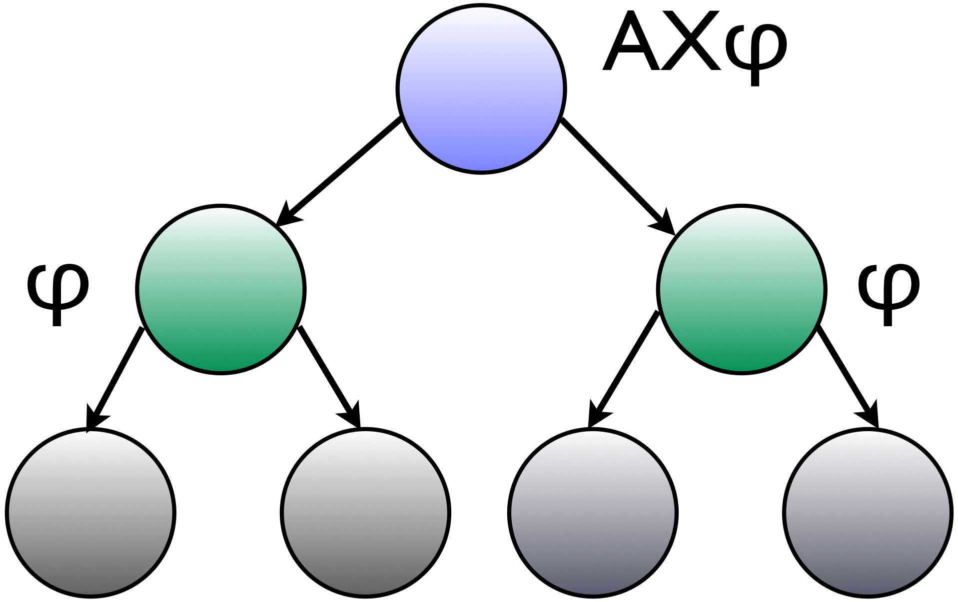
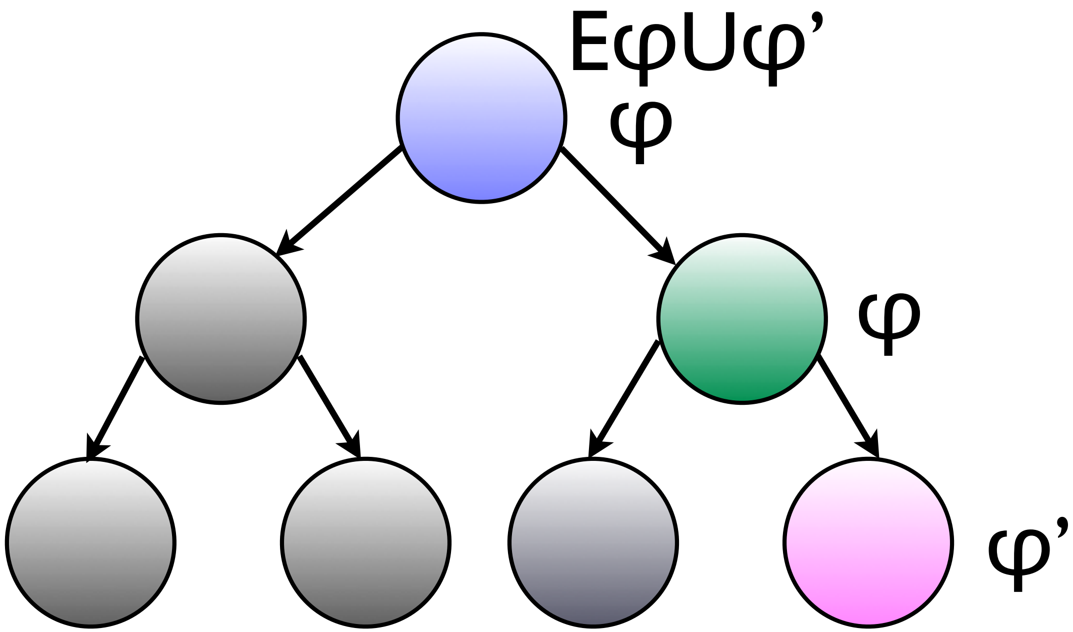
CTL: syntaxe et sémantique



s ⊧ AXφ ssi pour tout s', successeur de s, s' ⊧φ

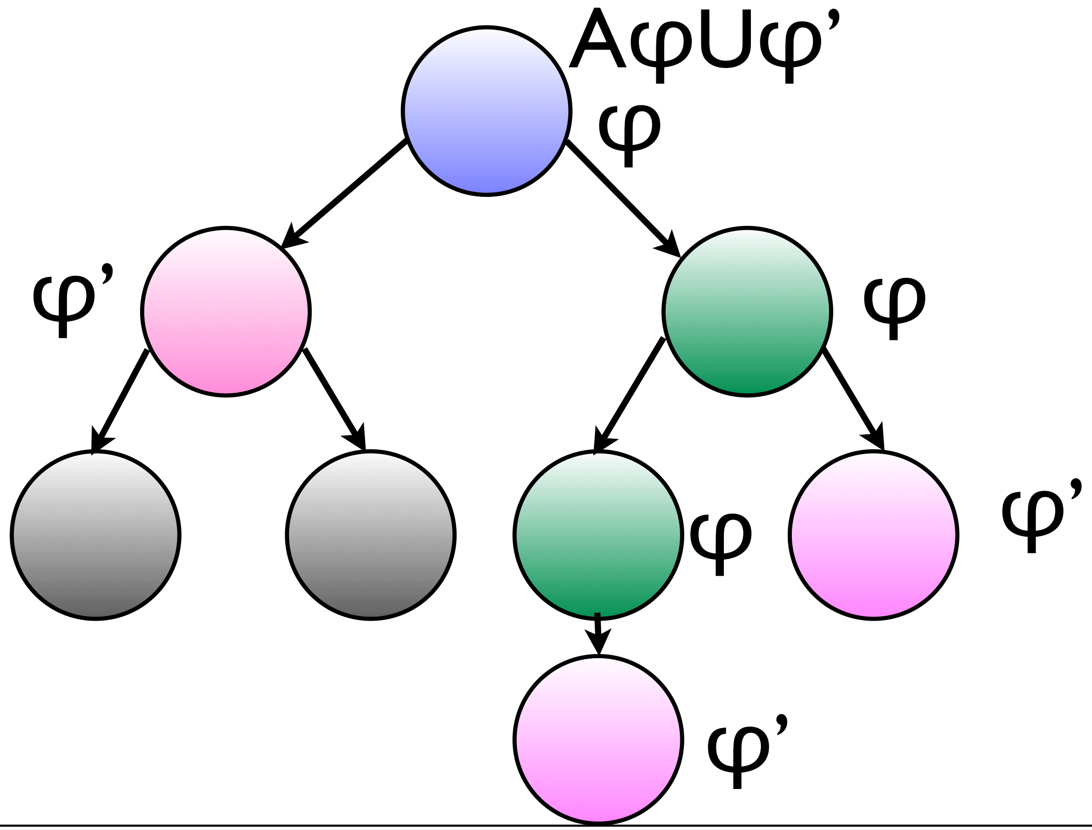
CTL: syntaxe et sémantique



s ⊧ EφUφ' ssi il existe une exécution $s_0 s_1 \ldots s_k$ telle que $s_0 = s$, $s_k \models \varphi'$ et pour tout $0 \leq i < k$, $s_i \models \varphi$.

CTL: syntaxe et sémantique

,



$s \models A\varphi U\varphi'$ ssi pour toute exécution $s_0 s_1 \ldots$ telle que $s_0 = s$, $\exists$ k t.q. $s_k \models \varphi'$ et pour tout $0 \leq i < k$, $s_i \models \varphi$.

CTL: syntaxe et sémantique

$\varphi ::= p \in AP \mid \neg\varphi \mid \varphi \vee \varphi$
$\mid EX\varphi \mid AX\varphi \mid E\varphi U\varphi \mid A\varphi U\varphi$

$s \models p$ ssi $p \in l(s)$

$s \models \neg\varphi$ ssi $s \not\models \varphi$

$s \models \varphi_1 \vee \varphi_2$ ssi $s \models \varphi_1$ ou $s \models \varphi_2$

$s \models EX\varphi$ ssi il existe s', successeur de s, t.q. $s' \models \varphi$

$s \models AX\varphi$ ssi s', pour tout s', successeur de s, $s' \models \varphi$

$s \models E\varphi_1 U\varphi_2$ ssi il existe une exécution $s_0 s_1 \ldots s_k$ tel que $s_0 = s$, $s_k \models \varphi_2$ et pour tout $0 \leq i \leq k$, $s_i \models \varphi_1$.

$s \models A\varphi_1 U\varphi_2$ ssi pour toute exécution $s_0 s_1 \ldots$ telle que $s_0 = s$, il existe k t.q. $s_k \models \varphi_2$ et pour tout $0 \leq i \leq k$, $s_i \models \varphi_1$.

CTL : macros

- $EF\varphi \equiv E\top U\varphi$
- $AF\varphi \equiv A\top U\varphi$
- $EG\varphi \equiv \neg AF\neg\varphi$
- $AG\varphi \equiv \neg EF\neg\varphi$

CTL : Equivalences de formules

- $AX\varphi = \neg EX \neg \varphi$
- $A\varphi U \varphi' = \neg E \neg (\varphi U \varphi') = \neg E(G \neg \varphi' \vee \neg \varphi' U(\neg \varphi \wedge \neg \varphi')) = \neg EG \neg \varphi' \wedge \neg E(\neg \varphi' U(\neg \varphi \wedge \neg \varphi'))$

CTL : Lois d'expansion

- $A\varphi U\varphi' = \varphi' \vee (\varphi \wedge AX(A\varphi U\varphi'))$
- $AF\varphi = \varphi \vee (AXAF\varphi)$
- $AG\varphi = \varphi \wedge AXAG\varphi$
- $E\varphi U\varphi' = \varphi' \vee (\varphi \wedge EXE(\varphi U\varphi'))$
- $EF\varphi = \varphi \vee EXEF\varphi$
- $EG\varphi = \varphi \wedge EXEG\varphi$

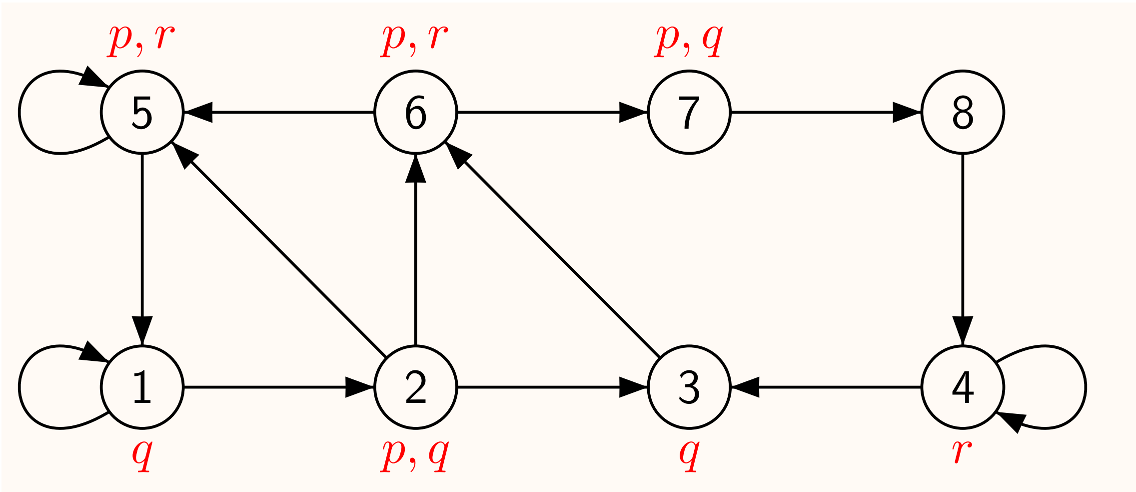
CTL : lois distributives

- $AG(\varphi \wedge \varphi') = AG\varphi \wedge AG\varphi'$
- $EF(\varphi \vee \varphi') = EF\varphi \vee EF\varphi'$

Exemples

- Accessibilité : $EF(x=0)$
- Invariance : $AG\neg(x=0)$
- Vivacité : $AGAF(active)$

Exercice



S(EXp)? S(AXp)?

S(EFp)? S(AFp)?

S(EqUr)?S(AqUr)?

Exercice

- Toute fraude est susceptible d'être détectée un jour(AP={fraude, detect})
- Deux processus ne sont jamais en section critique en même temps (AP={crit1,crit2})
- Toute requête sera un jour satisfaite (AP = {requete, reponse})
- Le processus est activé infiniment souvent (AP= {active})
- Il est possible qu'à partir d'un moment, l'alarme sonne continuellement (AP= {alarm})
- La lumière finit toujours par s'éteindre (AP= {off})
- La lumière finit toujours par s'éteindre et la ventilation tourne tant que la lumière est allumée (AP= {ventilation,off})

Comparaison LTL/CTL

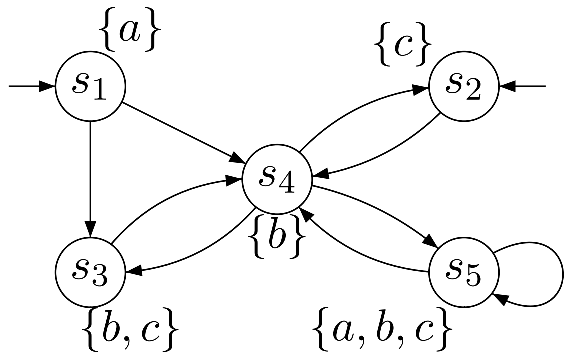
- La formule CTL $AF(a \wedge EXa)$ n'est pas exprimable en LTL
- La formule LTL FG request $\rightarrow$ GF response n'est pas exprimable en CTL
- LTL et CTL incomparables!
- LTL et CTL inclus dans CTL*

3. Algorithmes de Model-Checking

Model-Checking de LTL

- Données : Une structure de Kripke $M=(Q, T, A, q_0, AP, l)$ et une formule LTL φ.
- Question : Est-ce que $M \models \varphi$?
  - $M \models \varphi$ ssi $t,0 \models \varphi$ pour toute trace initiale t de M.

Exercice



$M \models \varphi$?

- $\varphi = FGc$
- $\varphi = GFc$
- $\varphi = Ga$
- $\varphi = aU(G(b \vee c))$
- $X \neg c \rightarrow XXc$

Model-Checking de LTL: principe

- Soit Σ un alphabet. On note Σ^* l'ensemble des mots finis et Σ^ω les mots infinis.
- Modèles de φ = mots infinis. Soit $[\![\varphi]\!]$ le langage des modèles de la formule : $[\![\varphi]\!]=\{t\in(2^{AP})^\omega \mid t, 0 \models \varphi\}$
- Soit $[\![M]\!]$ le langage des traces initiales de M : $[\![M]\!]=\{t\in (2^{AP})^\omega \mid t$ est une trace initiale de M$\}$
- Le problème du model-checking revient donc à vérifier si : $[\![M]\!]\subseteq[\![\varphi]\!]$

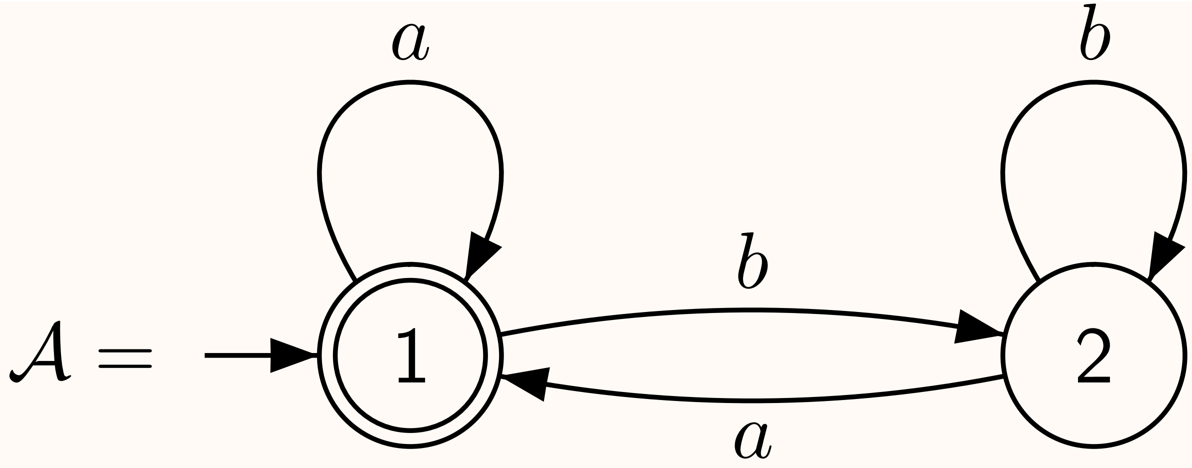
Outil : les automates de Büchi

- Définition : Un automate de Büchi est un n-uplet $A=(Q, \Sigma, I, T, F)$ avec
  - Q un ensemble fini d'états
  - Σ un alphabet fini
  - $I \subseteq Q$ les états initiaux
  - $T \subseteq Q \times \Sigma \times Q$ la relation de transition
  - $F \subseteq Q$ un ensemble d'états acceptants (ou répétés)

Outil : les automates de Büchi

- Une exécution de A sur un mot infini $w=w_0w_1w_2\ldots$ de Σ^ω est une séquence $r=q_0q_1q_2q_3\ldots$ telle que $q_0 \in I$ et $(q_i,w_i,q_{i+1}) \in T$, pour tout $i \geq 0$.
- r est acceptante si $q_i \in F$ pour un nombre infini de i.
- w est accepté par A s'il existe une exécution acceptante de A sur w.
- $L(A)=\{w \in \Sigma^\omega \mid w$ accepté par A$\}$.

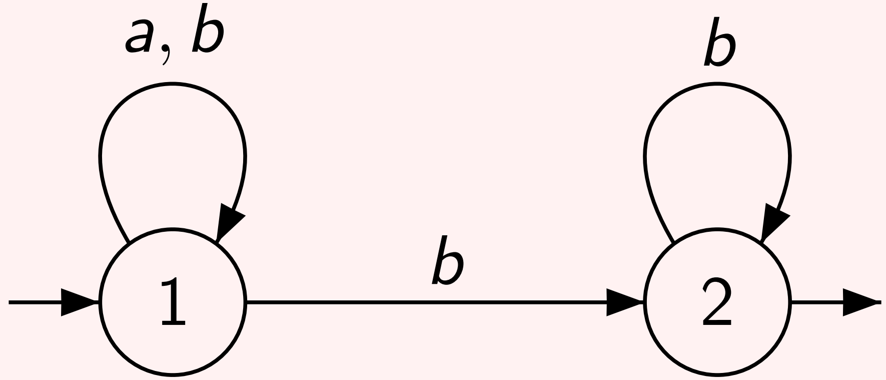
Automate de Büchi: exemple



$$\mathcal{L}(\mathcal{A}) = \{w \in \{a, b\}^\omega \mid |w|_a = \omega\}$$

Automate de Büchi: exemple

$\mathcal{A}=$



$\mathcal{L}(\mathcal{A})$?

Automates de Büchi non-déterministes

- Les automates de Büchi non déterministes sont plus expressifs que les automates de Büchi déterministes
- Les langages reconnus par un NBA forment les ω-réguliers
- Toute formule de LTL peut être reconnue par un NBA

Les automates de Büchi pour LTL

- Définition : Un automate de Büchi est un n-uplet $A=(Q, \Sigma, I, T, F)$ avec
  - Q un ensemble fini d'états
  - Σ un alphabet fini
  - $I \subseteq Q$ les états initiaux
  - $T \subseteq Q \times \Sigma \times Q$ la relation de transition
  - $F \subseteq Q$ un ensemble d'états acceptants (ou répetés)

Les automates de Büchi pour LTL

- Définition : Un automate de Büchi est un n-uplet $A=(Q, \Sigma, I, T, F)$ avec
  - Q un ensemble fini d'états
  - Σ un alphabet fini $\boxed{\Sigma = 2^{AP}}$
  - $I \subseteq Q$ les états initiaux
  - $T \subseteq Q \times \Sigma \times Q$ la relation de transition
  - $F \subseteq Q$ un ensemble d'états acceptants (ou répétés)

Exercice

- Exemple : automate de Büchi reconnaissant p, Xp.
- Construire des automates de Büchi reconnaissant Fp, XXp, Gp, FGp, GFp, pUq, pRq.

Automates de Büchi et LTL

- Les formules de LTL sont moins expressives que les automates de Büchi
- Exemple : «Un instant sur deux, l'événement a arrive.» est une propriété ω régulière non exprimable en LTL

Automates de Büchi

Théorème : Les automates de Büchi sont clos par union, intersection, et complément.

Théorème : on peut tester le vide d'un automate de Büchi.

Automates de Büchi - Test du vide

- Chercher si un état acceptant est accessible depuis l'état initial
- Chercher si cet état appartient à un cycle

Model-Checking LTL : approche par automates

- Donnée: Structure de Kripke M, formule LTL φ.
- Etapes de l'algorithme :
  - Transformer M en un automate A_M tel que $L(A_M)=[\![M]\!]$ (assez facile)
  - Transformer φ en un automate A_φ tel que $L(A_\varphi)=[\![\varphi]\!]$ (plus difficile)
  - Tester si $L(A_M) \subseteq L(A_\varphi)$, i.e., si $L(A_M) \cap L(A_\varphi)^c = \emptyset$.

Model-Checking LTL : approche par automates

- Donnée: Structure de Kripke M, formule LTL φ.
- Etapes de l'algorithme :
  - Transformer M en u… tel que $L(A_M)=[\![M]\!]$ (as…
  - Transf… automate A_φ tel que L(… difficile)
  - Tes… si $L(A_M) \subseteq L(A_\varphi)$, i.e., si $L(A_M) \cap L(A_\varphi)^c = \emptyset$.

Difficile de complémenter un automate de Büchi!!

Model-Checking LTL : approche par automates

- Donnée: Structure de Kripke M, formule LTL φ.
- Etapes de l'algorithme :
  - Transformer M en un automate A_M tel que $L(A_M)=[\![M]\!]$
  - Transformer φ en un automate $A_{\neg\varphi}$ tel que $L(A_{\neg\varphi})=[\![\neg\varphi]\!]$
  - Tester si $L(A_M) \cap L(A_{\neg\varphi})=\emptyset$.

Transformer φ en un automate de Büchi

I. Automates de Büchi généralisés
II. Réduire la formule
    I. Forme normale négative
    II. Réduire les connecteurs temporels
III. Construire un graphe
IV. Transformation en automate de Büchi

Transformer φ en un automate de Büchi

I. Automates de Büchi généralisés

II. Réduire la formule

   I. Forme normale négative

   II. Réduire les connecteurs temporels

III. Transformation en automate de Büchi généralisé

Automates de Büchi généralisés

- Définition : Un automate de Büchi généralisé est un n-uplet $A=(Q, \Sigma, I, T, F)$ avec
    - Q un ensemble fini d'états
    - Σ un alphabet fini
    - $I \subseteq Q$ les états initiaux
    - $T \subseteq Q \times \Sigma \times Q$ la relation de transition
    - $F=\{F_1,F_2,\ldots,F_k\} \subseteq 2^Q$ un ensemble d'ensemble d'états acceptants (ou répétés)

Automates de Büchi généralisés

- Une exécution de A sur un mot infini $w=w_0w_1w_2\ldots$ de Σ^ω est une séquence $r=q_0q_1q_2q_3\ldots$ telle que $q_0\in I$ et $(q_i,w_i,q_{i+1})\in T$, pour tout $i\geq 0$.

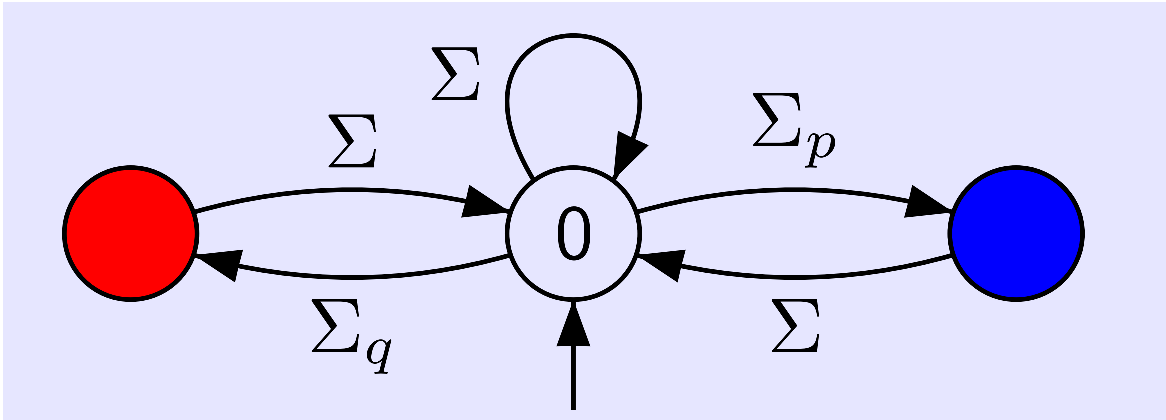
- r est acceptante si pour tout $\mathcal{F}\in F$, $q_i\in\mathcal{F}$ pour un nombre infini de i.

- w est accepté par A s'il existe une exécution acceptante de A sur w.

- $L(A)=\{w\in \Sigma^\omega \mid w \text{ accepté par } A\}$.

Automates de Büchi généralisés : exemple

GFp∧GFq:

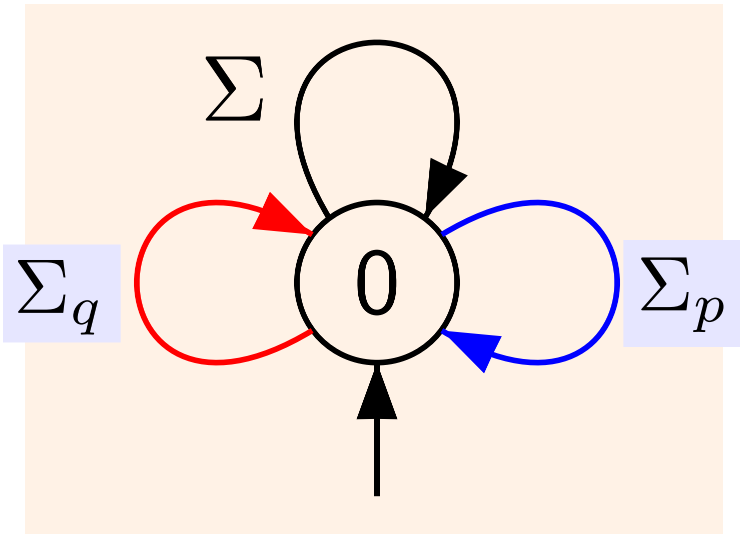


Automates de Büchi généralisés avec condition sur les transitions

- Définition : Un automate de Büchi généralisé avec condition sur les transitions est un n-uplet $A=(Q, \Sigma, I, T, F)$ avec
  - Q un ensemble fini d'états
  - Σ un alphabet fini
  - $I \subseteq Q$ les états initiaux
  - $T \subseteq Q \times \Sigma \times Q$ la relation de transition
  - $\mathcal{T}=\{T_1, T_2, \ldots, T_k\} \subseteq 2^T$ un ensemble d'ensemble de transitions acceptantes (ou répétées)

Automates de Büchi généralisés avec condition sur les transitions - exemple

GFp∧GFq:



Des ABG aux AB

Théorème : Tout automate de Büchi généralisé A peut être transformé en un automate de Büchi A' tel que L(A)=L(A')

Transformer φ en un automate de Büchi

I. Automates de Büchi généralisés

II. Réduire la formule

   I. Forme normale négative

   II. Réduire les connecteurs temporels

III. Transformation en automate de Büchi généralisé

Forme normale négative

$$\varphi ::= \bot \mid \top \mid p \mid \neg p \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid$$
$$X\varphi \mid \varphi U \varphi \mid \varphi R \varphi$$

- $\neg\neg p = p$
- $\neg(\varphi_1 \vee \varphi_2) = \neg\varphi_1 \wedge \neg\varphi_2$
- $\neg(\varphi_1 \wedge \varphi_2) = \neg\varphi_1 \vee \neg\varphi_2$
- $\neg(X\varphi) = X(\neg\varphi)$
- $\neg(\varphi_1 U \varphi_2) = \neg\varphi_1 R \neg\varphi_2$
- $\neg(\varphi_1 R \varphi_2) = \neg\varphi_1 U \neg\varphi_2$

Exercice

- Transformer $G(p \rightarrow Fq)$ en forme normale négative

Réduire les connecteurs temporels

- Idée : Un état de notre graphe va représenter l'ensemble des propositions atomiques vérifiées au prochain instant de la séquence, et l'ensemble des sous-formules qu'il «promet» de vérifier à l'état suivant.
- Pour cela, on ne veut que des propositions atomiques (ou négations), et des sous-formules commençant par X (next).

Réduire les connecteurs temporels

- Un ensemble Z de formules en forme normale négative est réduit si

  1. pour tout $z \in Z$, z est de la forme p, $\neg p$ ou $X(z')$
  2. il est cohérent : $\perp \notin Z$, $\{p, \neg p\} \not\subseteq Z$, pour tout $p \in AP$.