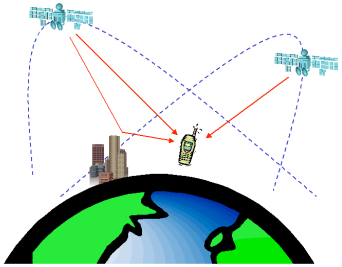
Vérification formelle de systèmes par Model-Checking

Nathalie Sznajder
Sorbonne Université, LIP6

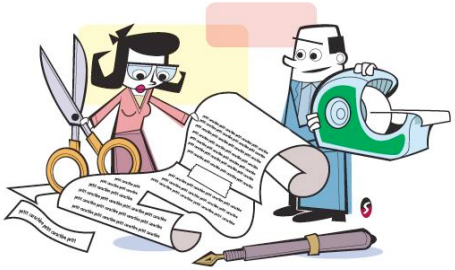
Les méthodes formelles

- Preuve assistée par ordinateur
- Test
- Model-Checking

Model-Checking

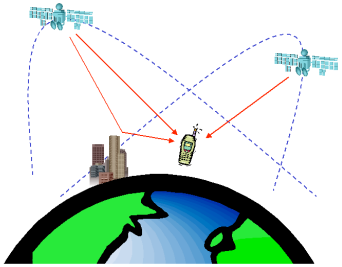
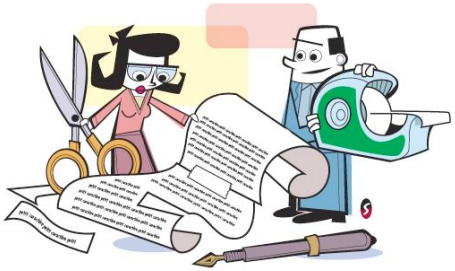


système



spécification

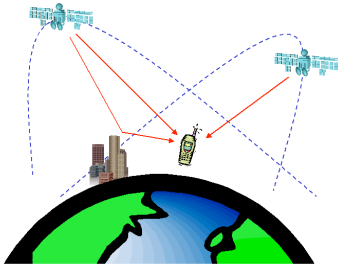
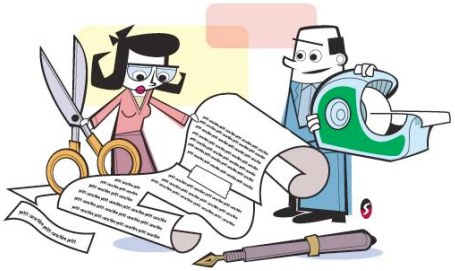
Model-Checking

Est-ce que le  satisfait  ?

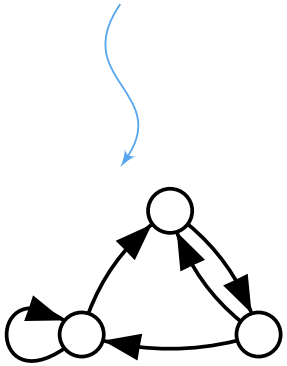
système

spécification

Model-Checking

Est-ce que le  satisfait  ?

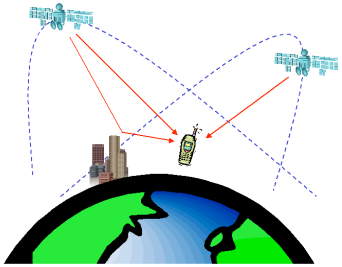
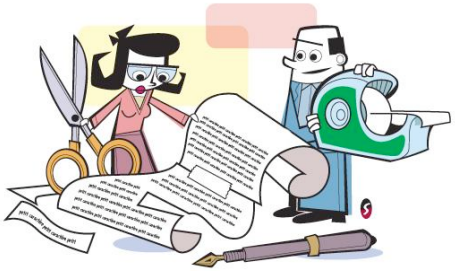
système



modèle

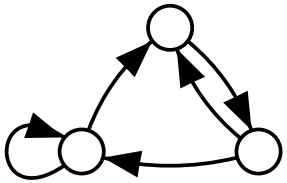
spécification

Model-Checking

Est-ce que le  satisfait  ?

système

spécification

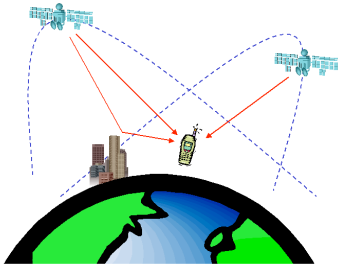
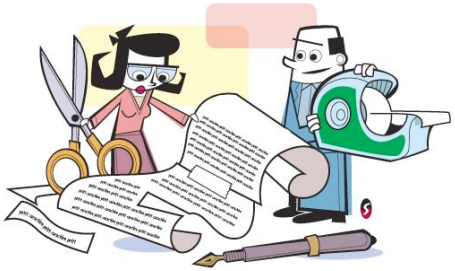


modèle

φ

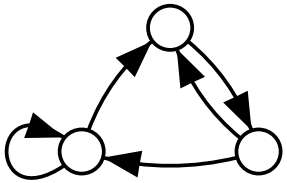
formule

Model-Checking

Est-ce que le  satisfait  ?

système

spécification



modèle

$\models$

algorithme de Model Checking

φ

formule

?

Références bibliographiques

- *Model Checking*, E. Clarke, O. Grumberg, D. Peled, MIT Press 99
- *Vérification de logiciels : techniques et outils du model-checking*, P. Schnoebelen, B. Bérard, M. Bidoit, F. Laroussinie, A. Petit, Vuibert 99
- *Principles of Model-Checking*, C. Baier, J.-P. Katoen, MIT Press 08

Les devises de Leslie Lamport

- A system specification consists of a lot of elementary mathematics glued together with a tiny bit of temporal logic
- Unfortunately some [•••] believe that fluency in C++ is more important than a sound education in elementary mathematics. So, some readers may be unfamiliar with the math needed to write specification.
- If If exposure to C++ has not destroyed your ability to think logically, you should have no trouble filling any gaps in your mathematics education

(Gérard Berry, leçon au collège de France, mars 2015)

Plan

1. Modélisation
2. Spécifications
    1. Généralités
    2. LTL
    3. CTL
3. Algorithmes de Model Checking
    1. LTL
    2. CTL
    3. Equité

1. Modélisation

- On veut vérifier comportement du système au cours du temps.
- Notion d'état à un instant donné
- Actions du système →changement d'état.
- →Système de transition
- Informations supplémentaires sur
  - communication (notion d'action)
  - propriétés vérifiées par les états (propositions atomiques)

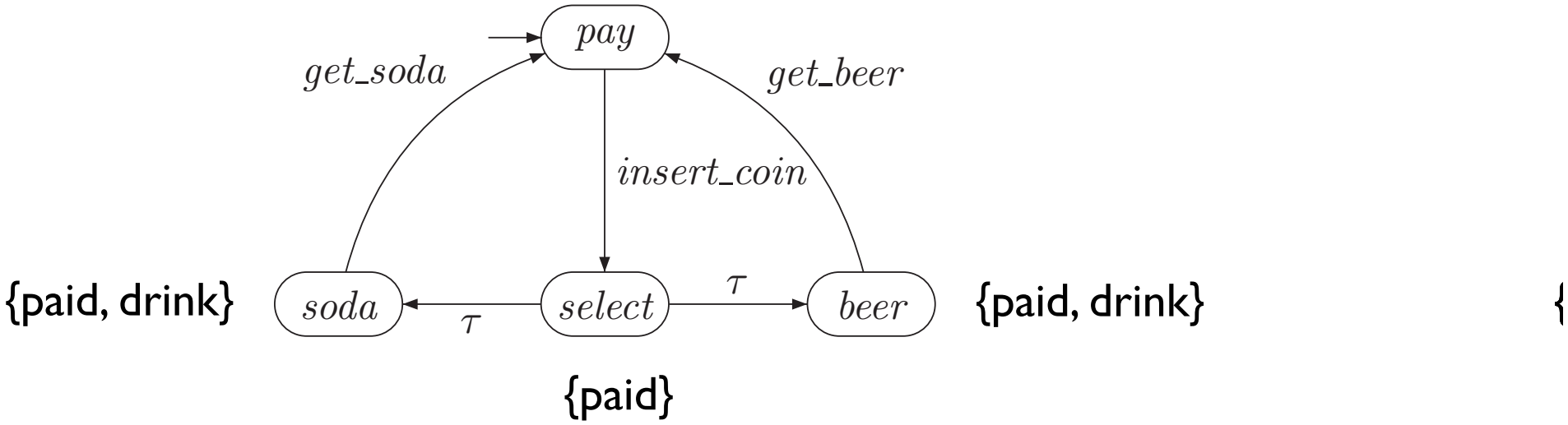
Structure de Kripke

- Définition: $M=(Q, T, A, q_0, AP, l)$
  - Q : ensemble fini d'états
  - A : alphabet d'actions (facultatif)
  - T : relation de transitions entre états
  - q_0 : état initial
  - AP : ensemble de propositions atomiques
  - $l : Q \rightarrow 2^{AP}$, étiquetage des états

Structure de Kripke

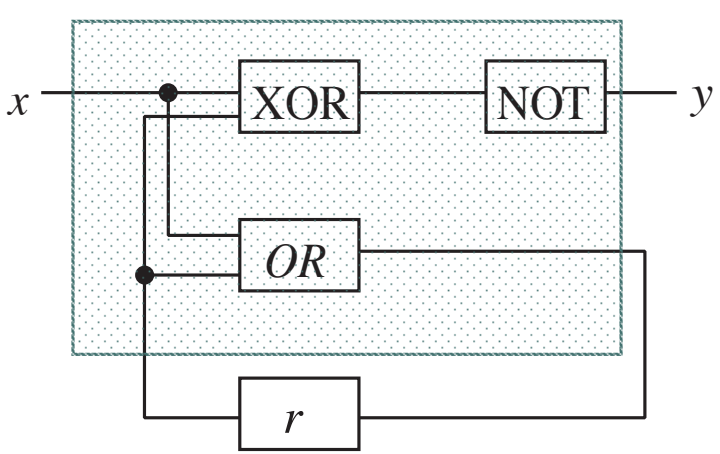
- Soit M=(Q, T, A, q_0, AP, l) une structure de Kripke.
- Soit q un état. L'ensemble {q'∈Q| il existe a ∈A, (q,a,q')∈T} est l'ensemble des successeurs de q.

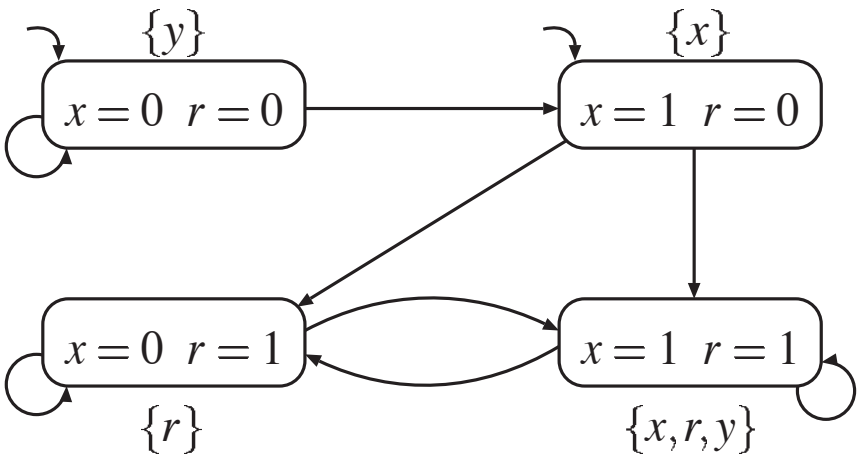
Exemple: distributeur



[Principles of Model-Checking, C. Baier, J.-P. Katoen]

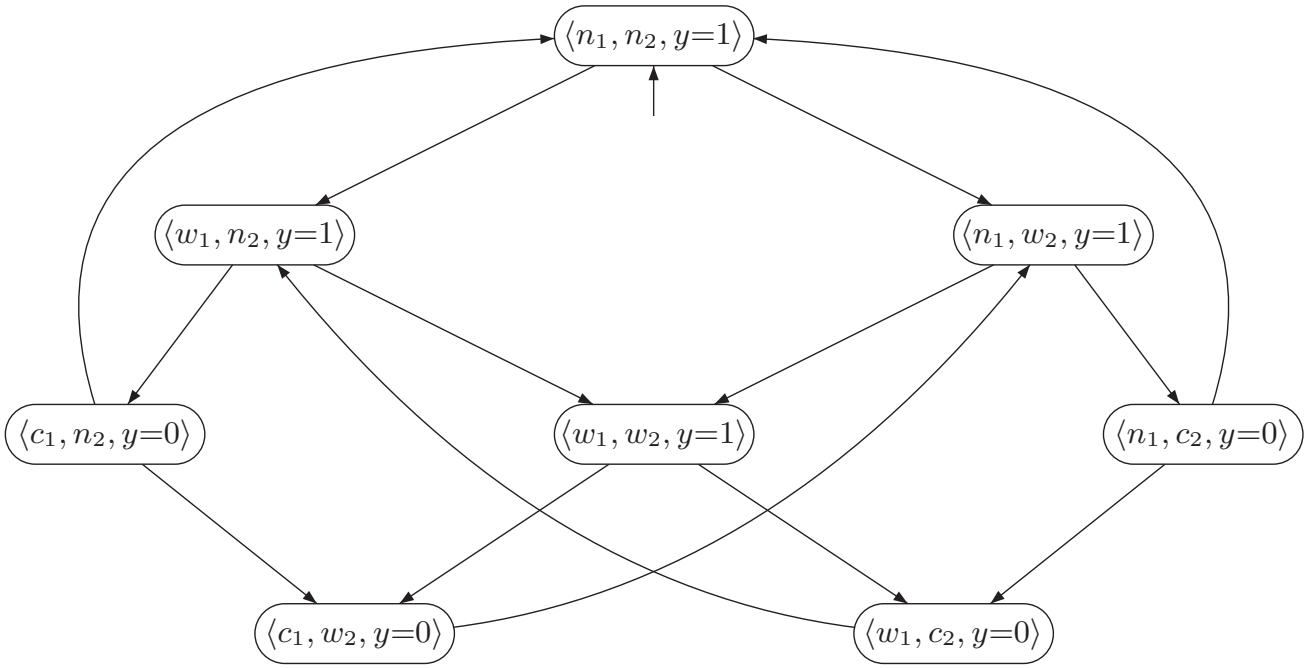
Exemple: circuit





[Principles of Model-Checking, C. Baier, J.-P. Katoen]

Exemple: exclusion mutuelle



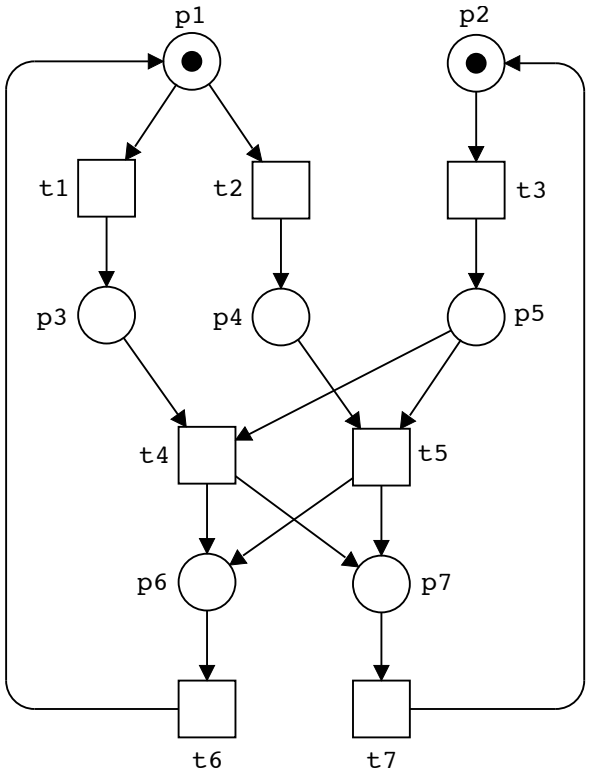
Structure de Kripke

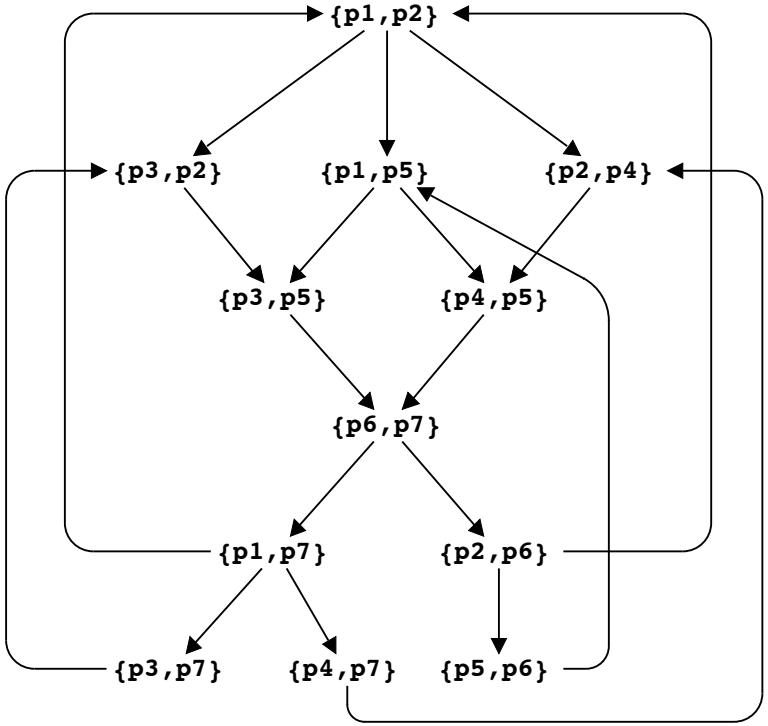
- Soit M= (Q, T, A, q_0, AP, l).
- On supposera que T est totale, i.e., chaque état a au moins un successeur.
- On peut compléter une structure de Kripke : on rajoute un état puits successeur des états dead-lock.

Descriptifs de haut niveau

- Programmes séquentiels
- Programmes concurrents
- Réseaux de Petri
- ...

Une structure de Kripke générée par un réseau de Petri





Systèmes concurrents

- Mode d'exécution : synchrone, asynchrone
- Mode de communication : mémoire partagée (variables partagées), envoi de messages (rendez-vous, broadcast…)
- Exemples : threads Java (asynchrone, variables partagées), processus (asynchrone, envoi de messages), composants électroniques,

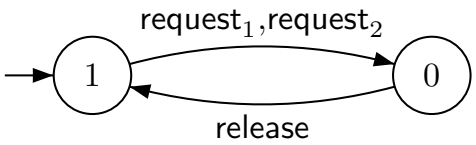
Systèmes concurrents

- Compositionnalité des modèles, description modulaire
- Différents modes de synchronisation
  - mode d'exécution : synchrone, asynchrone
  - mode de synchronisation : variables partagées, envoi de messages (rendez-vous, broadcast…)
  - ...
- explosion combinatoire

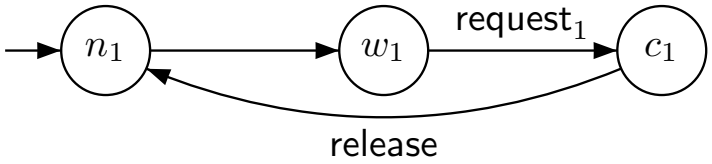
- Exemples : threads Java (asynchrone, variables partagées), processus (asynchrone, envoi de messages), composants électroniques dans un processeur (synchrone, variables partagées), agents d'un protocole de communication (asynchrone, envoi de messages)…

Exemple : exclusion mutuelle II

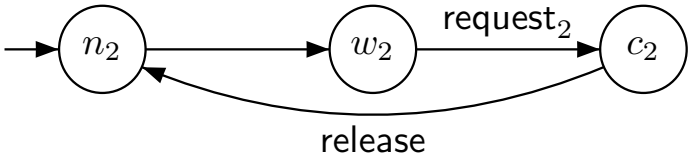
y :

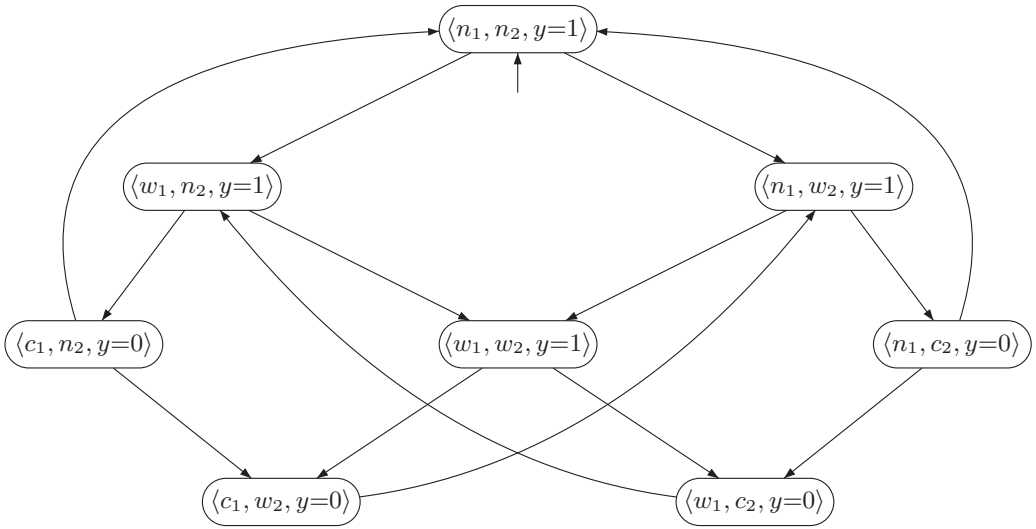


P1



P2

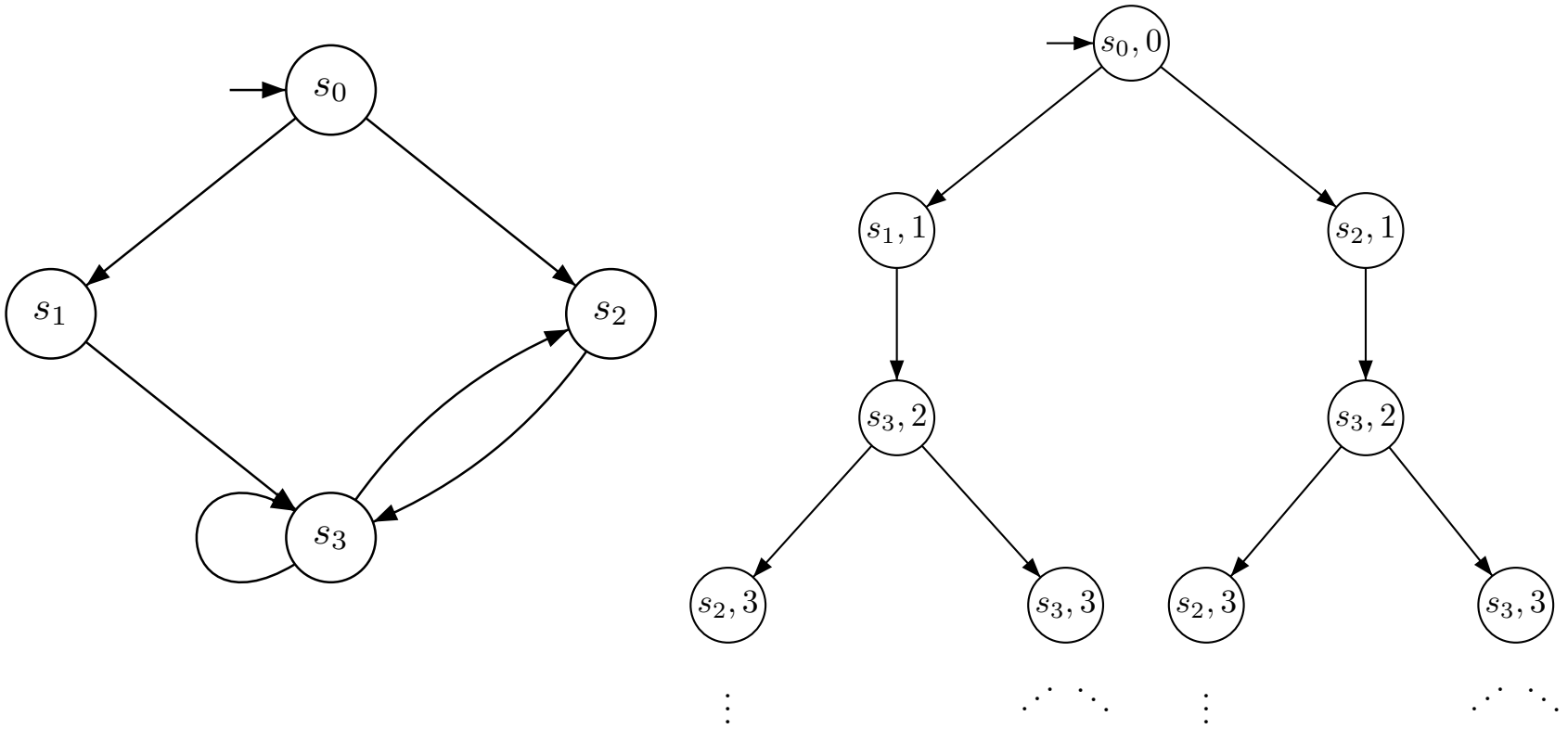




Exécutions et traces

- Soit $M=(Q,T,A,q_0,AP,l)$. Une exécution de M est une séquence infinie $r=q_0a_0q_1a_1q_2a_2...$ telle que $(q_i,a_i,q_{i+1})\in T$, pour tout $i\geq 0$.
- On peut omettre l'étiquetage des transitions : $r=q_0q_1q_2...$
- Une trace d'exécution de M est l'étiquetage d'une exécution: $l(r)=l(q_0)l(q_1)l(q_2)...$

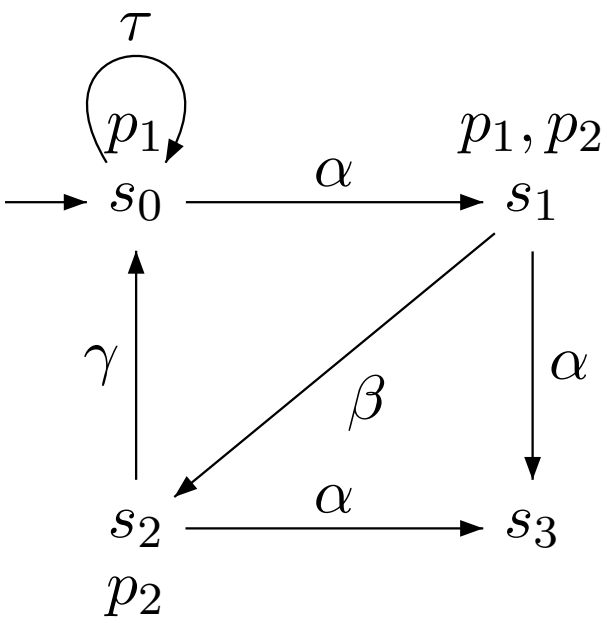
Arbre d'exécutions d'une structure de Kripke



Arbre d'exécutions d'une structure de Kripke

- Correspond au «dépliage» de la structure de Kripke
- Sa racine est l'état initial de la structure de Kripke
- Au niveau i, les fils d'un noeud sont les états successeurs au niveau i+ l
- La relation de transition est totale : l'arbre est infini

Exercice



- Décrire formellement la structure de Kripke ci-dessus.
- Donner une exécution, une trace d'exécution
- Dessiner l'arbre d'exécutions associé (3 premiers niveaux).

2. Spécifications

Propriétés sur les systèmes de transition (I)

- Invariance : tous les états du système vérifient une certaine propriété
- Sûreté : quelque chose de mauvais n'arrive jamais
- Accessibilité : un état donné est accessible depuis l'état initial

Propriétés sur les systèmes de transition (II)

- Vivacité : Quelque chose de «bon» finira par arriver
- Equité : Quelque chose se produira infiniment souvent

Logiques temporelles

- Permettent d'exprimer propriétés sur séquences d'observations
- Utilisation de connecteurs temporels et de quantificateurs sur les chemins

Logiques temporelles : pourquoi?

- On pourrait utiliser logique du premier ordre.
- Exemple : «toute requête sera un jour satisfaite»

$$\forall t \cdot \big(\text{requete} \rightarrow \exists t' \geq t \cdot (\text{reponse})\big)$$

Logiques temporelles : pourquoi?

- On pourrait utiliser [illegible] premier ordre.
- [illegible] requête sera un jour [illegible]

Difficile à écrire/comprendre
Vérification peu efficace

$$\forall t \cdot \big(\text{requete} \rightarrow \exists t' \geq t \cdot (\text{reponse})\big)$$

Logiques temporelles

- Pas de variable (instants implicites), mais modalités.
- Temporel ≠ temporisé : logiques temporelles ne quantifient pas écoulement du temps.

Logiques temporelles linéaires ou arborescentes

- 2 approches :
  - temps linéaire : propriétés des séquences d'exécutions (futur déterminé)
  - temps arborescent : propriétés de l' arbre d'exécutions (tous les futurs possibles)

2.2 La logique LTL

Logique temporelle linéaire : LTL

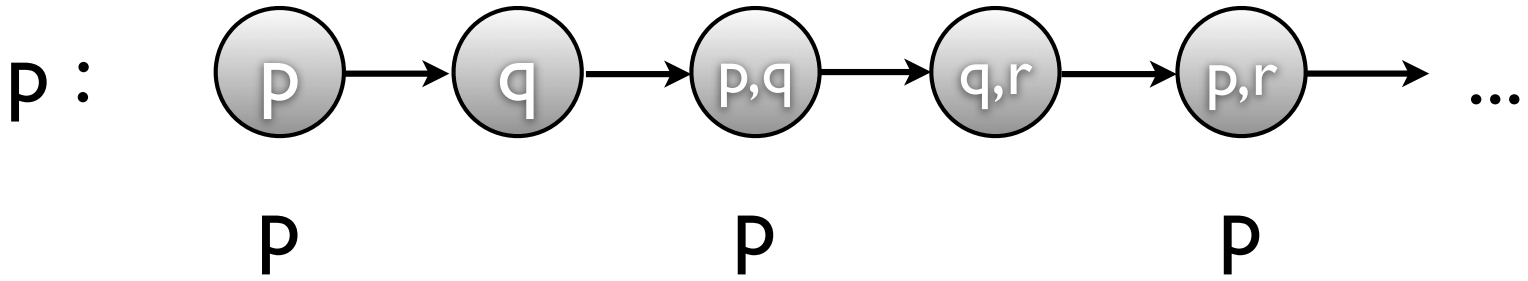
[Pnueli 77]

- Modèle des formules : une trace d'exécution infinie.
- $t, i \models \varphi$ ssi la formule φ est vérifiée à la position i de la trace.
- Défini inductivement sur la formule

Logique temporelle linéaire : LTL

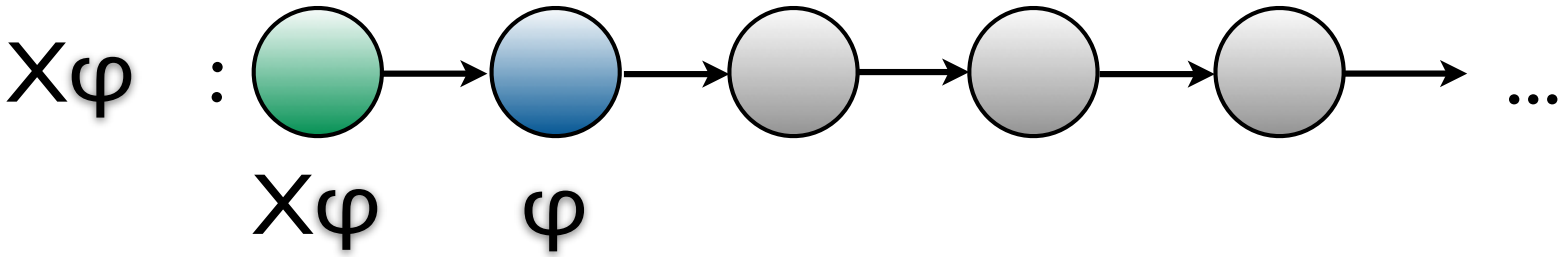
- Rappel : une trace d'exécution ≡ exécution dans laquelle seul l'étiquetage des états est visible
- → c'est un mot (infini) sur l'alphabet 2^{AP}.
- Soit t une trace, on note t(i) la «lettre» à la position $i \geq 0$, i.e. l'ensemble des propositions atomiques vraies.

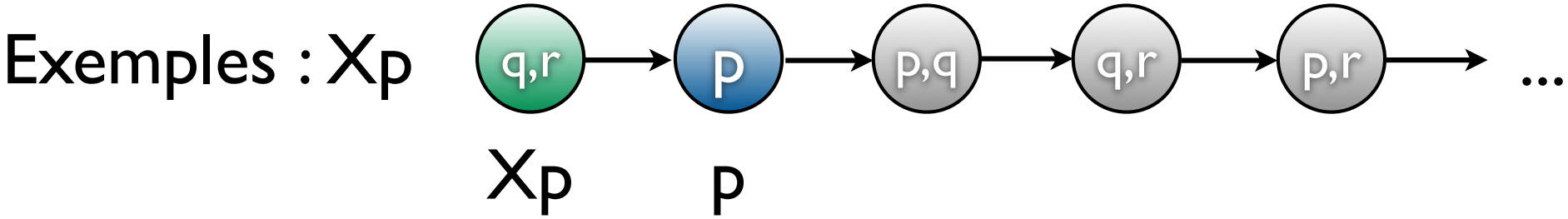
Logique temporelle linéaire : LTL

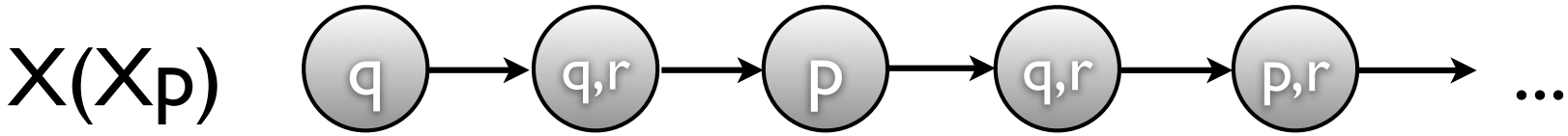


t,i ⊨ p ssi p∈t(i)

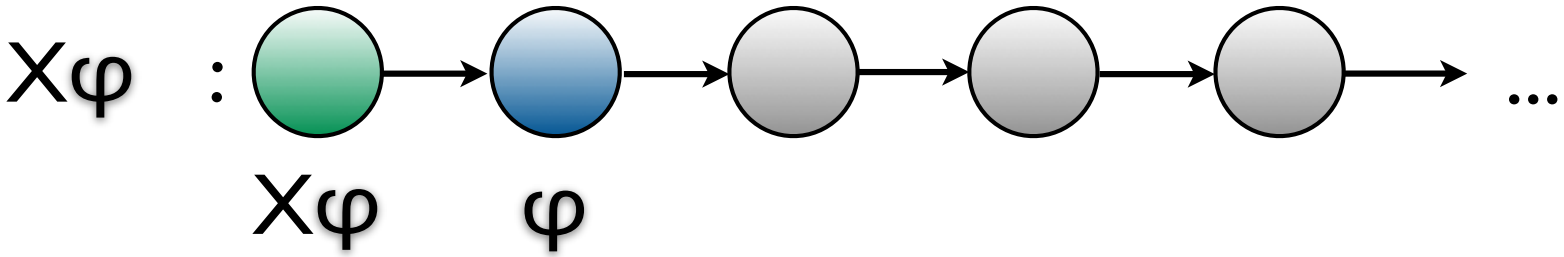
Logique temporelle linéaire : LTL

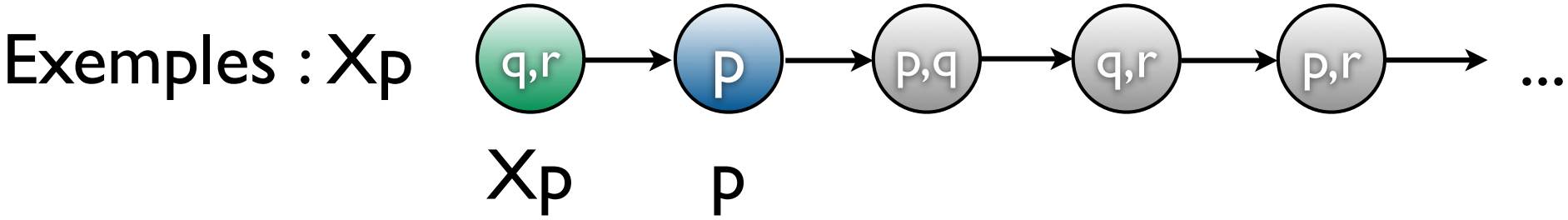


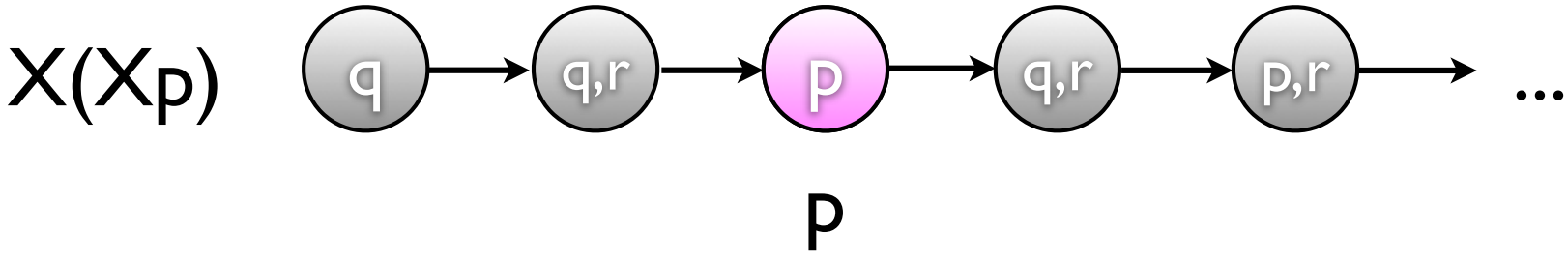




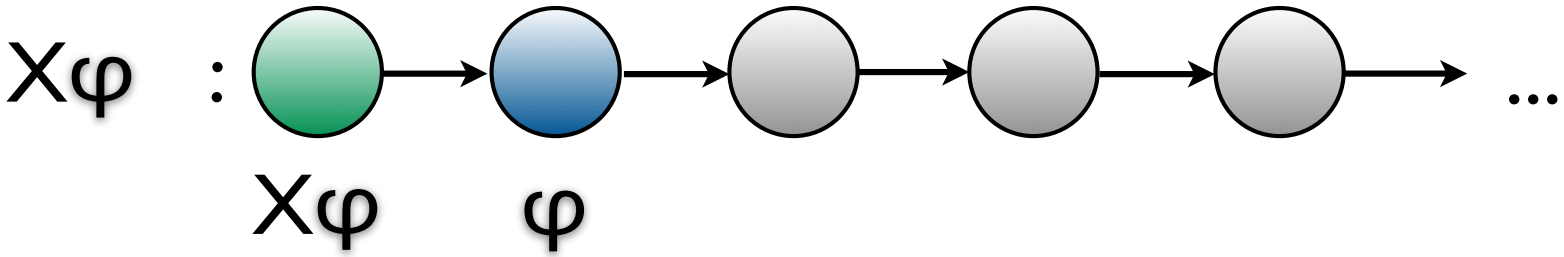
Logique temporelle linéaire : LTL

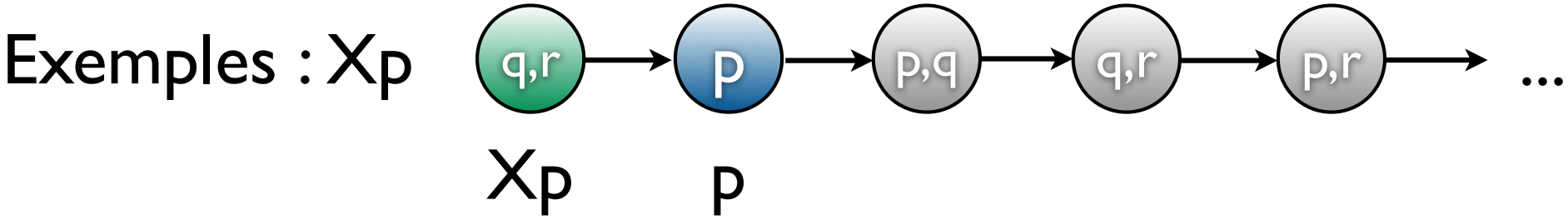


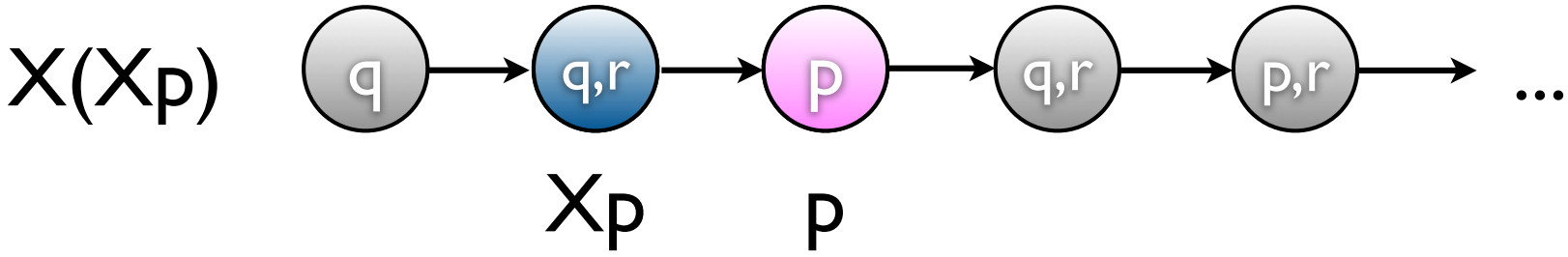




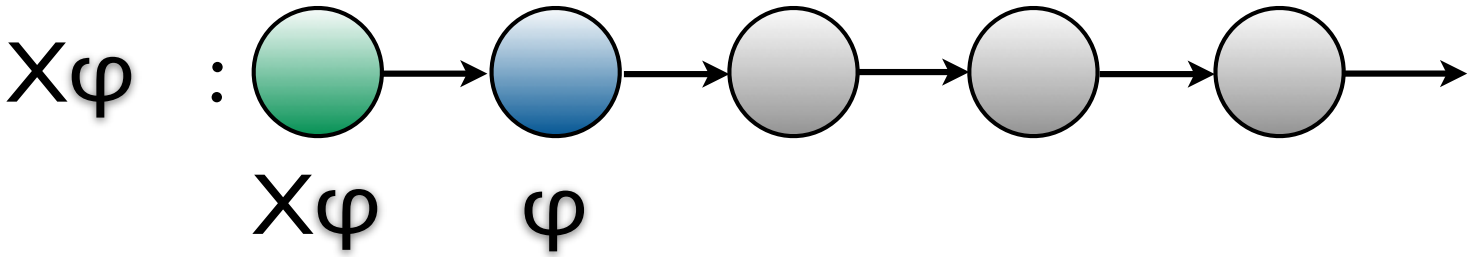
Logique temporelle linéaire : LTL

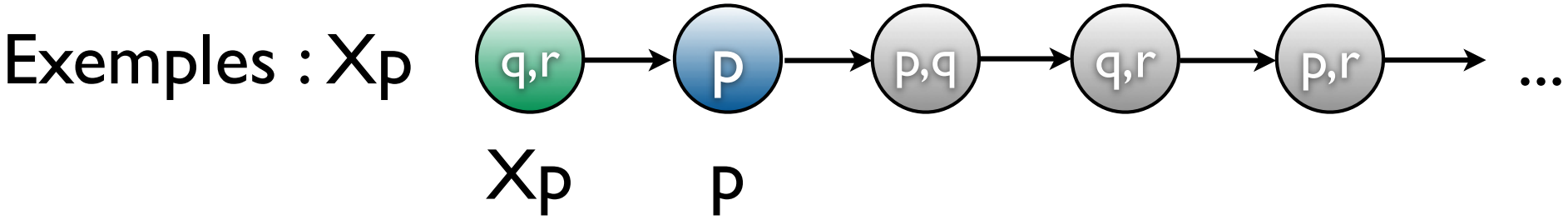


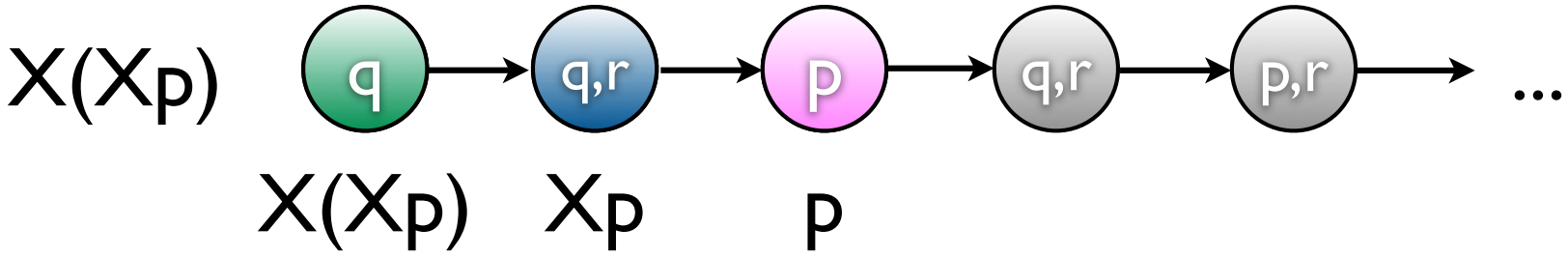




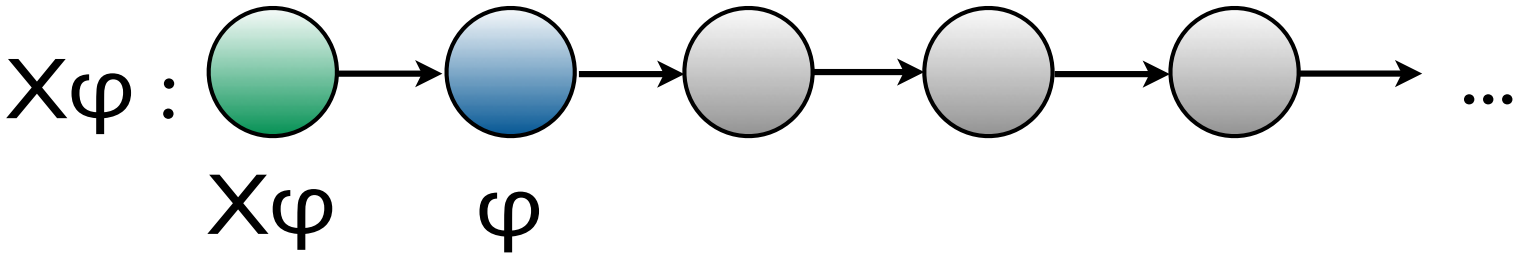
Logique temporelle linéaire : LTL





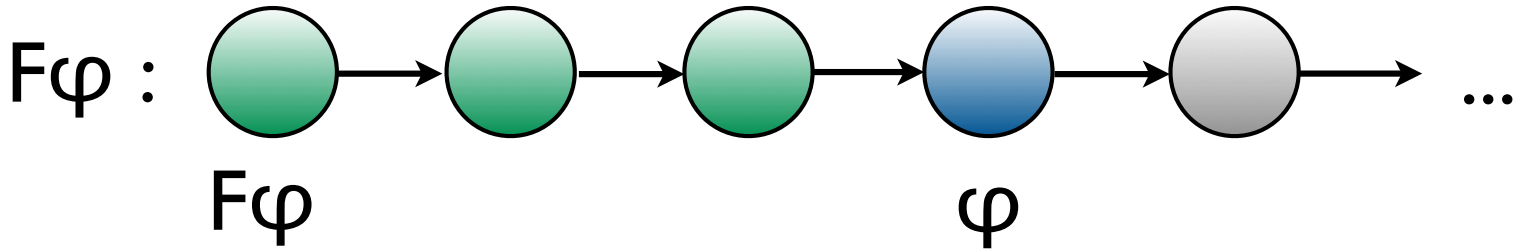


Logique temporelle linéaire : LTL

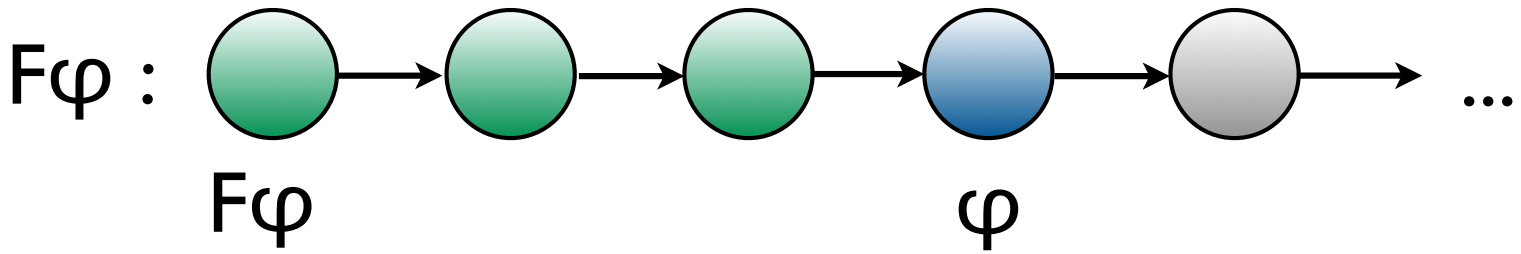


$t,i \models X\varphi$ ssi $t,i+1 \models \varphi$

Logique temporelle linéaire : LTL

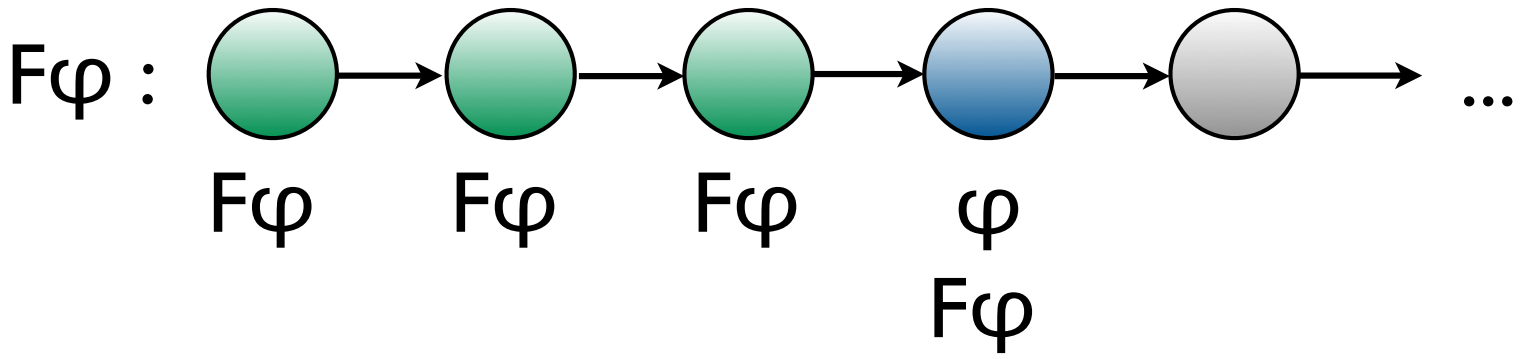


Logique temporelle linéaire : LTL



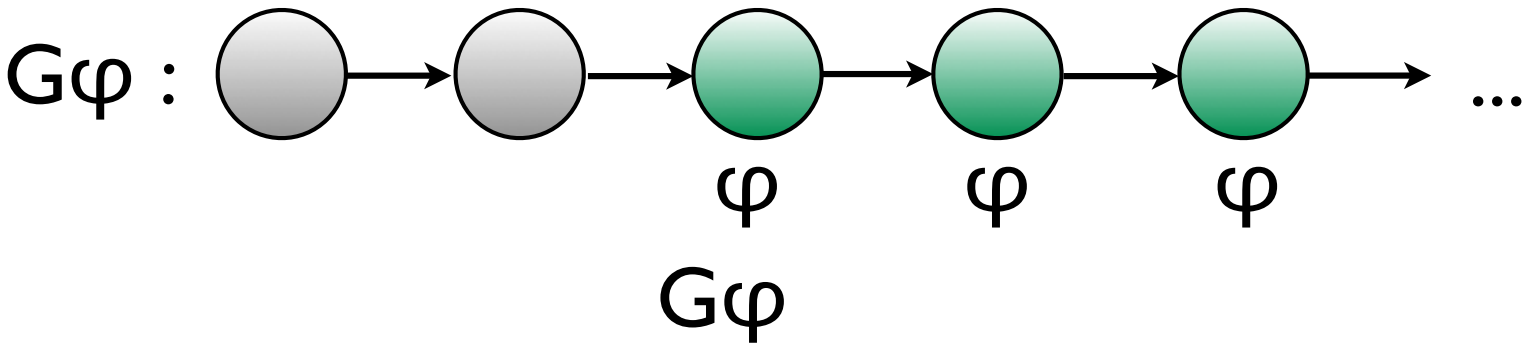
t,i ⊨ Fφ ssi il existe j≥i tel que t,j⊨φ

Logique temporelle linéaire : LTL

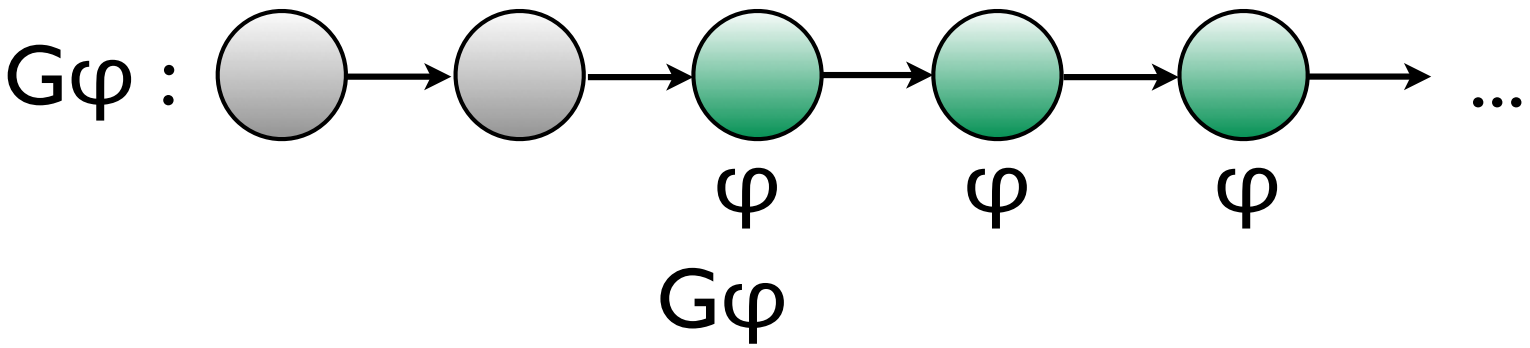


t,i ⊨ Fφ ssi il existe j≥i tel que t,j⊨φ

Logique temporelle linéaire : LTL

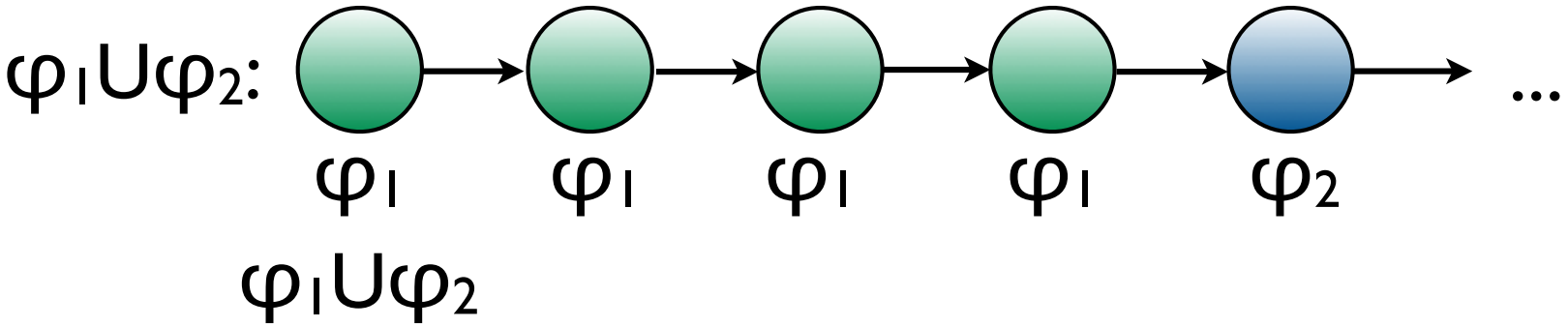


Logique temporelle linéaire : LTL

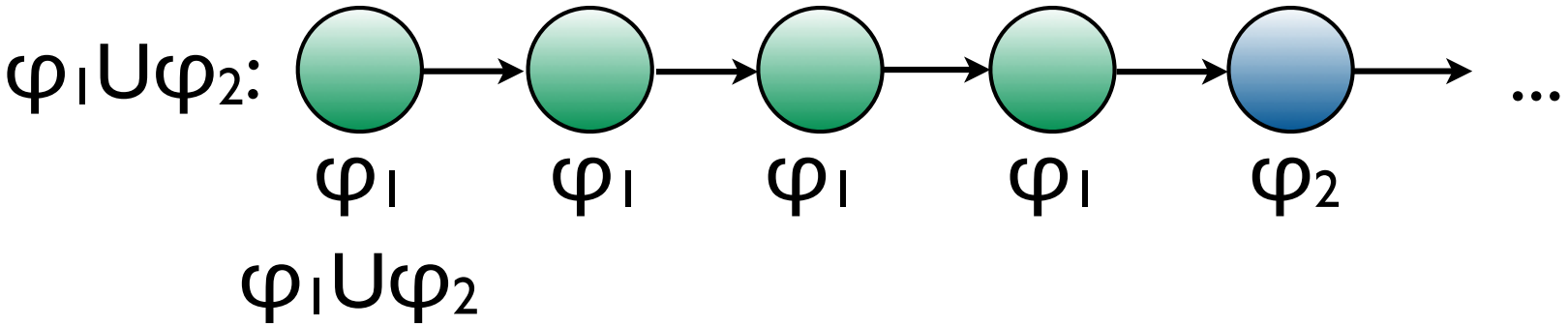


t,i ⊨ Gφ ssi pour tout j≥i, t,j⊨φ

Logique temporelle linéaire : LTL



Logique temporelle linéaire : LTL



t,i ⊨ $\varphi_1 U \varphi_2$ ssi il existe $j \geq i$, t,j ⊨ φ_2 et, pour tout $i \leq k < j$, t,k ⊨ φ_1

Logique temporelle linéaire : LTL

$$\varphi ::= p \in AP \mid \neg\varphi \mid \varphi \vee \varphi \mid X\varphi \mid F\varphi \mid G\varphi \mid \varphi U \varphi$$

$t,i \models p$ ssi $p \in t(i)$

$t,i \models \neg\varphi$ ssi $t,i \not\models \varphi$

$t,i \models \varphi_1 \vee \varphi_2$ ssi $t,i \models \varphi_1$ ou $t,i \models \varphi_2$

$t,i \models X\varphi$ ssi $t,i+1 \models \varphi$

$t,i \models F\varphi$ ssi il existe $j \geq i$ tel que $t,j \models \varphi$

$t,i \models G\varphi$ ssi pour tout $j \geq i$, $t,j \models \varphi$

$t,i \models \varphi_1 U \varphi_2$ ssi il existe $j \geq i$, $t,j \models \varphi_2$ et, pour tout $i \leq k < j$, $t,k \models \varphi_1$

Logique temporelle linéaire : LTL

$$\varphi ::= p \in AP \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid X\,\varphi \mid F\,\varphi \mid G\varphi \mid \varphi U \varphi$$

$\top \equiv p \vee \neg p$, pour $p \in AP$ quelconque

$\bot \equiv \neg \top$

$\varphi_1 \wedge \varphi_2 \equiv \neg(\neg\varphi_1 \vee \neg\varphi_2)$

$\varphi_1 \rightarrow \varphi_2 \equiv \neg\varphi_1 \vee \varphi_2$

Logique temporelle linéaire : LTL

En fait, Fφ et Gφ macros aussi :

- $F\varphi \equiv \top U \varphi$
- $G\varphi \equiv \neg F(\neg\varphi)$

Exercice : vérifier.

Logique temporelle linéaire : LTL

$$\varphi ::= p \in AP \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid X\,\varphi \mid \varphi U \varphi$$

- Autres macros utiles :
  - (Weak until) $\varphi_1 W \varphi_2 \equiv G\varphi_1 \vee \varphi_1 U \varphi_2$
  - (Release) $\varphi_1 R \varphi_2 \equiv \varphi_2 W (\varphi_1 \wedge \varphi_2) \equiv G\varphi_2 \vee \varphi_2 U (\varphi_1 \wedge \varphi_2)$

LTL : Exemples

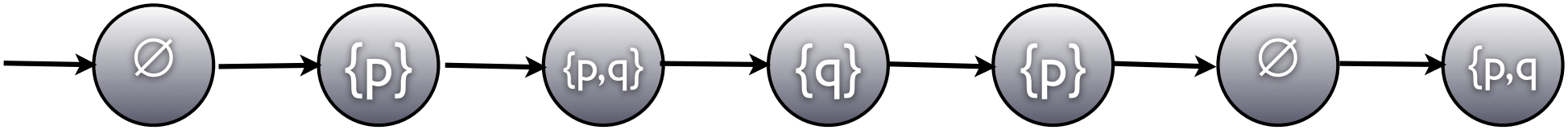
- Accessibilité : F (x=0)
- Invariance : G ¬(x=0)
- Vivacité : G(request → F response)
- Equité forte : GF enabled → GF scheduled
- Equité faible : FG enabled → GF scheduled
- Relâchement de contrainte : reset R alarm

LTL : Exercice 1

- Toute requête sera un jour satisfaite (AP = {requete, reponse})
- A chaque fois que de l'argent a été retiré, le code pin a été fourni (AP={cash-withdraw, pin-ok})
- Deux processus ne sont jamais en section critique en même temps (AP= {critique$_1$, critique$_2$})
- Si un processus demande l'accès en section critique, il l'obtiendra un jour (AP = {demande_crit, acc_crit})
- Une fois que le feu est vert, il ne peut pas devenir rouge immédiatement (AP= {vert, rouge})
- Lorsque le feu est rouge, il deviendra vert un jour
- Lorsque le feu est vert, il deviendra rouge un jour, après avoir été orange (AP= {vert, rouge, orange})

LTL : Exercice II

- Vérifier que $\neg X\varphi \equiv X\neg\varphi$, $\neg(\varphi_1 U \varphi_2) \equiv \neg\varphi_1 R \neg\varphi_2$
- Dites si, à chaque position de la trace ci-dessous, les propositions suivantes sont vérifiées : $p \wedge q$, $F(p \wedge q)$, pUq.



LTL : Exercice III

- Les équivalences suivantes sont-elles vraies?
  - $G(Fp \wedge Fq) \leftrightarrow GFp \wedge GFq$
  - $F(Gp \wedge Gq) \leftrightarrow FGp \wedge FGq$
  - $G(Fp \vee Fq) \leftrightarrow GFp \vee GFq$
  - $F(Gp \vee Gq) \leftrightarrow FGp \vee FGq$
  - $GF(p \wedge q) \leftrightarrow GFp \wedge GFq$
  - $GF(p \vee q) \leftrightarrow GFp \vee GFq$
  - $FG(p \wedge q) \leftrightarrow FGp \wedge FGq$
  - $FG(p \vee q) \leftrightarrow FGp \vee FGq$