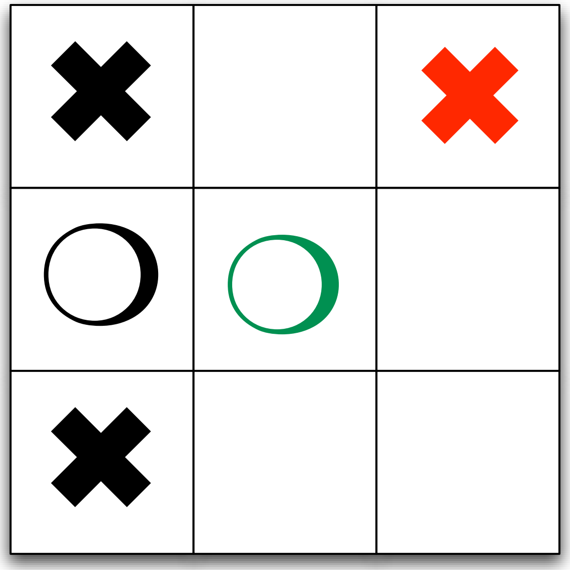
- Si ✖ joue correctement ses deux premiers coups, il est sûr de remporter la partie

Jeux



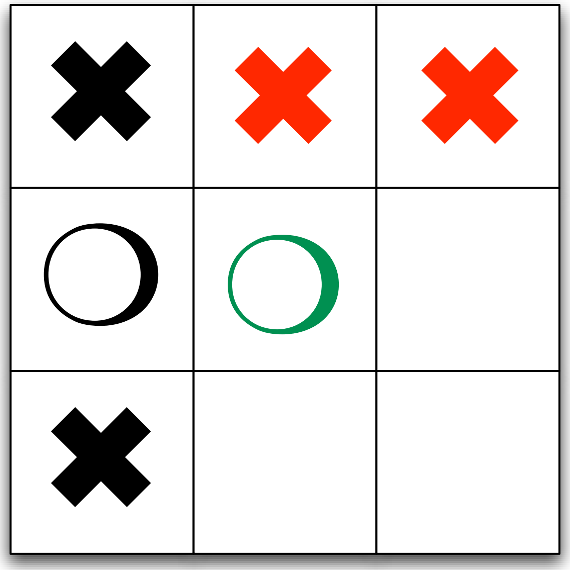
- Si ✖ joue correctement ses deux premiers coups, il est sûr de remporter la partie

Jeux



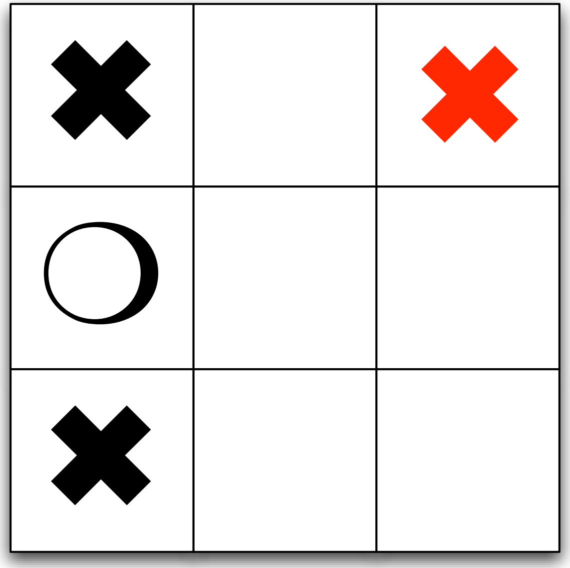
- Si ✖ joue correctement ses deux premiers coups, il est sûr de remporter la partie

Jeux



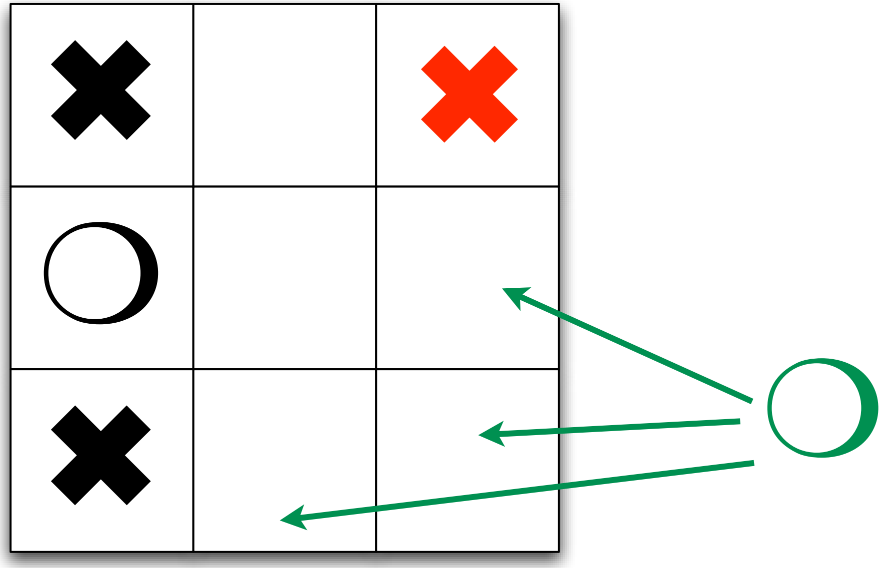
- Si ✖ joue correctement ses deux premiers coups, il est sûr de remporter la partie

Jeux



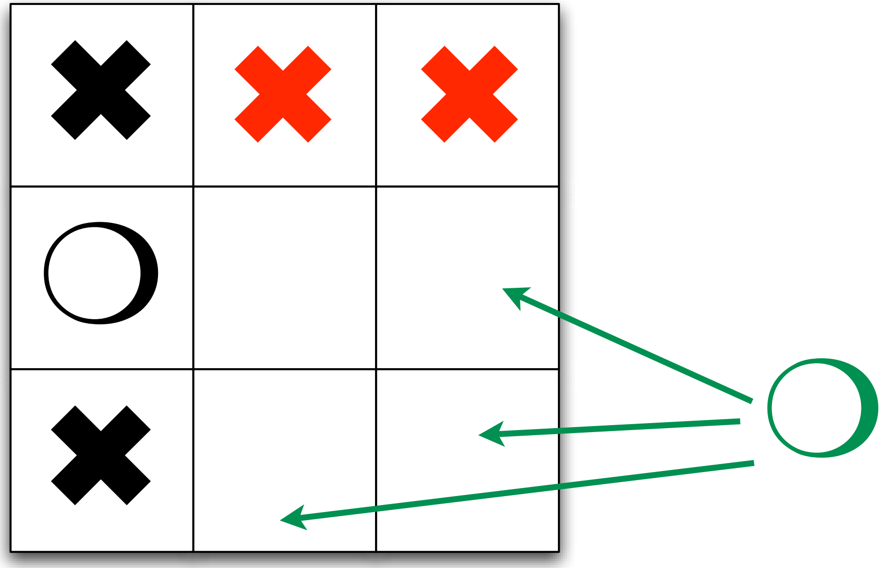
- Si ✖ joue correctement ses deux premiers coups, il est sûr de remporter la partie

Jeux



- Si ✖ joue correctement ses deux premiers coups, il est sûr de remporter la partie

Jeux



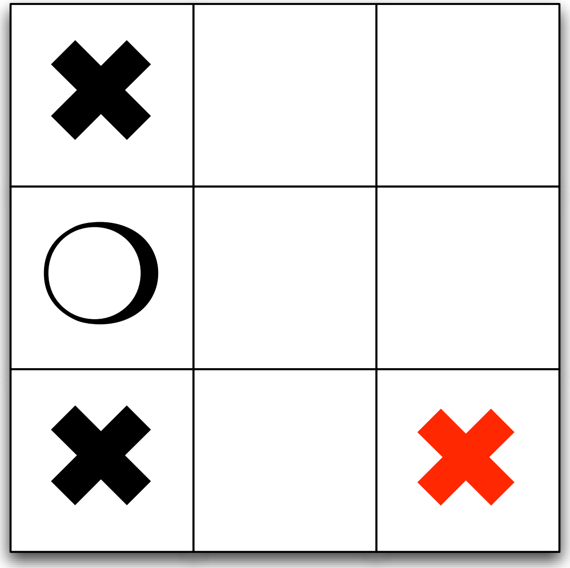
- Si ✖ joue correctement ses deux premiers coups, il est sûr de remporter la partie

Jeux

✖		
❍		
✖		

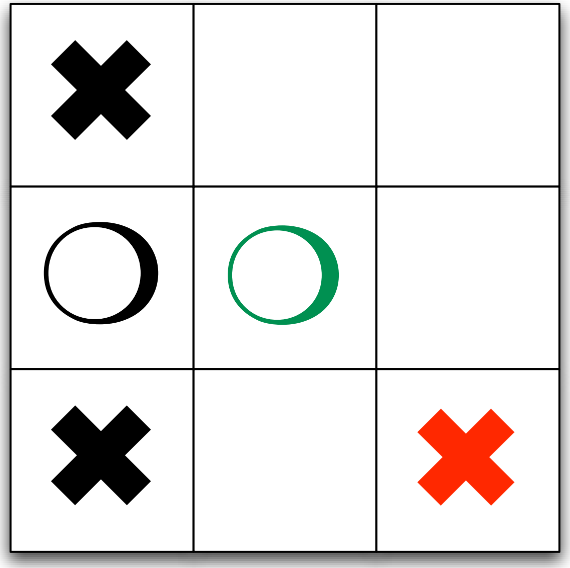
- Si ✖ joue correctement ses deux premiers coups , il est sûr de remporter la partie
- Sinon, il se peut que ❍ gagne

Jeux



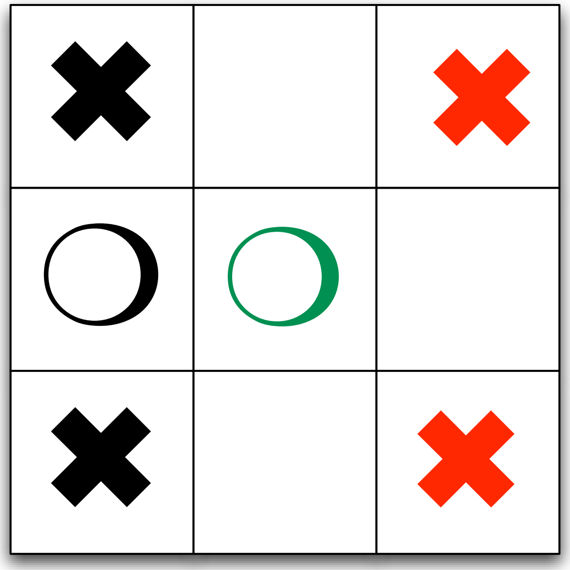
- Si ✖ joue correctement ses deux premiers coups , il est sûr de remporter la partie
- Sinon, il se peut que ❍ gagne

Jeux



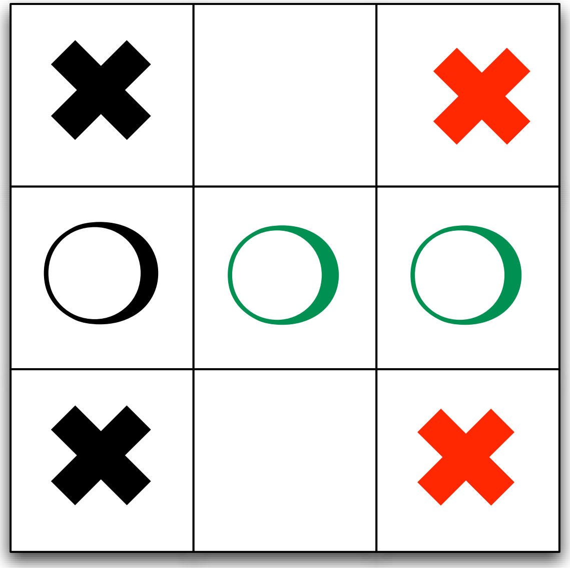
- Si ✖ joue correctement ses deux premiers coups , il est sûr de remporter la partie
- Sinon, il se peut que ❍ gagne

Jeux



- Si ✖ joue correctement ses deux premiers coups , il est sûr de remporter la partie
- Sinon, il se peut que ❍ gagne

Jeux



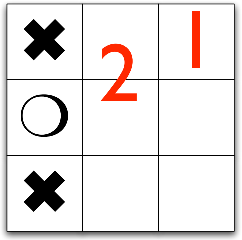
- Si ✖ joue correctement ses deux premiers coups , il est sûr de remporter la partie
- Sinon, il se peut que ❍ gagne

Jeux

✖	2	1
❍		
✖		

- Partant de cette configuration initiale un peu particulière, ✖ possède donc une stratégie gagnante:
  - D'abord, mettre une croix dans la case supérieure droite
  - Puis jouer une des deux premières cases de la deuxième colonne (une de celles qui sont vides)
- Quoi que fasse ❍, cette stratégie assure à ✖ la victoire du jeu.

Problème de synthèse



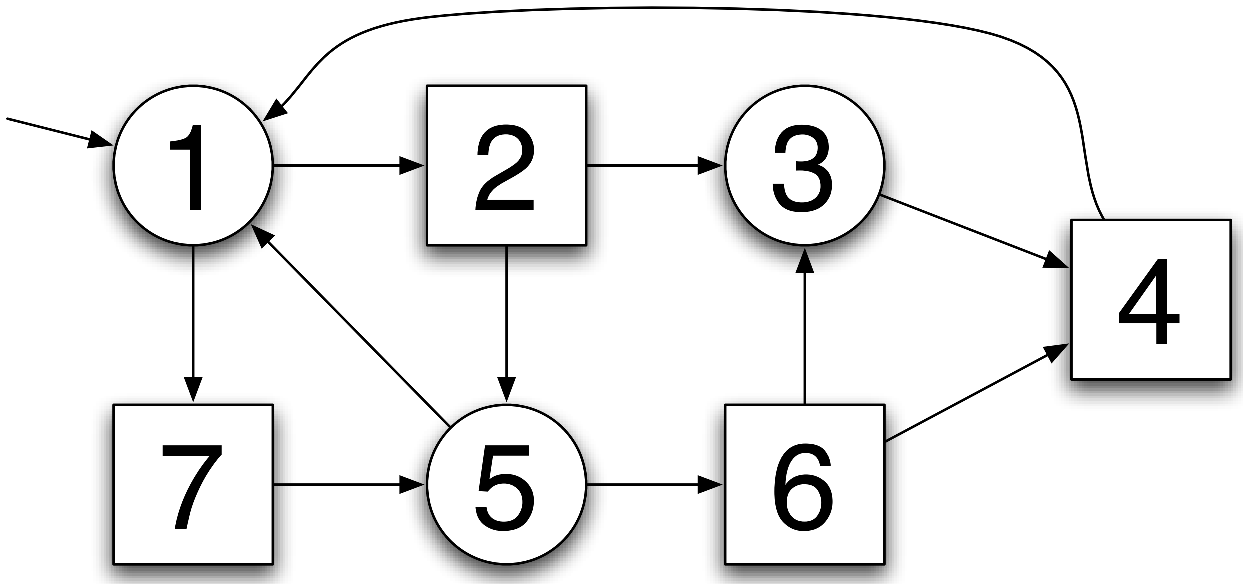
- Revenons à l'analogie de départ:
  - ❍ est l'environnement. On veut éviter qu'il gagne, car cela mettrait le système dans un mauvais état
  - ✖ est l'antagoniste. Grâce à sa stratégie gagnante, il peut toujours empêcher l'environnement de gagner
  - La stratégie gagnante de ✖ forme donc un contrôleur qui garantit la propriété "❍ ne gagne jamais".

Jeux

- Comment représenter ce jeu ?
- On va utiliser des graphes avec deux types de noeuds:
  - Des noeuds qui appartiennent au joueur A, et où lui seul peut décider quelle action effectuer (c-à-d quel est l'état suivant)
  - Des noeuds qui appartiennent au joueur B, symétriquement.

Jeux

- Exemple:
  - Le joueur B contrôle les noeuds ronds
  - Le joueur A contrôle les noeuds carrés



Exemple tiré (ainsi qu'une partie de la suite) d'un tutoriel de W. Thomas

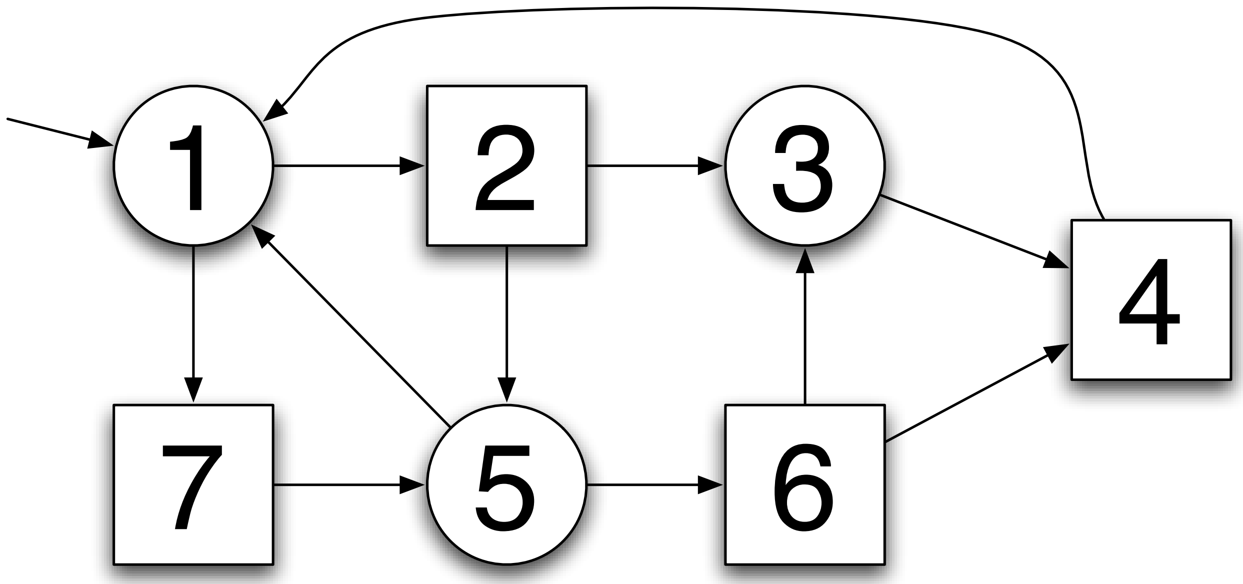
Jeux

- Définition: Une arène est un tuple $\langle Q, q_0, E\rangle$ où:
  - $Q=Q_A \cup Q_B$ (avec $Q_A \cap Q_B=\varnothing$) est l'ensemble des noeuds. Les noeuds dans Q_A (resp. Q_B) sont contrôlés par le joueur A (B)
  - $q_0 \in Q$ est le noeud initial
  - $E \subseteq Q \times Q$ est l'ensemble des arcs. Chaque noeud doit avoir au moins un successeur: $\forall q \in Q$, $\exists q' \in Q$: $(q,q') \in E$.
    - rem: le même joueur peut jouer plusieurs fois de suite.

Jeux

- **Définition**: Une partie dans une arène $\langle Q, q_0, E \rangle$

  est une séquence infinie $r_1r_2r_3\ldots$ telle que $r_1=q_0$ et $\forall i\geq 1: (r_i, r_{i+1})\in E$. C'est donc un chemin infini dans le graphe, partant de q_0.



1 2 3 4 1 7 5 1 7 5 1 7 5 1...

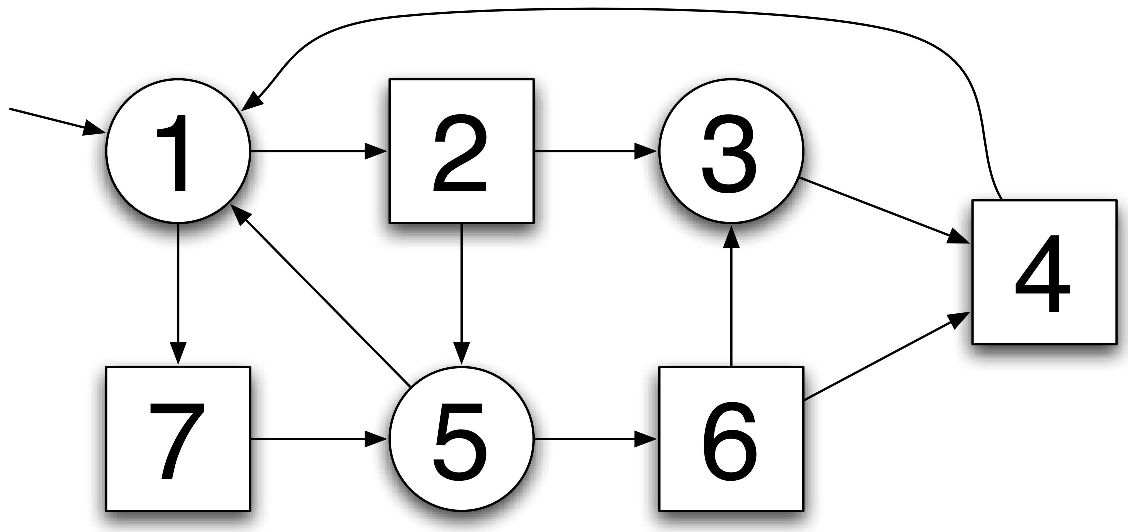
Jeux

- Pour déterminer qui gagne, on va utiliser des conditions inspirées des automates de Muller.
- Etant donné une partie p dans une arène:
  - Inf(p) = ensemble des noeuds qui apparaissent infiniment souvent dans p
  - Occ(p) = ensemble des noeuds qui apparaissent dans p
- Exemple: pour $p = 1\ 2\ 3\ 4\ 1\ 7\ 5\ 1\ 7\ 5\ 1\ (7\ 5\ 1)^\omega$
  - $Inf(p)=\{1,5,7\}$
  - $Occ(p)= \{1,2,3,4,5,7\}$

Jeux

- Fixons un ensemble F d'ensembles de noeuds de l'arène et une partie p
- On va regarder deux types de conditions de Muller:
  - Les conditions faibles: p est une partie gagnante ssi $Occ(p) \in F$
  - Les conditions fortes: p est une partie gagnante ssi $Inf(p) \in F$

Jeux



- Exemple:
  - $123(157)^\omega$ gagne pour la condition forte {{1,3}, {1,5,7}} et pour la condition faible {{1,2,3,5,7},{1,2}}
  - $123(157)^\omega$ perd pour la condition forte {{1,2,3}, {1,2,3,5,7}} et pour la condition faible {{1,4}}

Jeux

- Définition: Un jeu infini est une paire $\langle G,\varphi\rangle$ où:
  - G est une arène
  - φ est une condition de Muller (forte ou faible) pour un des joueurs

Jeux

- Cas particuliers: Si la condition de Muller est:
  - une condition faible de la forme $\{S|q\in S\}$ pour un certain état q, on a un jeu d'accessibilité (pour q).
    - e.g.: $F=\{\{1\},\{1,2\},\{1,3\}, \{1,2,3\}\}$, avec $Q=\{1,2,3\}$. On gagne si on arrive à forcer le jeu à passer par 1 (on atteint 1).
    - cas pratique: un système qui doit s'initialiser et dont on veut vérifier qu'il passe au moins par l'état "initialisation terminée", afin d'être sûr qu'il ne reste pas bloqué dans le phase d'initialisation.

Jeux

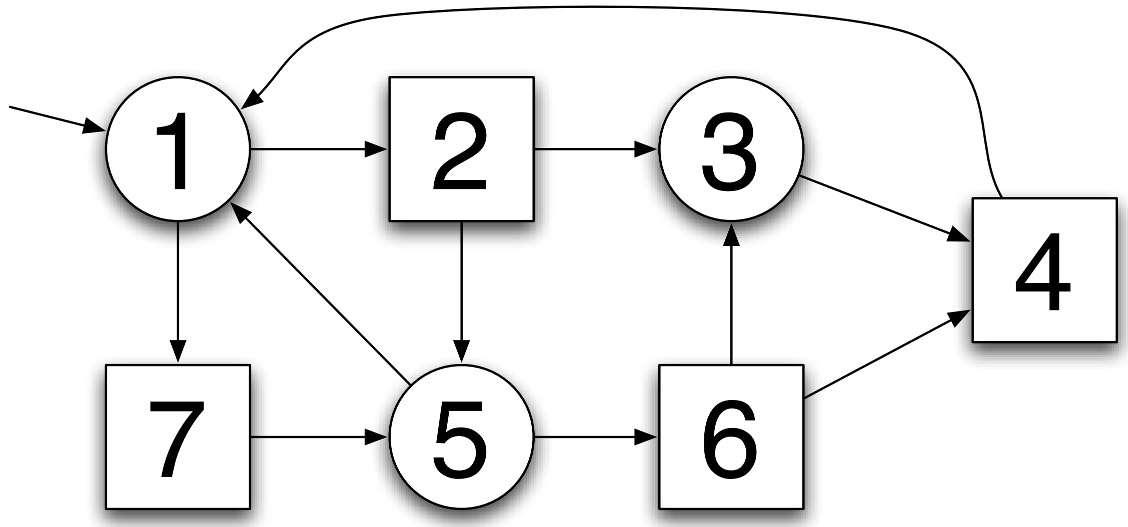
- Cas particuliers: Si la condition de Muller est:
  - une condition faible de la forme $\{S' | S' \subseteq S\}$ pour un certain ensemble S, on a un jeu de sûreté (pour S).
    - e.g.: F={{1,2,3},{1,2},{1,3},{2,3},{1},{2},{3}}, avec Q={1,2,3,4,5}. On gagne si on ne visite que les états 1, 2 ou 3 (mais on n'est pas obligé de passer par tous les états)
    - cas pratique: une pompe doit maintenir un certain niveau de liquide dans une cuve, d'où le liquide est consommé par l'environnement. Les bons états dont ceux où le niveau est suffisant. On veut synthétiser un contrôleur qui assure qu'on reste dans ces bons états

Jeux

- Cas particuliers: Si la condition de Muller est:
  - une condition forte de la forme $\{S|q\in S\}$ pour un certain état q, on a un jeu de récurrence (ou jeu de Büchi).
    - e.g.: $F=\{\{1\},\{1,2\},\{1,3\}, \{1,2,3\}\}$, pour $Q=\{1,2,3\}$. On gagne si on arrive à forcer le jeu à passer infiniment souvent par 1.
    - cas pratique: un système qui traite des requêtes, et dont on veut s'assurer qu'il revient perpétuellement dans l'état "en attente d'une requête" (afin d'éviter le déni de service).

Jeux

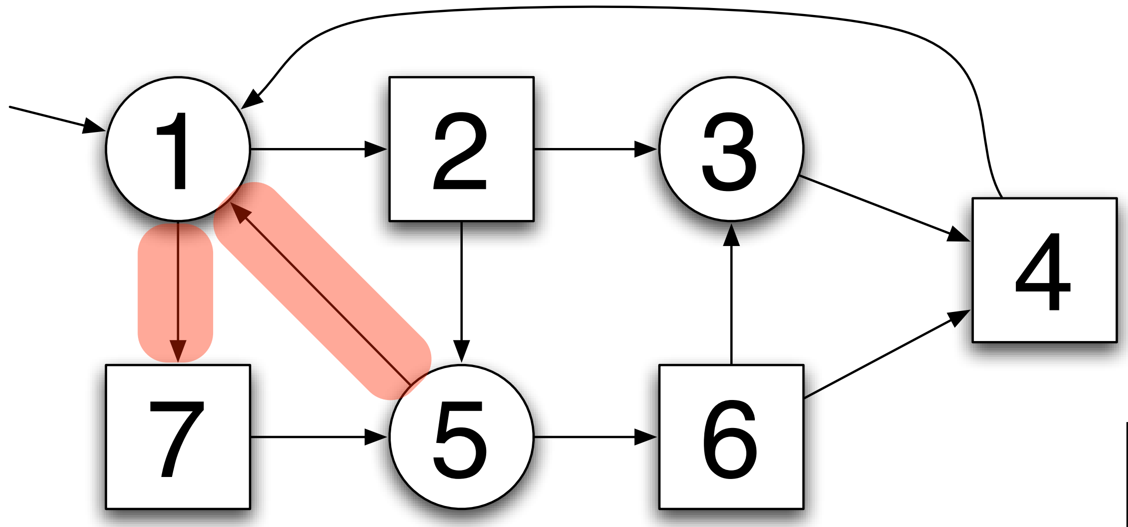
- Exemple de jeu:
  - On considère l'arène ci-dessous et la condition forte {{1,5,7}} pour le joueur B



Le joueur B a-t-il une stratégie gagnante ?

Jeux

- Exemple de jeu:
  - On considère l'arène ci-dessous et la condition forte {{1,5,7}} pour le joueur B

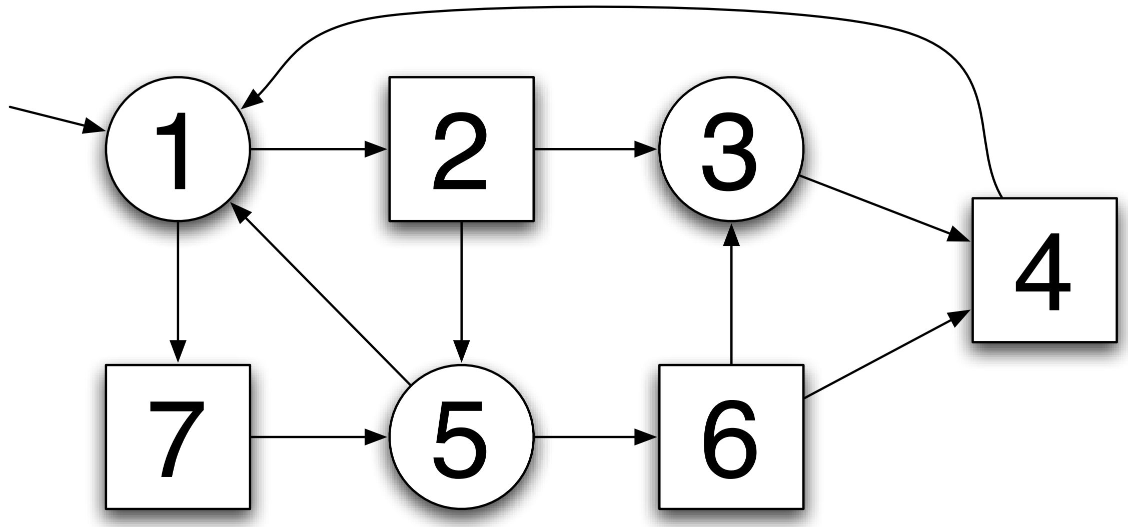


oui !

Le joueur B a-t-il une stratégie gagnante ?

Jeux

- Exemple de jeu:
  - On considère l'arène ci-dessous et la condition forte $\{S | \{2,7\} \subseteq S\}$ pour le joueur B

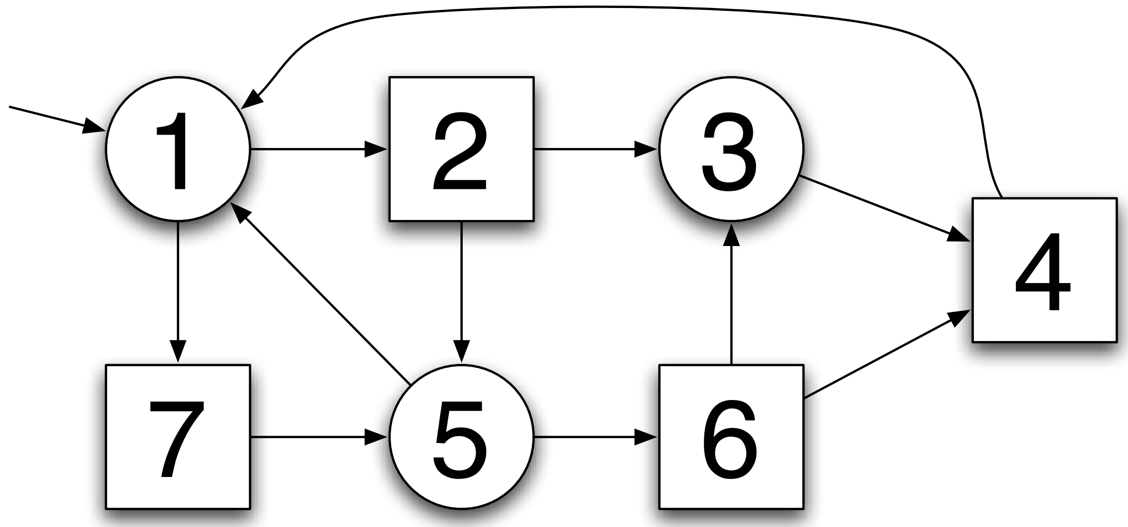


Le joueur B a-t-il une stratégie gagnante ?

Jeux

- Exemple de jeu:
  - On considère l'arène ci-dessous forte $\{S|\{2,7\}\subseteq S\}$ pour le joueur

oui !
à partir de 1:
aller 1 fois sur 2 vers 2 et l'autre fois vers 7



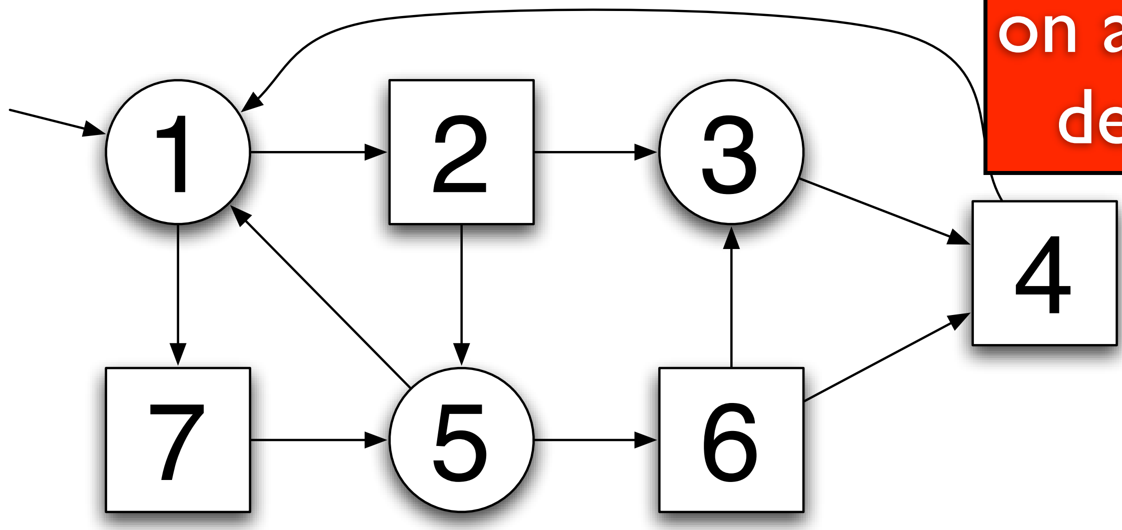
Le joueur B a-t-il une stratégie gagnante ?

Jeux

oui !
à partir de 1:
aller 1 fois sur 2
vers 2 et l’autre
fois vers 7

- Exemple de jeu:
  - On considère l’arène ci-dessous
    forte $\{S|\{2,7\}\subseteq S\}$ pour le joueur

on a donc besoin
de mémoire !



Le joueur B a-t-il une stratégie gagnante ?

Jeux - stratégies

- Définition: Une stratégie pour le joueur X (resp. B) dans une arène $\langle Q, q_0, E\rangle$ est une fonction

  $f: Q^*Q_X \rightarrow Q$ telle que, pour tout $\sigma q \in Q^*Q_X$: $(q,f(\sigma q)) \in E$.

- Autrement dit, pour tout préfixe de partie σq qui se termine dans un noeud contrôlé par le joueur X, $f(\sigma q)$ indique quel est le prochain état à jouer.

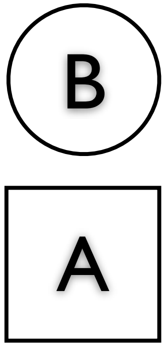
- Cela n'est possible que s'il existe un arc entre q et $f(\sigma q)$.

Jeux - stratégies

- Définition: Une partie $p=r_1r_2r_3\ldots$ est jouée selon une stratégie f (pour le joueur X) ssi:
pour tout $r_i \in Q_X$: $r_{i+1} = f(r_1r_2\ldots r_i)$
- Autrement dit, chaque fois qu'on arrive dans un noeud appartenant à X, on choisit comme noeud suivant celui qui est indiqué par la stratégie.

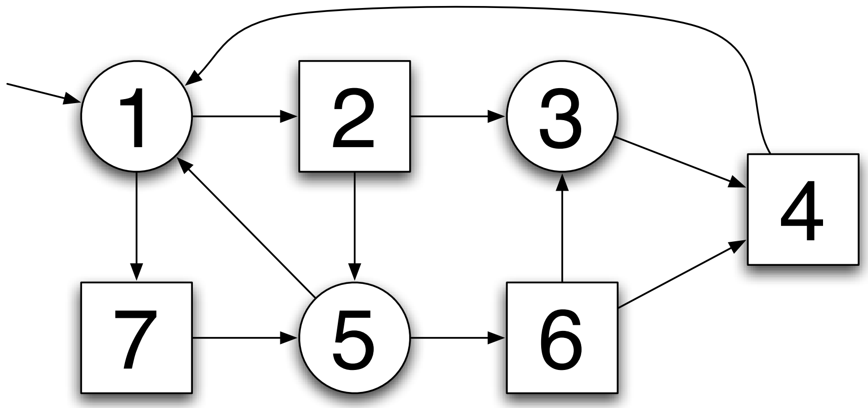
Jeux - stratégies

- Définition: Une stratégie f (pour le joueur X) dans une arène A est gagnante pour X dans le jeu G= $\langle A, \varphi \rangle$ ssi chaque partie jouée dans G selon f est gagnante pour X, par rapport à l'objectif φ.
- Autrement dit, quoique fasse l'adversaire de X, si X joue selon sa stratégie, il remporte le jeu car l'objectif φ est atteint.



Jeux - stratégies

- Exemple: On considère l'arène ci-dessous et la condition forte {{1,5,7}} pour le joueur B



- Stratégie gagnante:
  - $\forall \sigma \in Q^*: f(\sigma 1)=7$
  - $\forall \sigma \in Q^*: f(\sigma 5)=1$
  - $\forall \sigma \in Q^*: f(\sigma 3)=4$

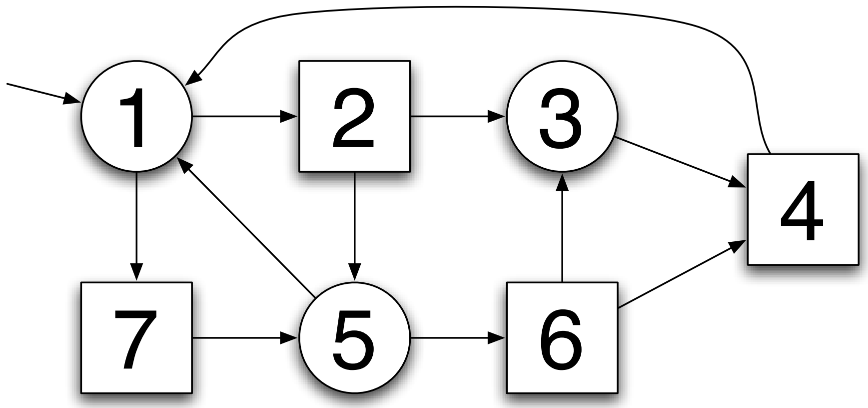
Indifférent, car on ne passera jamais par 3

B

A

Jeux - stratégies

- Exemple: On considère l'arène ci-dessous et la condition forte {{1,5,7}} pour le joueur B



- Stratégie gagnante:
  - $\forall \sigma \in Q^*: f(\sigma 1)=7$
  - $\forall \sigma \in Q^*: f(\sigma 5)=1$
  - $\forall \sigma \in Q^*: f(\sigma 3)=4$

La stratégie ne dépend que de l'état courant !

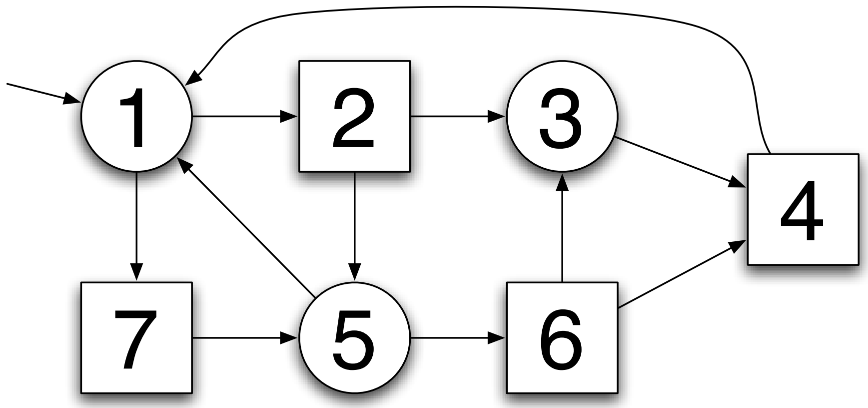
Indifférent, car on ne passera jamais par 3

B

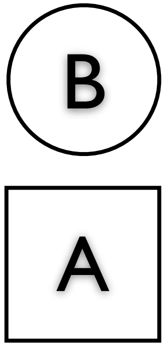
A

Jeux - stratégies

- Exemple: On considère l'arène ci-dessous et la condition forte $\{S|\{2,7\}\subseteq S\}$ pour le joueur B

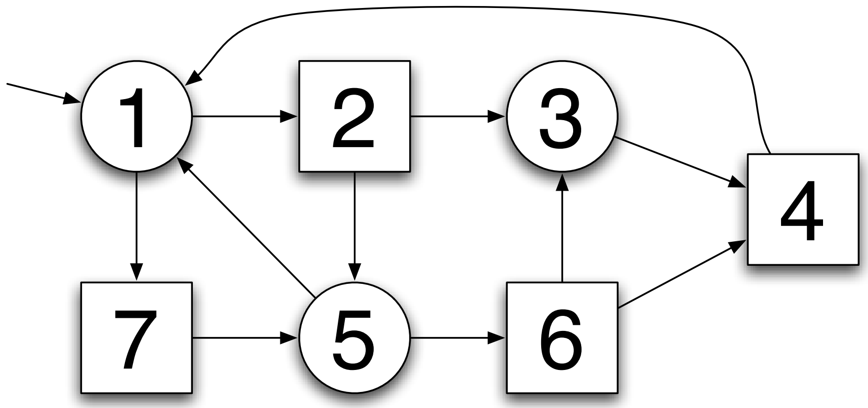


- Stratégie gagnante pour B:
  - $\forall\sigma\in Q^*: f(\sigma 1)=7$ si $\exists$ i: $\sigma_i=2$ et $\forall$ j>i: $\sigma_j \notin \{2,7\}$; $f(\sigma 1)=2$ sinon
  - $\forall\sigma\in Q^*: f(\sigma 5)=1$
  - $\forall\sigma\in Q^*: f(\sigma 3)=4$



Jeux - stratégies

- Exemple: On considère l’arène ci-dessous et la condition forte $\{S|\{2,7\}\subseteq S\}$ pour le joueur B



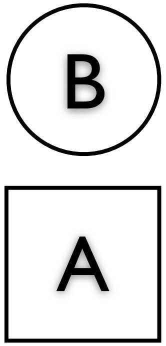
- Stratégie gagnante pour B:
  - $\forall\sigma\in Q^*$:$f(\sigma 1)=7$ si $\exists$ i: $\sigma_i=2$ et $\forall$ j>i: $\sigma_j \notin \{2,7\}$; $f(\sigma 1)=2$ sinon
  - $\forall\sigma\in Q^*$:$f(\sigma 5)=1$
  - $\forall\sigma\in Q^*$:$f(\sigma 3)=4$

La stratégie dépend de l’historique !

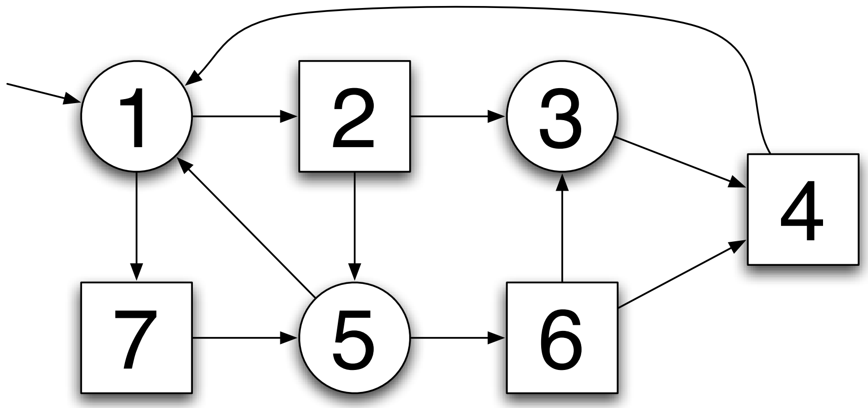
Jeux - stratégies

- De manière générale, une stratégie peut donc utiliser toute l'information donnée par le préfixe déjà joué.
- On aimerait au moins avoir une stratégie calculable, mais certains cas particuliers (simples) sont plus favorables:
  - Si la stratégie peut être calculée par un automate fini, on parle de stratégie à états finis
  - Si la stratégie ne dépend que de l'état en cours, on parle de stratégie positionelle

Jeux - stratégies



- Exemple: On considère l'arène ci-dessous et la condition forte $\{\{1,5,7\}\}$ pour le joueur B



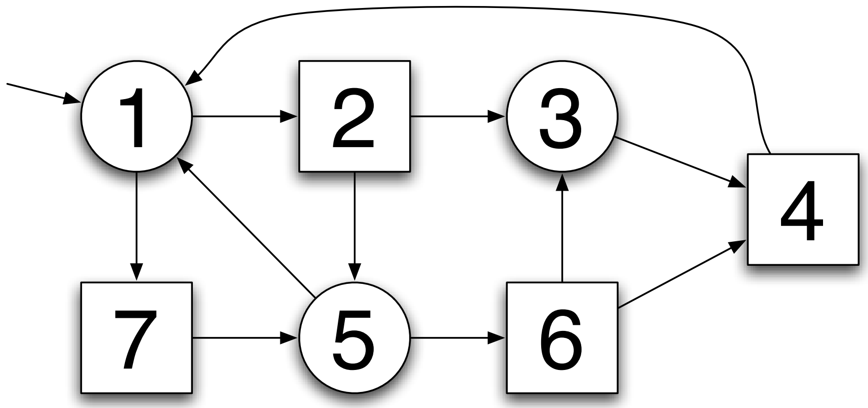
- Stratégie gagnante:
  - $\forall \sigma \in Q^*: f(\sigma 1)=7$
  - $\forall \sigma \in Q^*: f(\sigma 5)=1$
  - $\forall \sigma \in Q^*: f(\sigma 3)=4$

B

A

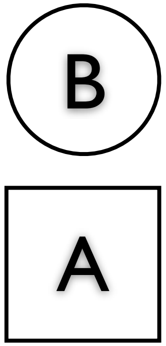
Jeux - stratégies

- Exemple: On considère l'arène ci-dessous et la condition forte {{1,5,7}} pour le joueur B



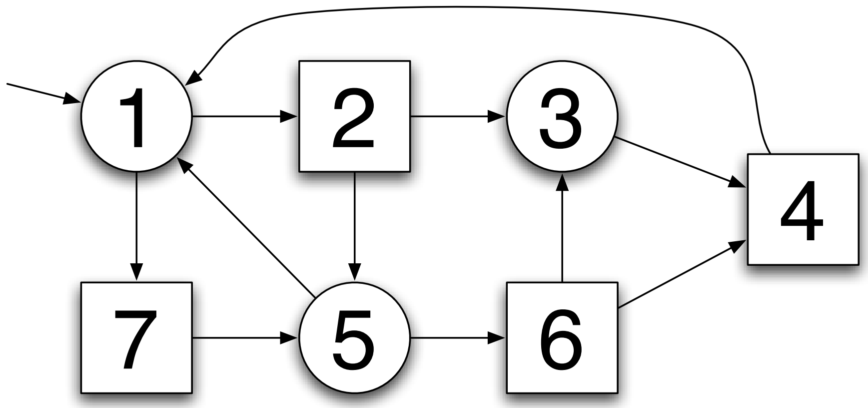
- Stratégie gagnante:
  - $\forall \sigma \in Q^*: f(\sigma 1)=7$
  - $\forall \sigma \in Q^*: f(\sigma 5)=1$
  - $\forall \sigma \in Q^*: f(\sigma 3)=4$

Stratégie positionelle



Jeux - stratégies

- Exemple: On considère l'arène ci-dessous et la condition forte $\{S \mid \{2,7\} \subseteq S\}$ pour le joueur B



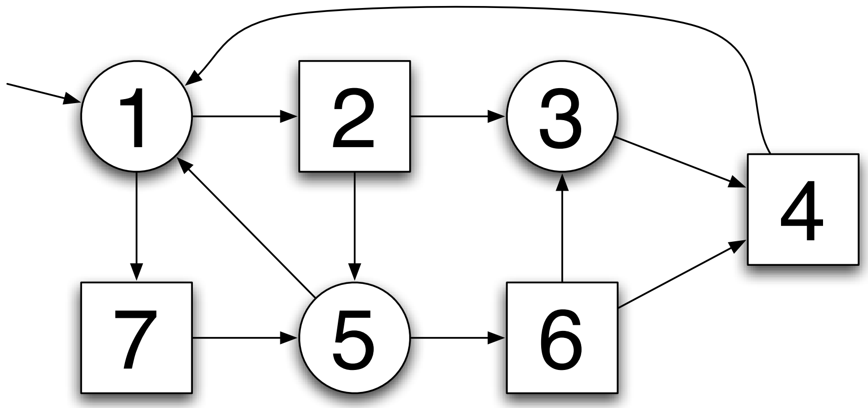
- Stratégie gagnante pour B:
  - $\forall \sigma \in Q^*: f(\sigma 1)=7$ si $\exists\, i: \sigma_i=2$ et $\forall\, j>i: \sigma_j \notin \{2,7\}$; $f(\sigma 1)=2$ sinon
  - $\forall \sigma \in Q^*: f(\sigma 5)=1$
  - $\forall \sigma \in Q^*: f(\sigma 3)=4$

B

A

Jeux - stratégies

- Exemple: On considère l'arène ci-dessous et la condition forte $\{S|\{2,7\}\subseteq S\}$ pour le joueur B



- Stratégie gagnante pour B:
  - $\forall\sigma\in Q^*$:$f(\sigma 1)=7$ si $\exists$ i: $\sigma_i=2$ et $\forall$ j>i: $\sigma_j \notin \{2,7\}$; $f(\sigma 1)=2$ sinon
  - $\forall\sigma\in Q^*$:$f(\sigma 5)=1$
  - $\forall\sigma\in Q^*$:$f(\sigma 3)=4$

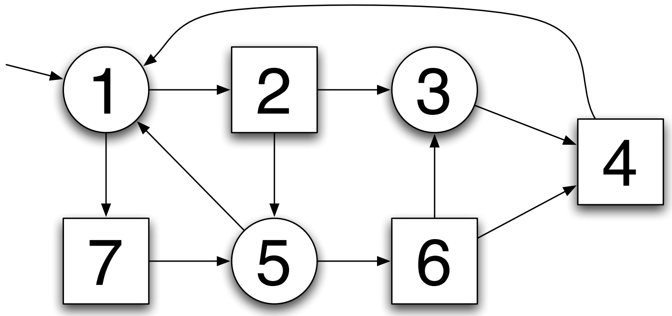
Stratégie à états finis

B

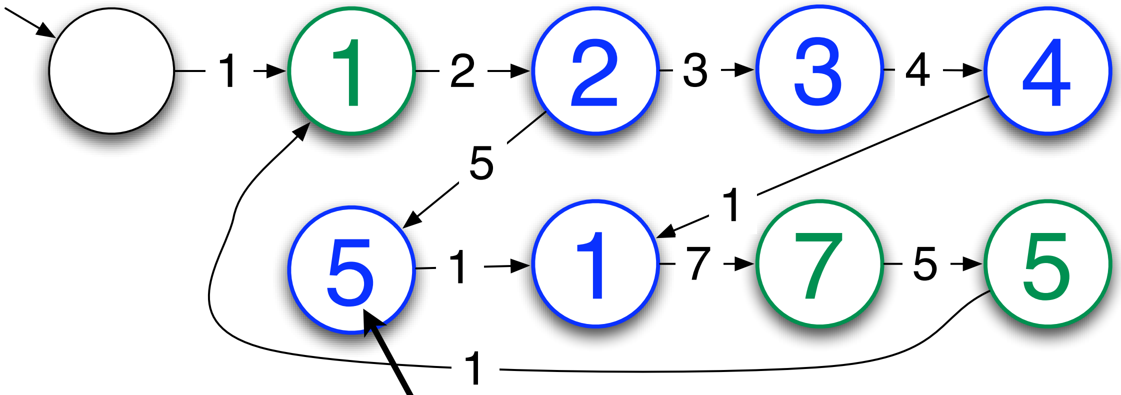
A

Jeux - stratégies

- Exemple: On considère l'arène ci-dessous et la condition forte {S|{2,7}⊆S} pour le joueur B



- Stratégie gagnante pour B:



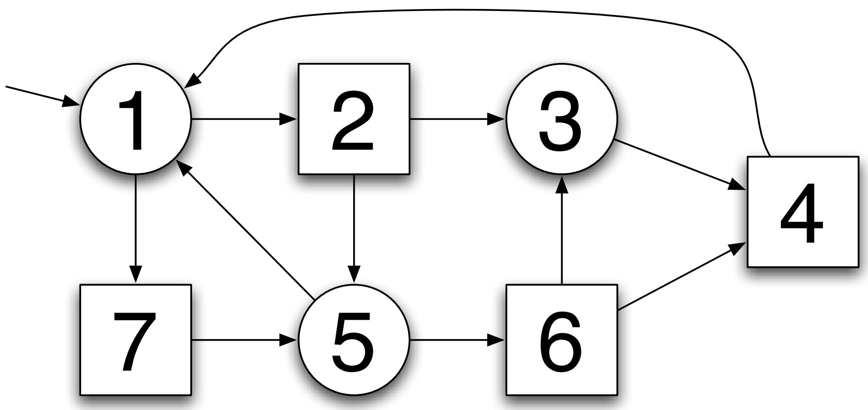
retient l'état courant

vert: on joue 2 après le prochain 1

bleu: on joue 7 après le prochain 1

Jeux - stratégies

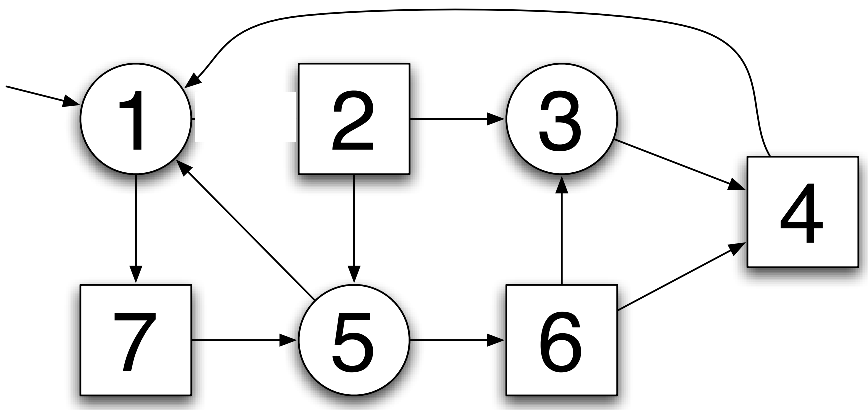
- Remarque sur les stratégies positionelles:
  - Un stratégie positionelle f pour le joueur X est en fait une fonction qui associe à chaque noeud de X un noeud successeur (pas besoin de passer tout l'historique à la fonction)
  - On peut aussi voir une stratégie positionelle comme une sélection des arcs du jeu: pour chaque noeud q de X, on ne garde que l'arc sortant (q,f(q))



- $\forall \sigma \in Q^*: f(\sigma 1)=7$
- $\forall \sigma \in Q^*: f(\sigma 5)=1$
- $\forall \sigma \in Q^*: f(\sigma 3)=4$

Jeux - stratégies

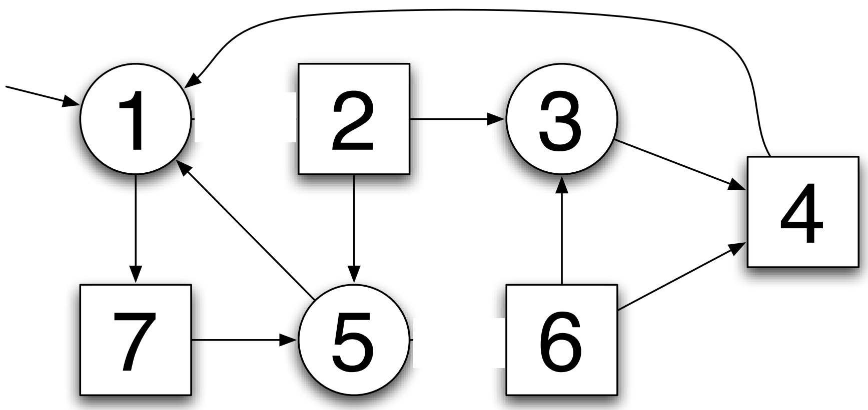
- Remarque sur les stratégies positionelles:
  - Un stratégie positionelle f pour le joueur X est en fait une fonction qui associe à chaque noeud de X un noeud successeur (pas besoin de passer tout l'historique à la fonction)
  - On peut aussi voir une stratégie positionelle comme une sélection des arcs du jeu: pour chaque noeud q de X, on ne garde que l'arc sortant (q,f(q))



- $\forall \sigma \in Q^*: f(\sigma 1)=7$
- $\forall \sigma \in Q^*: f(\sigma 5)=1$
- $\forall \sigma \in Q^*: f(\sigma 3)=4$

Jeux - stratégies

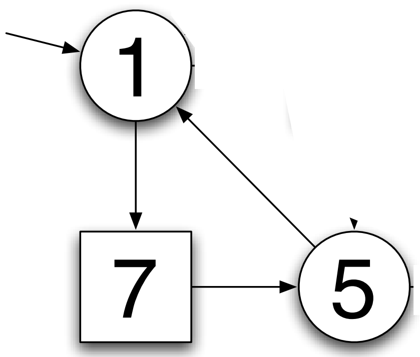
- Remarque sur les stratégies positionelles:
  - Un stratégie positionelle f pour le joueur X est en fait une fonction qui associe à chaque noeud de X un noeud successeur (pas besoin de passer tout l'historique à la fonction)
  - On peut aussi voir une stratégie positionelle comme une sélection des arcs du jeu: pour chaque noeud q de X, on ne garde que l'arc sortant (q,f(q))



- $\forall \sigma \in Q^*: f(\sigma 1)=7$
- $\forall \sigma \in Q^*: f(\sigma 5)=1$
- $\forall \sigma \in Q^*: f(\sigma 3)=4$

Jeux - stratégies

- Remarque sur les stratégies positionelles:
  - Un stratégie positionelle f pour le joueur X est en fait une fonction qui associe à chaque noeud de X un noeud successeur (pas besoin de passer tout l'historique à la fonction)
  - On peut aussi voir une stratégie positionelle comme une sélection des arcs du jeu: pour chaque noeud q de X, on ne garde que l'arc sortant (q,f(q))



- $\forall \sigma \in Q^*: f(\sigma 1)=7$
- $\forall \sigma \in Q^*: f(\sigma 5)=1$
- $\forall \sigma \in Q^*: f(\sigma 3)=4$

Jeux déterminés

- Afin de résoudre le problème de synthèse, on va s'intéresser à deux ensembles:
  - W_A = l'ensemble des positions du jeu à partir desquelles le joueur A a une stratégie gagnante
  - W_B = l'ensemble des positions du jeu à partir desquelles le joueur B a une stratégie gagnante
- Clairement $W_A \cap W_B = \varnothing$
- Par contre on pourrait imaginer des positions à partir desquelles les deux joueurs peuvent se contrer l'un l'autre, de manière à ce qu'aucun des deux n'ait une stratégie gagnante.

Jeux déterminés

- Définition: Un jeu (dont l'ensemble des positions est Q) est déterminé si et seulement si $W_A \cup W_B = Q$.
- Théorème (Borel - Martin): Les jeux infinis que nous avons définis ici (avec les objectifs de Muller) sont toujours déterminés.

E. Borel (1871-1956)

D. Martin (1940-)

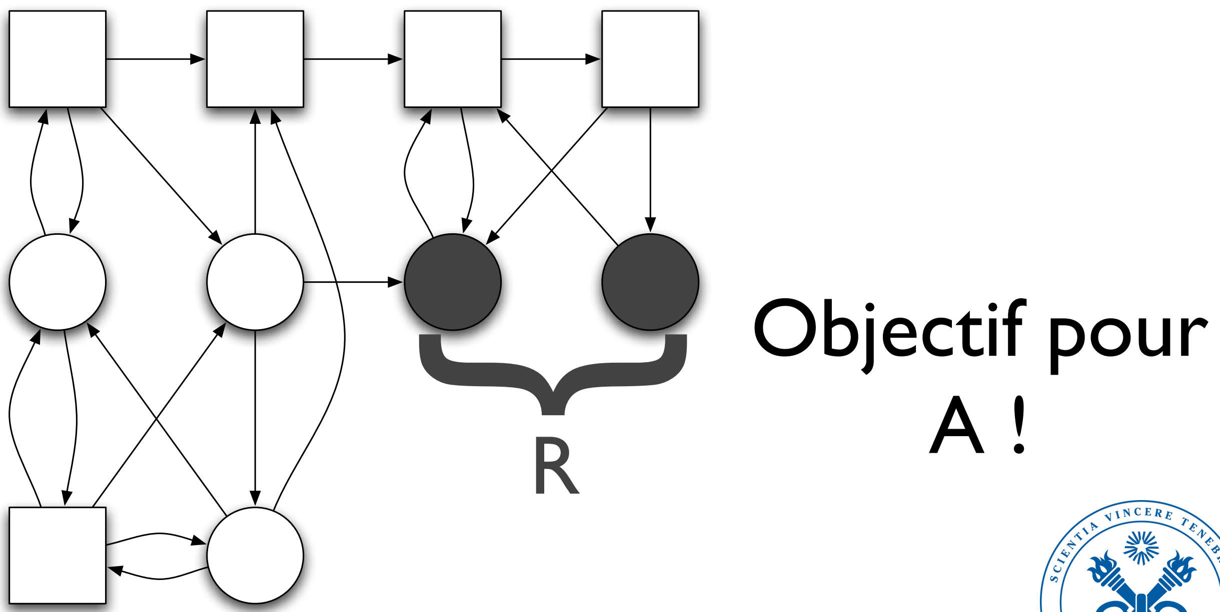
Problème de synthèse

- On peut maintenant ré-exprimer le problème de synthèse en termes de jeu.
- Etant donné un jeu $G = \langle A, \varphi \rangle$ avec $A = \langle Q, q_0, E \rangle$ et φ un objectif pour le joueur X on voudrait
  - Déterminer si $q_0 \in W_X$
  - Si oui: construire une stratégie gagnante pour le joueur X

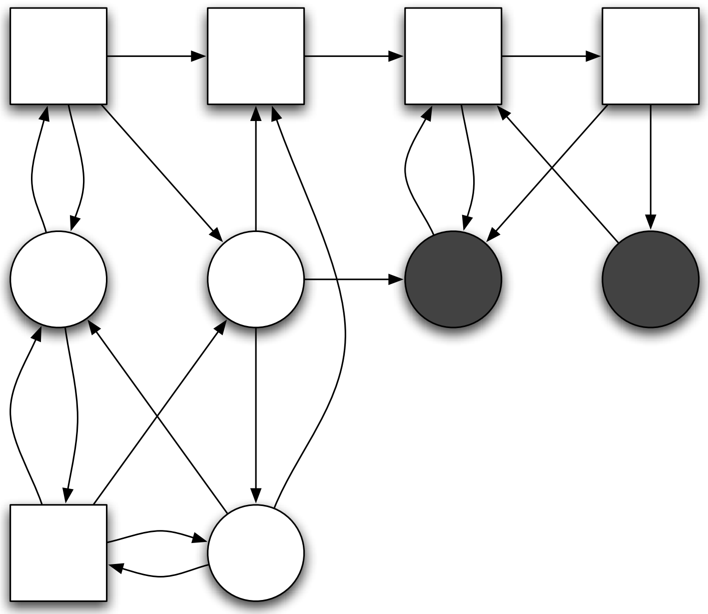
Théorie des jeux
Algorithme pour les jeux d'accessibilité

Jeux d'accessibilité

- Rappel: dans un jeu d'accessibilité, on veut être certain que, dans toute partie $p=r_1r_2r_3\ldots$, il existe une position i telle que $r_i \in$ un ensemble donné R.
- En terme de condition de Muller faible: $F = \{ S \mid S \cap R \neq \varnothing \}$
- Exemple:



Jeux d'accessibilité



- La joueur A veut forcer le jeu à atteindre les noeuds gris
- Une fois que le jeu a atteint ces noeuds, peu importe comment la partie continue (condition faible).
- On va calculer un attracteur... (= les noeuds à partir desquels on peut attirer le jeu vers certains noeuds)

Jeux d'accessibilité

- Définition: Etant donné un ensemble R de noeuds de l'arène, et un joueur X, l'attracteur de R pour X $Attr_X(R)$ est l'ensemble des noeuds à partir desquels X peut forcer le jeu à atteindre R
- A partir de ces noeuds, X possède donc une stratégie gagnante pour l'objectif "accéder à R"
- Pour calculer cet ensemble, on va procéder par étapes (point fixe).
- Définition: $Attr_X^i(R)$ est l'ensemble des noeuds à partir desquels X peut forcer le jeu à atteindre R en i coups au plus

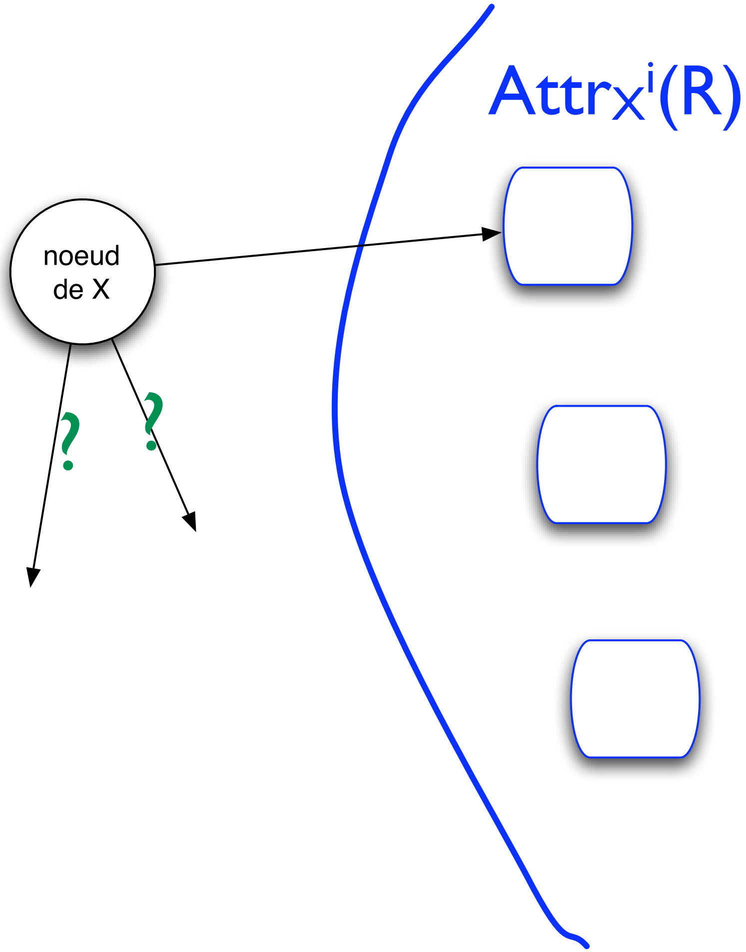
Jeux d'accessibilité

- Définition: $Attr_X^i(R)$ est l'ensemble des noeuds à partir desquels X peut forcer le jeu à atteindre R en i coups au plus
- Clairement $Attr_X^0(R) = R$. Pour être sûr d'atteindre R sans jouer de coups, il faut déjà s'y trouver.
- Comment fait-on pour calculer $Attr_X^{i+1}(R)$ à partir de $Attr_X^i(R)$?
  - Clairement $Attr_X^i(R) \subseteq Attr_X^{i+1}(R)$, mais comment sélectionner les noeuds à ajouter ?

Jeux d'accessibilité

- Comment fait-on pour calculer $Attr_X^{i+1}(R)$ à partir de $Attr_X^i(R)$?

Cas 1:

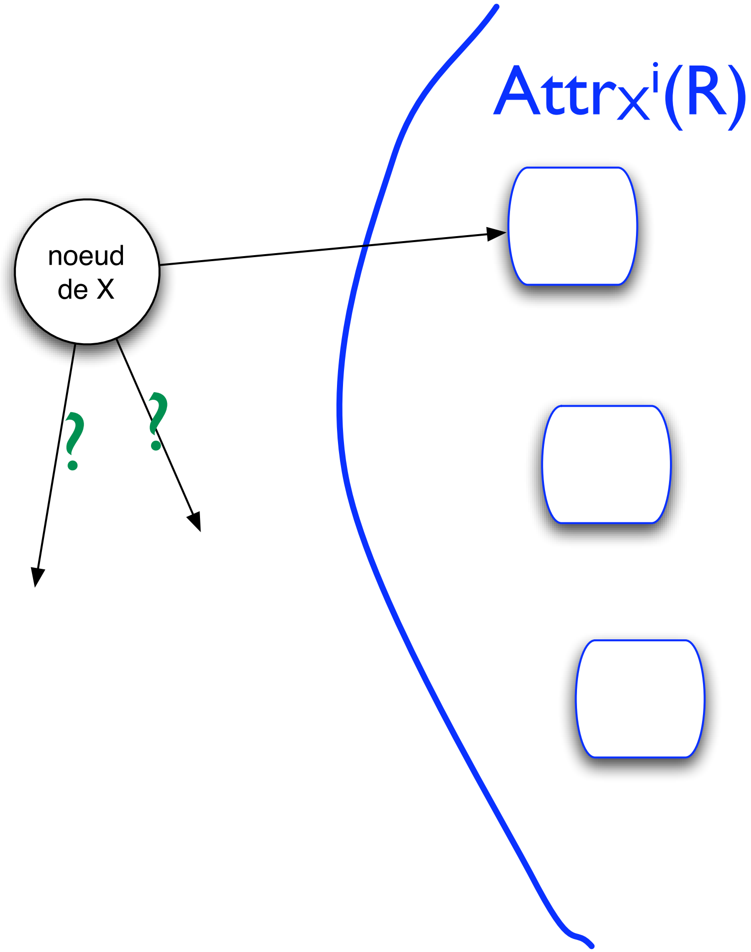


Jeux d'accessibilité

- Comment fait-on pour calculer $Attr_X^{i+1}(R)$ à partir de $Attr_X^i(R)$?

Cas 1:

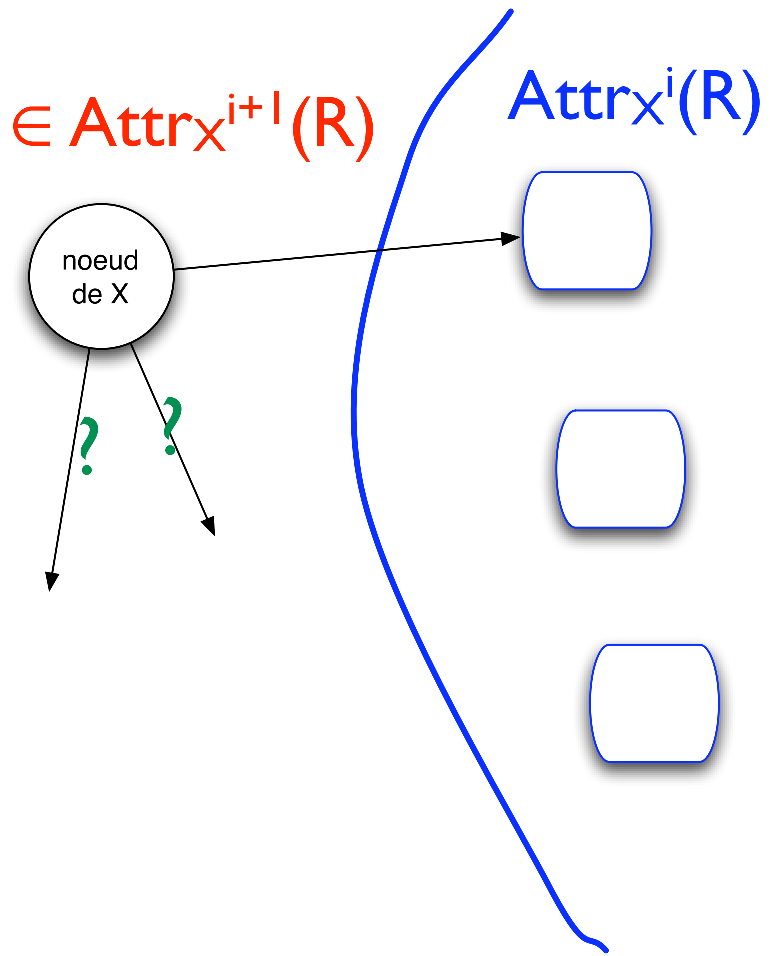
Comme c'est X qui définit la stratégie, il peut toujours choisir d'aller vers $Attr_X^i(R)$



Jeux d'accessibilité

- Comment fait-on pour calculer $Attr_X^{i+1}(R)$ à partir de $Attr_X^i(R)$?

Cas 1:

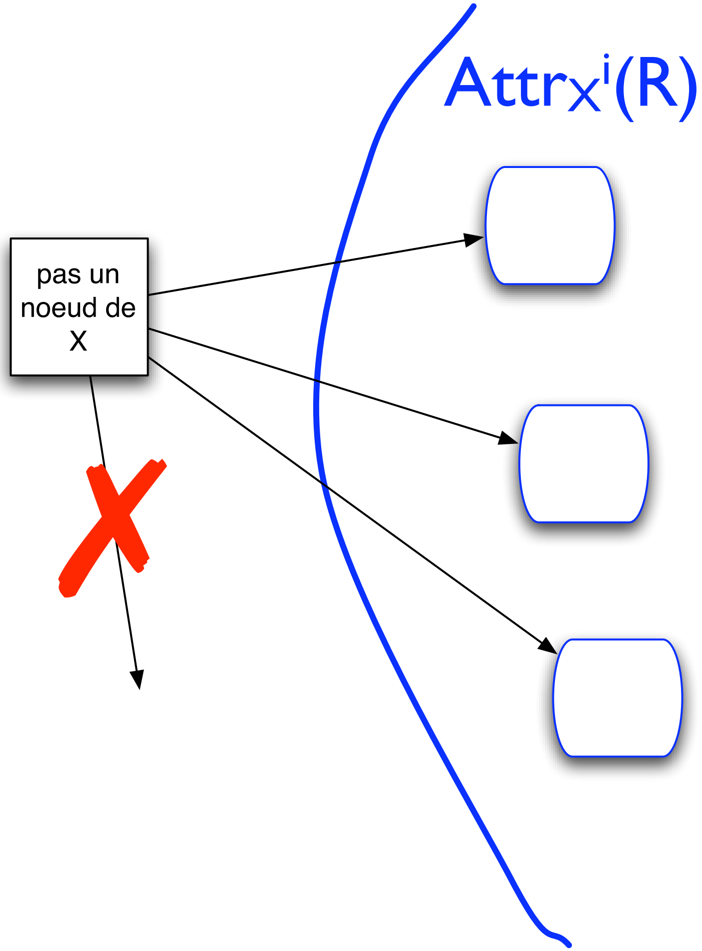


Comme c'est X qui définit la stratégie, il peut toujours choisir d'aller vers $Attr_X^i(R)$

Jeux d'accessibilité

- Comment fait-on pour calculer $Attr_X^{i+1}(R)$ à partir de $Attr_X^i(R)$?

Cas 2:

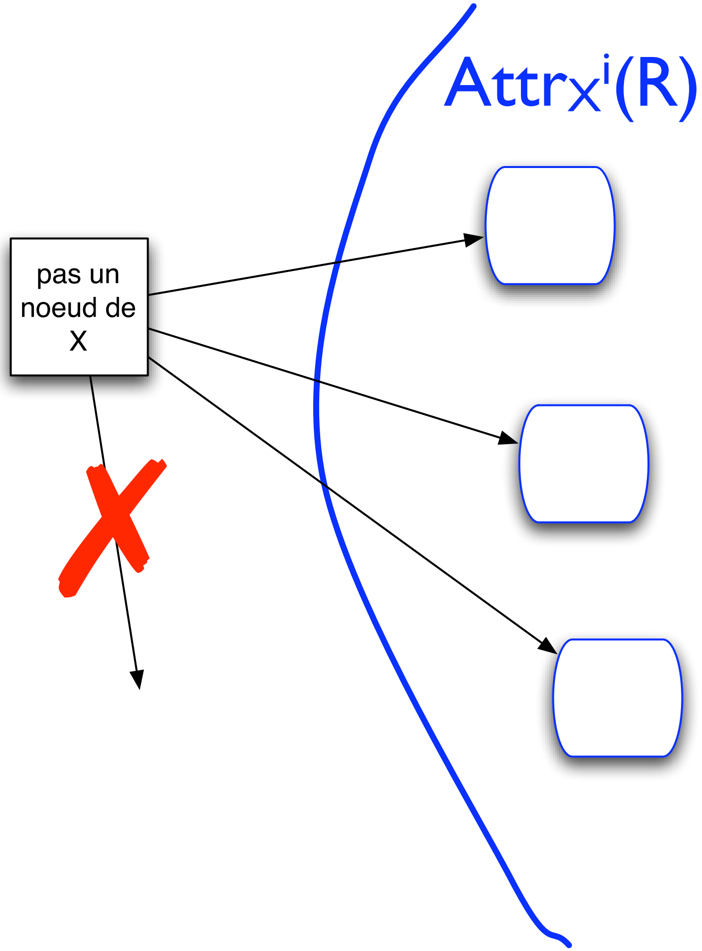


Jeux d'accessibilité

- Comment fait-on pour calculer $Attr_X^{i+1}(R)$ à partir de $Attr_X^i(R)$?

Cas 2:

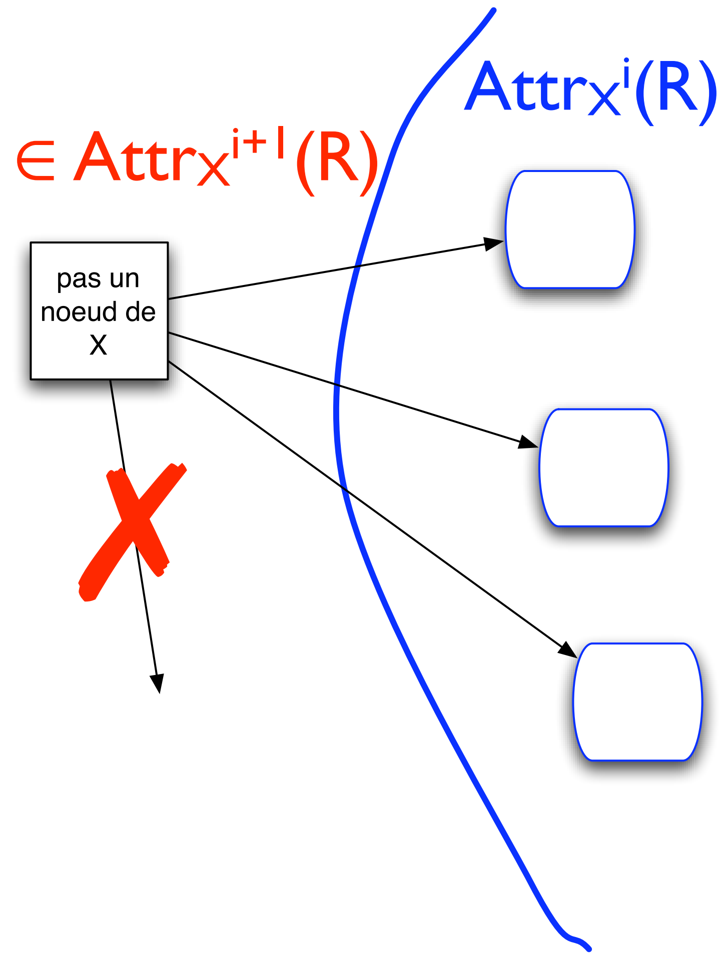
L'adversaire ne peut choisir que des successeurs dans $Attr_X^i(R)$



Jeux d'accessibilité

- Comment fait-on pour calculer $Attr_X^{i+1}(R)$ à partir de $Attr_X^i(R)$?

Cas 2:

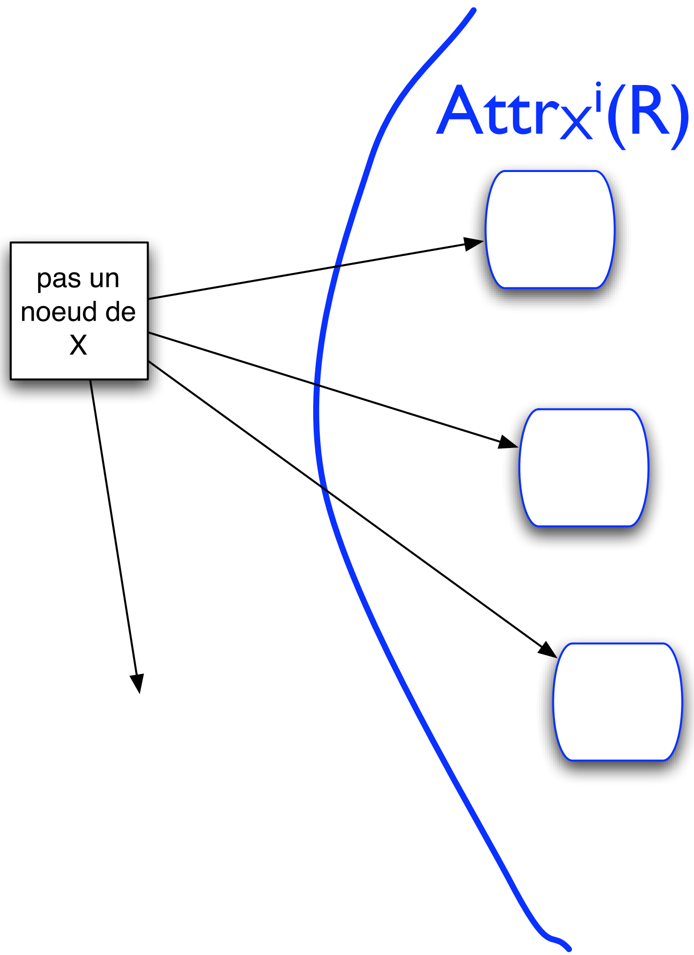


L'adversaire ne peut choisir que des successeurs dans $Attr_X^i(R)$

Jeux d'accessibilité

- Comment fait-on pour calculer $Attr_X^{i+1}(R)$ à partir de $Attr_X^i(R)$?

Cas 3:

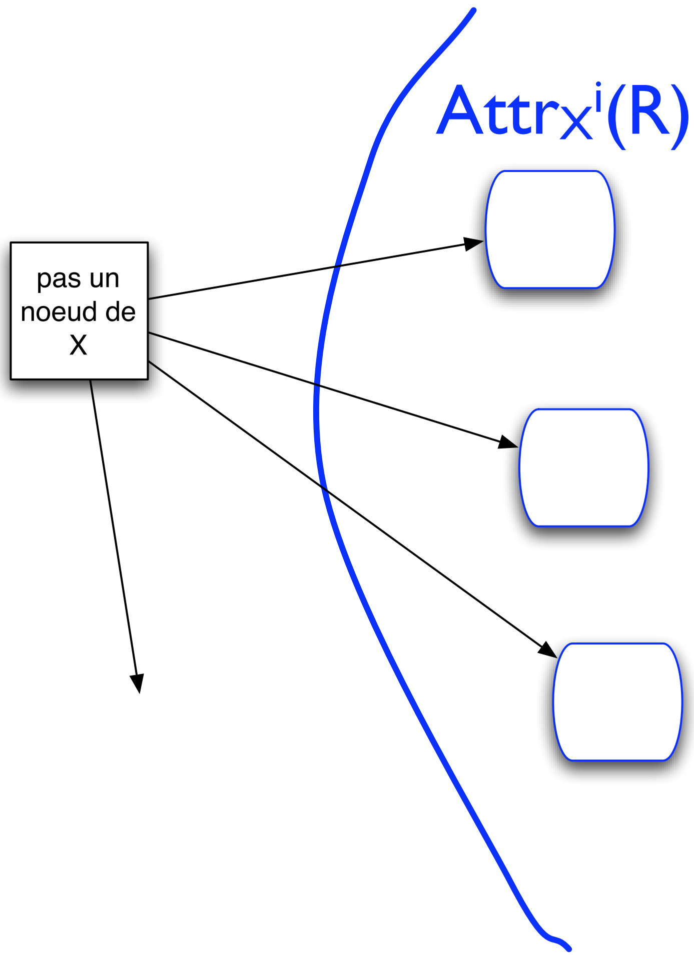


Jeux d'accessibilité

- Comment fait-on pour calculer $Attr_X^{i+1}(R)$ à partir de $Attr_X^i(R)$?

Cas 3:

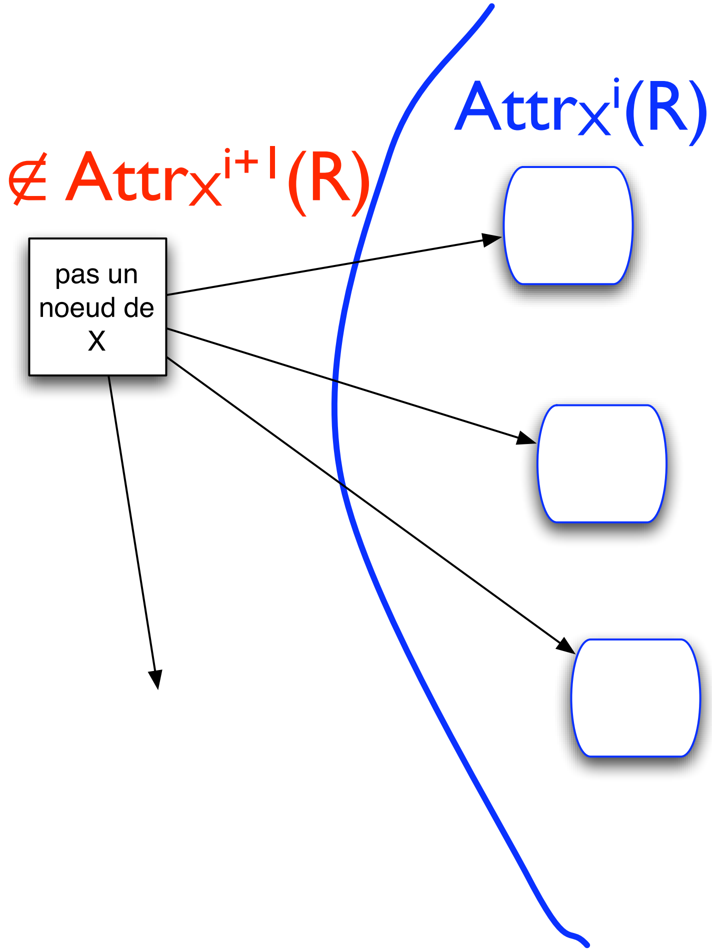
L'adversaire peut choisir un successeur hors de $Attr_X^i(R)$



Jeux d'accessibilité

- Comment fait-on pour calculer $Attr_X^{i+1}(R)$ à partir de $Attr_X^i(R)$?

Cas 3:



L'adversaire peut choisir un successeur hors de $Attr_X^i(R)$

Jeux d'accessibilité

- On obtient donc:

$$\begin{array}{rcl} \mathrm{Attr}_X^0(R) & = & R \\ \mathrm{Attr}_X^{i+1}(R) & = & \mathrm{Attr}_X^i \\ & & \cup\{q \in Q_X \mid \exists (q,r) \in E : r \in \mathrm{Attr}_X^i(R)\} \\ & & \cup\{q \in Q \setminus Q_X \mid \forall (q,r) \in E : r \in \mathrm{Attr}_X^i(R)\} \end{array}$$

Mais cela définit une séquence infinie d'ensembles !

Jeux d'accessibilité

$Attr_X^0(R) \subseteq \quad Attr_X^1(R) \subseteq \quad Attr_X^2(R) \subseteq \; \ldots \; \subseteq \; Attr_X^k(R) \; \ldots$

- Théorème: La séquence des $Attr_X^i(R)$ converge
  - Preuve: La séquence est croissante, et chaque $Attr_X^i(R)$ est contenu dans Q, qui est fini.
- On va donc considérer la première position k telle que $Attr_X^k(R) = Attr_X^{k+1}(R)$
- On a donc: $Attr_X(R) = Attr_X^k(R)$

Jeux d'accessibilité

$Attr_X^0(R)$ $Attr_X^1(R)$ $Attr_X^2(R)$ ... $Attr_X^k(R)$ ...

- Théorème: La séquence des $Attr_X^i(R)$ converge
  - Preuve: La séquence est croissante, et chaque $Attr_X^i(R)$ est contenu dans Q, qui est fini.
- On va donc considérer la première position k telle que $Attr_X^k(R) = Attr_X^{k+1}(R)$
- On a donc: $Attr_X(R) = Attr_X^k(R)$

Jeux d'accessibilité

$$\mathrm{Attr}_X^0(R) \subset \mathrm{Attr}_X^1(R) \subset \mathrm{Attr}_X^2(R) \subset \ldots \subset \mathrm{Attr}_X^k(R) = \ldots$$

- Théorème: La séquence des $\mathrm{Attr}_X^i(R)$ converge
  - Preuve: La séquence est croissante, et chaque $\mathrm{Attr}_X^i(R)$ est contenu dans Q, qui est fini.
- On va donc considérer la première position k telle que $\mathrm{Attr}_X^k(R) = \mathrm{Attr}_X^{k+1}(R)$
- On a donc: $\mathrm{Attr}_X(R) = \mathrm{Attr}_X^k(R)$

Jeux d'accessibilité

$\mathrm{Attr}_X^0(R) \subset \mathrm{Attr}_X^1(R) \subset \mathrm{Attr}_X^2(R) \subset \ldots \subset \mathrm{Attr}_X^k(R) \ldots =$

- Théorème: $W_X = \mathrm{Attr}_X(R)$
  - Preuve (1): $\mathrm{Attr}_X(R) \subseteq W_X$ (il n'y a que des positions gagnantes dans l'attracteur)
    Clairement, on n'a mis dans $\mathrm{Attr}_X(R)$ que des positions gagnantes pour X (voir définition de Attr). Cette direction est donc triviale.

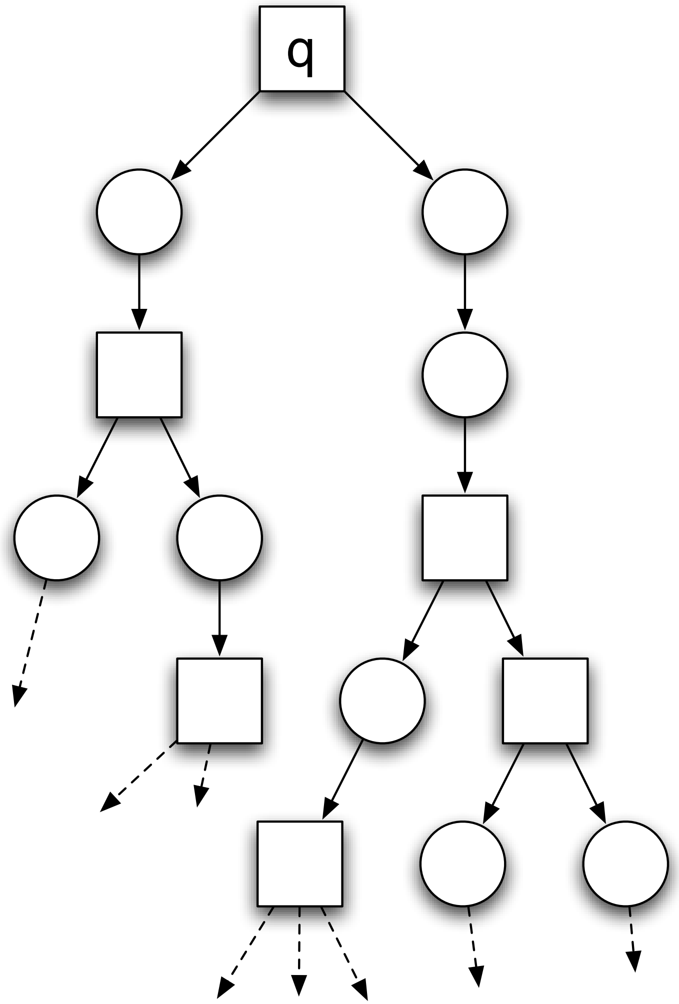
Jeux d'accessibilité

$$Attr_X^0(R) \subset Attr_X^1(R) \subset Attr_X^2(R) \subset \ldots \subset Attr_X^k(R) = \ldots$$

- Théorème: $W_X = Attr_X(R)$
  - Preuve (2): $Attr_X(R) \supseteq W_X$ (toutes les positions gagnantes sont dans l'attracteur)
    - Par contradiction: considérons une position gagnante q pour X ($q \in W_X$) et supposons que $q \notin Attr_X(R) = Attr_X^k(R)$.
    - Comme $q \in W_X$, X a une stratégie gagnante f
    - On va construire l'arbre de toutes les parties possibles à partir de q, selon f

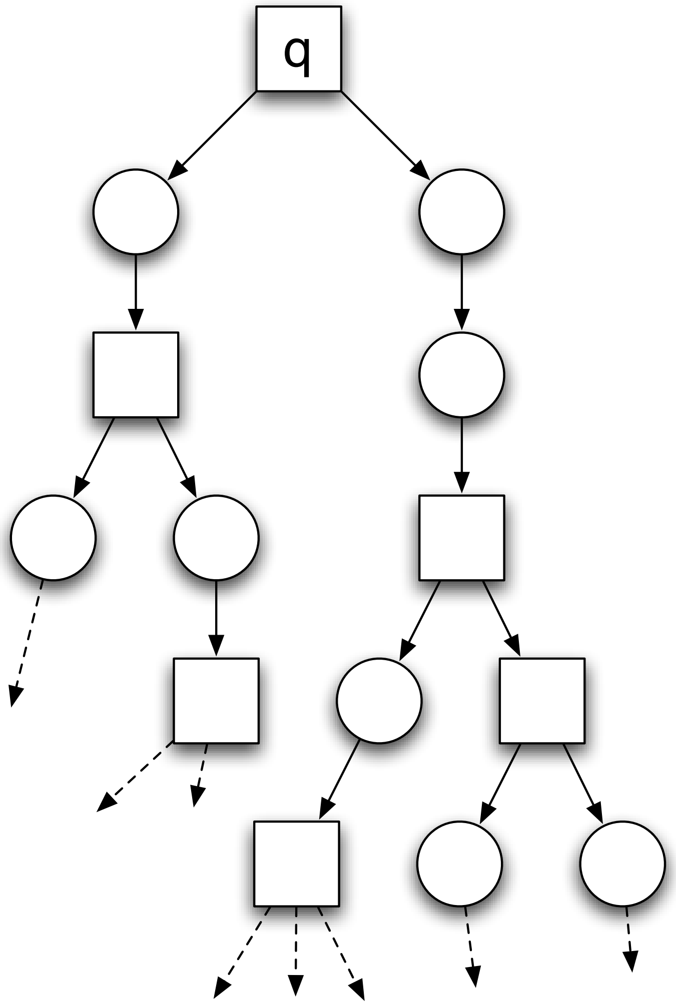
Jeux d'accessibilité

- Chaque position q' de X a un seul fils: f(q')
- L'ensemble des fils de chaque position q' de l'adversaire est l'ensemble des successeurs de q' dans le jeu
- L'arbre est infini



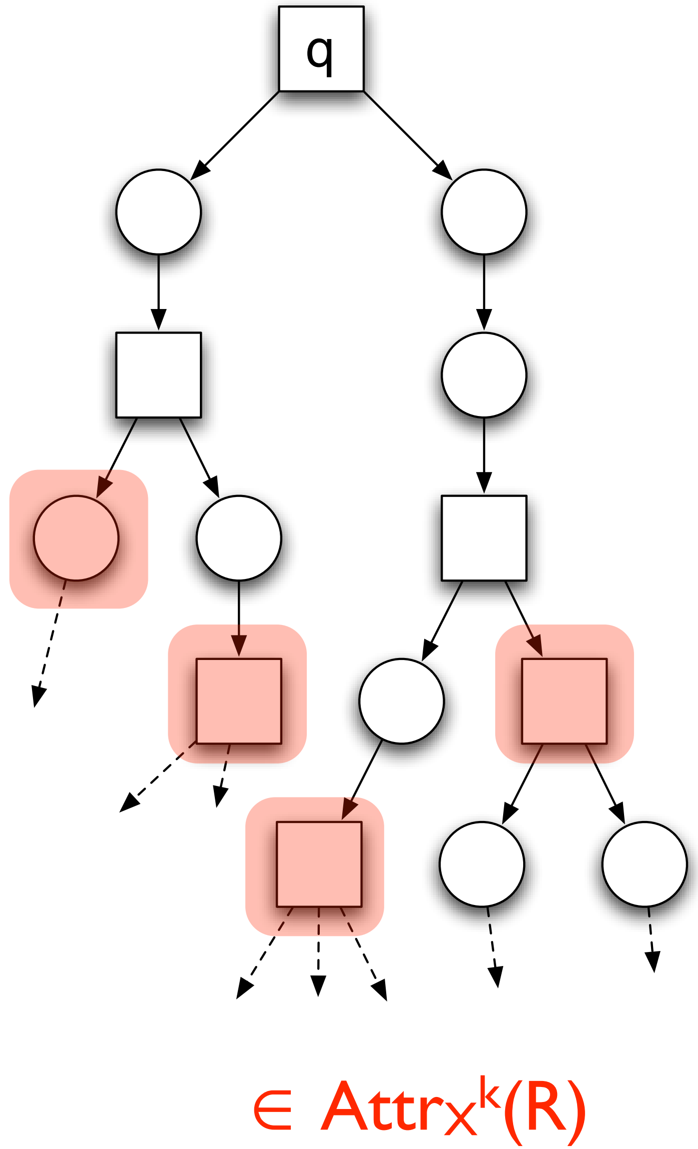
Jeux d'accessibilité

- Chaque branche passe obligatoirement par une position $\in Attr_X^k(R)$ car:
  - f est une strat. gagnante
  - $R \subseteq Attr_X^k(R)$
- On peut donc séparer les noeuds de l'arbre en deux:
  - Ceux qui sont au-dessus des noeuds $\in Attr_X k(R)$
  - Ceux en-dessous

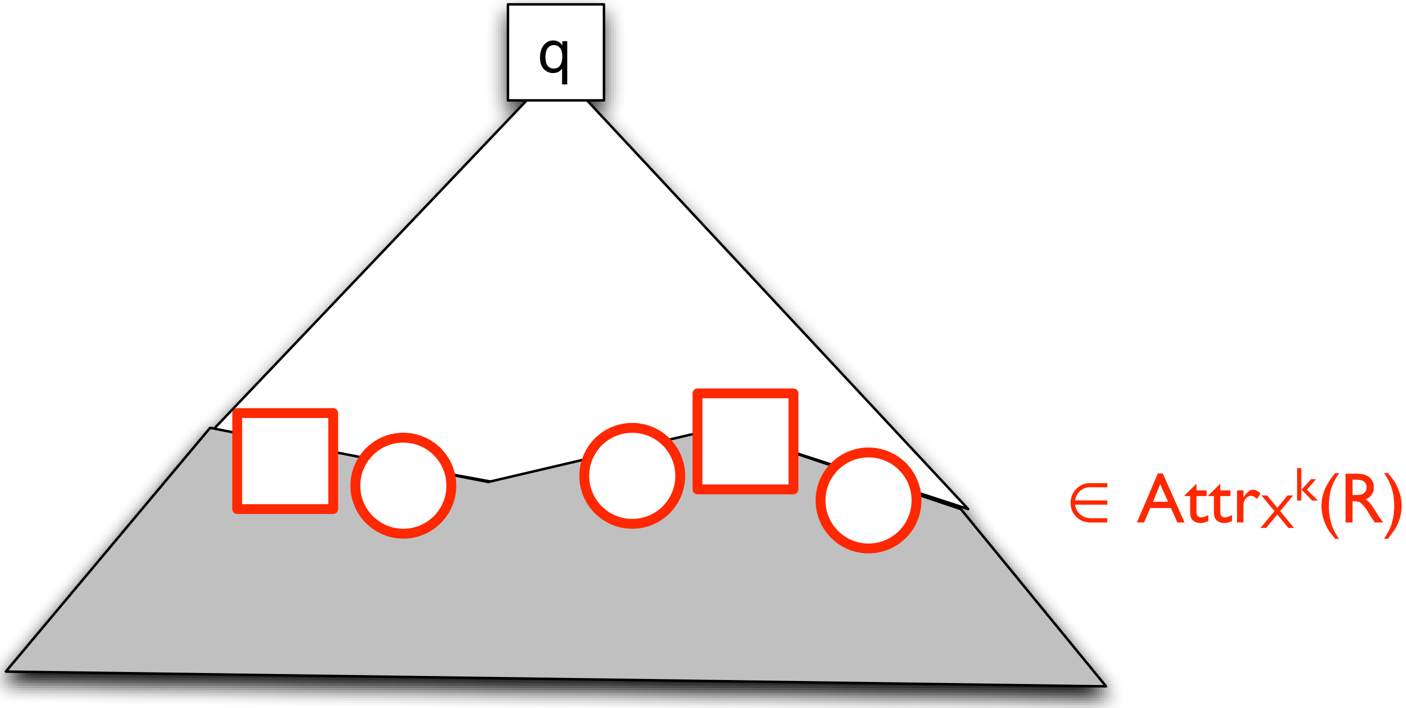


Jeux d'accessibilité

- Chaque branche passe obligatoirement par une position $\in \mathrm{Attr}_X^k(R)$ car:
  - f est une strat. gagnante
  - $R \subseteq \mathrm{Attr}_X^k(R)$
- On peut donc séparer les noeuds de l'arbre en deux:
  - Ceux qui sont au-dessus des noeuds $\in \mathrm{Attr}_X k(R)$
  - Ceux en-dessous



Jeux d'accessibilité

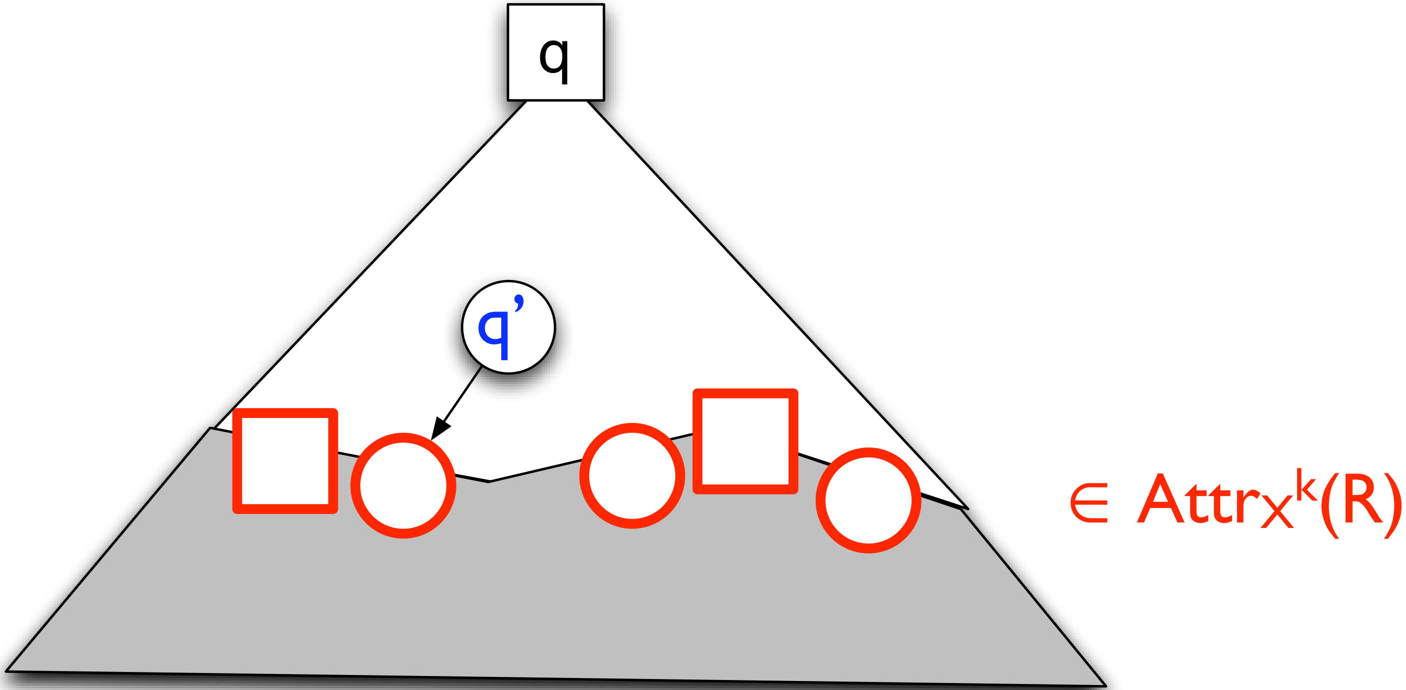


- Regardons les pères des noeuds "rouges" $\in Attr_X^k(R)$.

  Ces pères $\notin Attr_X^k(R)$

- S'il y a un père q' de X, alors q' doit appartenir à $Attr_X^{k+1}(R)$. Comme $q' \notin Attr_X^k(R)$, on a $Attr_X^k(R) \subset Attr_X^{k+1}(R)$. Contradiction.

Jeux d'accessibilité

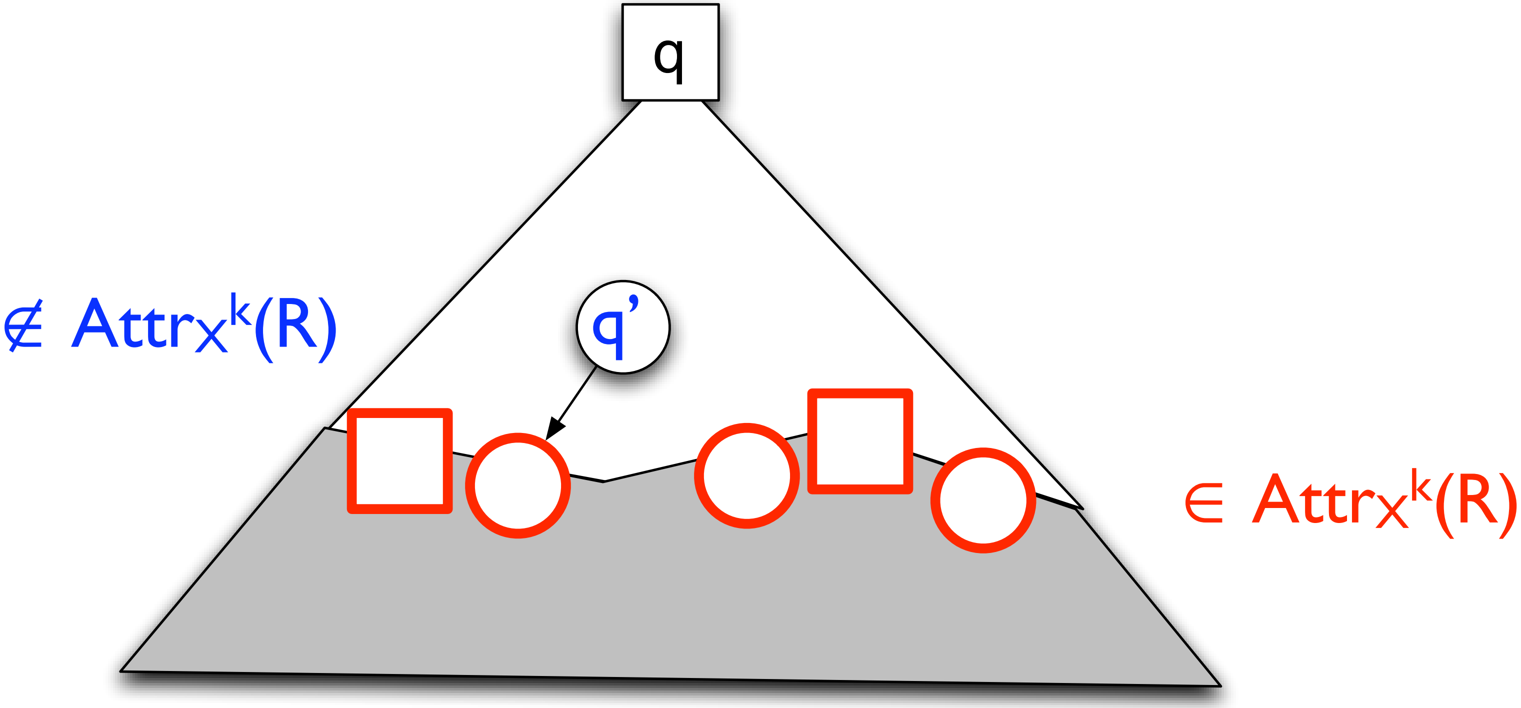


- Regardons les pères des noeuds "rouges" $\in Attr_X^k(R)$.

  Ces pères $\notin Attr_X^k(R)$

- S'il y a un père q' de X, alors q' doit appartenir à $Attr_X^{k+1}(R)$. Comme q'$\notin Attr_X^k(R)$, on a $Attr_X^k(R) \subset Attr_X^{k+1}(R)$. Contradiction.

Jeux d'accessibilité

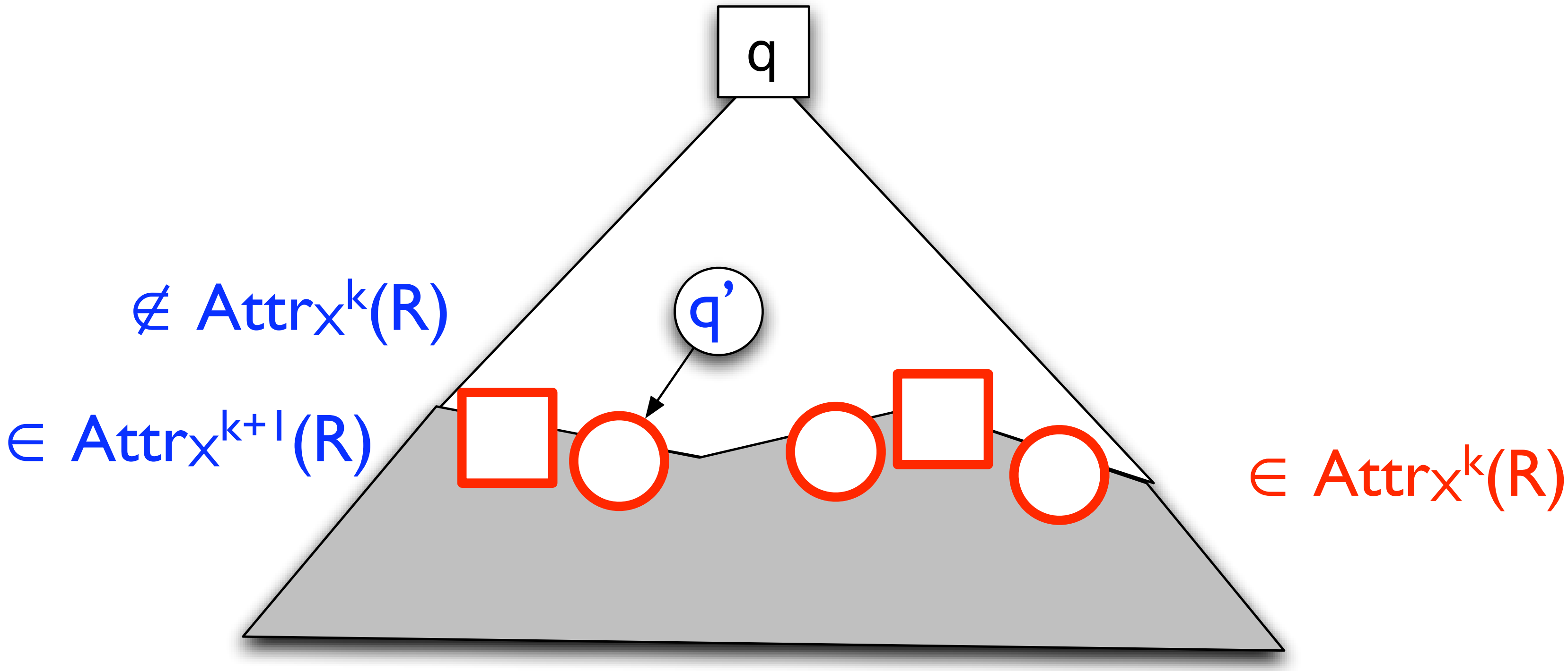


- Regardons les pères des noeuds "rouges" $\in \mathrm{Attr}_X^k(R)$.

  Ces pères $\notin \mathrm{Attr}_X^k(R)$

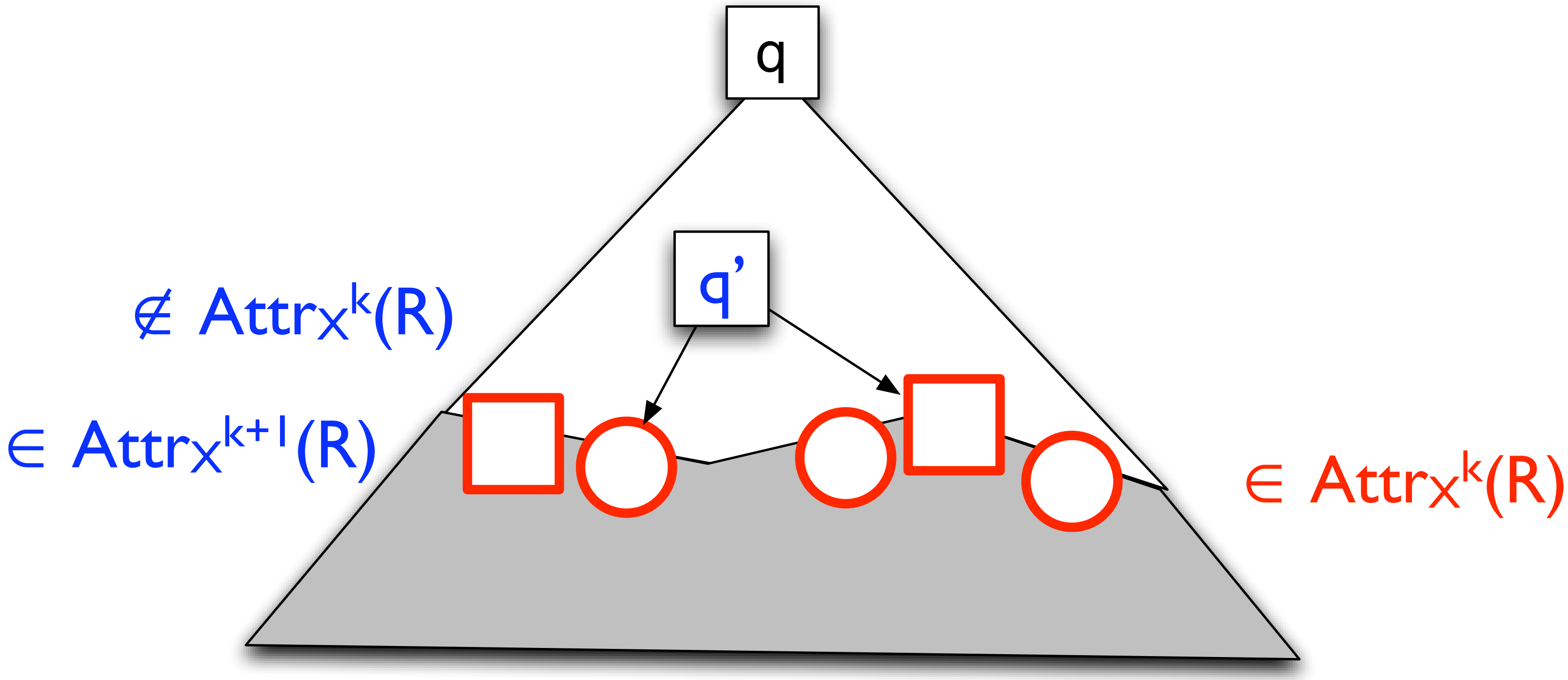
- S'il y a un père q' de X, alors q' doit appartenir à $\mathrm{Attr}_X^{k+1}(R)$. Comme q'$\notin \mathrm{Attr}_X^k(R)$, on a $\mathrm{Attr}_X^k(R) \subset \mathrm{Attr}_X^{k+1}(R)$. Contradiction.

Jeux d'accessibilité



- Regardons les pères des noeuds "rouges" $\in \mathrm{Attr}_X^k(R)$.
  Ces pères $\notin \mathrm{Attr}_X^k(R)$
- S'il y a un père q' de X, alors q' doit appartenir à $\mathrm{Attr}_X^{k+1}(R)$. Comme $q' \notin \mathrm{Attr}_X^k(R)$, on a $\mathrm{Attr}_X^k(R) \subset \mathrm{Attr}_X^{k+1}(R)$. Contradiction.

Jeux d'accessibilité



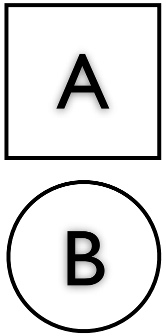
- Sinon, tous les pères appartiennent à l'adversaire et ont tous leurs fils dans Attrx^k(R). Ils doivent donc appartenir à Attrx^{k+1}(R). Donc Attrx^k (R) $\subset$ Attrx^{k+1}(R). Contradiction.

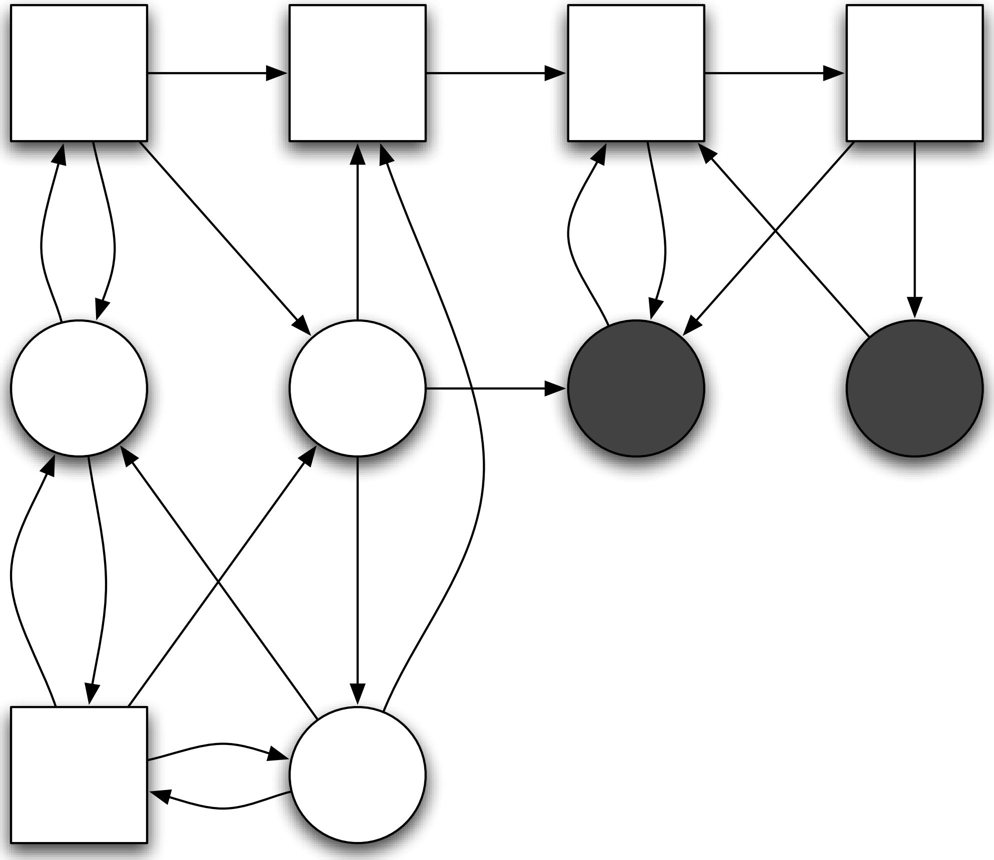
CQFD

Jeux d'accessibilité

- Nous pouvons maintenant calculer l'ensemble des positions gagnantes W_X d'un joueur X pour un objectif d'accessibilité R:
  - Il suffit de calculer $Attr_X(R)$ (point fixe)
- X possède donc une stratégie gagnante ssi la position initiale $q_0 \in W_X$
- Comment calculer cette stratégie ?
  - Pour chaque position $q \in W_X$, on choisit f(q) parmi les successeurs de q qui appartiennent à W_X.
  - Le point fixe nous donne en fait une famille de stratégies positionelles.

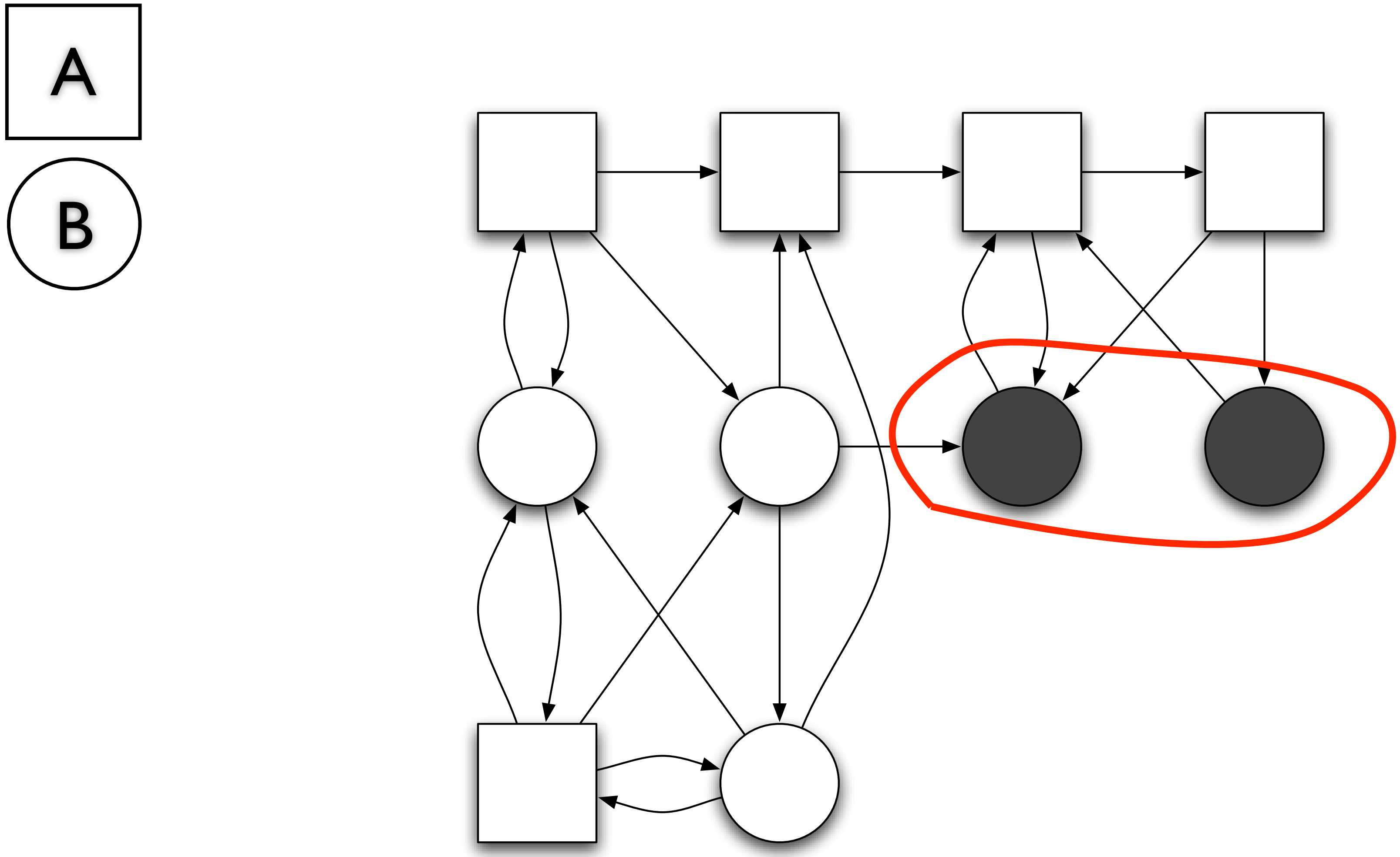
Exemple - accessibilité





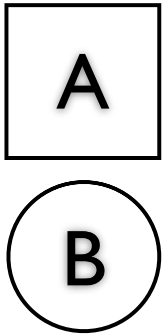
Les noeuds en gris sont un objectif pour B qui joue avec les ronds.

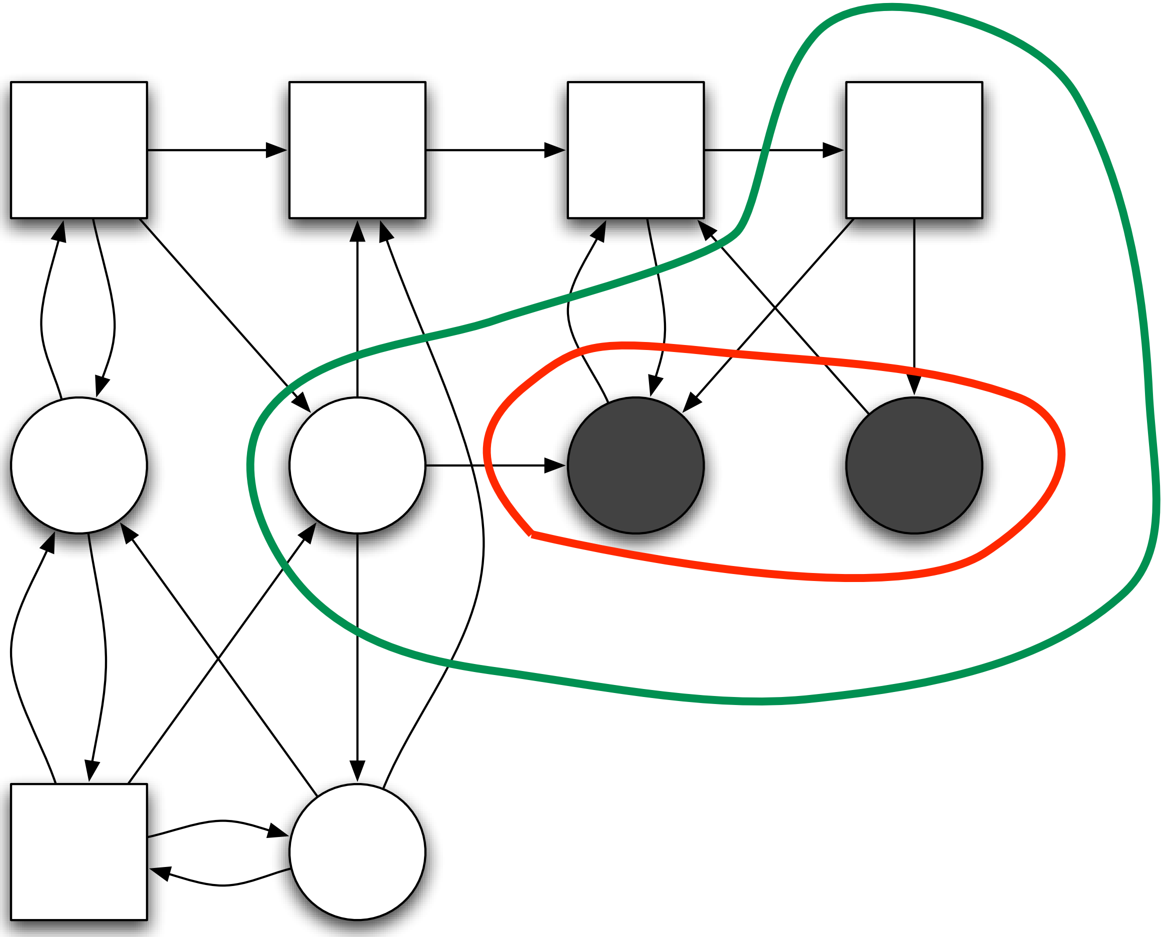
Exemple - accessibilité



Les noeuds en gris sont un objectif pour B qui joue avec les ronds.

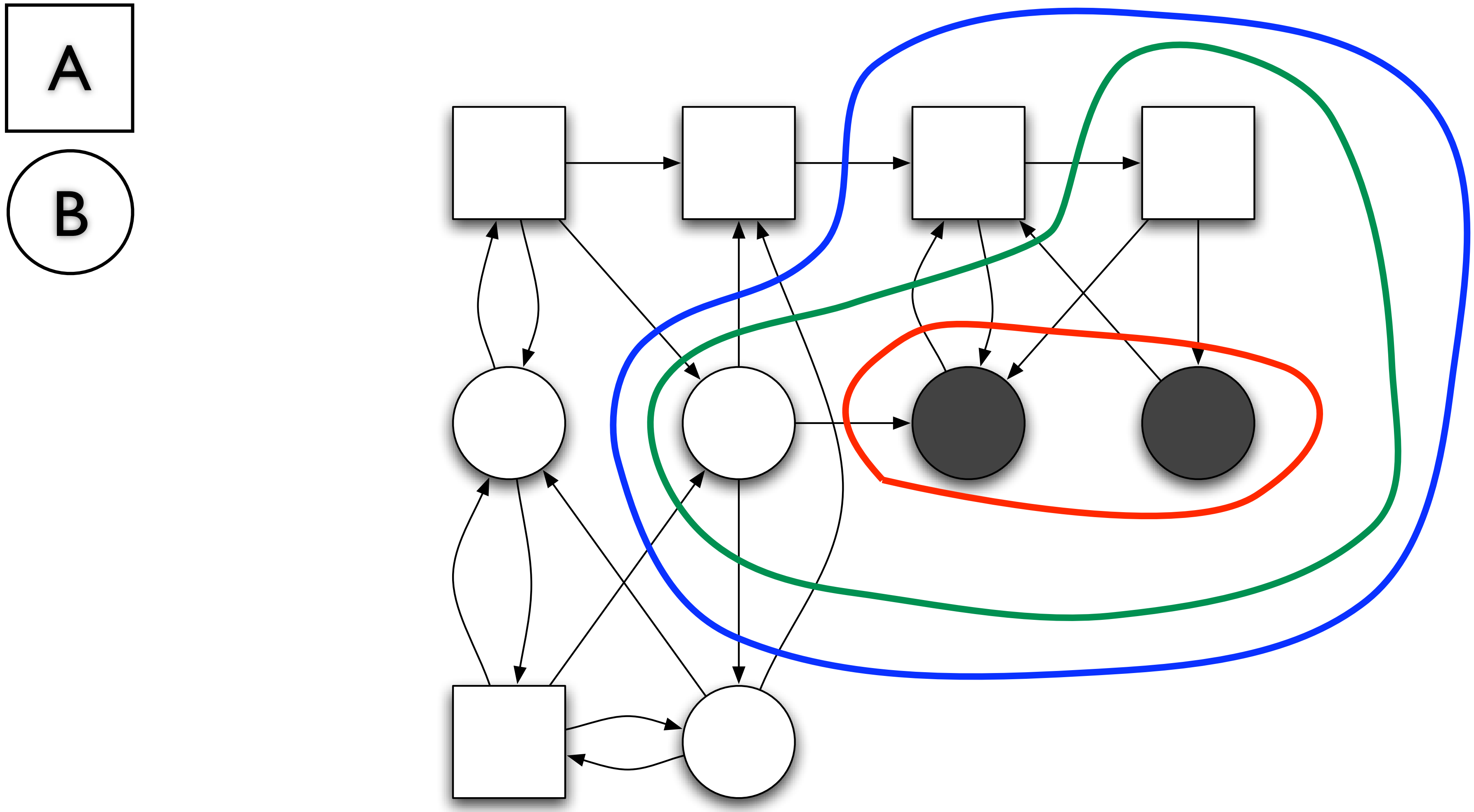
Exemple - accessibilité





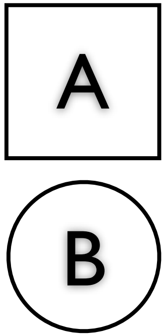
Les noeuds en gris sont un objectif pour B qui joue avec les ronds.

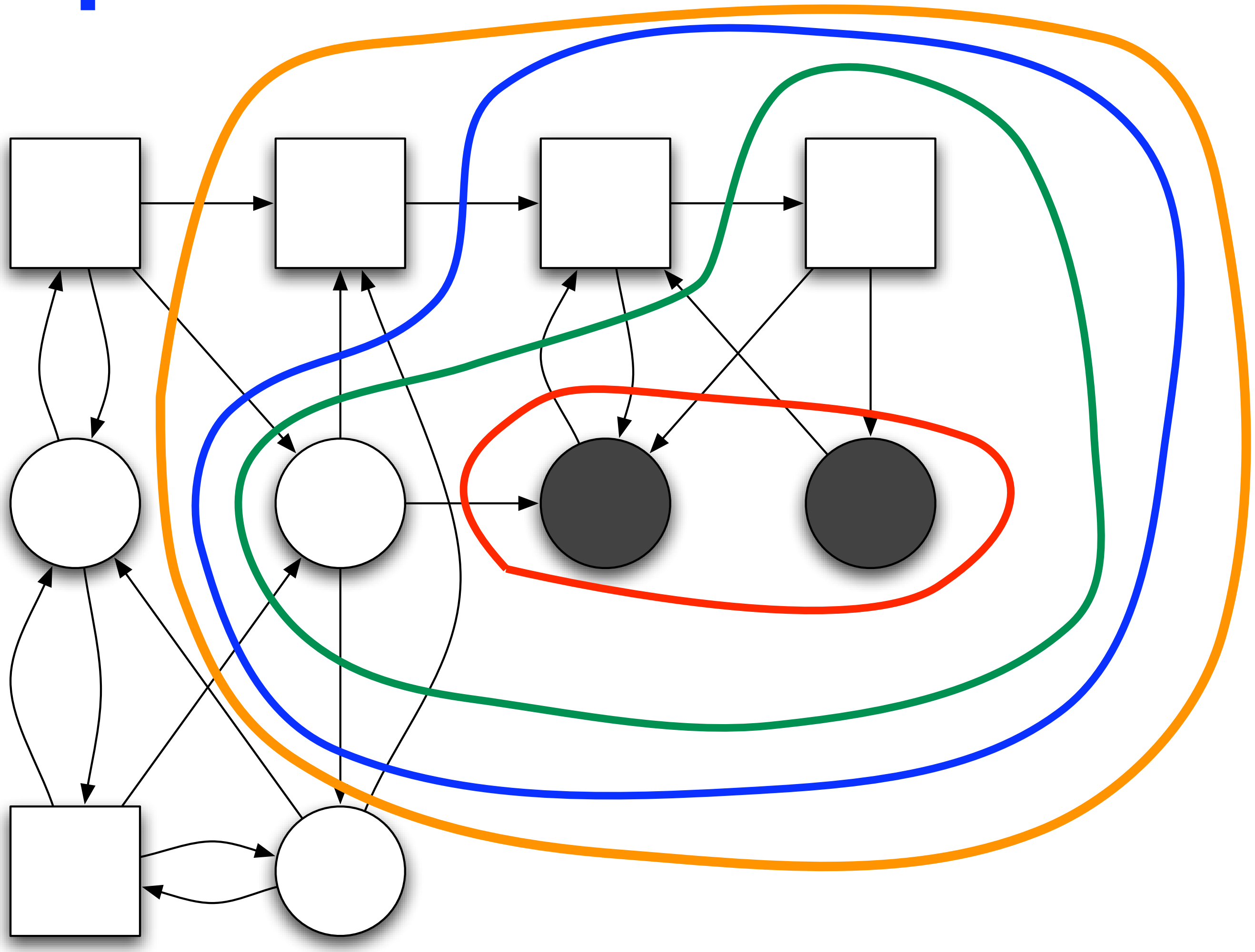
Exemple - accessibilité



Les noeuds en gris sont un objectif pour B qui joue avec les ronds.

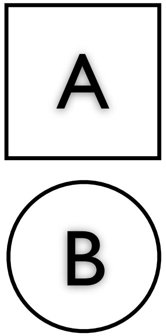
Exemple - accessibilité

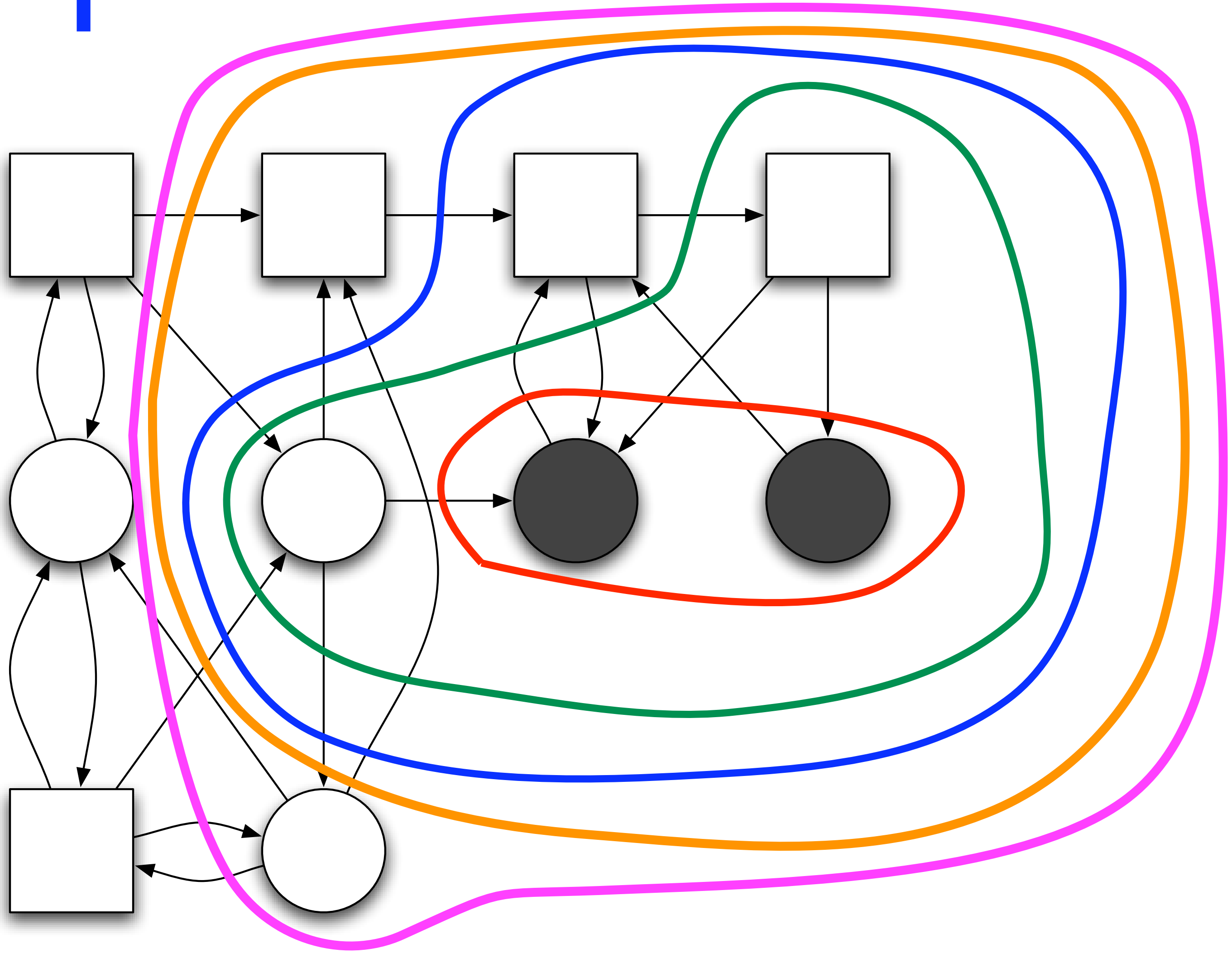




Les noeuds en gris sont un objectif pour B qui joue avec les ronds.

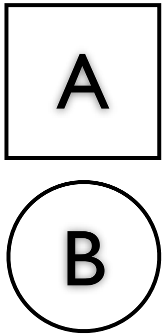
Exemple - accessibilité

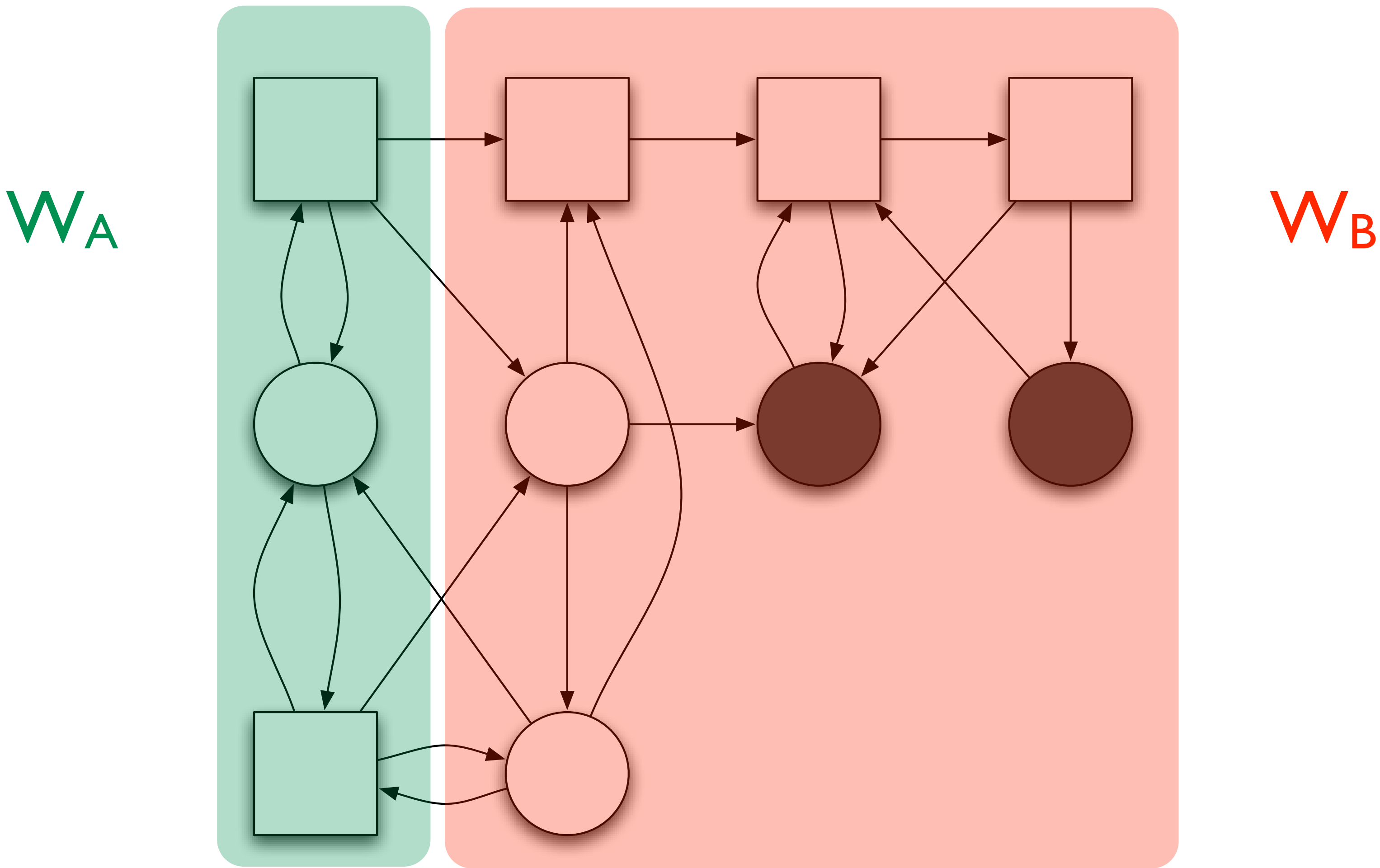




Les noeuds en gris sont un objectif pour B qui joue avec les ronds.

Exemple - accessibilité

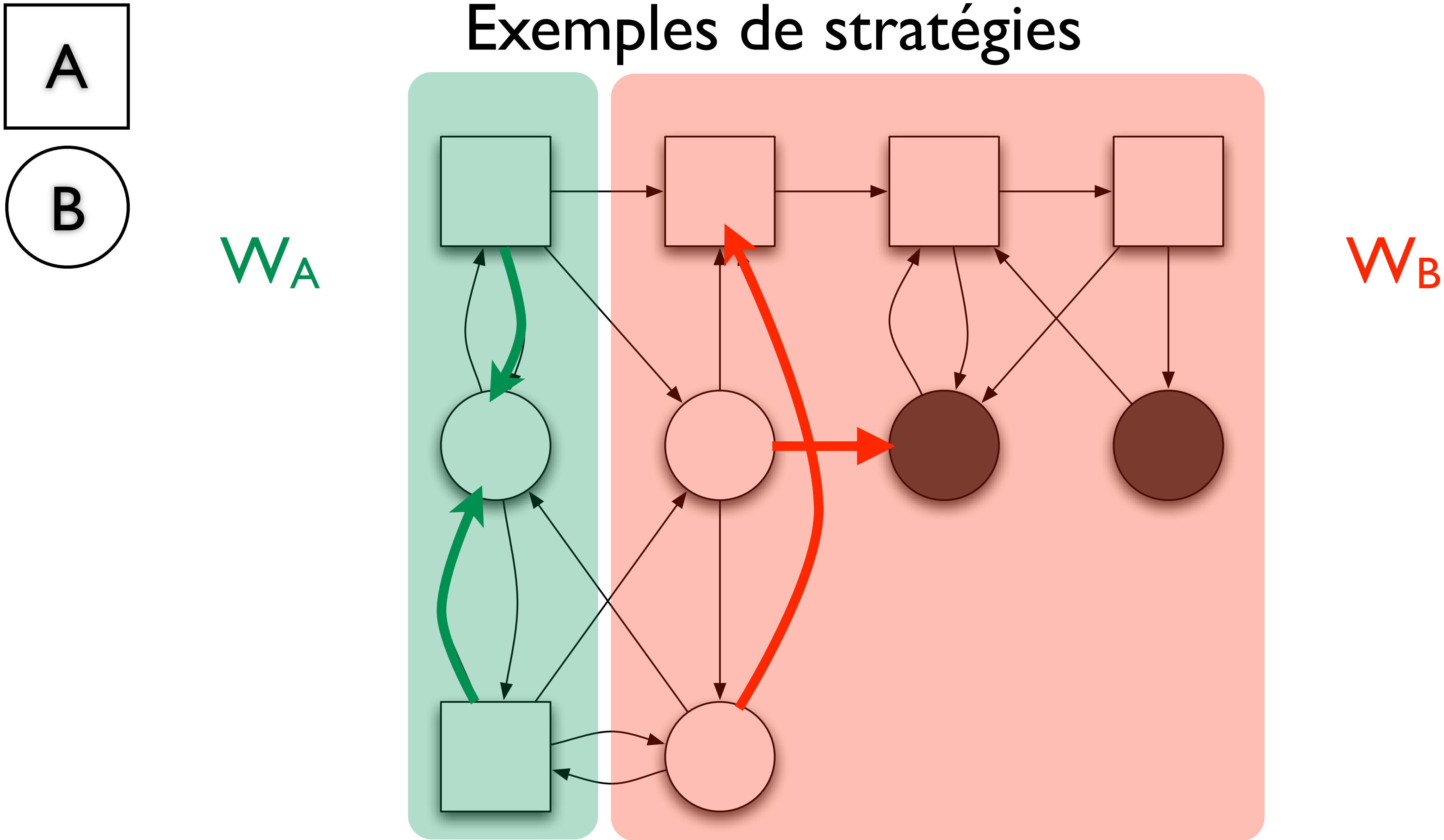




Les noeuds en gris sont un objectif pour B qui joue avec les ronds.

Exemple - accessibilité

Exemples de stratégies



Les noeuds grisés sont un objectif pour B qui joue avec les ronds.

Théorie des jeux
Algorithme pour les jeux de Muller faibles

Rappel

- Condition de Muller faible:
- Fixons un ensemble F d'ensembles de noeuds de l'arène et une partie p
- p est une partie gagnante ssi Occ(p)∈F
  - Occ(p) = ensemble de positions qui apparaissent finiment ou infiniment souvent dans p

Jeux de Muller faibles

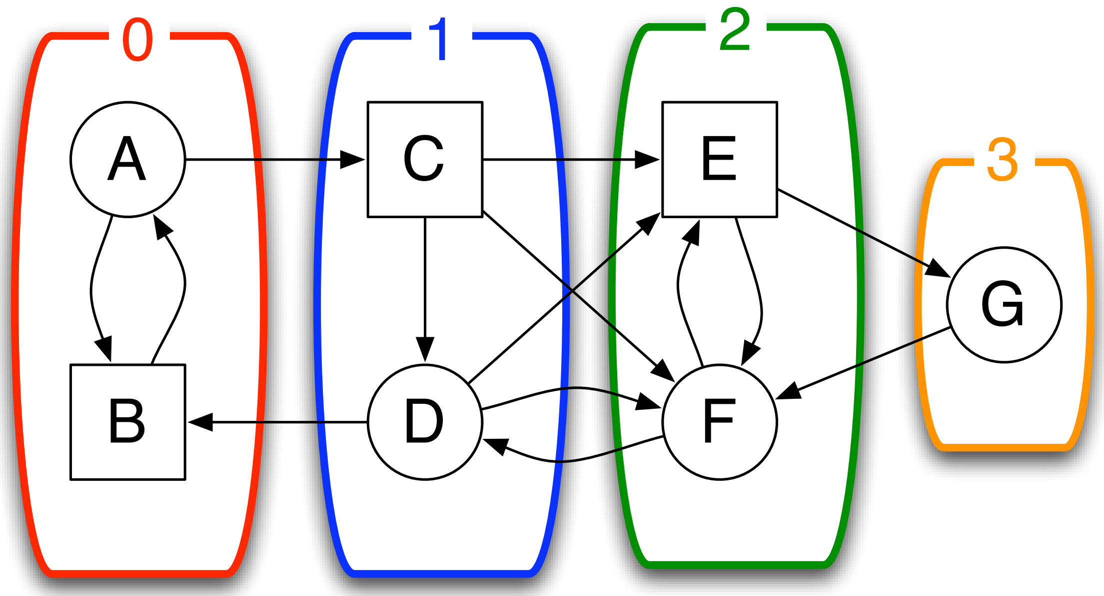
- Ces jeux généralisent les jeux d'accessibilité. On a donc des résultats un peu plus généraux:
  - Théorème: Etant donné un jeu de Muller faible, on peut déterminer, pour chaque joueur, à partir de quelle position il possède un stratégie gagnante.
  - Théorème: Si un joueur possède une stratégie gagnante dans un jeu de Muller faible, alors il possède en particulier une stratégie gagnante à mémoire finie. On peut la calculer.

Jeux de parité

- Pour résoudre ces jeux de Muller faibles, on va utiliser un outil classique de la théorie des jeux: les jeux de parité (faibles)
- Définition: un jeu de parité est un jeu $\langle Q,E \rangle$ auquel on ajoute une fonction de coloration des positions $c: Q \rightarrow \{1,\ldots,k\}$
  - Ici, les "couleurs" sont en fait des nombres naturels. On étend cette fonction aux parties: $c(q_1q_2q_3\ldots)=c(q_1)c(q_2)c(q_3)\ldots$
- Définition: une partie p dans un jeu de parité est gagnée par B ssi max(Occ(c(p))) est pair

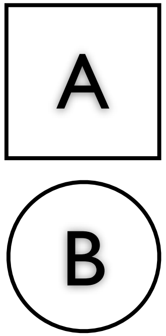
Jeux de parité - exemple

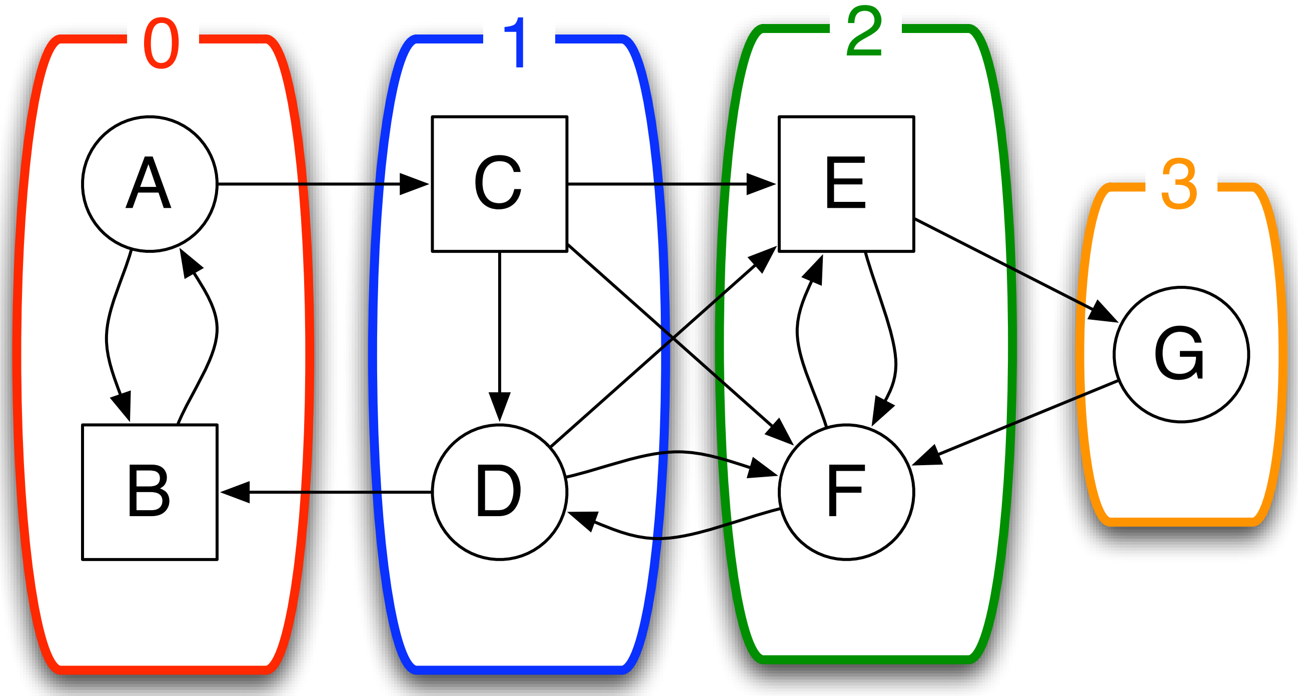
A: impair
B: pair



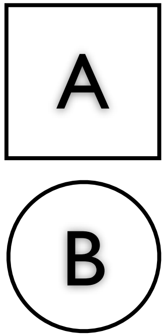
- La partie p=(ACDB)$^\omega$ est gagnée par A. max(Occ(c(p)))=1.
- La partie p=ACEFDB(ACDB)$^\omega$ est gagnée par B. max(Occ(c(p)))=2.

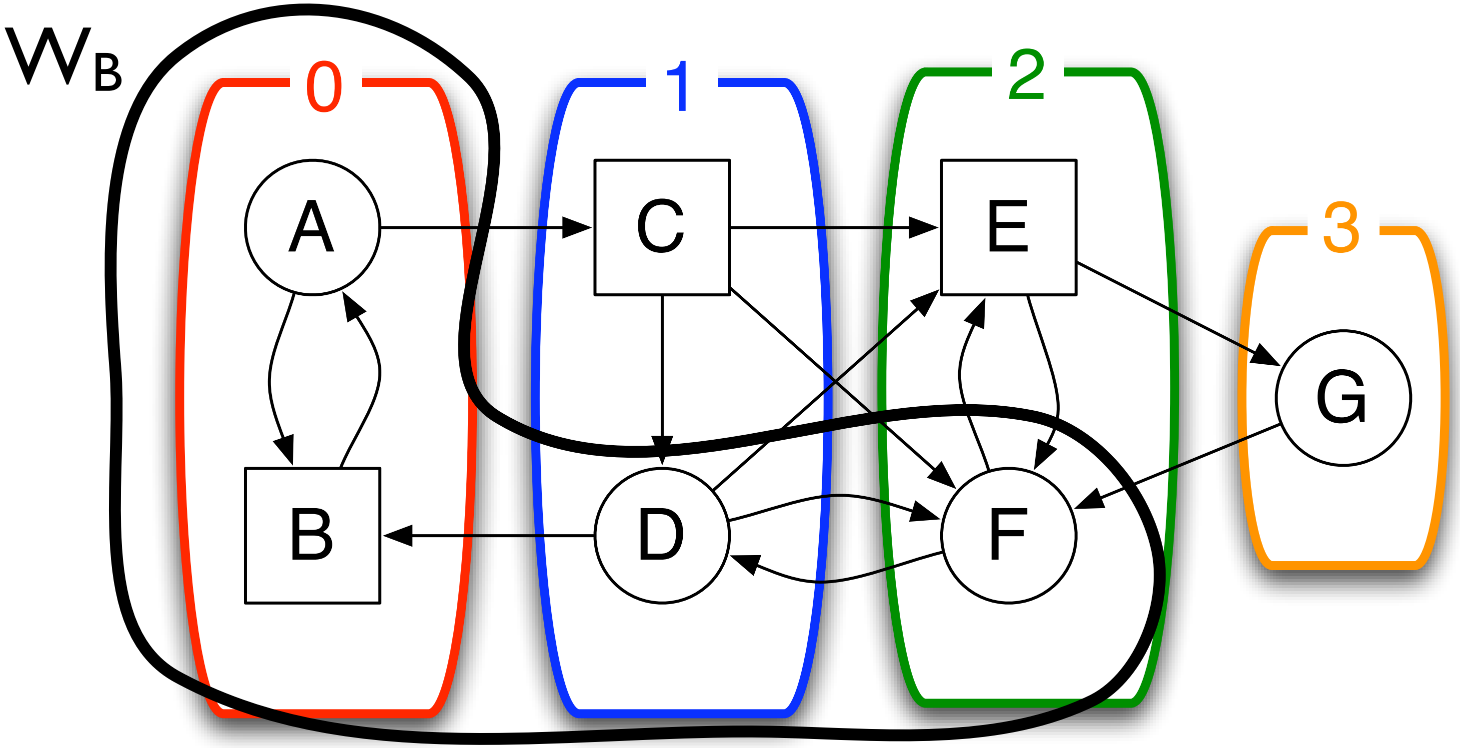
Jeux de parité - exemple de stratégies



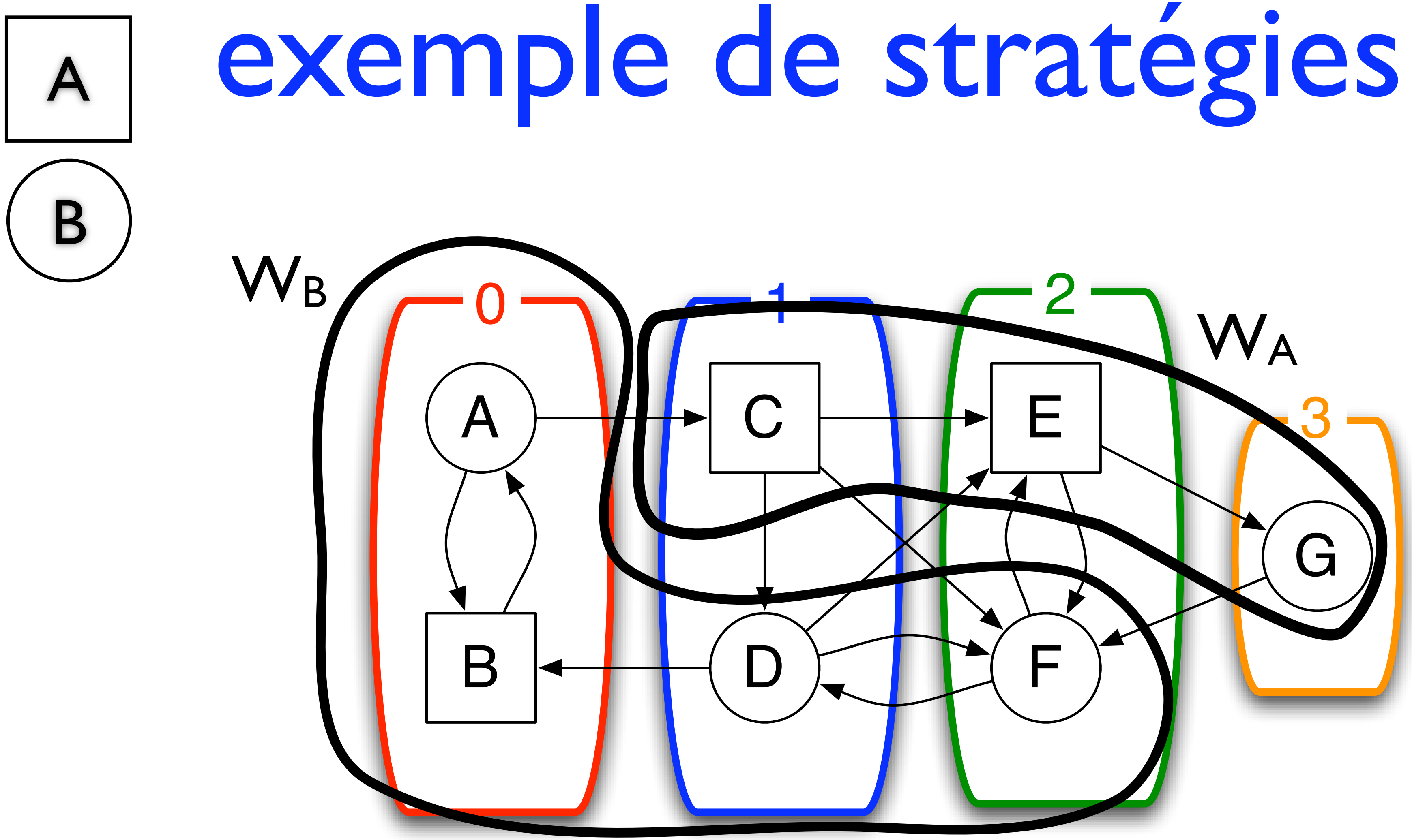


Jeux de parité - exemple de stratégies

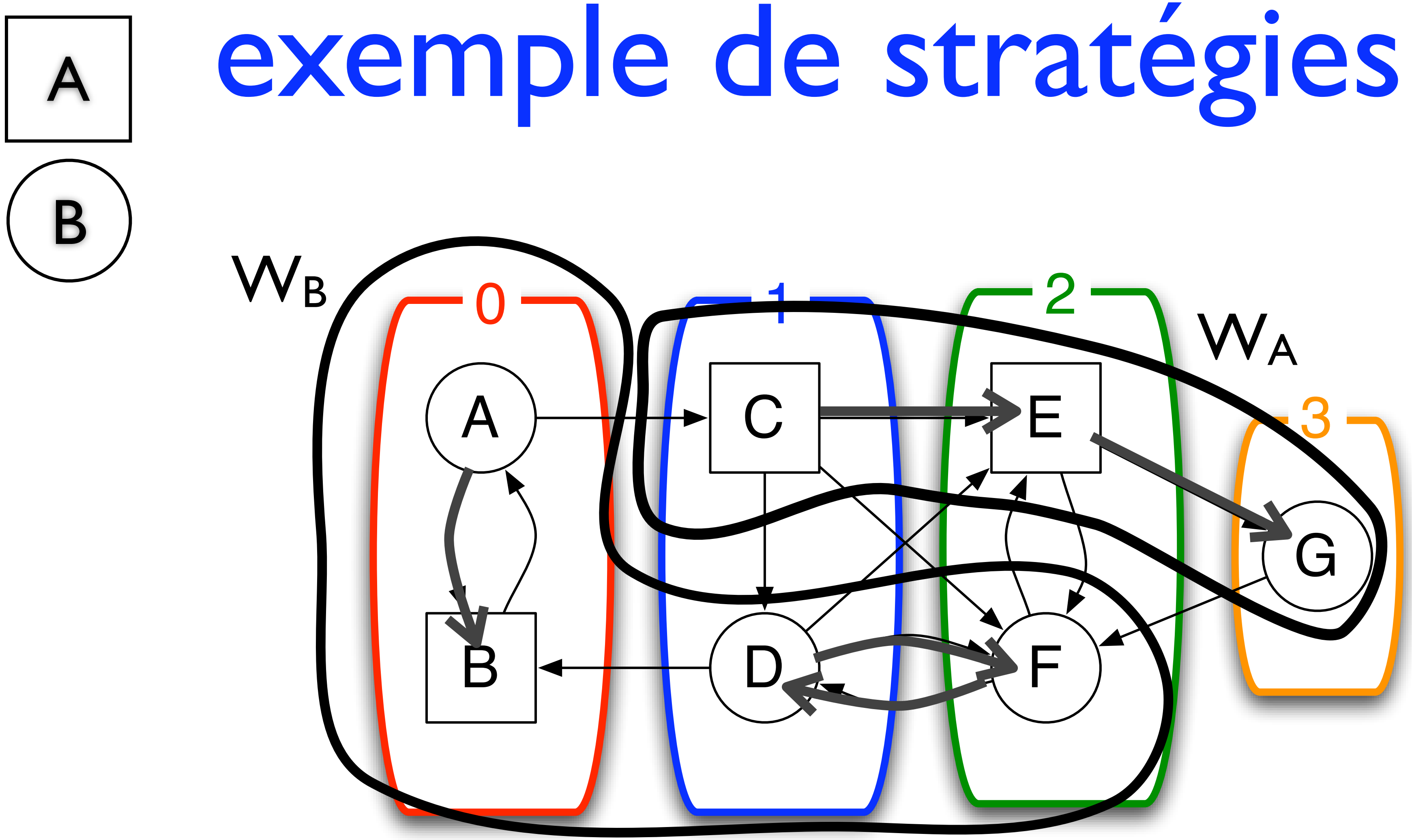




Jeux de parité - exemple de stratégies



Jeux de parité - exemple de stratégies



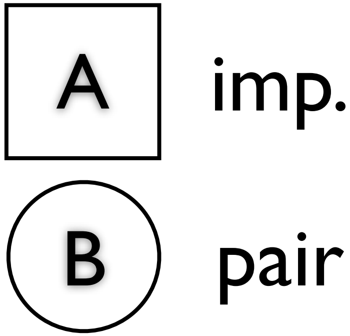
Jeux de parité - résultat principal

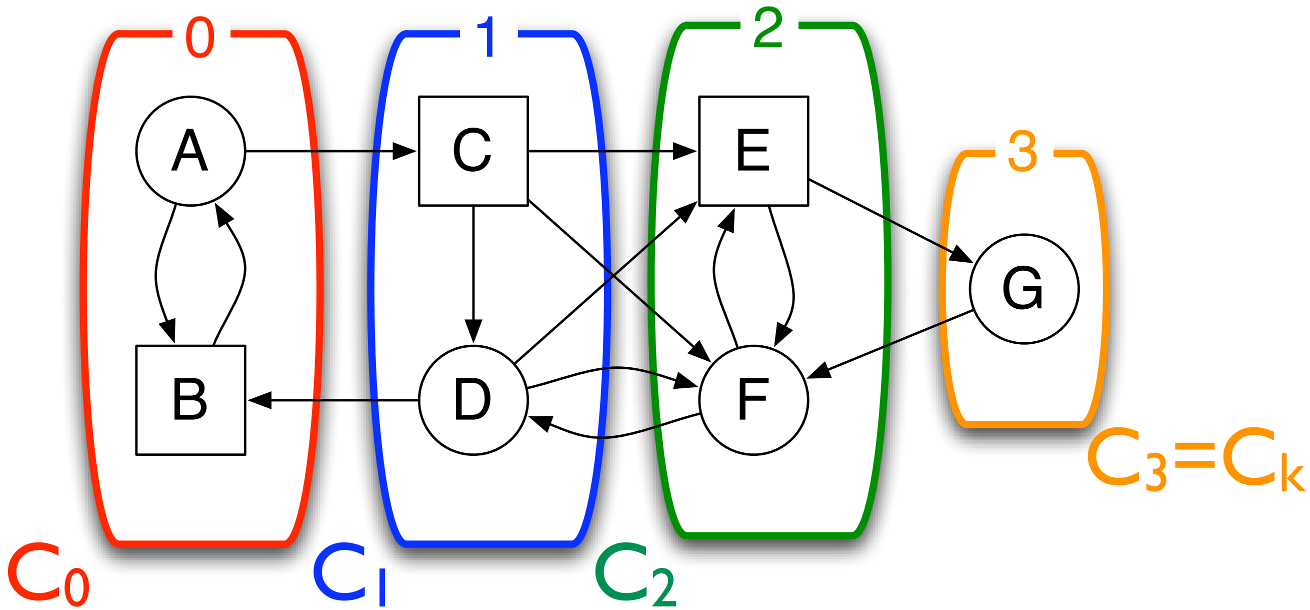
- Nous allons montrer que:
- Théorème: Dans un jeu de parité faible, on peut construire les régions gagnantes pour les deux joueurs, ainsi que des stratégies gagnantes correspondantes. Ces stratégies sont positionelles.

Jeux de parité - algorithme

- On considère un jeu $G = \langle Q, E \rangle$ et une fonction de coloration $c: Q \to \{1, \ldots, k\}$.
- On suppose que k est impair.
  - Ce n'est pas restrictif, car si k est pair, on incrémente toutes les couleurs de 1, et on inverse les positions des deux joueurs. On obtient un nouveau jeu dans lequel A gagne ssi B gagne dans le jeu original.
- On appelle C_i l'ensemble $\{q \in Q \text{ t.q. } c(q) = i\}$

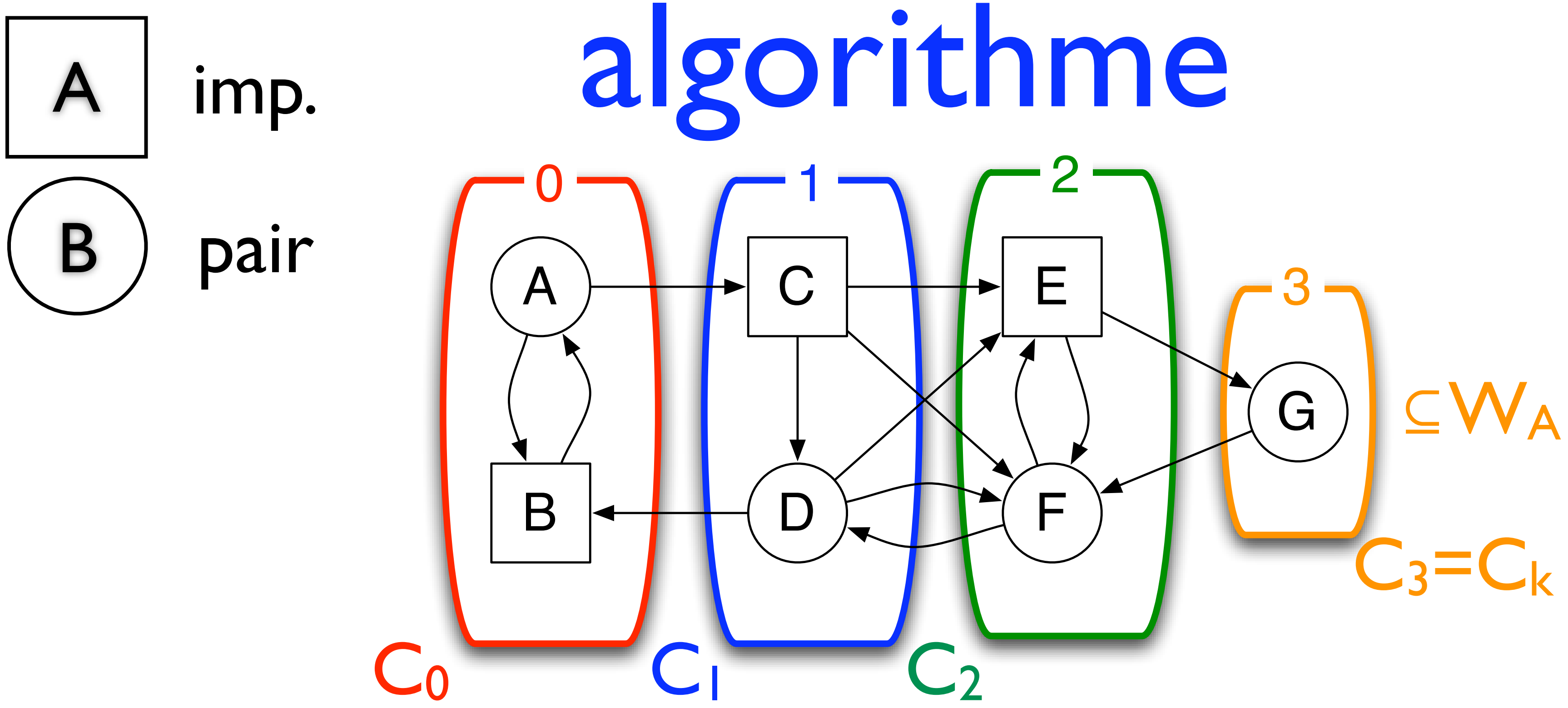
Jeux de parité - algorithme





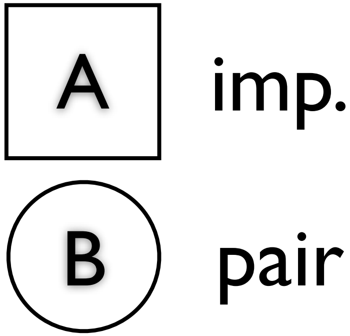
- Première constatation: A gagne si le jeu commence dans un noeud de C_k
    - k est la couleur maximale: B ne peut pas forcer le jeu à aller dans une couleur plus grande

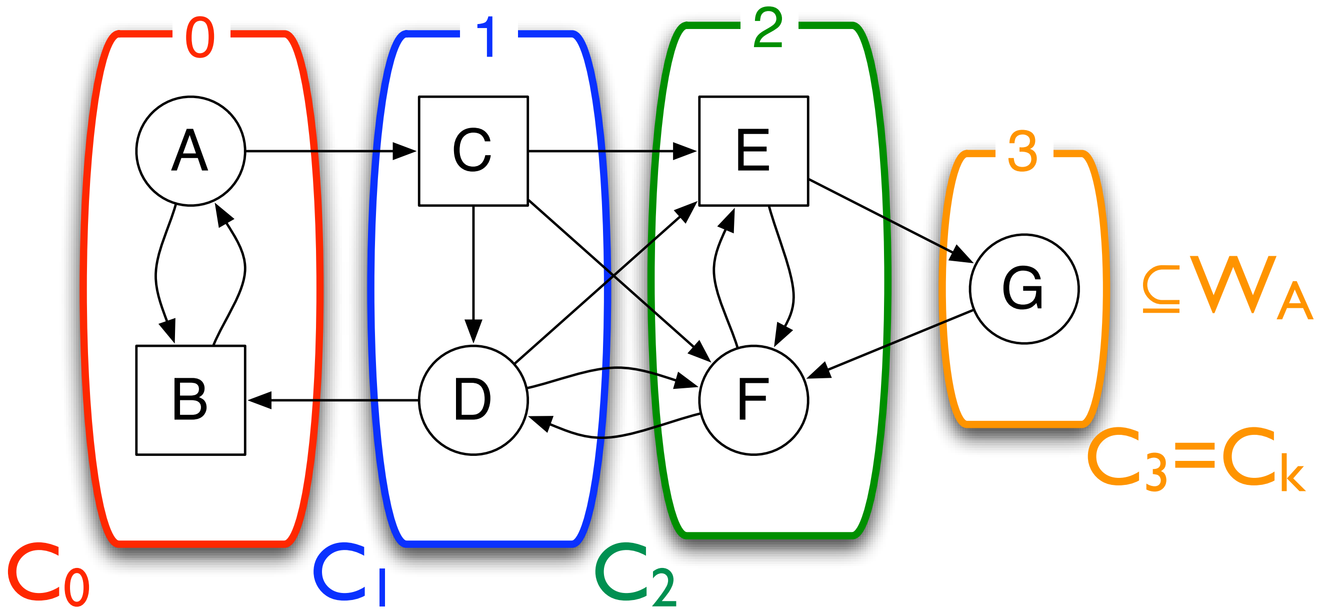
Jeux de parité - algorithme



- Première constatation: A gagne si le jeu commence dans un noeud de C_k
  - k est la couleur maximale: B ne peut pas forcer le jeu à aller dans une couleur plus grande

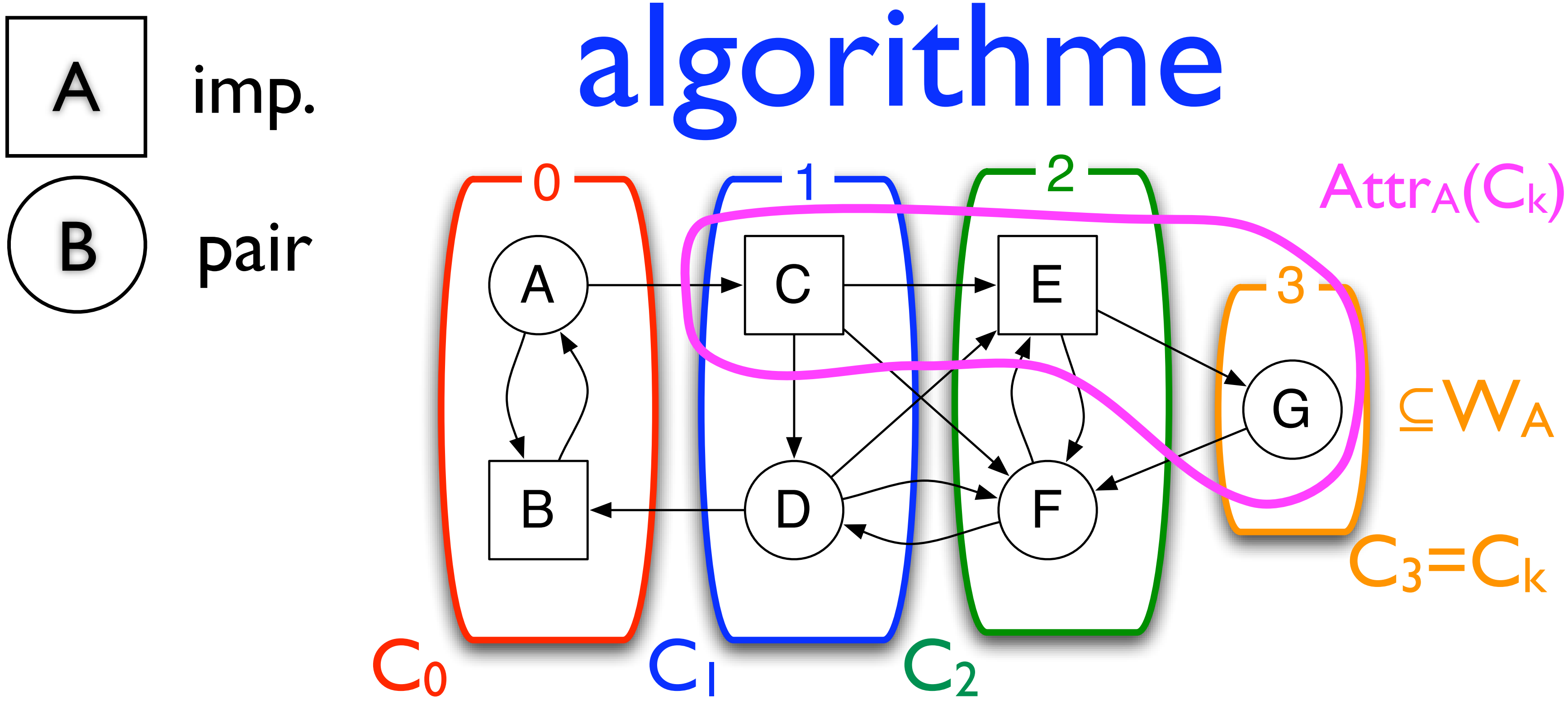
Jeux de parité - algorithme





- Deuxième constatation: A gagne si le jeu commence dans E par exemple, car A peut forcer le jeu à rejoindre C_k
  - De façon plus générale: A gagne si le jeu commence en $Attr_A(C_k)$.

Jeux de parité - algorithme



- Deuxième constatation: A gagne si le jeu commence dans E par exemple, car A peut forcer le jeu à rejoindre C_k
  - De façon plus générale: A gagne si le jeu commence en $Attr_A(C_k)$.

Jeux de parité - algorithme

- A gagne si le jeu commence en $Attr_A(C_k)$.
- Mais ce n'est pas suffisant !
  - A pourrait aussi gagner en forçant le jeu à aller dans une couleur impaire différente de k, et en empêchant B d'amener le jeu dans une couleur plus grande.
  - Par exemple: A force le jeu à aller en k-2, et empêche B d'amener le jeu dans k-1.

Jeux de parité - algorithme

- De manière générale, A gagne:
  - s'il peut forcer le jeu à aller dans une couleur impaire,...
  - tout en évitant les positions dans lesquelles B peut forcer le jeu à aller dans une couleur paire plus grande.
- Et symétriquement pour B.

Jeux de parité - algorithme

- Idée de l'algorithme:
  - On va calculer, pour chaque couleur i, un ensemble A_i
  - Cet ensemble contiendra toutes les positions dans lesquelles le joueur auquel appartient la couleur i gagne en forçant le jeu à passer par i ou par une couleur plus grande qui lui appartient
- Exemple: A_2 = l'ensemble des positions dans lesquelles B gagne (2 est pair) parce qu'il force le jeu à passer par 2 ou par 4 ou par 6, etc.

Jeux de parité - algorithme

- Idée de l'algorithme:
  - On va calculer, pour chaque couleur i, un ensemble A_i
- Comme le nombre de couleurs est fini, il n'y a qu'un nombre fini d'ensembles à calculer
- Il faut les calculer dans le bon ordre ! A_i dépend de A_{i+1} (positions permettant à l'adversaire d'atteindre une couleur plus grande) et de A_{i+2} (pour gagner grâce à une couleur plus grande que i).

Jeux de parité - algorithme

- On définit donc la séquence suivante:
  - $A_k = Attr_A(C_k)$: A gagne s'il peut forcer le jeu à aller en k (impair)
  - $A_{k-1} = Attr_B(C_{k-1} \setminus A_k)$: B gagne s'il peut forcer le jeu à aller en k-1 (pair) tout en évitant les positions où A peut forcer le jeu à aller en k

Jeux de parité - algorithme

- $A_{k-2} = Attr_A(C_{k-2} \setminus A_{k-1} \cup A_k)$: A gagne soit:
  - s'il peut forcer le jeu à aller en k-2 (impair) tout en évitant les positions où B peut forcer le jeu à aller en k-1
  - s'il peut forcer le jeu à aller dans une position d'où il est sûr de pouvoir forcer le jeu à aller en k (impair).

Jeux de parité - algorithme

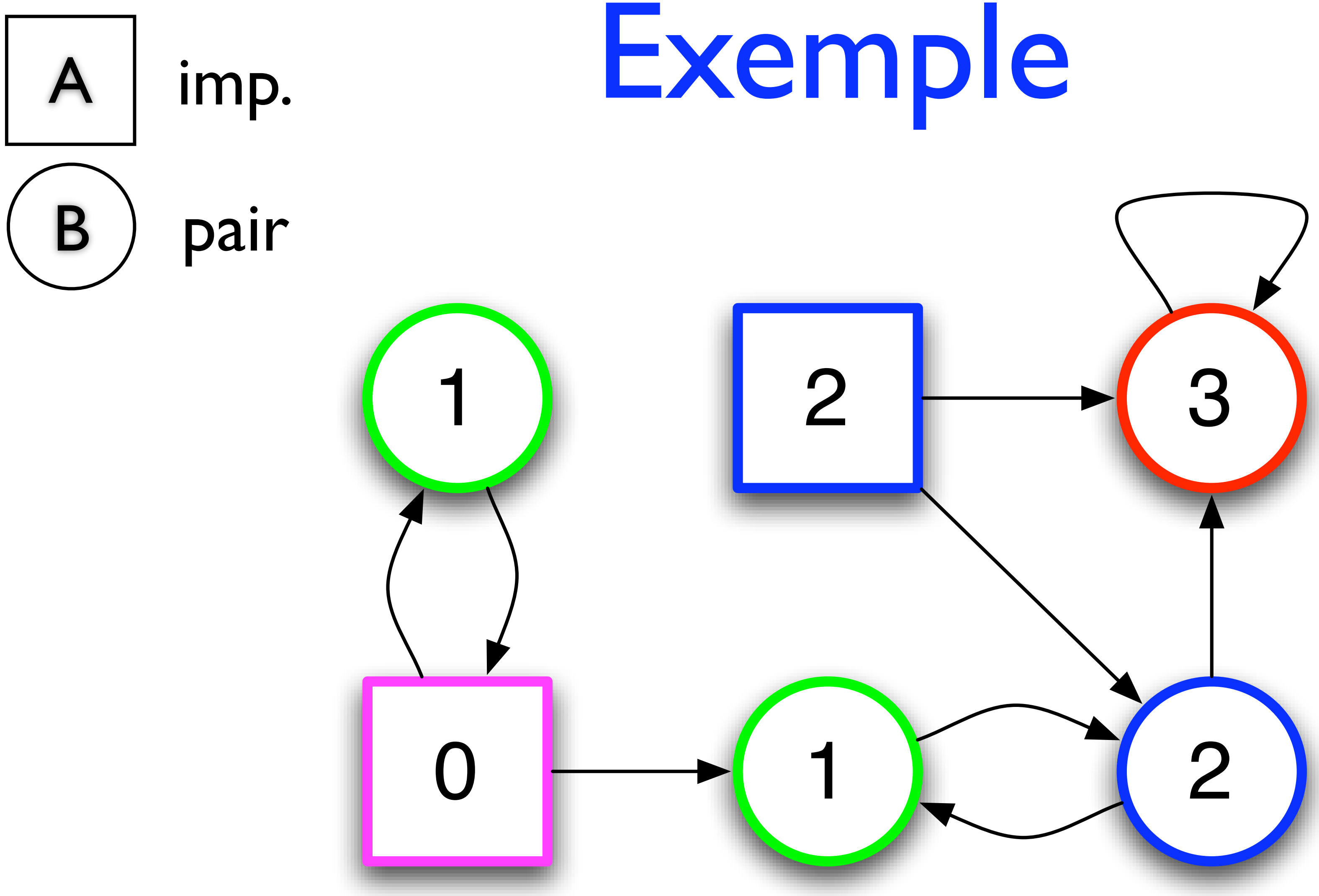
- $A_{k-3} = Attr_B(C_{k-3} \setminus A_{k-2} \cup A_{k-1})$: B gagne soit:
  - s'il peut forcer le jeu à aller en k-3 (pair) tout en évitant les positions où A peut forcer le jeu à aller en k-2 ou en k
  - s'il peut forcer le jeu à aller en k-1 (pair) tout en évitant les positions où A peut forcer à aller en k.

Jeux de parité - algorithme

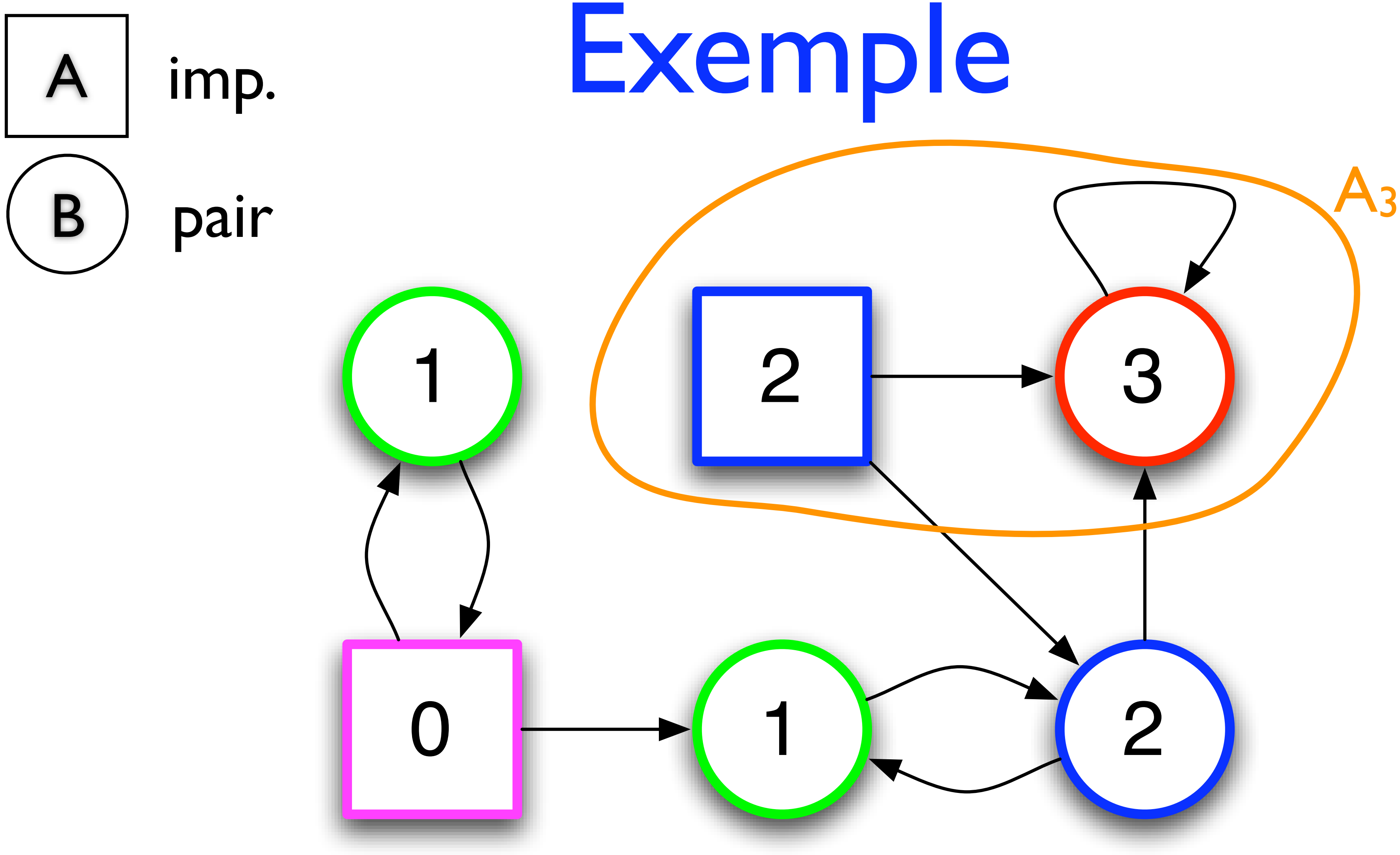
$$
\begin{array}{rcl}
A_k & = & \mathrm{Attr}_A(C_k) \\
A_{k-1} & = & \mathrm{Attr}_B(C_{k-1} \setminus A_k) \\
\forall 1 \leq i < k-1 : & & \\
A_i & = & \begin{cases} \mathrm{Attr}_A(C_i \setminus A_{i+1} \cup A_{i+2}) & \text{si } i \text{ est impair} \\ \mathrm{Attr}_B(C_i \setminus A_{i+1} \cup A_{i+2}) & \text{si } i \text{ est pair} \end{cases}
\end{array}
$$

- On a : $A_1 = W_A$ et $A_2 = W_B$
- Pour obtenir les stratégies, on procède comme dans les jeux d'accessibilité (stratégies positionelles).

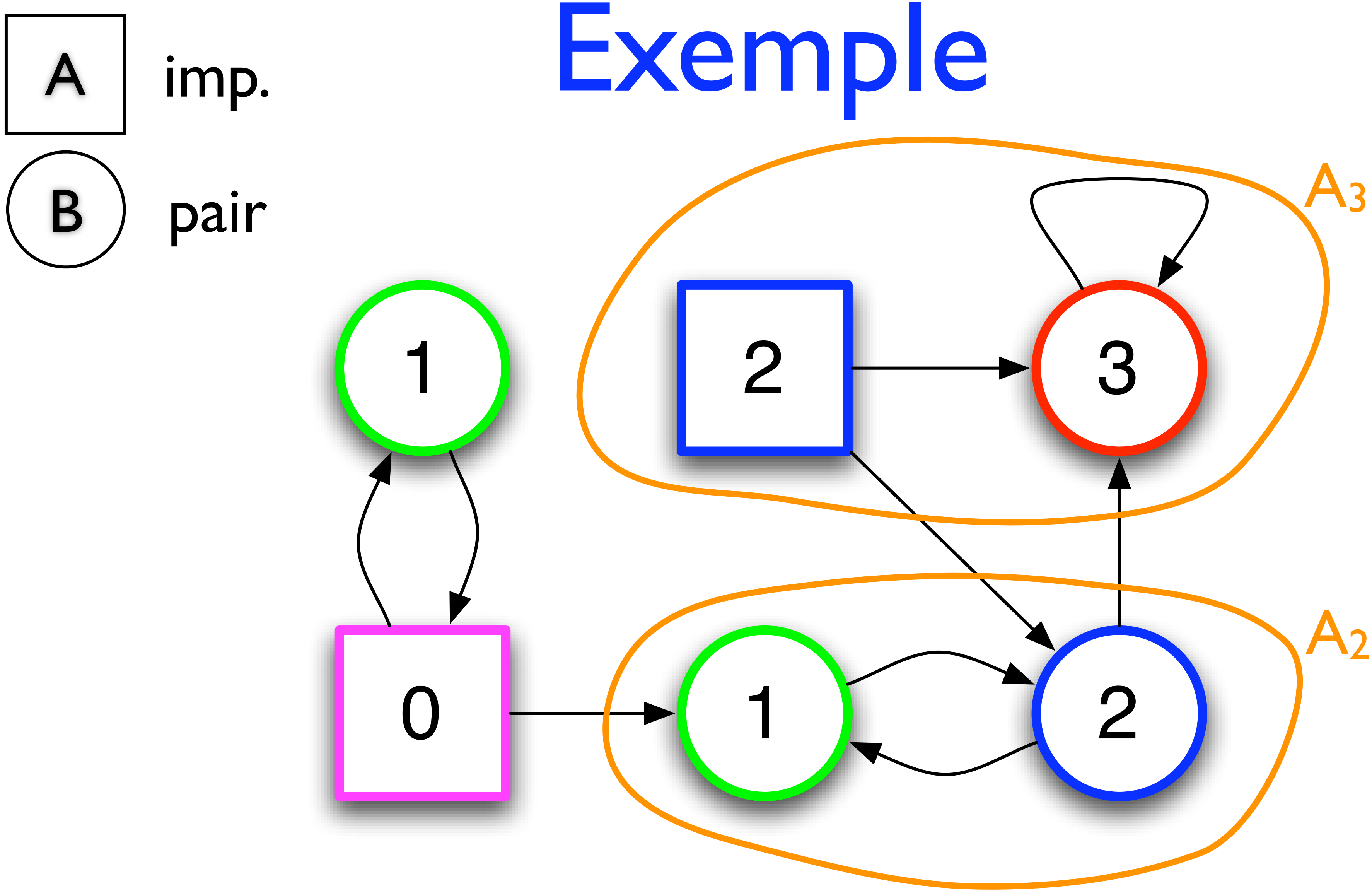
Jeux de parité - Exemple



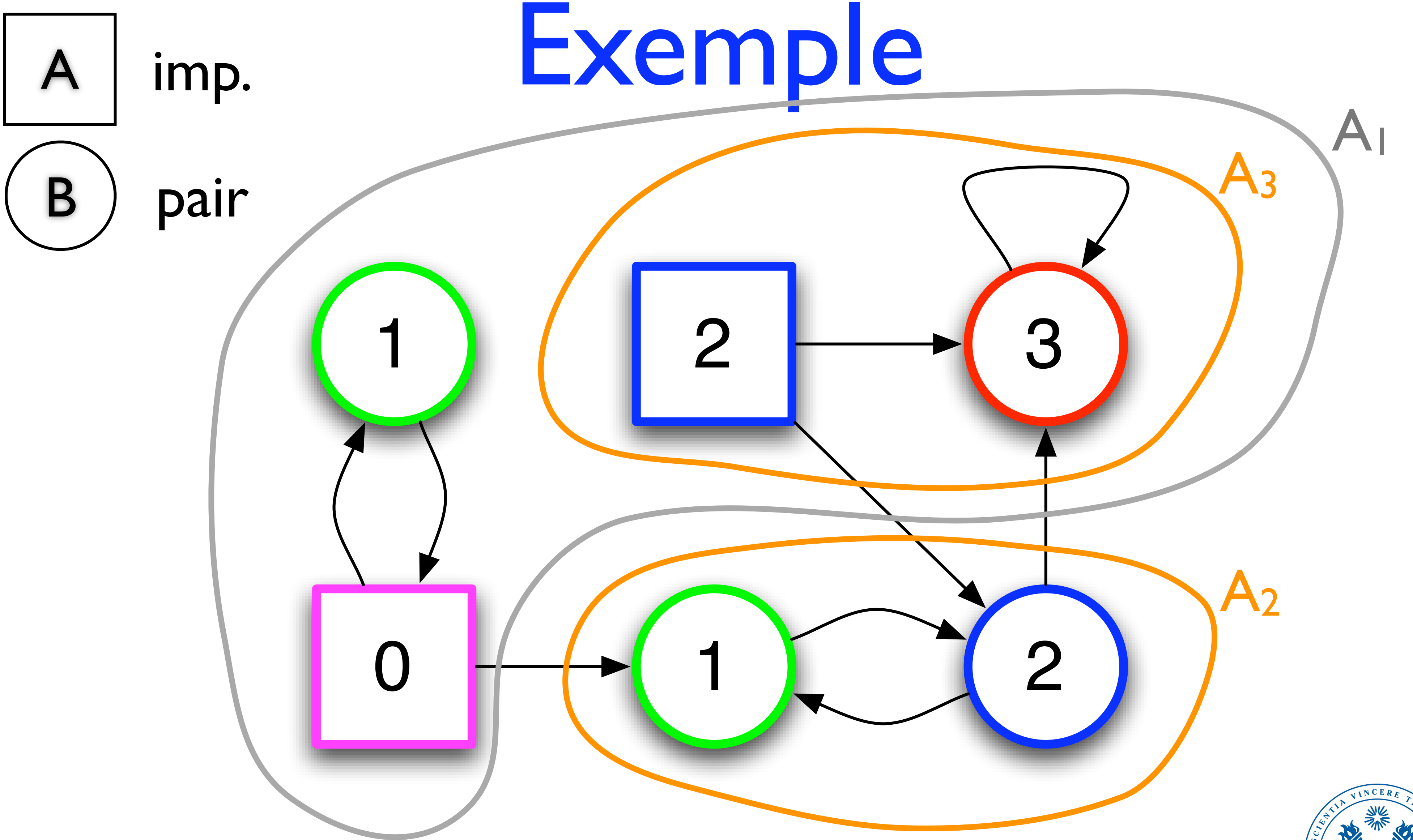
Jeux de parité - Exemple



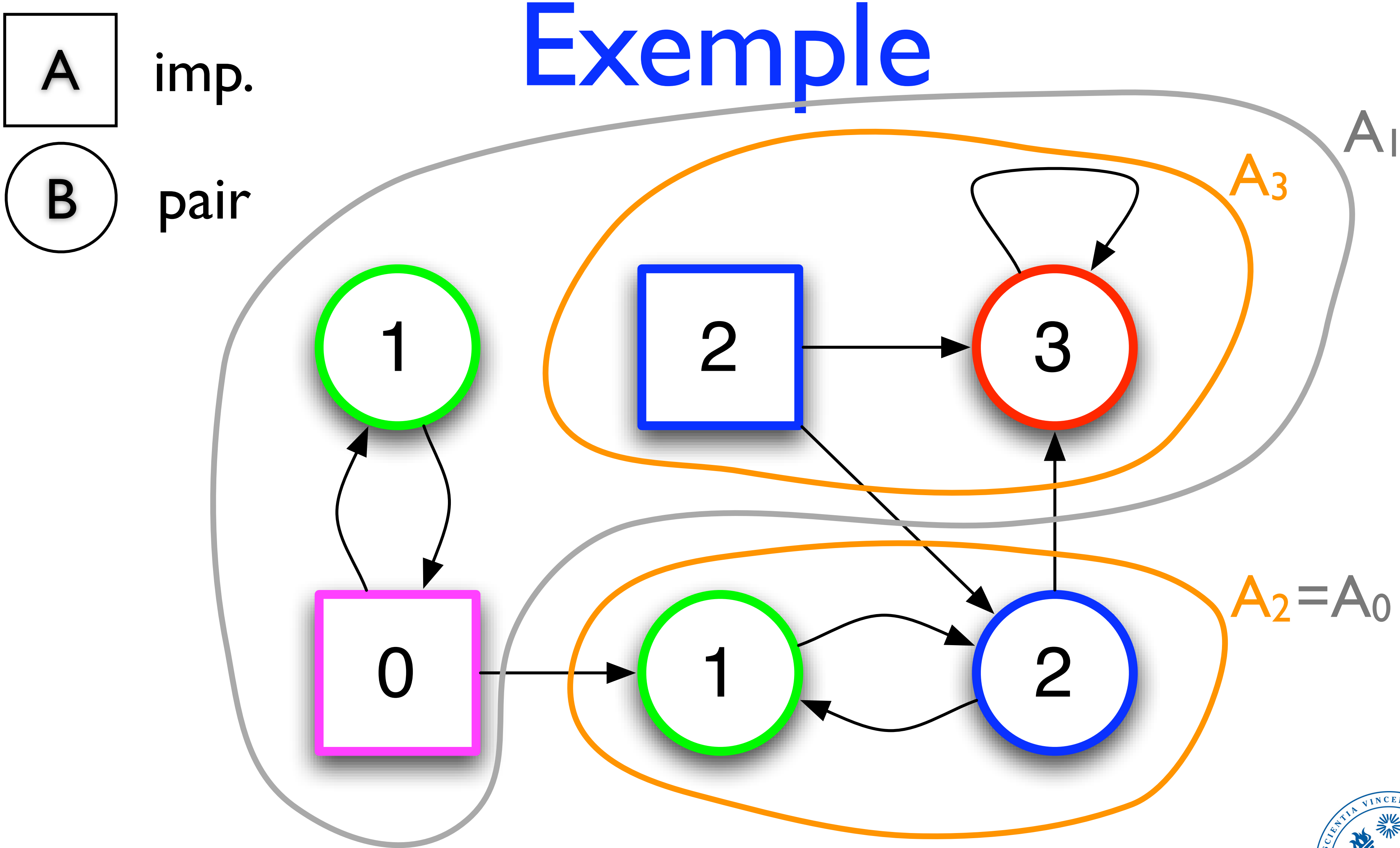
Jeux de parité - Exemple



Jeux de parité - Exemple



Jeux de parité - Exemple



Jeux de Muller faibles

- Nous allons maintenant réduire le problème de résolution d'un jeu de Muller faible à la résolution d'un jeu de parité, et utiliser l'algorithme que nous venons de présenter.

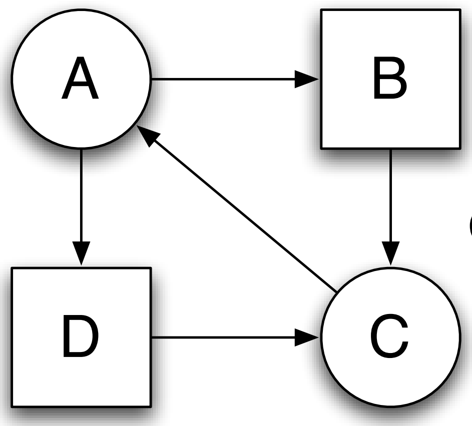
Réduction Muller-parité

- Dans un jeu de Muller faible, on fixe les ensembles de positions par lesquels on passe pour gagner le jeu.
  - e.g. F={{A,B,C}, {D,B,E}}, et le joueur gagne la partie si le jeu passe uniquement par A,B et C ou uniquement par D,B et E
- Intuitivement, cela signifie que pour "bien jouer", il faut retenir par quels états on est déjà passé
  - Par exemple, si on est en B, qu'on est déjà passé par A, et qu'on peut choisir entre C et E comme successeurs, il faut choisir C.

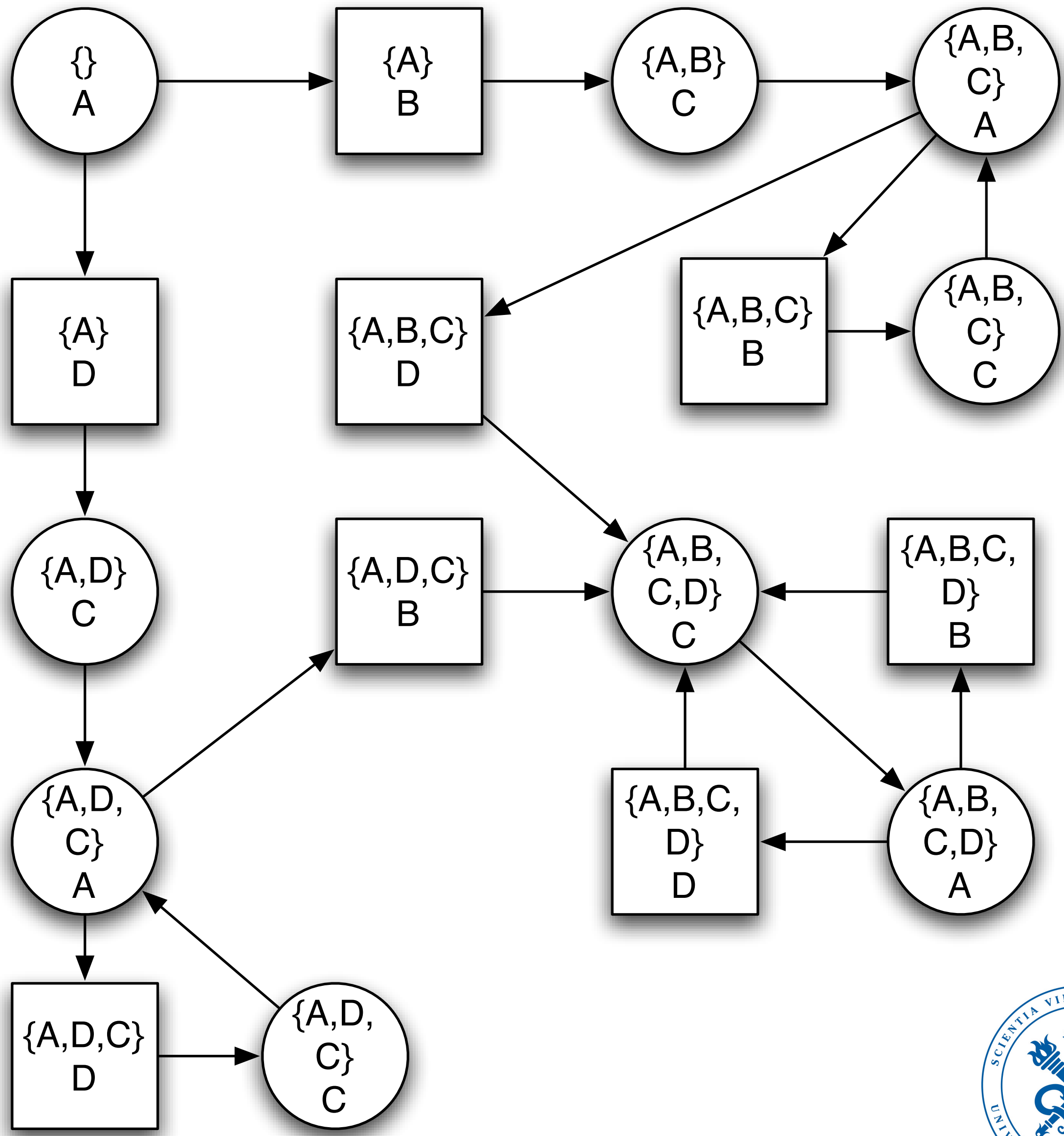
Réduction Muller-parité

- On va d'abord transformer le jeu de Muller faible $G = \langle Q,E \rangle$ en un autre jeu $G' = \langle Q',E' \rangle$ dans lequel on retient les états par lesquels on est déjà passé.
- Les états de G' seront de la forme (S,q), où
  - $q \in Q$ est un état du jeu de départ
  - $S \subseteq Q$ retient les états par lesquels on est déjà passé
  - (S,q) appartient au joueur auquel appartient q
- On construit E' en conséquence: $((S,q), (S',q')) \in E'$ ssi $(q,q') \in E$ et $S' = S \cup \{q\}$.

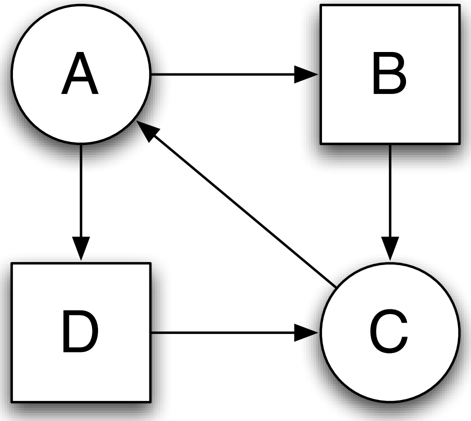
Réduction Muller-parité

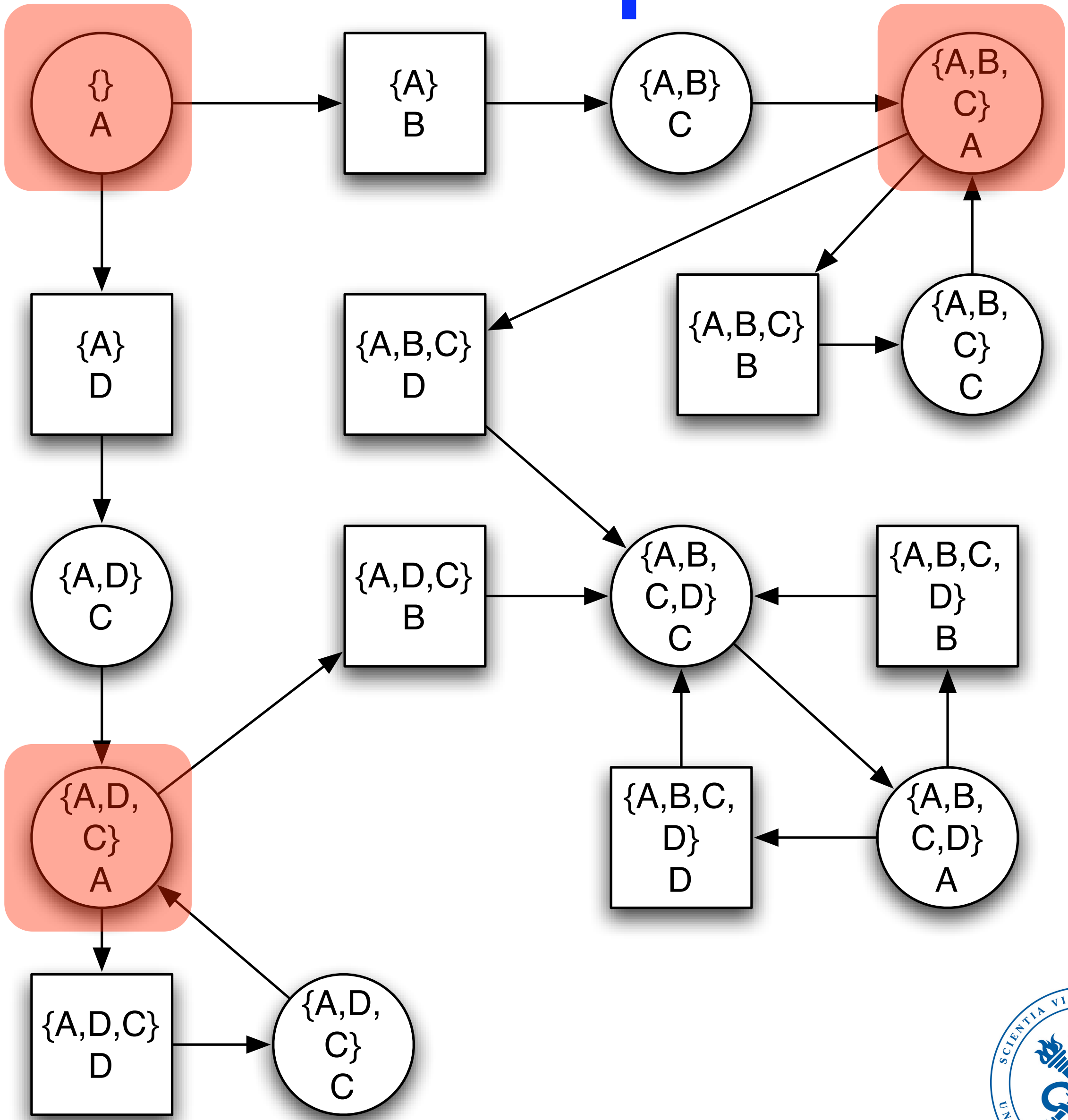


devient:



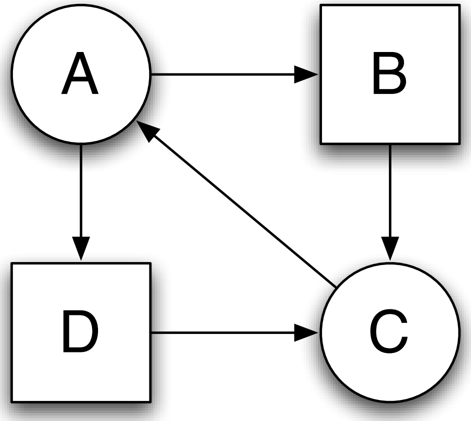
Réduction Muller-parité



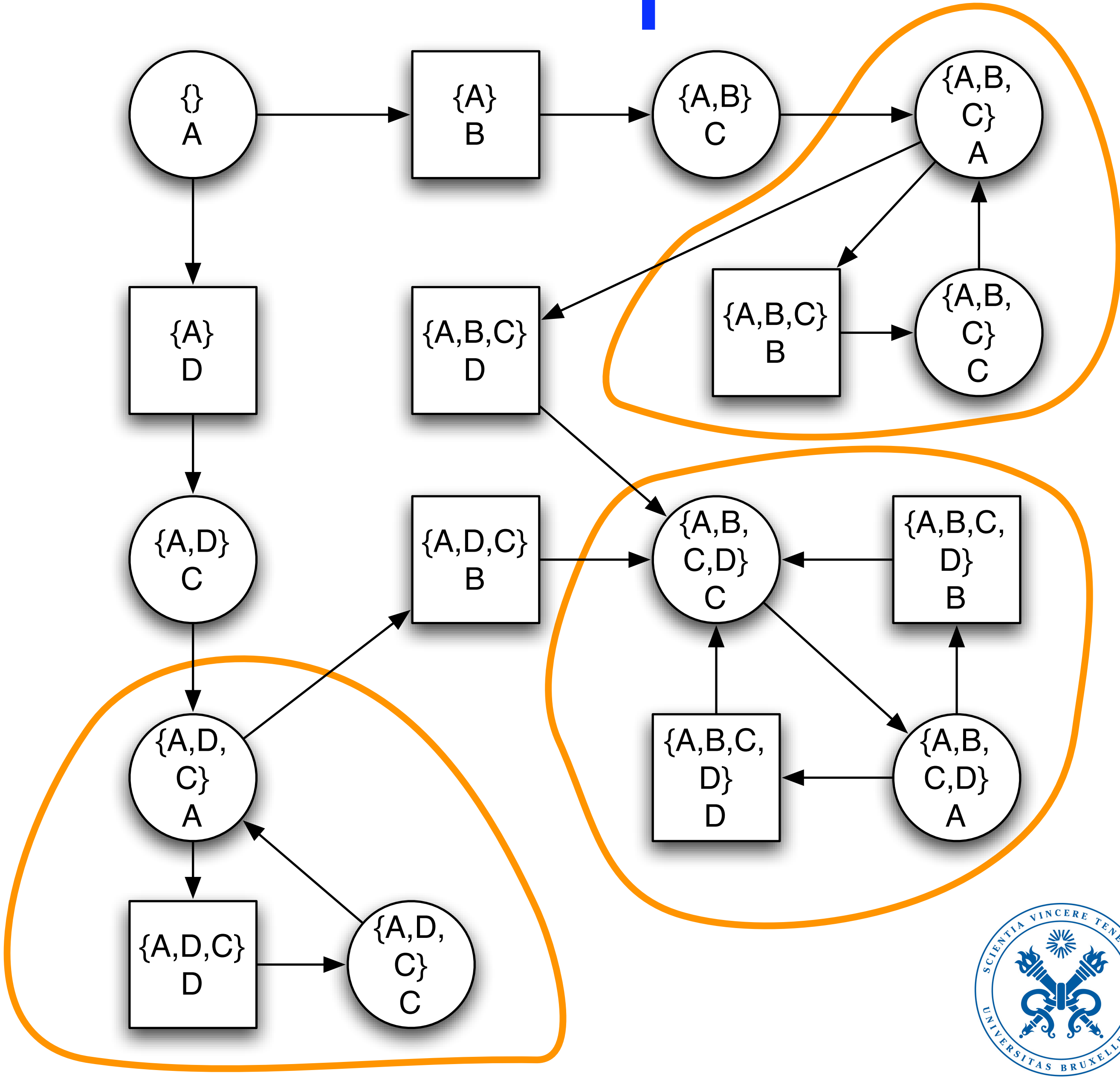


Si on joue une partie p àpd A, le joueur “rond” peut fixer Occ(p): soit ABC, soit ADC, soit ABCD

Réduction Muller-parité



Le jeu finira "**bloqué**" dans une de ces trois composantes. La "**mémoire**" de ces états correspond à Occ(p)



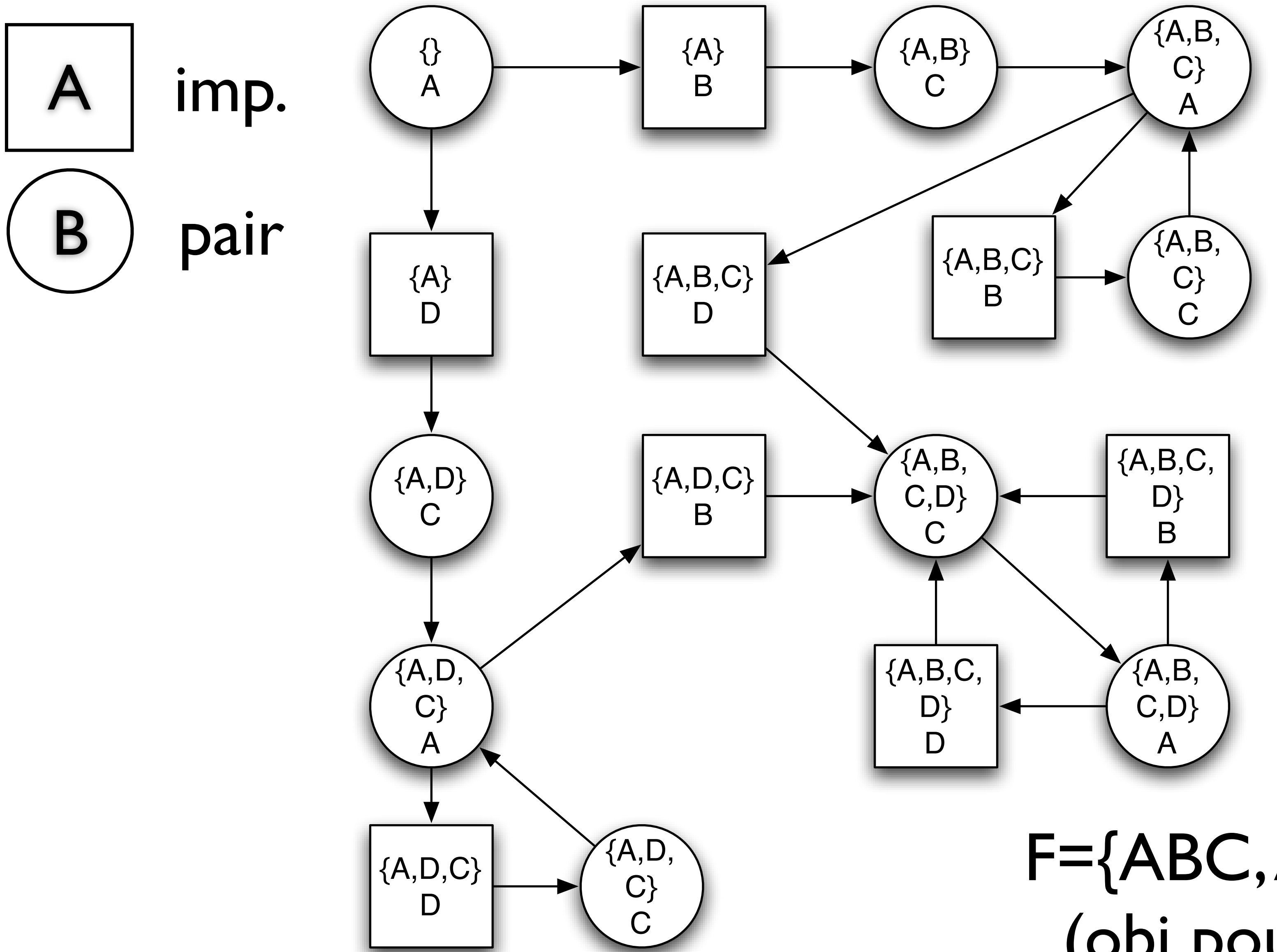
Réduction Muller-parité

- On constate que la mémoire augmente tout au long de la partie et se stabilise sur Occ(p).
- Pour obtenir un jeu de parité, on va associer une couleur à chaque sous-ensemble de Q (c'est-à-dire chaque contenu de la mémoire dans G').
    - Plus la mémoire contient d'élément, plus la couleur est grande (pour que le max des couleurs correspondent à Occ(p)).
    - Si F est un objectif pour B, alors les contenus de mémoire gagnants (ceux dans F) seront associés à une couleur paire. Symétriquement pour A.

Réduction Muller-parité

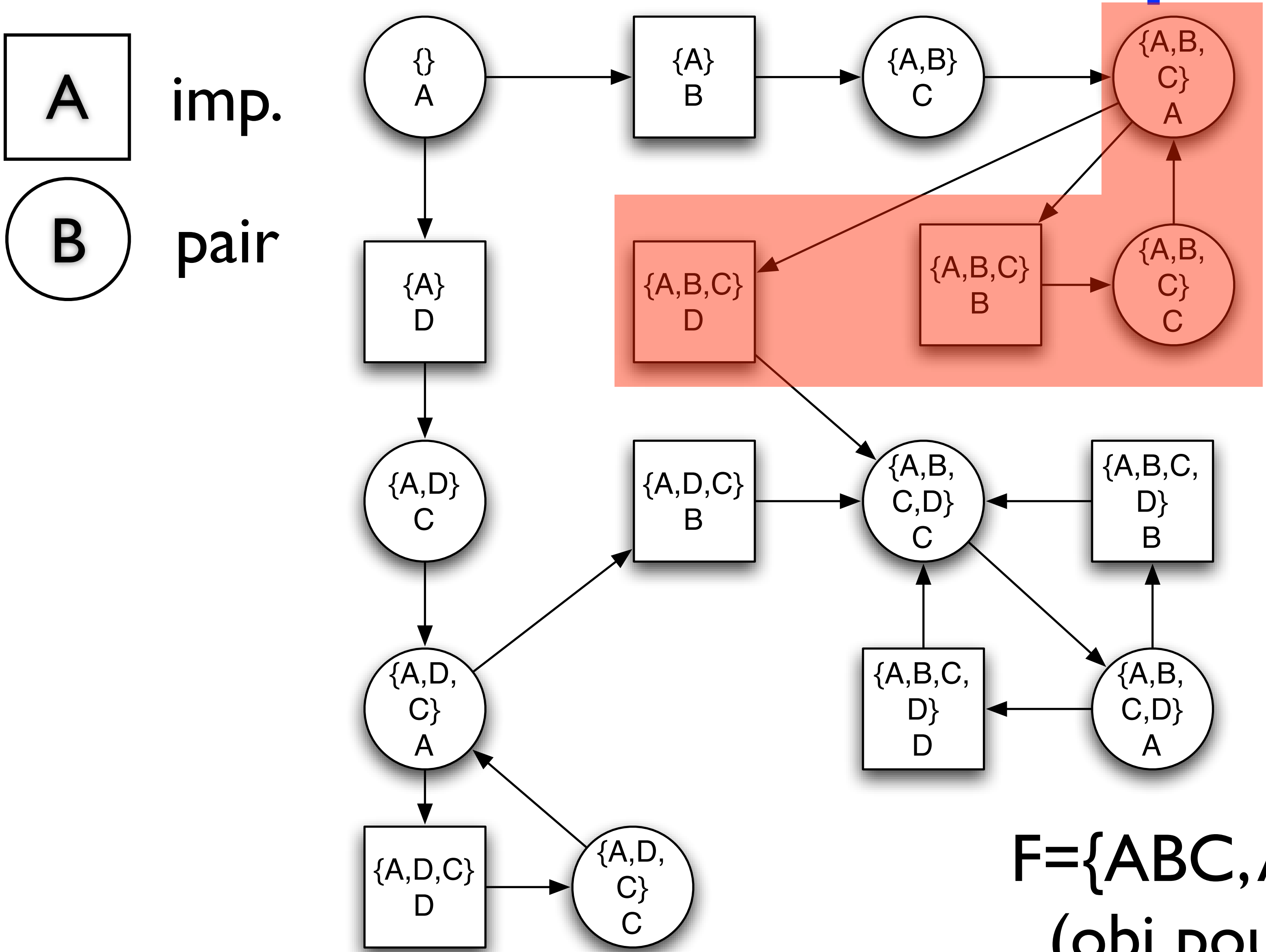
- En pratique, on peur procéder ainsi, pour un objectif F pour le joueur B (qui gagne avec les couleurs paires)
  - $c((S,q)) = 2\times|S|$ si $S \in F$
  - $c((S,q)) = 2\times|S|+1$ si $S \notin F$

Réduction Muller-parité



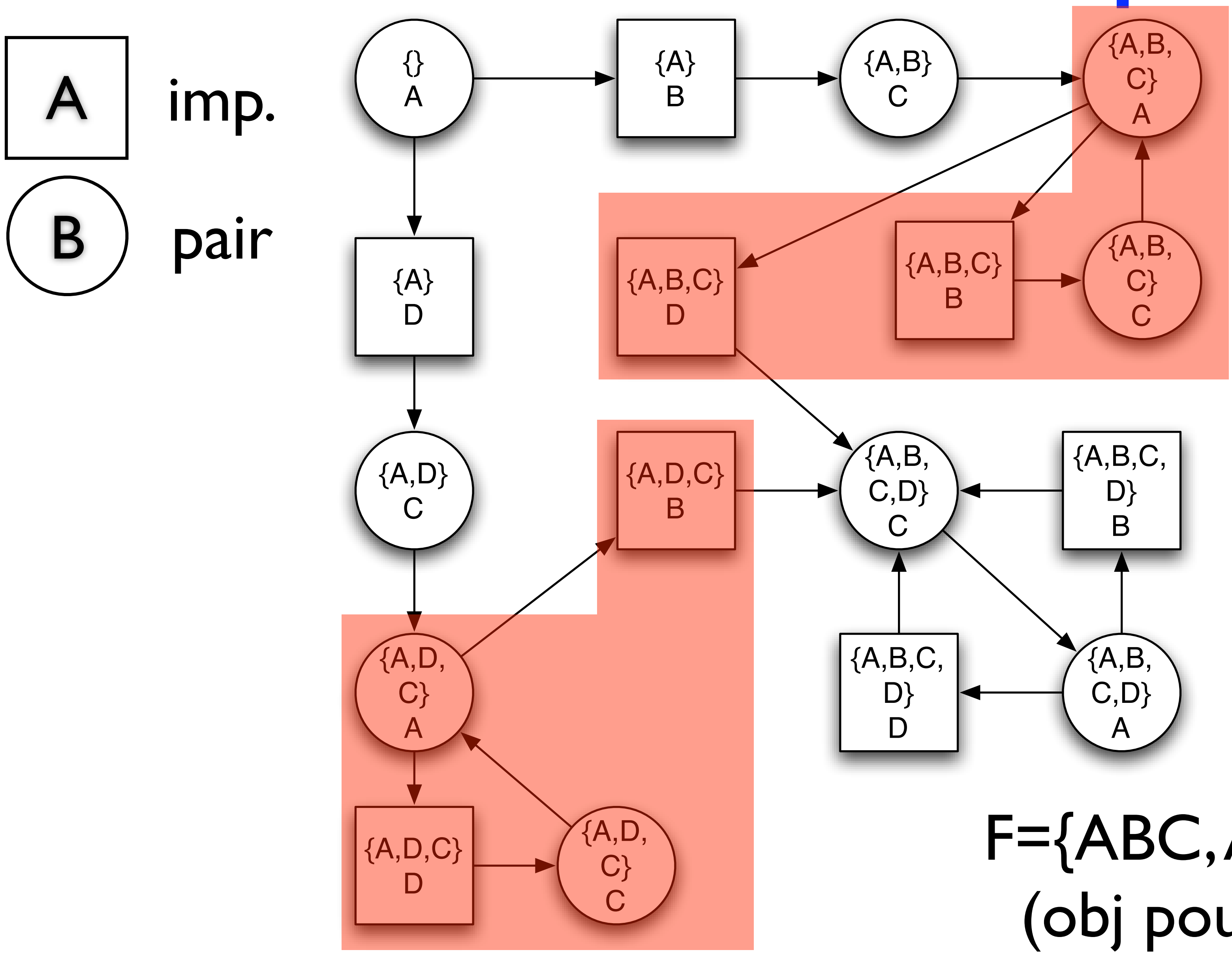
F={ABC,ADC}
(obj pour B)

Réduction Muller-parité



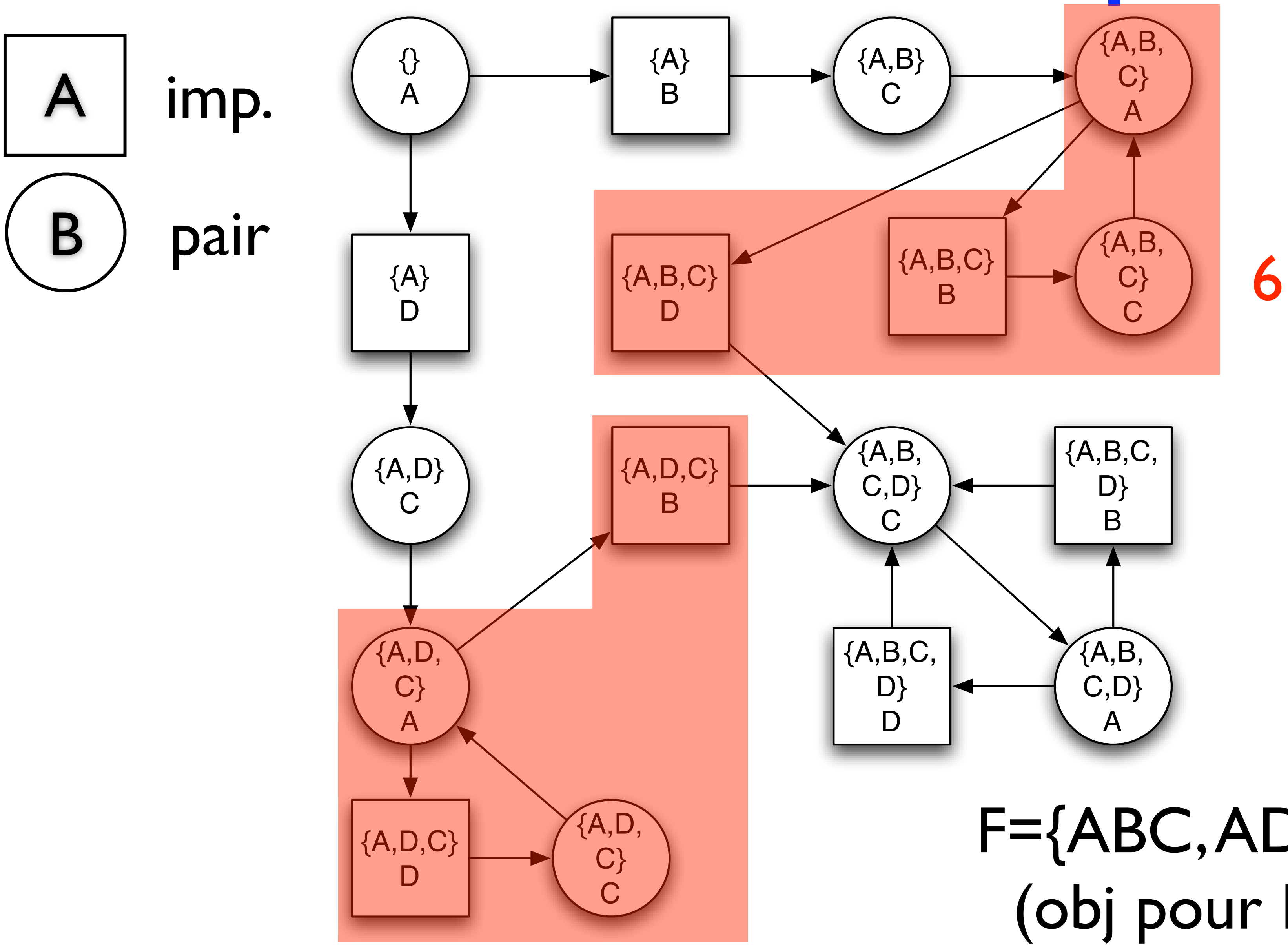
F={ABC,ADC}
(obj pour B)

Réduction Muller-parité



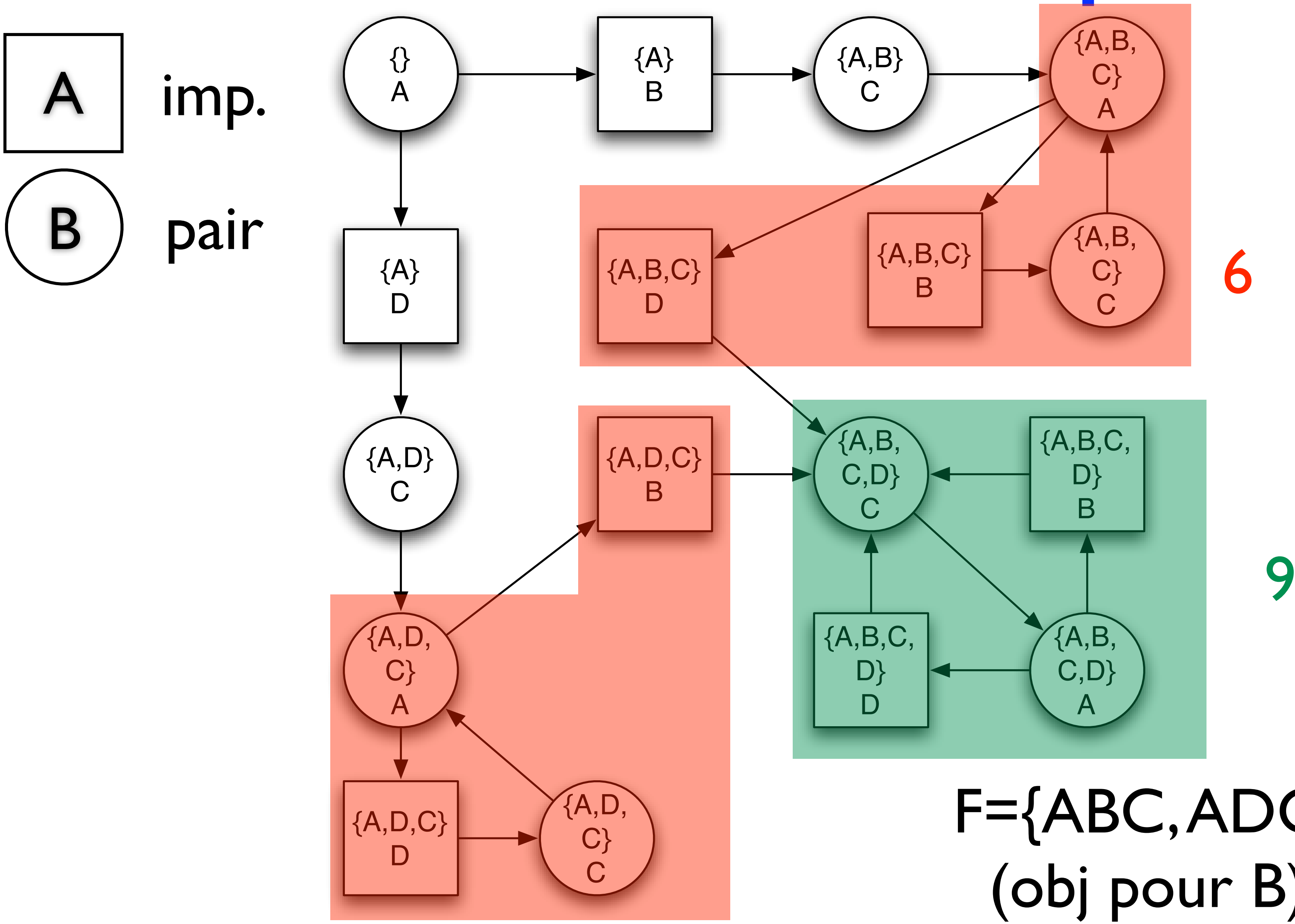
F={ABC,ADC}
(obj pour B)

Réduction Muller-parité



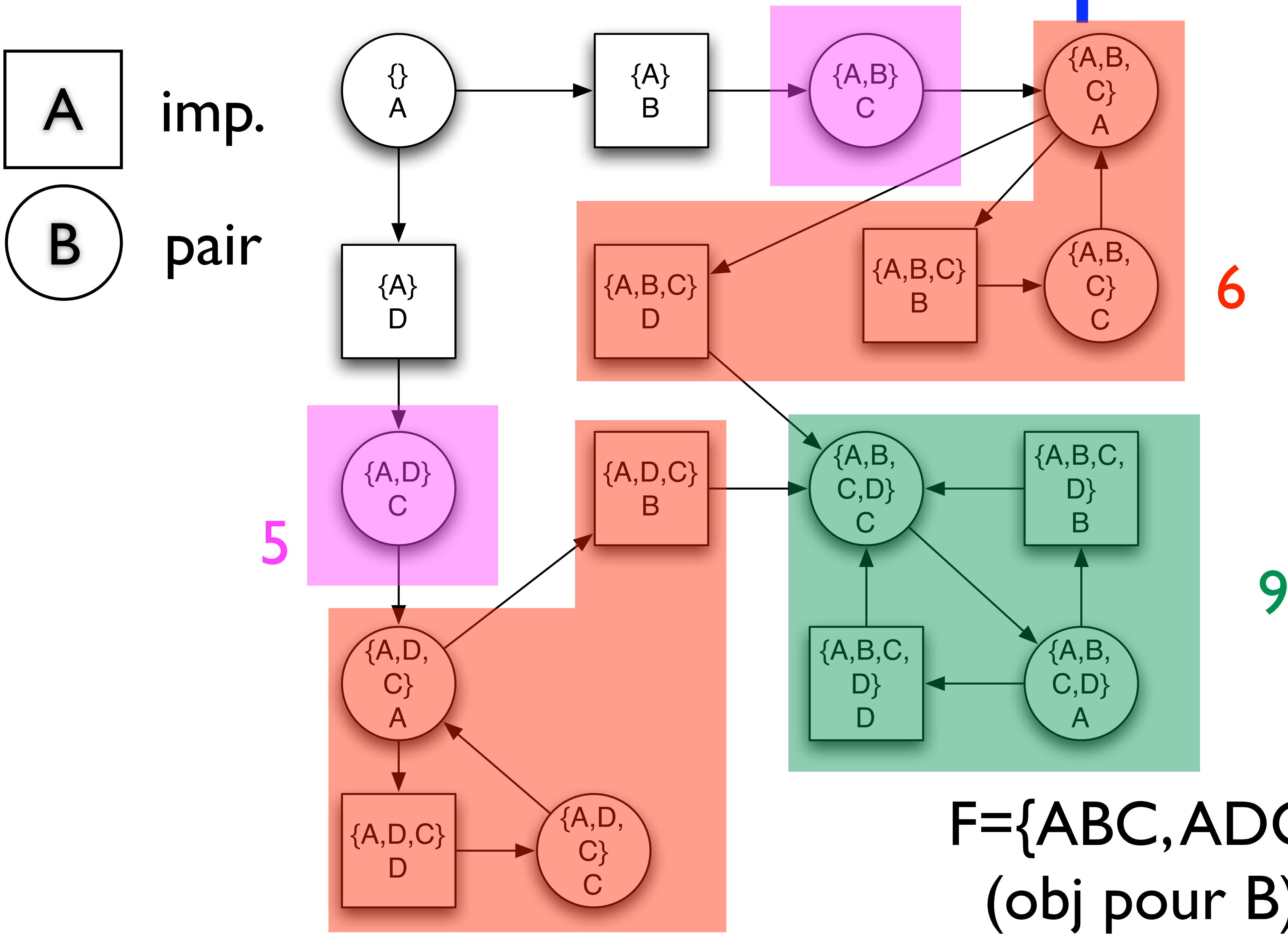
F={ABC,ADC}
(obj pour B)

Réduction Muller-parité



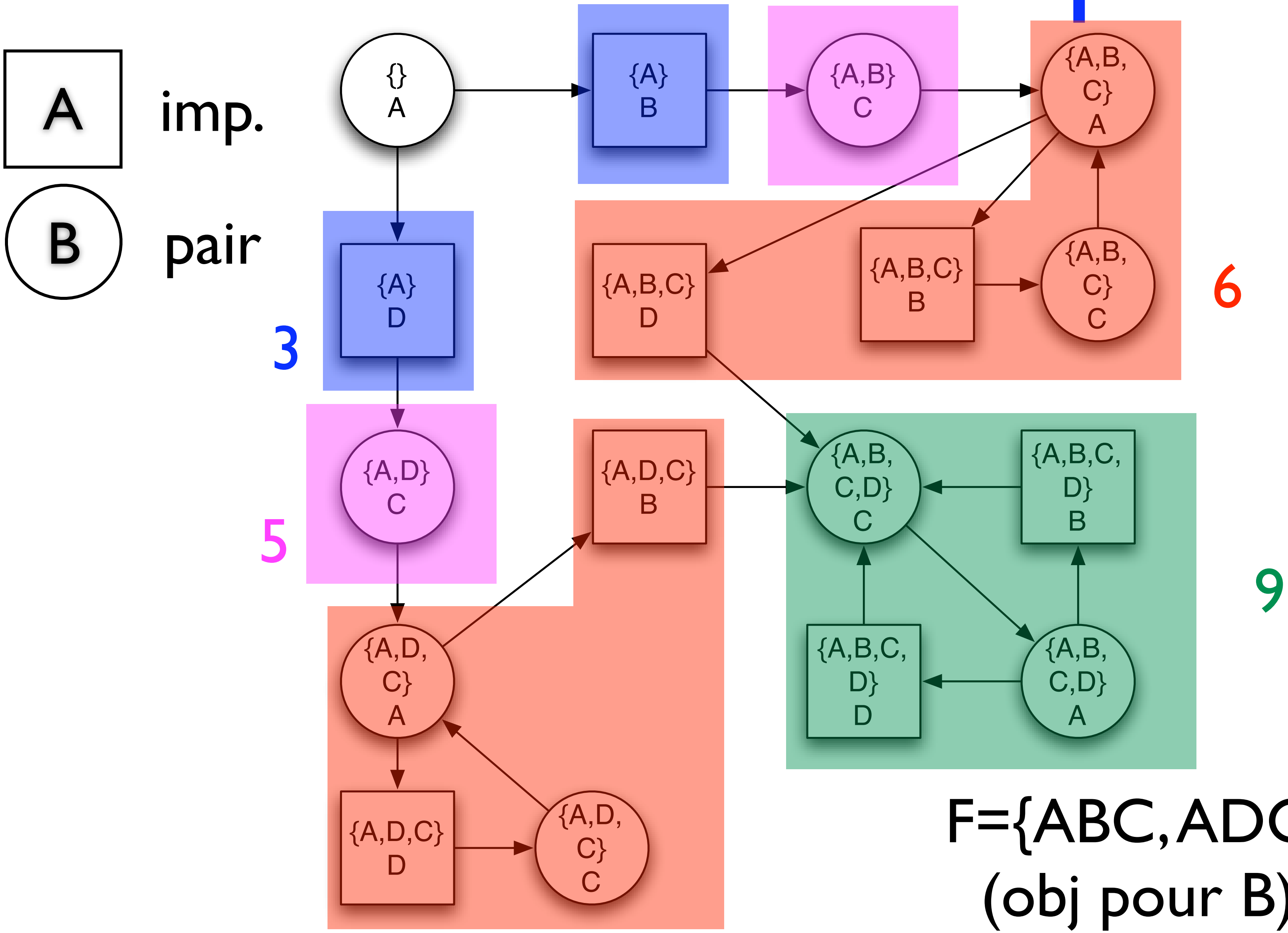
F={ABC,ADC}
(obj pour B)

Réduction Muller-parité



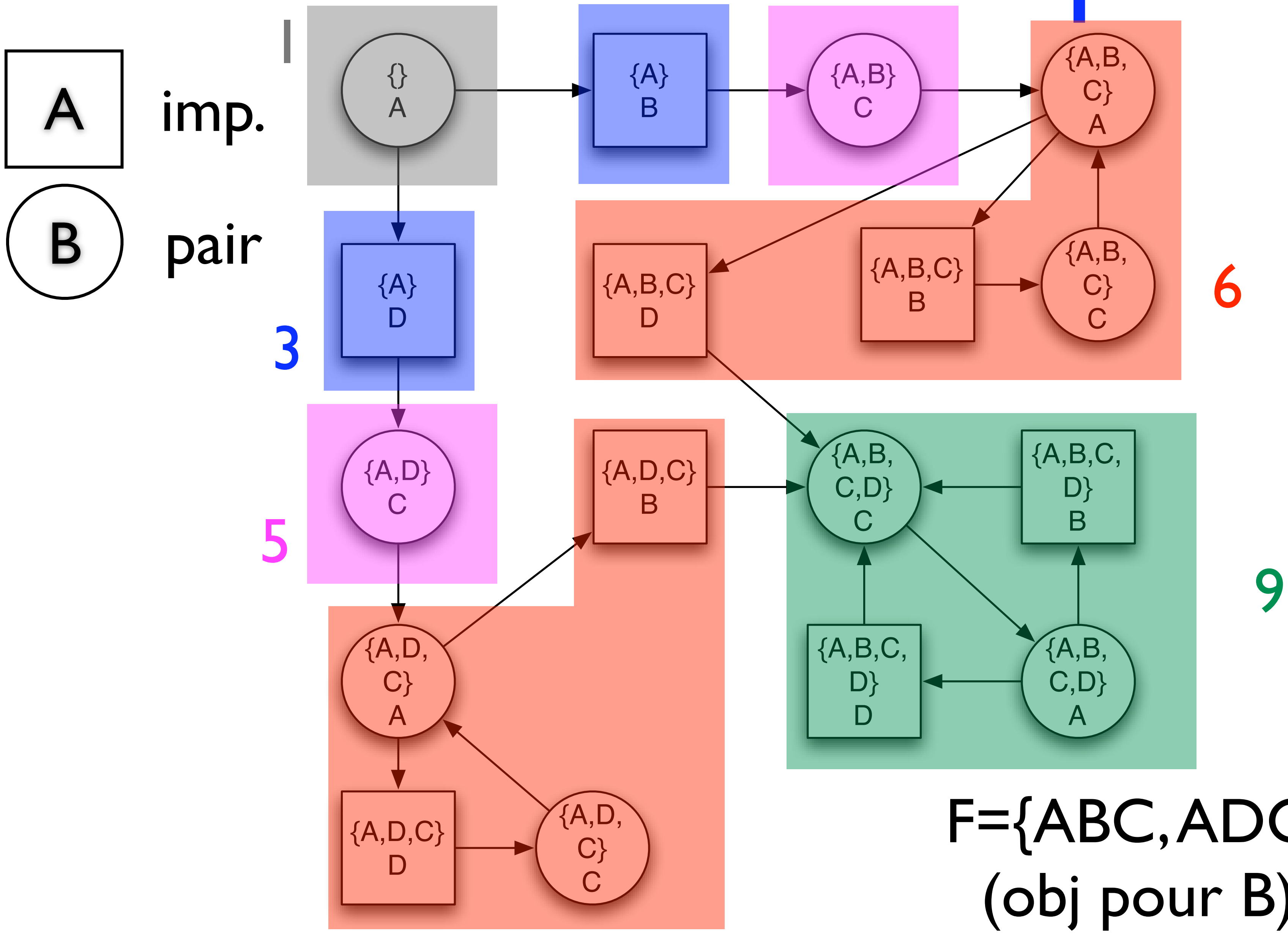
F={ABC,ADC}
(obj pour B)

Réduction Muller-parité



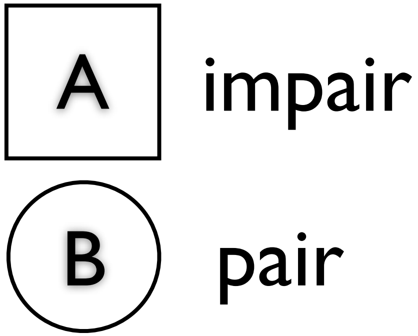
F={ABC, ADC}
(obj pour B)

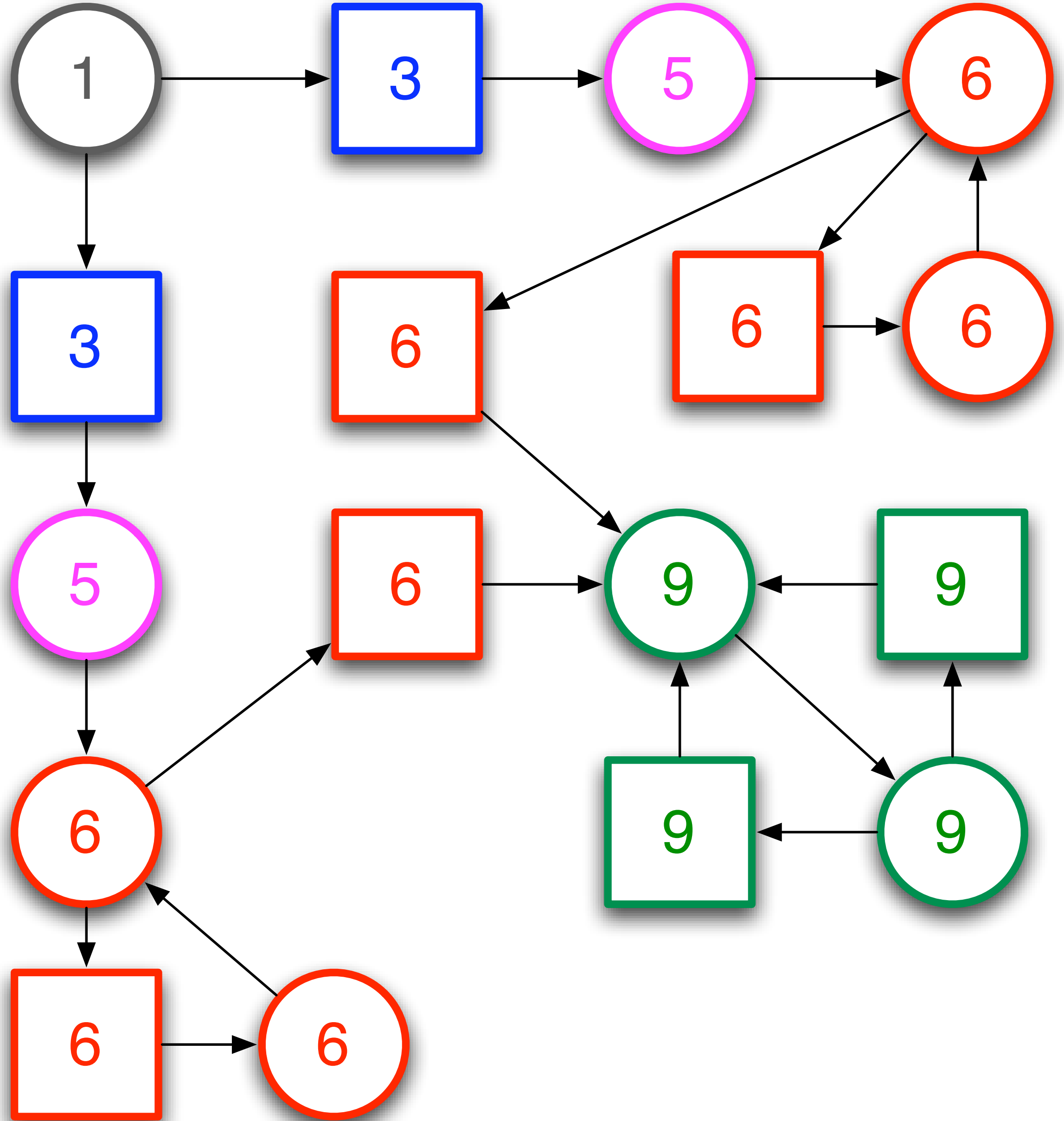
Réduction Muller-parité



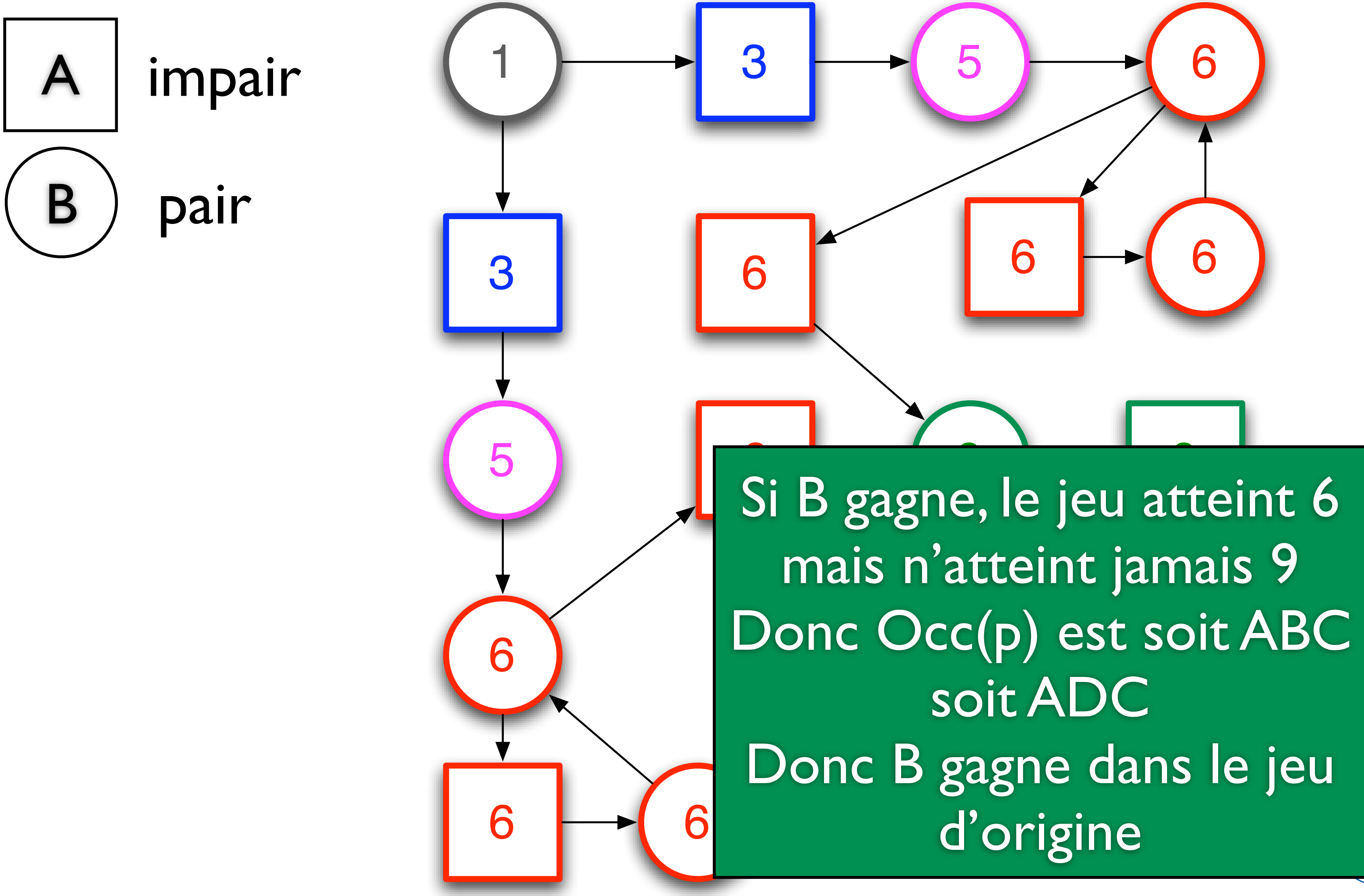
F={ABC,ADC}
(obj pour B)

Réduction Muller-parité





Réduction Muller-parité



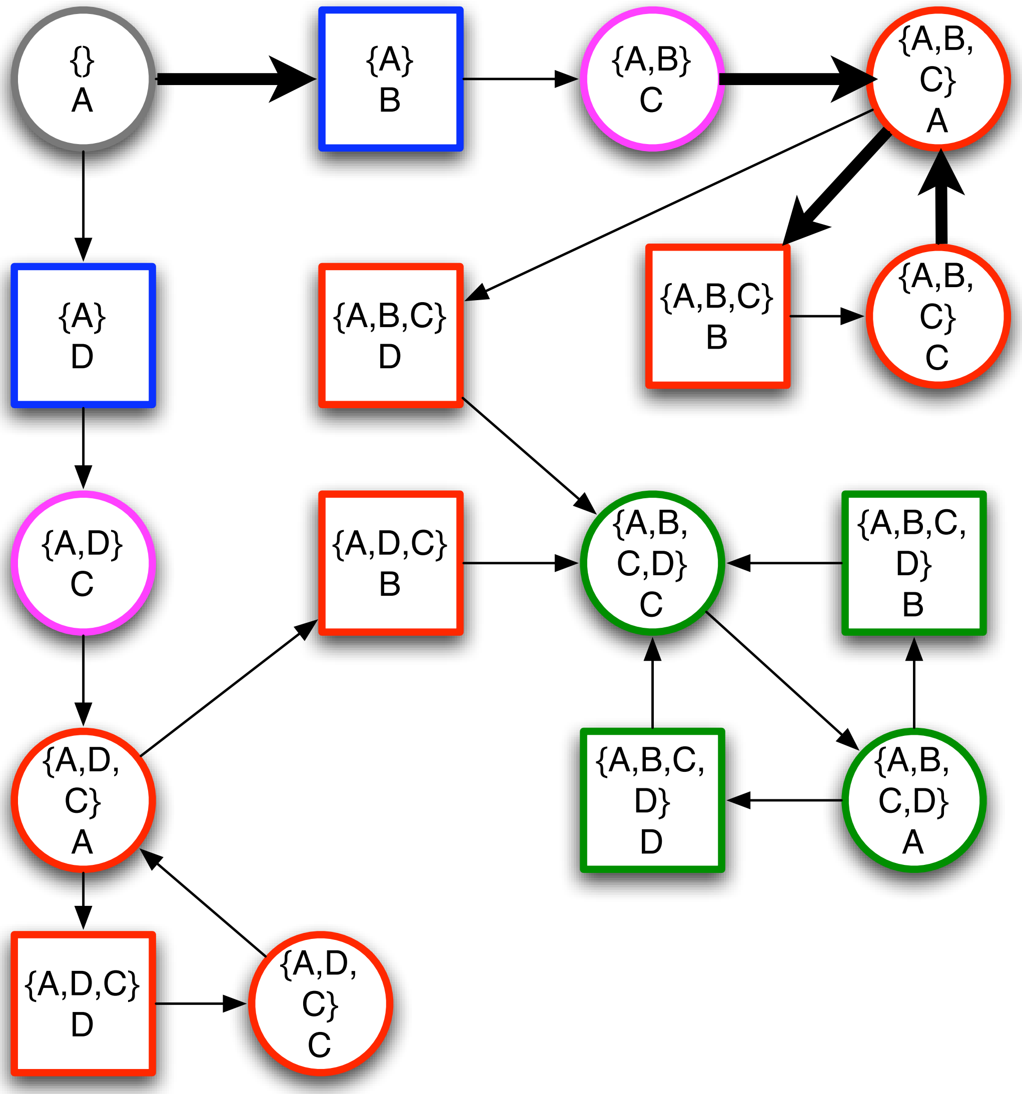
Si B gagne, le jeu atteint 6
mais n'atteint jamais 9
Donc Occ(p) est soit ABC
soit ADC
Donc B gagne dans le jeu
d'origine

Réduction Muller-parité

- On sait maintenant déterminer si un joueur possède une stratégie gagnante dans un jeu de Muller faible (à partir d'une position initiale donnée)
  - D'abord construire le jeu G' avec mémoire
  - Ensuite en déduire un jeu de parité qu'on résout
  - Le joueur a une stratégie depuis la position initiale du jeu de Muller faible ssi il a une stratégie depuis la position initiale du jeu de parité
- Question: comment calculer la stratégie associée ?

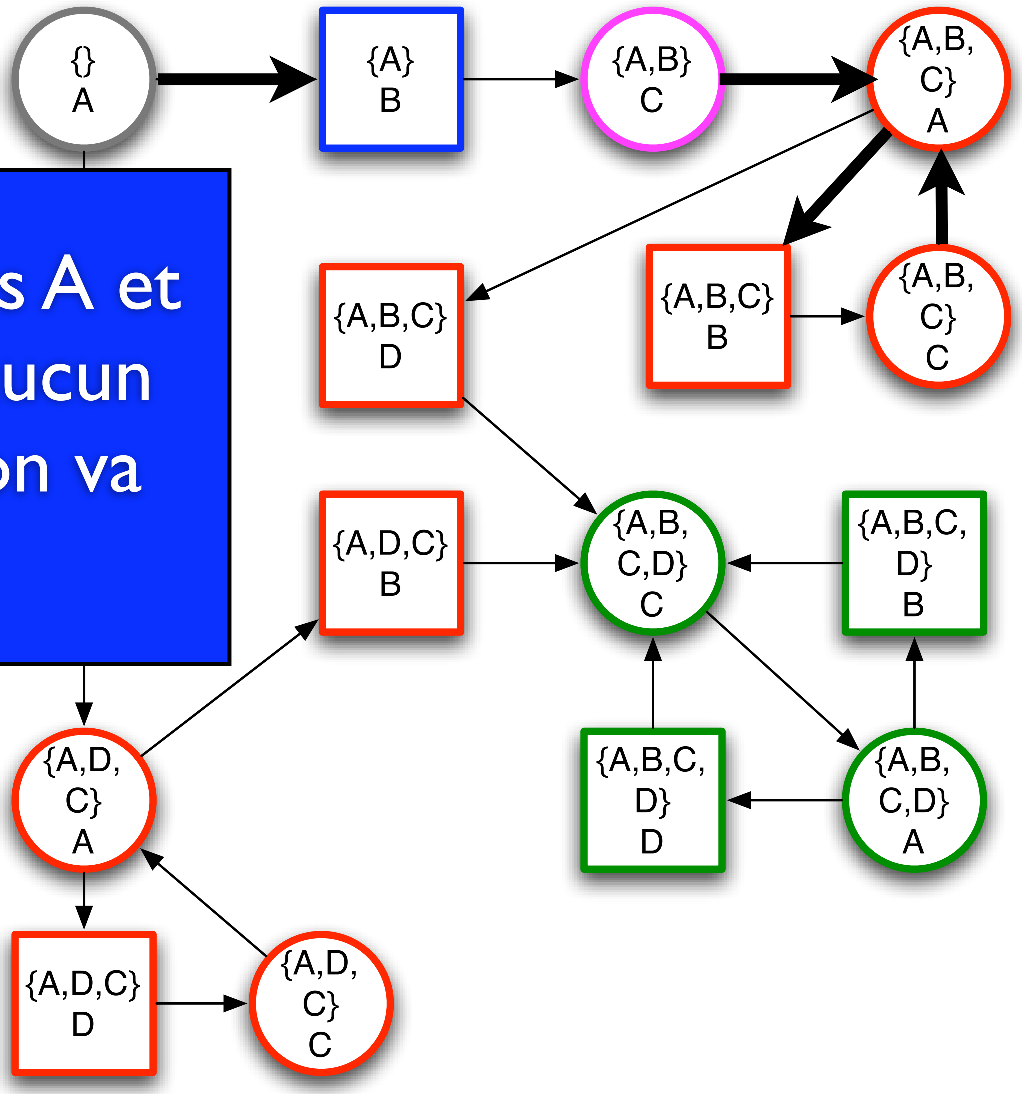
Réduction Muller-parité

- Question: comment calculer la stratégie associée ?



Réduction Muller-parité

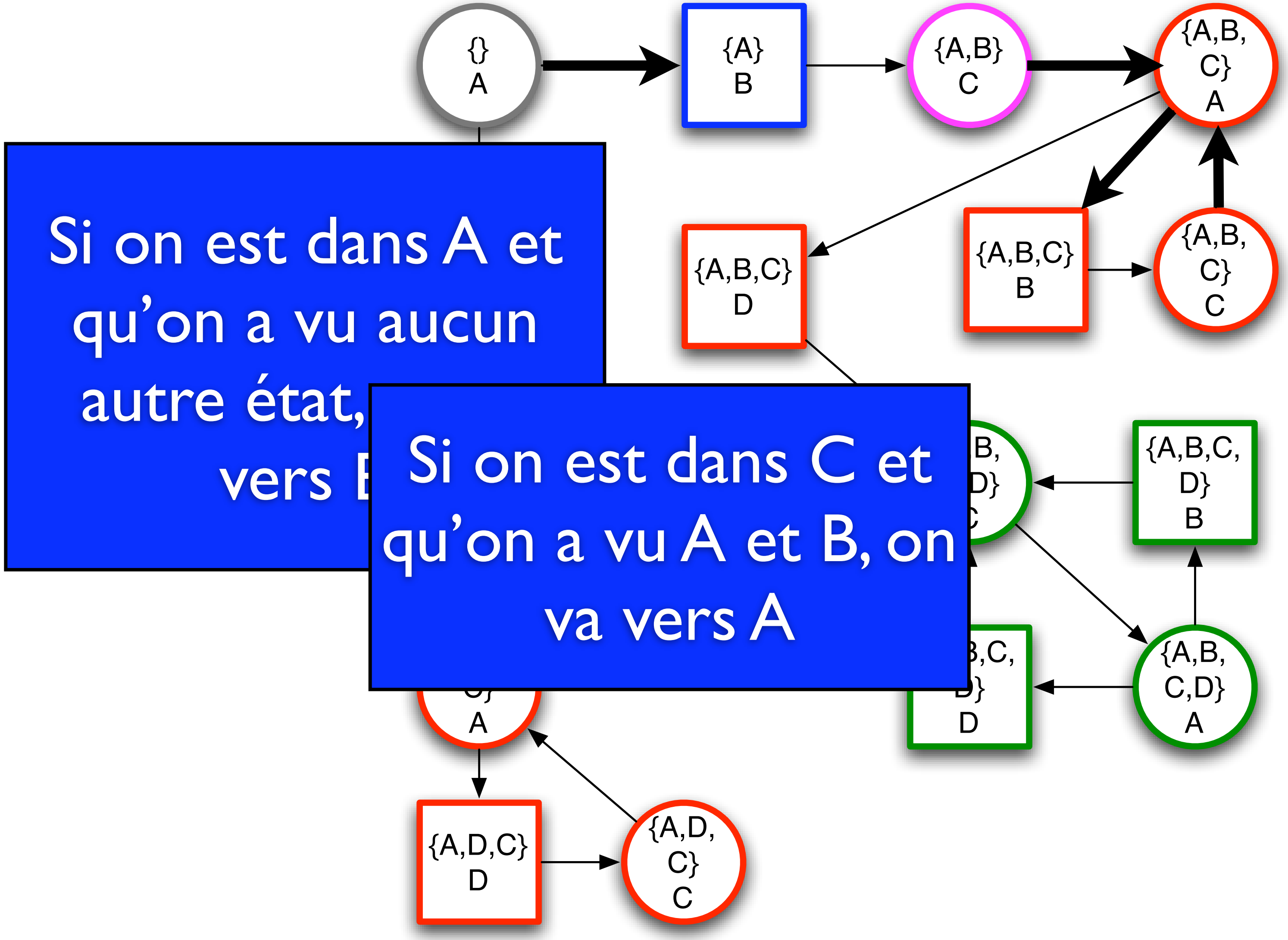
- Question: comment calculer la stratégie associée ?



Si on est dans A et qu'on a vu aucun autre état, on va vers B

Réduction Muller-parité

- Question: comment calculer la stratégie associée ?

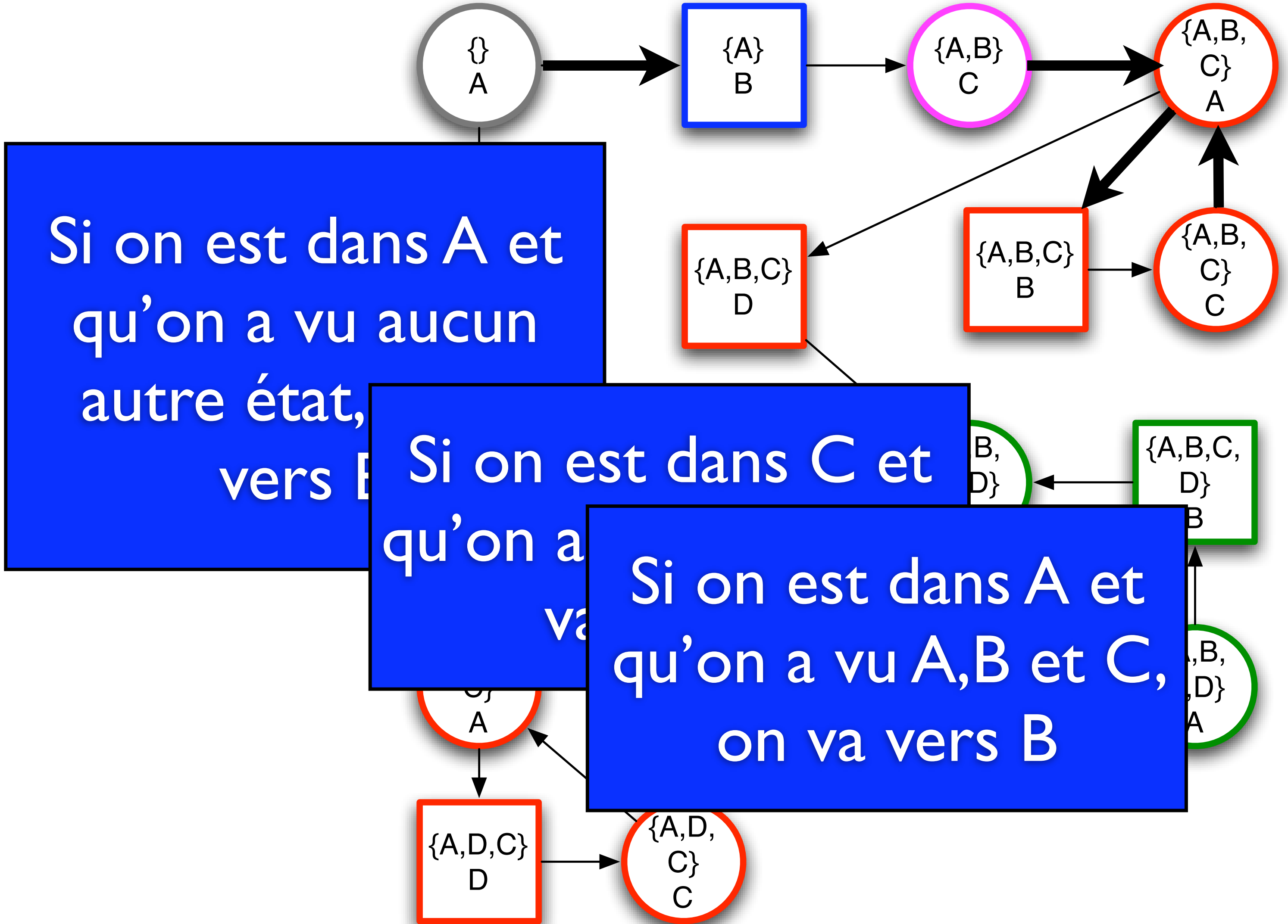


Si on est dans A et qu'on a vu aucun autre état, vers

Si on est dans C et qu'on a vu A et B, on va vers A

Réduction Muller-parité

- Question: comment calculer la stratégie associée ?



Réduction Muller-parité

- Question: comment calculer la stratégie associée ?