Spécifier pour vérifier

INTRODUCTION
PROPRIÉTÉS DES SYSTÈMES
LOGIQUES TEMPORELLES
VÉRIFICATION

PROPRIÉTÉS

2

Motivations

- Nous avons besoin de nous assurer que les systèmes que nous utilisons sont exempts d'erreurs.
- Il nous faut donc vérifier qu'ils respectent certaines propriétés qui nous intéressent : « *la porte de l'ascenseur s'ouvrira un jour* »
- Comment classifier ces propriétés ?

Propriétés élémentaires des systèmes

- Accessibilité une situation particulière est atteinte
  - chaque processus peut entrer dans sa section critique,
  - On ne peut atteindre un état « crash ».
- Invariance Tous les états du système satisfont une certaine propriété :
  - Exclusion mutuelle.
  - Respect de pré-condition
  - Pas de division par zéro,

Propriétés générales des systèmes

- Sûreté (*safety*) Quelque chose de mauvais ne se produit pas (l'invariance est un cas particulier) :
  - *à tout moment il n'y a qu'un processus en section critique*,
  - *absence de blocage*,
  - *dans tout état, x est différent de zéro* (*invariance*).
- Vivacité (*liveness*) Quelque chose de bon se produira :
  - tout message envoyé est reçu.
  - Lorsqu'une impression est lancée, elle finira par s'achever

Propriétés générales des systèmes

- Équité (*fairness*) quelque chose aura lieu (ou n'aura pas lieu) infiniment souvent.
  - l'ordre des entrées dans la section critique respecte l'ordre des demandes,
  - Si un processus demande continuellement son exécution, il finira par l'avoir (équité faible)
  - Si un processus demande infiniment souvent son exécution, il finira par l'avoir (équité forte)
- Absence de blocage Le système ne se bloque pas :
  - Il n'y a pas d'état bloquant

Propriétés générales des systèmes

- Equivalence comportementale est-ce que deux systèmes se comportent de la même manière ?
  - l'ordre des entrées dans la section critique respecte l'ordre des demandes,
  - Si un processus demande continuellement son exécution, il finira par l'avoir (équité faible)
  - Si un processus demande infiniment souvent son exécution, il finira par l'avoir (équité forte)

MAIS COMMENT FORMALISER CES BESOINS ?

Motivations

- Rappelez vous de de l'exemple de l'ascenseur, parmi les propriétés souhaitées du système :
  - « *Tout appel à l'ascenseur doit finir par être satisfait* ».
- Cette propriété dynamique du système peut être formalisée en considérant la position du temps t et en adoptant la logique du premier ordre :
  - $\forall t, \forall n\ [appel(n,t) \Rightarrow \exists t'>t : descend(n,t')]$

Étage n ↗ ↖ Temps t

- Mais :
  - Notation lourde et difficile à écrire.
  - Vérification peu efficace.

LOGIQUES TEMPORELLES

10

Définition

- La logique temporelle est utilisée pour spécifier l'ordonnancement dans le temps.
- Enonce formellement des propriétés portant sur des exécutions du système (suite d'états).
- Livrée avec une syntaxe et une sémantique formelles (nécessaires pour les langages de spécification).
- Punelli en 1977 a proposé son utilisation pour la première fois afin de spécifier formellement des propriétés de systèmes.

Eléments

- Des propositions atomiques (AP) : *p* ,*q* , *vrai*, *faux*.
- Des connecteurs (opérateurs) logiques : $\neg$, $\vee$, $\wedge$, $\Rightarrow$, $\Leftrightarrow$.
- Connecteur temporels (modalités) exprimant l'enchainement des états le long d'une exécution : toujours, tant que, …(X, F, G) :
  - **X***p* (next *p*) : *p* est vérifié à l'état suivant.
  - **F***p* (futur) : *p* est vérifié dans le futur.
  - **G***p* (globally) : tout état vérifie *p*.

Exercices

- Formaliser les propriétés suivantes :
  - *P1* : « *Si nous sommes dans un état d'alerte, nous serons plus tard dans un état d'arrêt* ».
  - Nous désirons que *P1* soit toujours vérifiée.

Remarque

Temporelle ≠ Temporisé

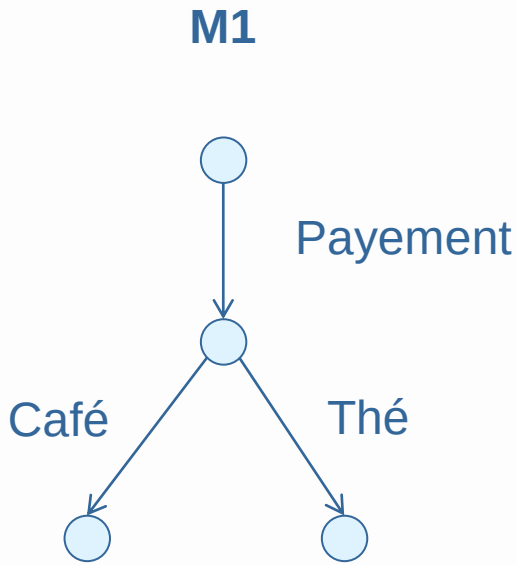
Ne quantifie pas l'écoulement du temps

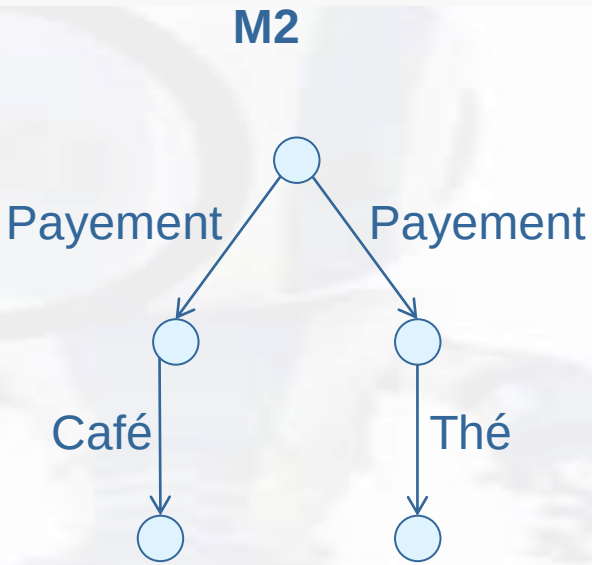
Quantifie l'écoulement du temps

Approches pour les logiques temporelles

- Etat ou action :
  - Logiques basées sur les états.
  - Logiques basées sur les actions.
- Temps linéaire ou arborescent :
  - Temps linaire (trace) : un système vérifie une formule de temps linéaire si tous ses comportements la vérifient (propriétés portant sur les chemins individuels).
  - Temps arborescent (arbre) : spécifient les propriétés directement sur le système de transitions (propriétés portant sur les arbres d'exécution).

Exemple : Machine à café

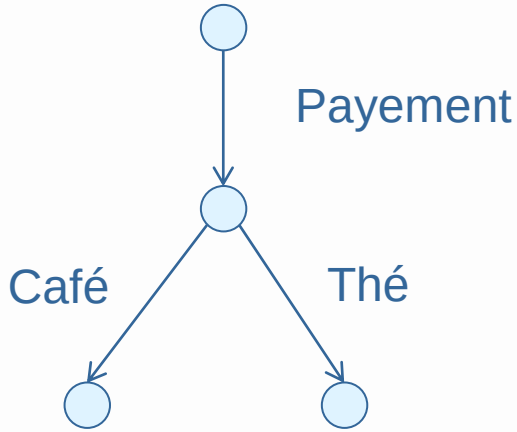


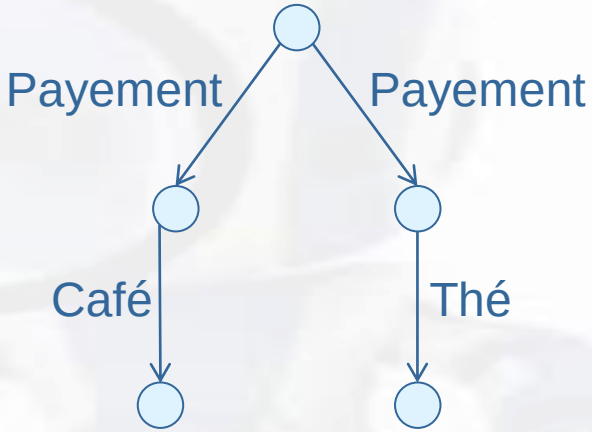


- Ces modèles sont équivalents en terme de séquences d'exécution {Payement.Café,Payement.Thé}, donc équivalent de point de vue logique temporelle linéaire.
- Mais leur interaction avec l'utilisateur est différente.

Exemple : Machine à café

M1



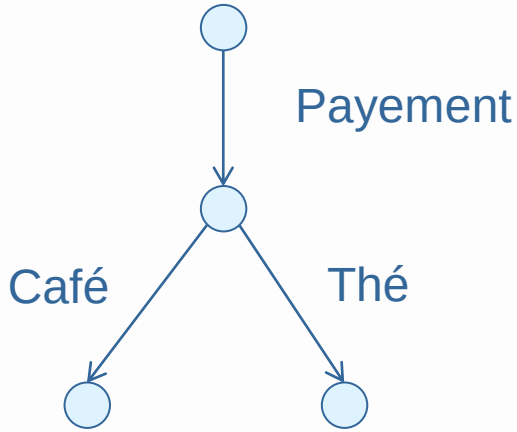


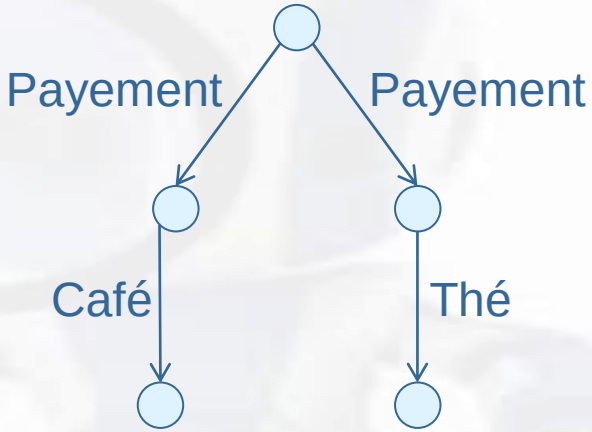
M2

- M1 permet à l'utilisateur de choisir la boisson après avoir payé.
- M2 choisit lui-même, de façon aléatoire, la boisson après le payement.
- La propriété : « *à chaque fois qu'un payement est fait, il est possible d'obtenir un café* » n'est pas exprimable en logique de temps linéaire.

Exemple : Machine à café

M1





M2

- La différence dans le comportement de ces deux modèle ne peut se voir que par le biais d'une logique de temps arborescent permettant un choix non déterministe après le payement.

LOGIQUES LTL (LINEAR TEMPORAL LOGIC)

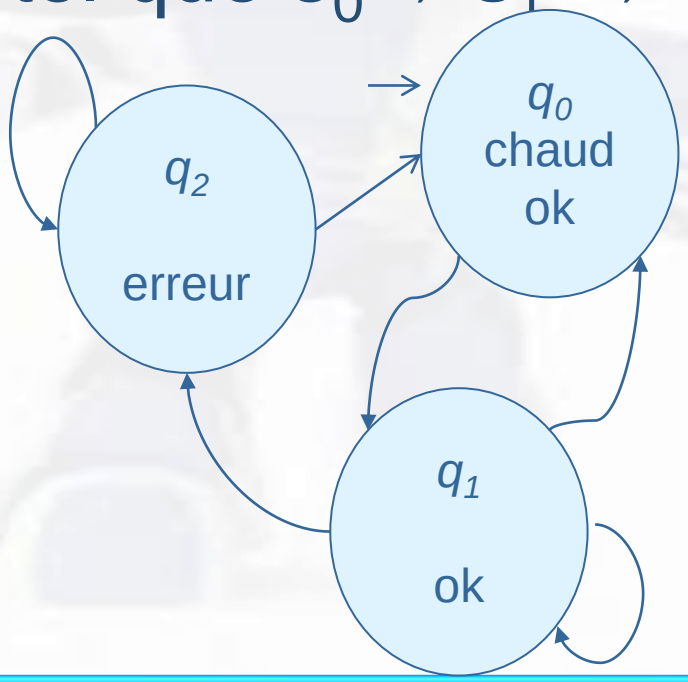
19

Rappel sur la sémantique d'une Structure de Kripke

- Etant donné un ensemble de variables propositionnelles *Prop*, les formules LTL expriment des propriétés sur 2^{Prop}.
- Une formule LTL est interprétée sur une structure de Kripke K : K satisfait une formule LTL ϕ si toutes les exécutions complètes (partielles et maximales) de T sont des modèles de ϕ.
- Rappelons :
  - Une exécution partielle est un chemin partant de l'état initial.
  - Une exécution maximale est une exécution qu'on ne peut pas prolonger (elle est soit infinie soit terminée dans un état duquel n'est issue aucune transition).

Rappel sur la sémantique d'une Structure de Kripke

- Un exécution (ou trace ou chemin) complète est une séquence initiale et maximale $s=s_0s_1s_2\ldots$ tel que $s_0 \rightarrow s_1 \rightarrow s_2\ldots$
- Exemples :
  - $\sigma_1 : q_0q_1q_0q_1q_0q_1 \ldots$
  - $\sigma_2 : q_0q_1q_2q_0q_1q_2 \ldots$
  - $\sigma_3 : q_0q_1q_2q_2q_2q_2 \ldots$



Syntaxe formelle de LTL

- $\phi ::= true | false | a \in AP \mid$ *propositions atomiques*

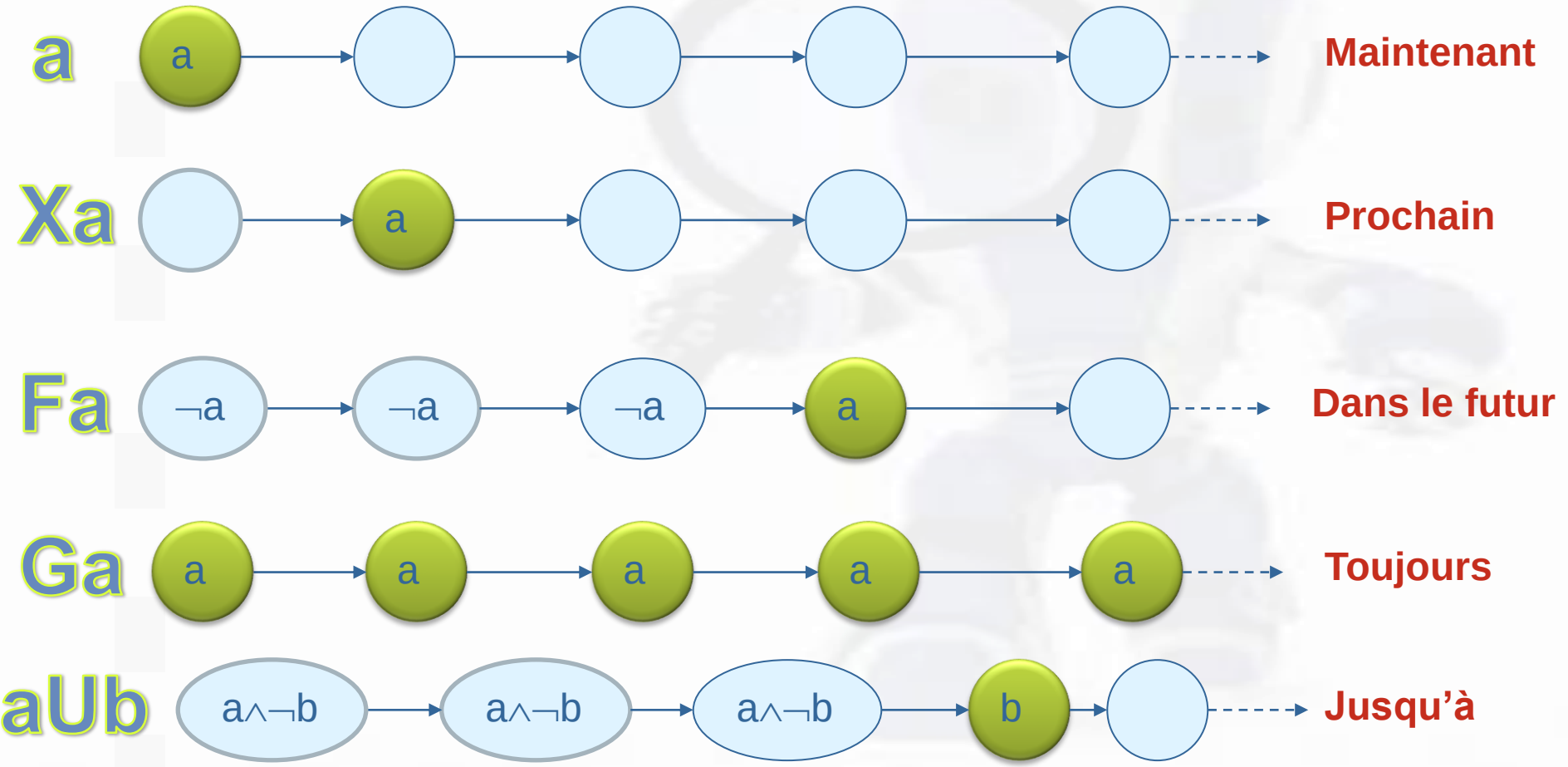
  $\neg\phi \mid \phi_1 \wedge \phi_2 | \phi_1 \vee \phi_2 | \phi_1 \Rightarrow \phi_2 |$ *combinateurs booléens*

  $\mathbf{X}\phi \mid \mathbf{F}\phi | \mathbf{G}\phi | \phi_1 \mathbf{U} \phi_2$ *opérateurs temporels*

- **X** (next) est souvent noté ◯
- **F** est souvent noté ◊
- **G** est souvent noté □
- La logique LTL considère le modèle comme un ensemble d'exécutions individuelles où chaque état a un unique successeur.

Sémantique informelle

V={a,b}
L'état courant est le premier état

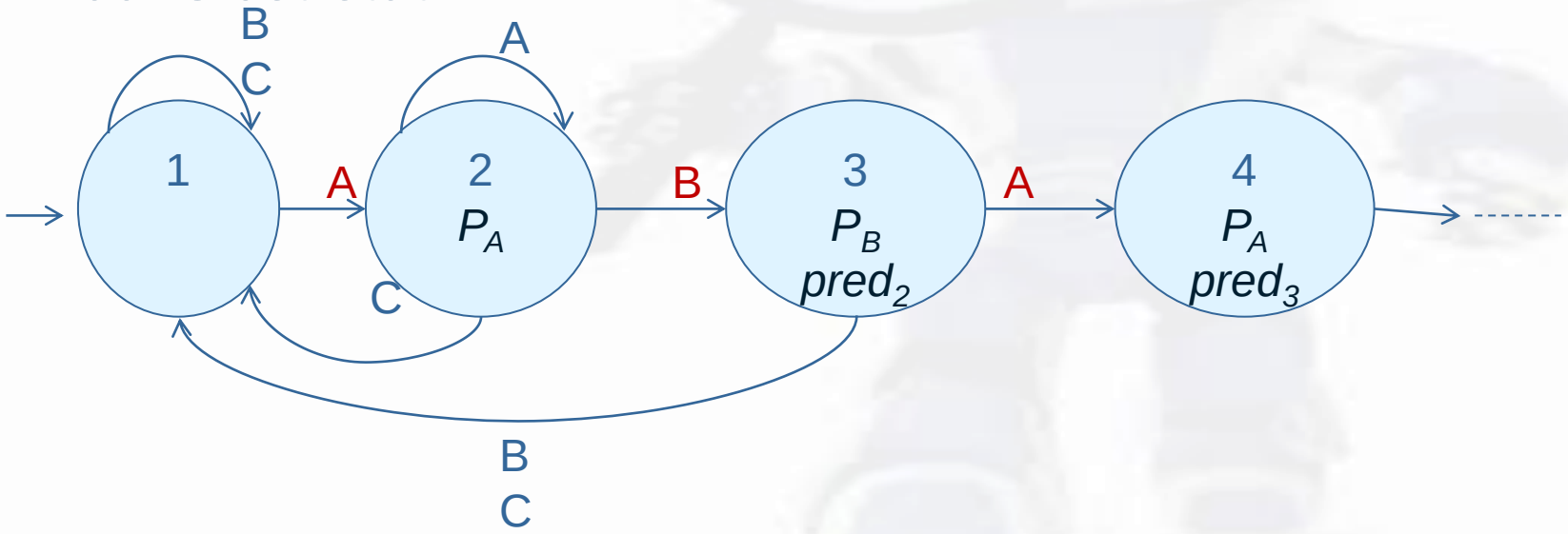


Sémantique formelle (notation)

- Soit A une structure de Kripke A= (Q,q_0,E,T,$Prop$,l).
- Une formule LTL est interprétée (par la relation de satisfaction l=) sur un triplet (A,σ,i).
- On écrira $A, \sigma, i \models \phi$ pour signifier qu'à l'instant i de l'exécution σ de A, ϕ est vraie.
- A est généralement omis.

Rappel sur I

- Rappelez vous de l'application I, définie comme élément d'un structure de Kripke et qui associe à tout élément de A l'ensemble fini des propriétés élémentaires vérifiables dans cet état.



- $I = \{1 \rightarrow \varnothing, 2 \rightarrow \{P_A\}, 3 \rightarrow \{P_B, pred_2\}, 4 \rightarrow \{P_A, pred_3\}\}$

Sémantique formelle

- La sémantique formelle est définie inductivement sur la structure de ϕ.

Sémantique formelle (LTL)

$\sigma, i \vDash true$

$\sigma, i \nvDash false$

$\sigma, i \vDash a \; ssi \; a \in l(\sigma(i))$

$\sigma, i \vDash \neg\phi \; ssi \; \sigma, i \nvDash \phi$

$\sigma, i \vDash \phi \vee \phi' ssi \; \sigma, i \vdash \phi \; ou \; \sigma, i \vdash \phi'$

$\sigma, i \vDash \phi \wedge \phi' ssi \; \sigma, i \vdash \phi \; et \; \sigma, i \vdash \phi'$

$\sigma, i \vDash \mathbf{X}\phi \; ssi \; i < |\sigma| \; et \; \sigma, i+1 \vDash \phi$

$\sigma, i \vDash \mathbf{F}\phi \; ssi \; \exists j | i \leq j \leq |\sigma| \; et \; \sigma, j \vDash \phi$

$\sigma, i \vDash \mathbf{G}\phi \; ssi \; \forall j | i \leq j \leq |\sigma| \; on \; a \; \sigma, j \vDash \phi$

$\sigma, i \vDash \phi \mathbf{U} \phi' \; ssi \; \exists j | i \leq j \leq |\sigma| \; et \; \sigma, j \vDash \phi'$

$et \; \forall k | i \leq k < j \; on \; a \; \sigma, k \vDash \phi$

Généralisation de la satisfaction

- Dans le cas général, notons une exécution par σ, on dit que le système A satisfait une formule ϕ :
  - $A \models \phi$ *ssi* $\sigma, 0 \models \phi \; \forall \sigma \; de \; A$
- On peut aussi vérifier si un état $s \in Q$ de A satisfait une formule ϕ (ϕ est vraie à cet état) :
  - $s \models \phi$ ssi pour toutes les exécutions σ commençant à l'état s : $\sigma \models \phi$.

Exemples de formules LTL

- P1 : *Si nous sommes dans un état alerte, nous serons plus tard dans un état d'arrêt* :
  - alerte $\Rightarrow$**F**arrêt
- P 2 : *La propriété P1 est toujours vérifiée* :
  - **G**(Alerte $\Rightarrow$**F**arrêt)
- P3 : *A partir d'une alerte, l'alarme est en marche jusqu'à l'arrêt qui suivra forcément :*
  - **G**(alerte $\Rightarrow$(alarme**U**arrêt))

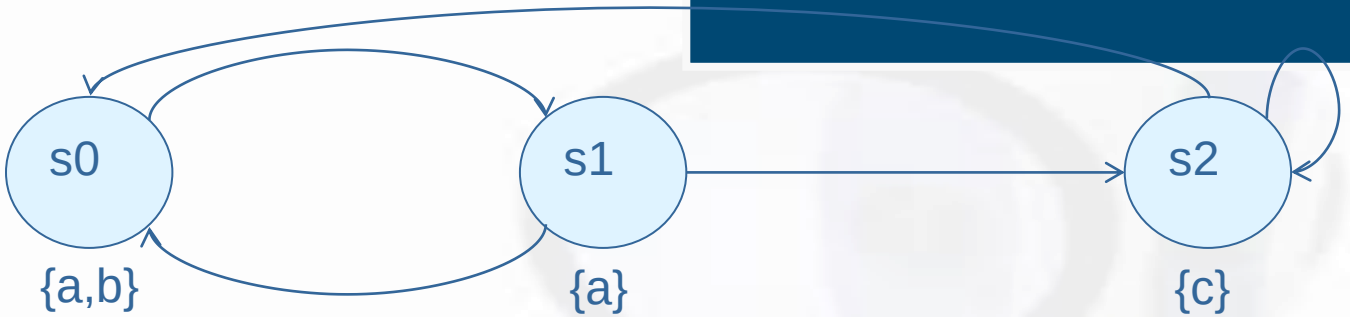
Combinaison des modalités

- **GF**ϕ : toujours il y aura un jour un état tel que ϕ (ϕ est vérifiée un nombre infini de fois le long d'une exécution).
- **FG**ϕ : tout le temps à partir d'un certain moment.
- Il existe un opérateur **W** dénotant un *jusqu'à faible* (on n'exige pas que ψ finisse par avoir lieu et si ψ n'a jamais lieu, φ reste vraie jusqu'à la fin) : $\varphi \mathbf{W} \psi \equiv (\varphi \mathbf{U} \psi) \vee \mathbf{G} \varphi$

Combinaison des modalités

- L'opérateur $\psi \mathbf{R} \varphi$ (*release*) : φ est vraie jusqu'à ce que ψ devienne vraie (φ doit être vraie également dans l'état où ψ est vrai). Si ψ ne devient jamais vraie, φ restera vrai pour toujours :
  - $\varphi \mathbf{R} \psi \equiv \psi \vee \mathbf{W}(\varphi \wedge \psi)$

Application



- Soient les exécutions :
  - $\sigma_1 : (s_0 s_1)^n, \sigma_2 : (s_0 s_1 s_2)^m$, $\sigma_3 : s_0 s_1 s_2^n$.
- A-t-on :
  - $\sigma_1, 0 \models a \wedge b$
  - $\sigma_2, 0 \models a \wedge b$
  - $\sigma_3, 0 \models a \wedge b$
  - $\sigma_1, 0 \models Xa$
  - $\sigma_2, 0 \models Xa$
  - $\sigma_3, 0 \models Xa$
  - $\sigma_1, 0 \models XXc$
  - $\sigma_2, 0 \models XXc$
  - $\sigma_3, 0 \models XXc$
  - $\sigma_1, 0 \models XXXc$
  - $\sigma_2, 0 \models XXXc$
  - $\sigma_3, 0 \models XXXc$
  - $\sigma_1, 0 \models XXc \vee XXXa$
  - $\sigma_2, 0 \models XXc \vee XXXa$
  - $\sigma_3, 0 \models XXc \vee XXXa$

Equivalences importantes

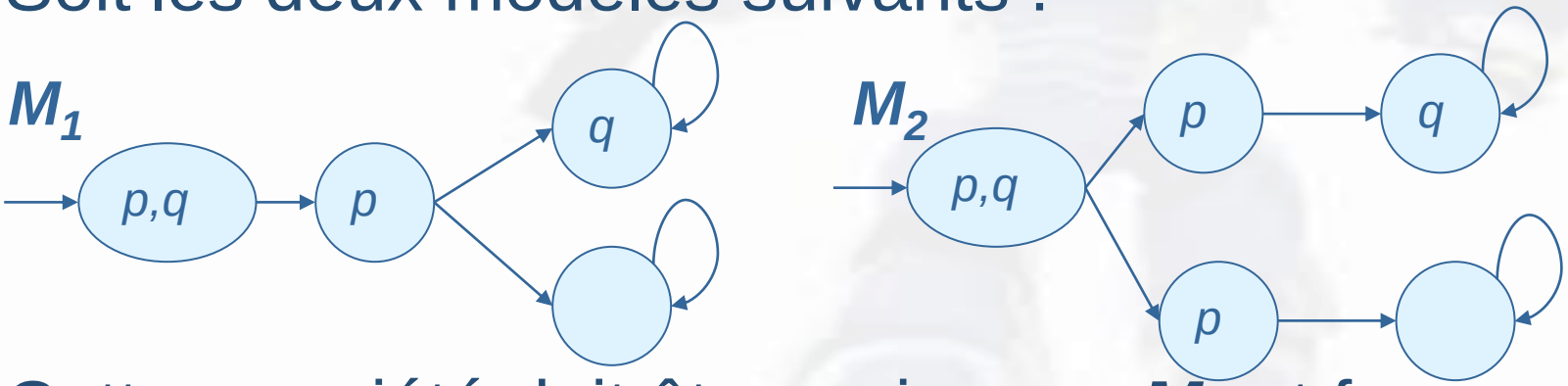
- Deux formules LTL φ et ψ sont dites sémantiquement équivalentes, et on écrit $\varphi \equiv \psi$, ssi pour toute exécution σ et pour tout modèle A :
  - $A,\sigma \models \varphi$ ssi $A,\sigma \models \psi$

Equivalences importantes

- $\neg(\varphi \wedge \phi) \equiv \neg\varphi \vee \neg\phi$
  $\neg(\varphi \vee \phi) \equiv \neg\varphi \wedge \neg\phi$
- $\neg \mathbf{X}\varphi \equiv \mathbf{X}\neg\varphi$
- $\neg \mathbf{G}\varphi \equiv \mathbf{F}\neg\varphi$
  $\neg \mathbf{F}\varphi \equiv \mathbf{G}\neg\varphi$
- $\neg(\varphi \mathbf{U} \phi) \equiv \neg\varphi \mathbf{R} \neg\phi$
  $\neg(\varphi \mathbf{R} \phi) \equiv \neg\varphi \mathbf{U} \neg\phi$
- $\mathbf{F}\varphi \equiv \top \mathbf{U} \varphi$
- $\mathbf{F}\varphi \equiv \top \mathbf{U} \varphi$
- $\mathbf{G}\varphi \equiv \bot \mathbf{R} \varphi \equiv \neg(\top \mathbf{U} \neg\varphi)$

LTL : expressivité

- Supposant qu'il existe une formule LTL exprimant la propriété P : « *A chaque fois que p est vraie , il est possible d'atteindre un état dans lequel q est vrai* ».
- Soit les deux modèles suivants :



- Cette propriété doit être vraie pour M_1 et fausse pour M_2

LTL : expressivité

- De point de vue LTL, ces deux automates correspondent à un même ensemble de chemins :
  - $\sigma_1 = \{p,q\}.\{p\}.\{q\}.\ \{q\}.\ \{q\}\ldots\ldots$
  - $\sigma_2 = \{p,q\}.\{p\}\ldots\ldots$
- Donc si la propriété *P* est vraie pour l'un elle est nécessairement vraie pour l'autre.
- On peut donc conclure qu'il existe des propriétés non exprimables en LTL.

Propriétés non exprimables en LTL

- P1 : « *Il est possible que p devienne vrai dans le futur* » : il existe une exécution du modèle pour laquelle *p* sera vraie (Notez que **F***p* indique que p deviendra vraie pour toutes les exécutions).
- P2 : « *pour toutes les exécutions du modèle, dès lors que alerte est vraie, il existe des exécutions dans lesquelles action devienne vraie* ».

Conclusion sur la logique LTL

- La logique LTL permet d'exprimer des propriétés qui s'intéressent à l'ensemble des exécutions et pas à la façon dont elles sont organisées en arbre : elle exprime des formules sur les chemins.
- LTL ne permet pas d'exprimer les différentes possibilités d'évolution d'un système à partir d'une configuration.
- LTL ne permet pas de discerner entre l'atteignabilité potentielle de l'atteignabilité inévitable.

PTLT

- La logique PLTL est une logique LTL augmentée par des opérateurs du passé : **Y** et **S** :
  - **Y**φ : hier φ (*yesterday*)
  - φ**S**ψ : φ depuis ψ (*since*).

LOGIQUES CTL (COMPUTATION TREE LOGIC)

40

Introduction

- CTL est une logique temporelle arborescente qui a été proposée en 1981 par E.M. Clarke & E.A. Emmerson.
- Permet d'exprimer des propriétés sur des arbres d'exécution.
- La vérification de propriétés exprimées en CTL se fait en temps polynomial en la taille de la formule : elle est P-complète.

Syntaxe informelle

- La différence avec LTL réside dans l'apparition de nouveaux opérateurs **A** et **E** temporels de quantification (de chemin) sur les exécutions (traces).
- **A**φ : (*all*) énonce que toutes les exécutions partant de l'état courant satisfont φ.
- **E**φ : (*exist*) à partir de l'état courant, il existe une exécution satisfaisant φ.

Syntaxe formelle

- $\phi ::= \mathit{true} | \mathit{flase} | a \in AP \mid$ *propositions atomiques*

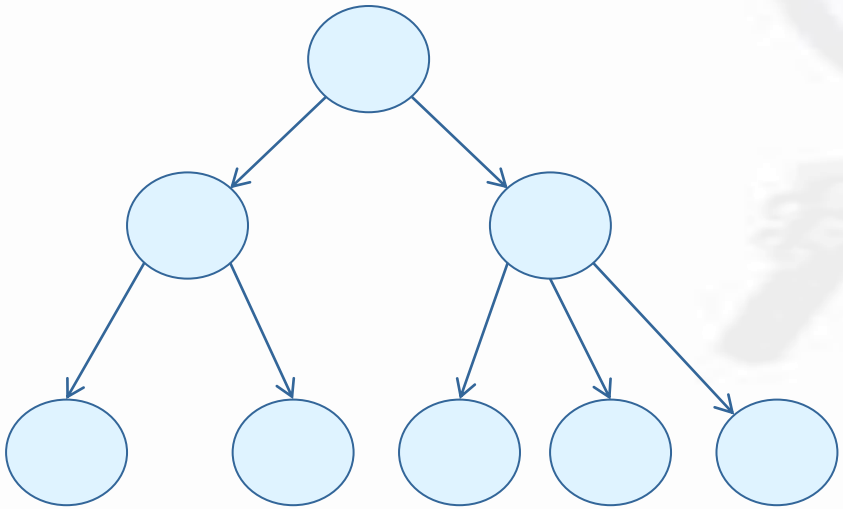
  $\neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid$ *combinateurs booléens*

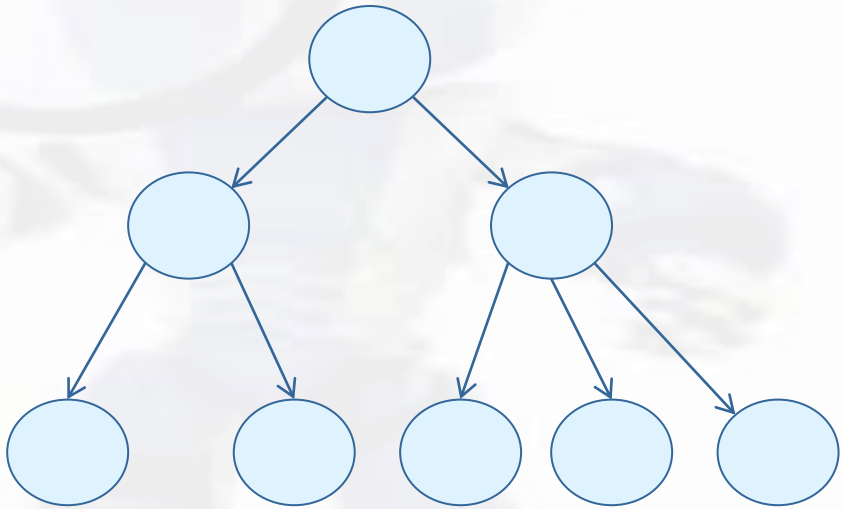
  $\varphi_1 \vee \varphi_2 \mid \varphi_1 \Rightarrow \varphi_2 \mid$

  $\mathbf{EX}\ \varphi \mid \mathbf{EF}\varphi \mid \mathbf{EG}\varphi \mid \varphi\mathbf{EU}\psi \mid$ *opérateurs temporels*

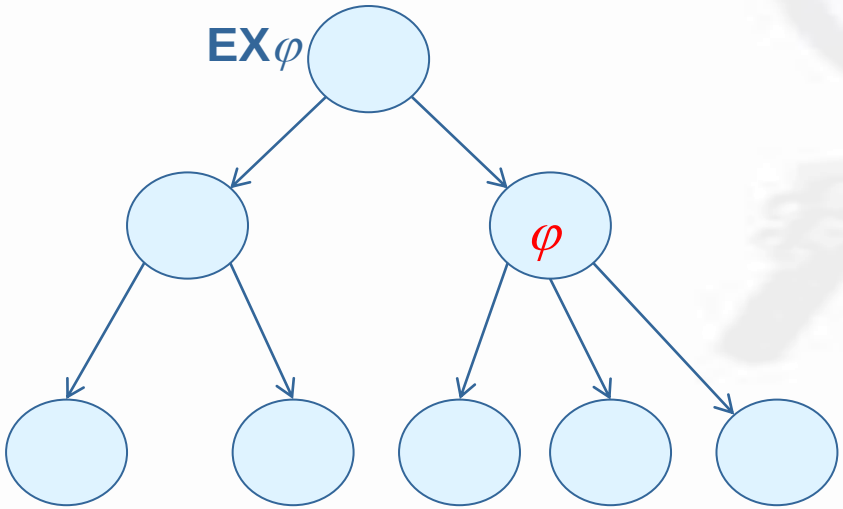
  $\mathbf{AX}\ \varphi \mid \mathbf{AF}\varphi \mid \mathbf{AG}\varphi \mid \varphi\mathbf{AU}\psi$
- **A** est souvent noté $\forall$
- **E** est souvent noté $\exists$
- Une propriété CTL s'interprète sur les états du système : chaque état peut avoir plusieurs successeurs.

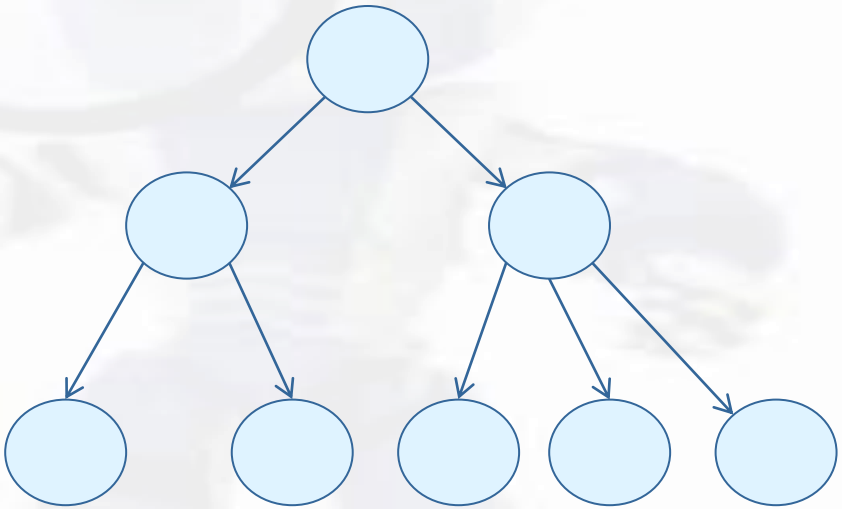
Sémantique informelle



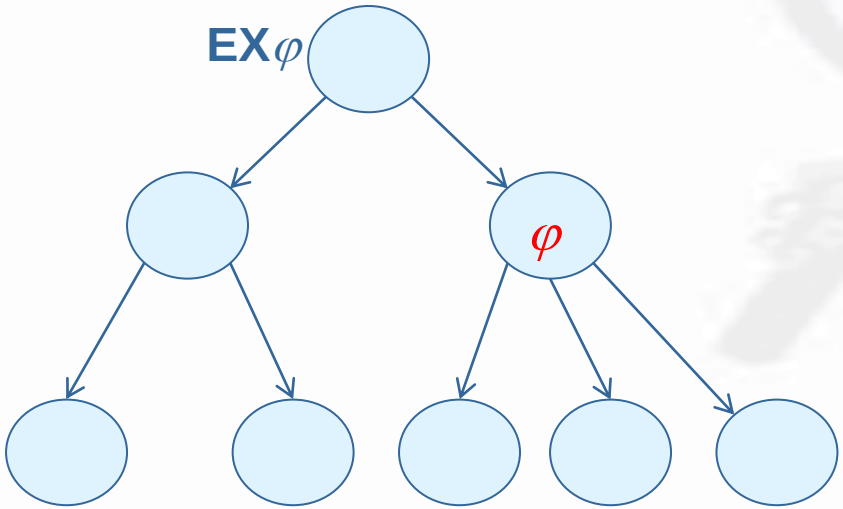


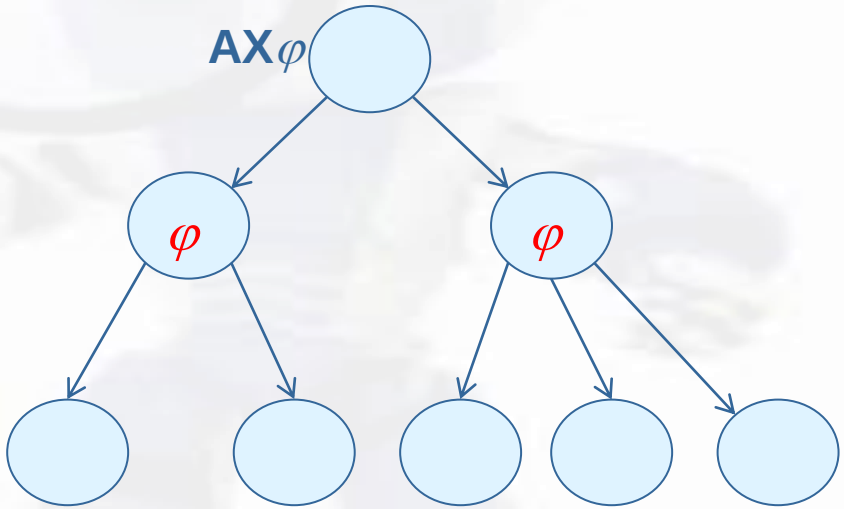
Sémantique informelle



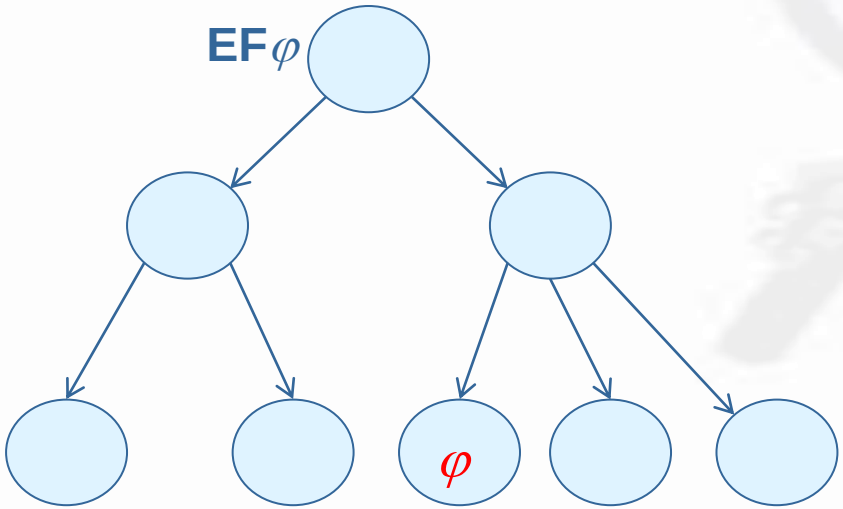


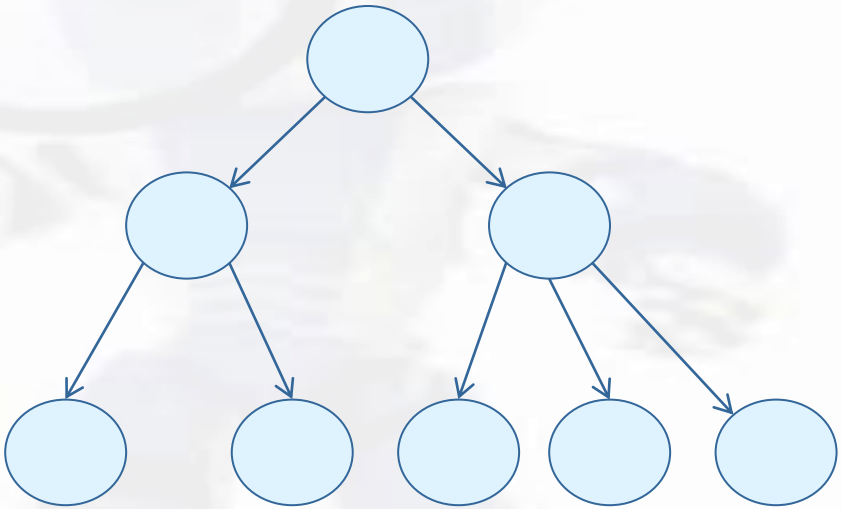
Sémantique informelle



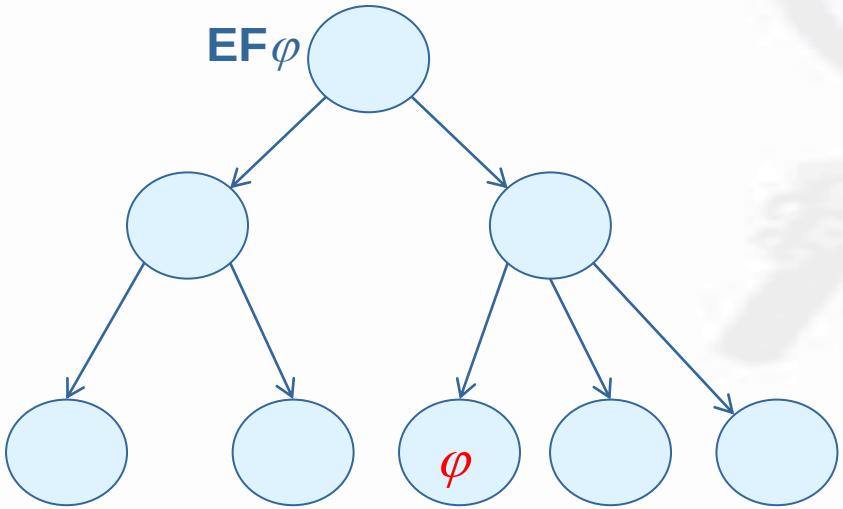


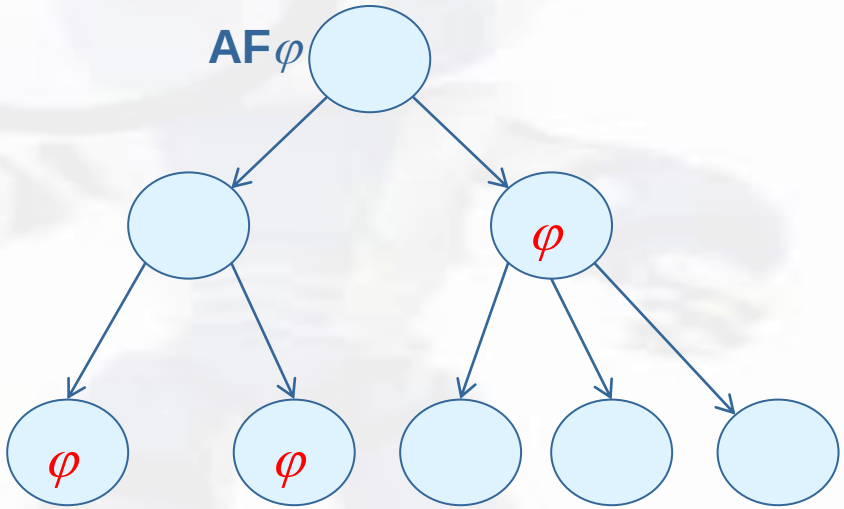
Sémantique informelle



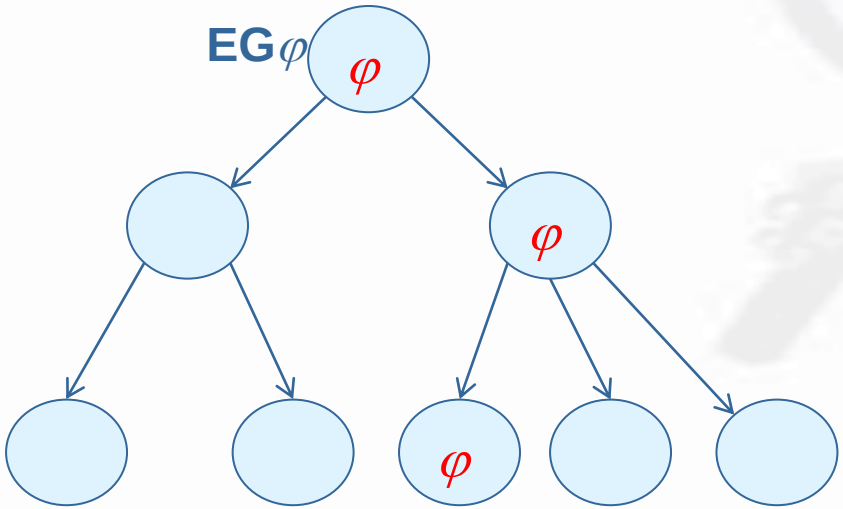


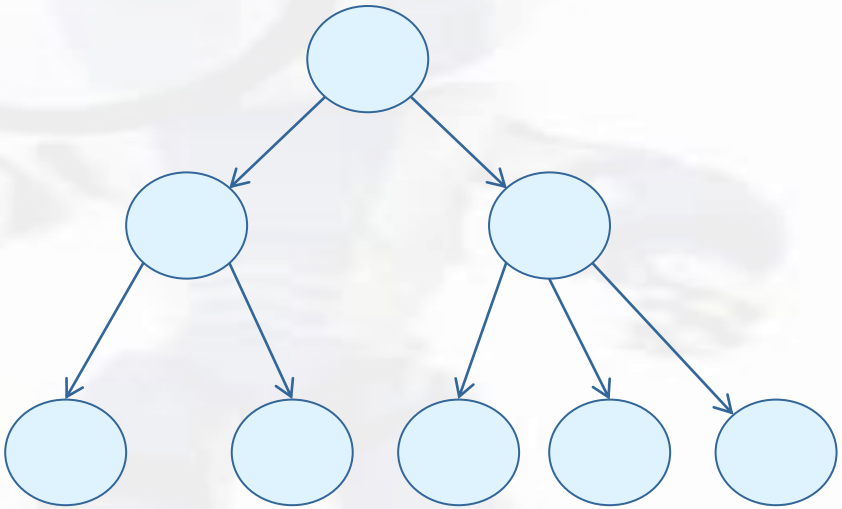
Sémantique informelle



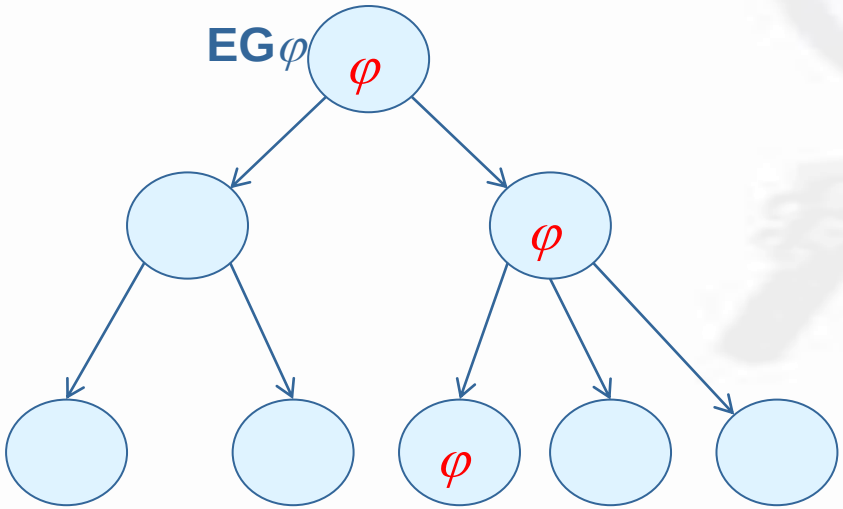


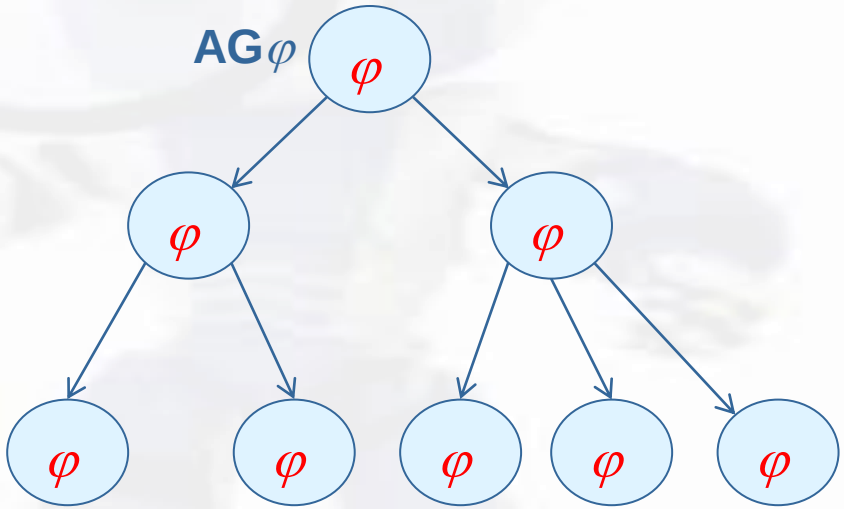
Sémantique informelle



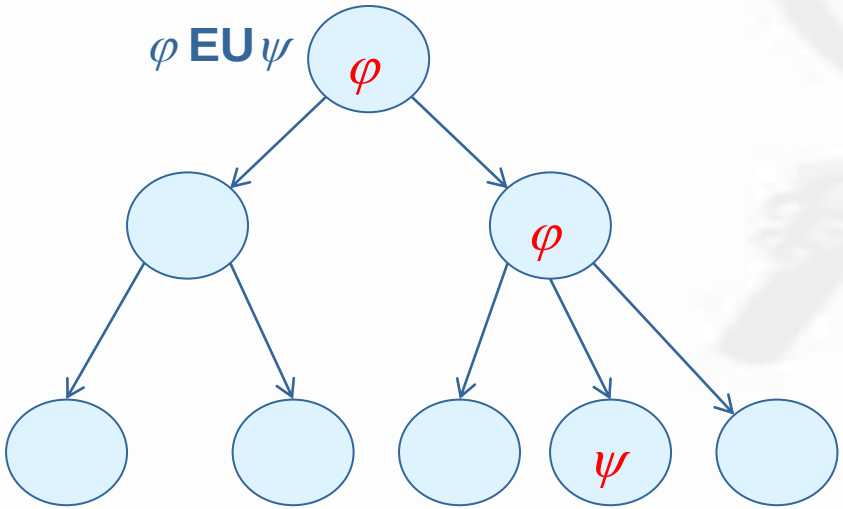


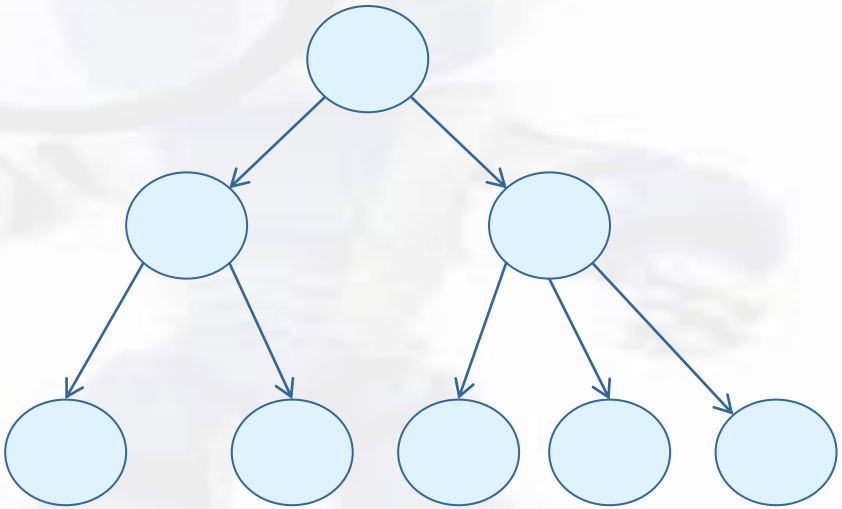
Sémantique informelle



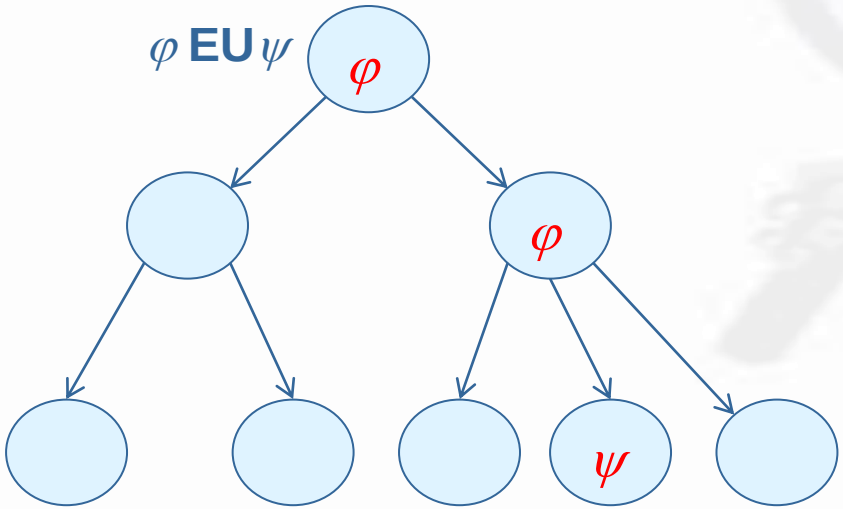


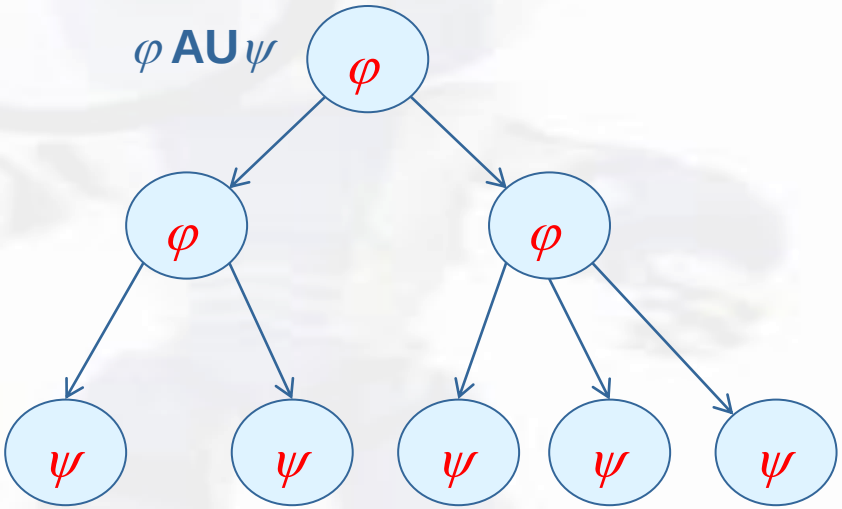
Sémantique informelle





Sémantique informelle





Opérateurs de chemin VS opérateurs d'état

- Les connecteurs A et G sont différents :
  - **A**φ exprime que toutes les exécutions possibles à partir de l'état courant satisfont φ.
  - **G**φ exprime que φ est vraie dans tous les états de l'exécution considérée.
- Les connecteurs **E** et **F** sont différents :
  - **E**φ exprime l'existence d'une exécution satisfaisant φ.
  - **F**φ exprime que φ est satisfaite au moins une fois le long d'une exécution donnée.

Opérateurs de chemin VS opérateurs d'état

- Les opérateurs **A** et **E** introduisent alors de nouvelles formules appelées des formules d'états appliquées à des formules de chemins (introduites par **F** et **G**).
- On interprète donc les formules CTL sur un état de l'automate et non une exécution.

Sémantique formelle

- Soit K une structure de Kripke K= (Q,q_0,E,T,$Prop$,l).
- Une formule CTL est interprétée (par la relation de satisfaction ⊨) sur une paire (K,q) : K une structure de Kripke et q un état de Q.
- Si $q \in Q$ et φ une formule de CTL :
  - K, $q \models \varphi$: *φ est vraie à l'état q.*
  - K $\models \varphi$ ssi K, $q_0 \models \varphi$.

Sémantique formelle de CTL

- On défini la relation de satisfaction inductivement sur la structure de ϕ comme suit :

$$K, q \models true$$

$$K, q \not\models false$$

$$K, q \models a \; ssi \; a \in l(q)$$

$$K, q \models \neg\phi \; ssi \; K, q \not\models \phi$$

$$K, q \models \phi \vee \phi' ssi \; K, q \models \phi \; ou \; K, q \models \phi'$$

$$K, q \models \phi \wedge \phi' ssi \; K, q \models \phi \; et \; K, q \models \phi'$$

Sémantique formelle de CTL

$$K, q \vDash \boldsymbol{EX}\phi \; ssi \; \exists\sigma | \sigma(0) = q \; et \; K, \sigma(1) \vDash \phi$$

$$K, q \vDash \boldsymbol{EF}\phi \; ssi \; \exists\sigma | \sigma(0) = q \; et \; \exists i | \; K, \sigma(i) \vDash \phi$$

$$K, q \vDash \boldsymbol{EG}\phi \; ssi \; \exists\sigma | \sigma(0) = q \; et \; \forall i \; on \; a \; K, \sigma(i) \vDash \phi$$

$$K, q \vDash \phi \boldsymbol{EU} \psi \; ssi \; \exists\sigma | \sigma(0) = q \; et \; \exists i \; tel \; que$$

$$((\forall j < i \; on \; a \; K, \sigma(j) \vDash \phi) \; et \; K, \sigma(i) \vDash \psi)$$

$$K, q \vDash \boldsymbol{AX}\phi \; ssi \; \forall\sigma | \sigma(0) = q \; on \; a \; K, \sigma(1) \vDash \phi$$

$$K, q \vDash \boldsymbol{AF}\phi \; ssi \; \forall\sigma | \sigma(0) = q, \exists i \; tel \; que \; K, \sigma(i) \vDash \phi$$

$$K, q \vDash \boldsymbol{AG}\phi \; ssi \; \forall\sigma | \sigma(0) = q, \forall i : \; K, \sigma(i) \vDash \phi$$

$$K, q \vDash \phi \boldsymbol{AU} \psi \; ssi \; \forall\sigma | \sigma(0) = q, \exists i \; tel \; que$$

$$((\forall j < i \; on \; K, \sigma(j) \vDash \phi) \; et \; K, \sigma(i) \vDash \psi))$$

Formules équivalentes

- Deux formules CTL φ et ψ sont dites sémantiquement équivalentes, et on écrit $\varphi \equiv \psi$, ssi pour tout état q et pout tout modèle K :
  - $K,q \models \varphi$ ssi $K,q \models \psi$

Equivalences connues

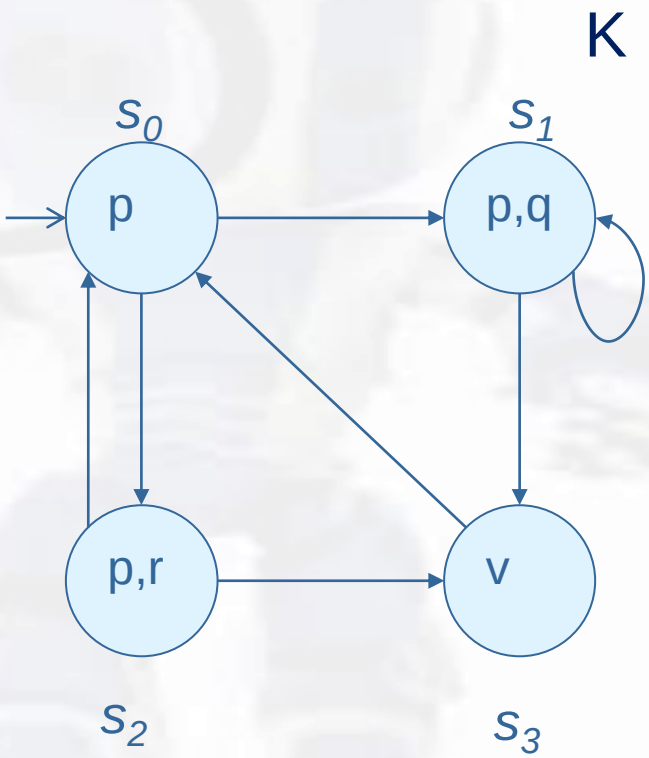
$$AX(\phi \wedge \psi) \equiv AX\phi \wedge AX\psi$$
$$EX(\phi \vee \psi) \equiv EX\phi \vee EX\psi$$
$$\neg AX\phi \equiv EX\neg\phi$$

$$EF(\phi \vee \psi) \equiv EF\phi \vee EF\psi$$
$$AG(\phi \wedge \psi) \equiv AG\phi \wedge AG\psi$$
$$\neg AF\phi \equiv EG\neg\phi$$
$$\neg EF\phi \equiv AG\neg\phi$$

$$AFAF\phi \equiv AF\phi$$
$$EFEF\phi \equiv EF\phi$$
$$AGAG\phi \equiv AG\phi$$
$$EGEG\phi \equiv EG\phi$$

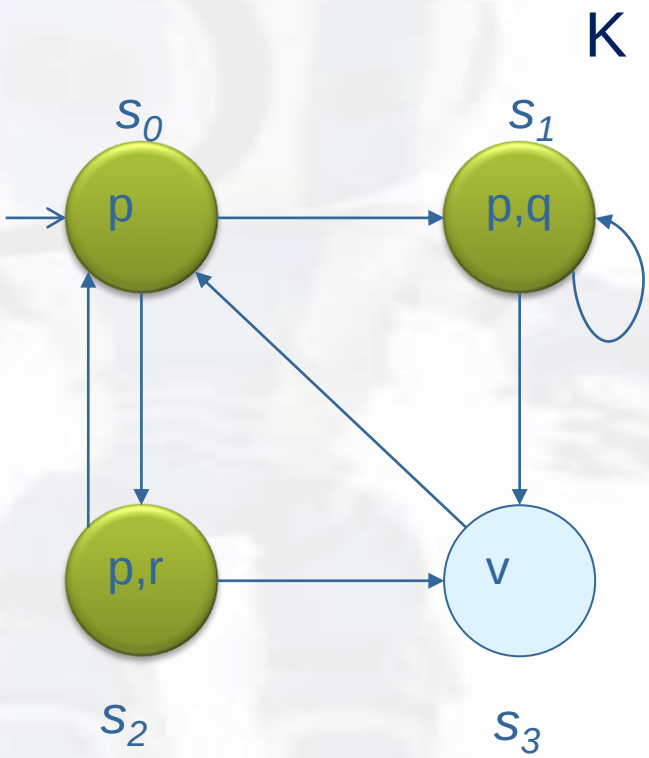
Exemples

- $K, s_0 \stackrel{?}{\models} \mathbf{AX}p$



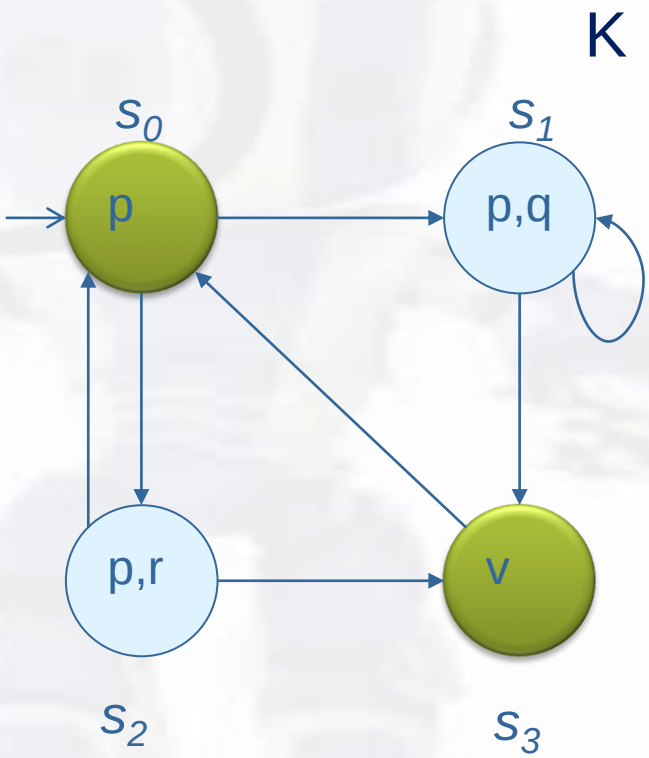
Exemples

- s_i tel que $K, s_i \vDash p$



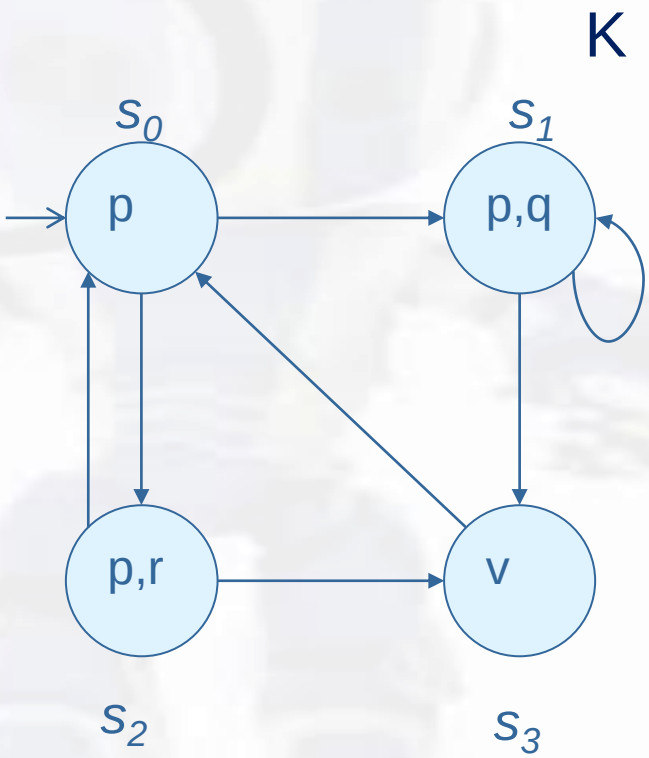
Exemples

- s_i tel que $K,s_i \vDash$ **AX**p
- Les états qui satisfont **AX**p sont $\{s_0, s_3\}$.
- Donc $K,s_0 \vDash$ **AX**p



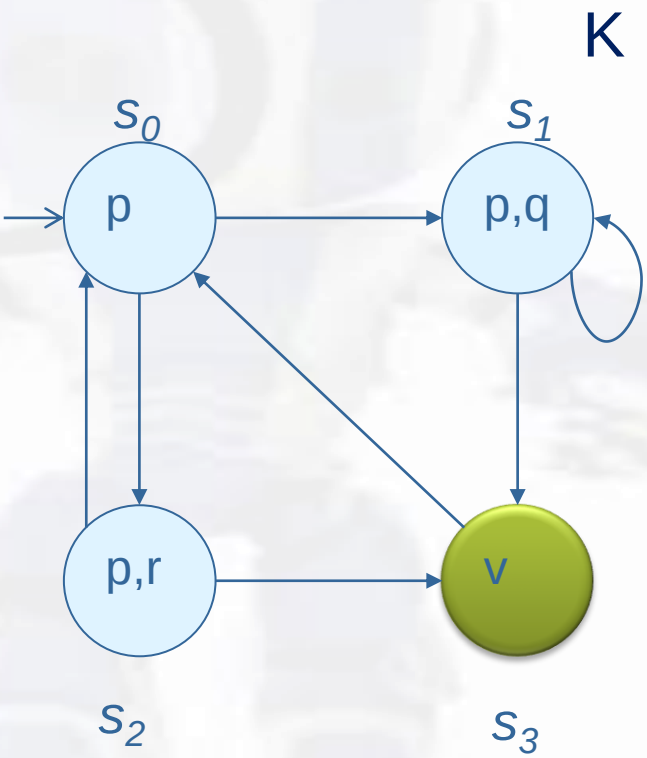
Exemples

- $K, s_0 \stackrel{?}{\models} \mathbf{EF}v$



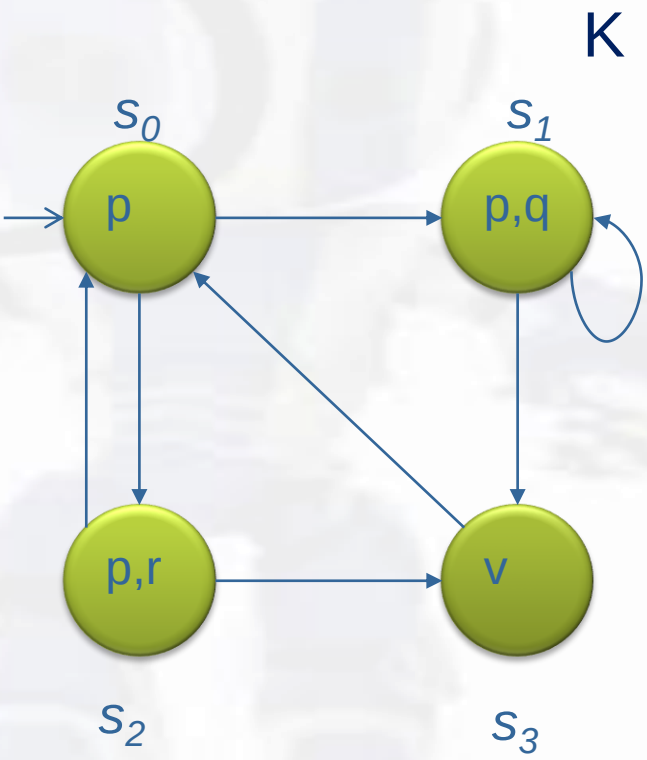
Exemples

- s_i tel que $K, s_i \models v$



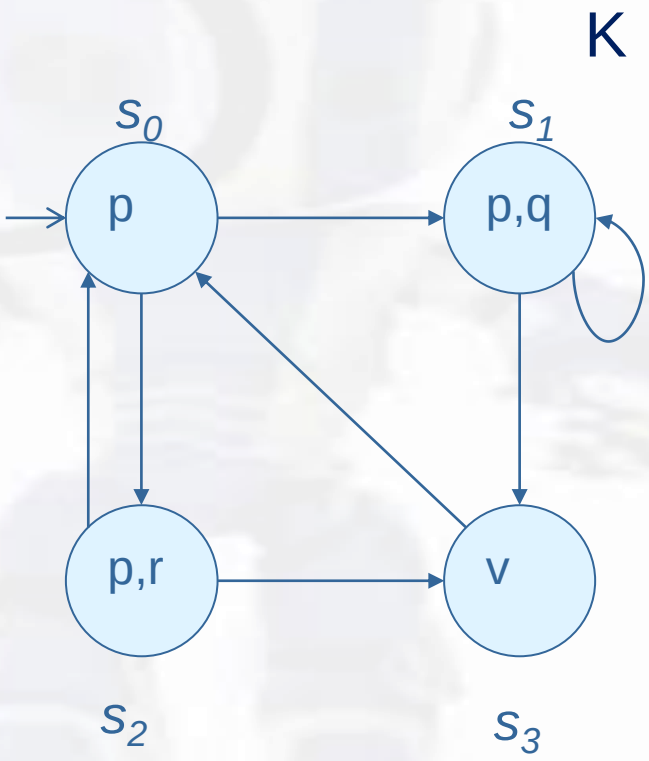
Exemples

- s_i tel que $K,s_i \models$ **EF**v
- Les états qui satisfont **EF**v sont $\{s_0,s_1,s_2, s_3\}$.
- Donc $K,s_0 \models$ **EF**v



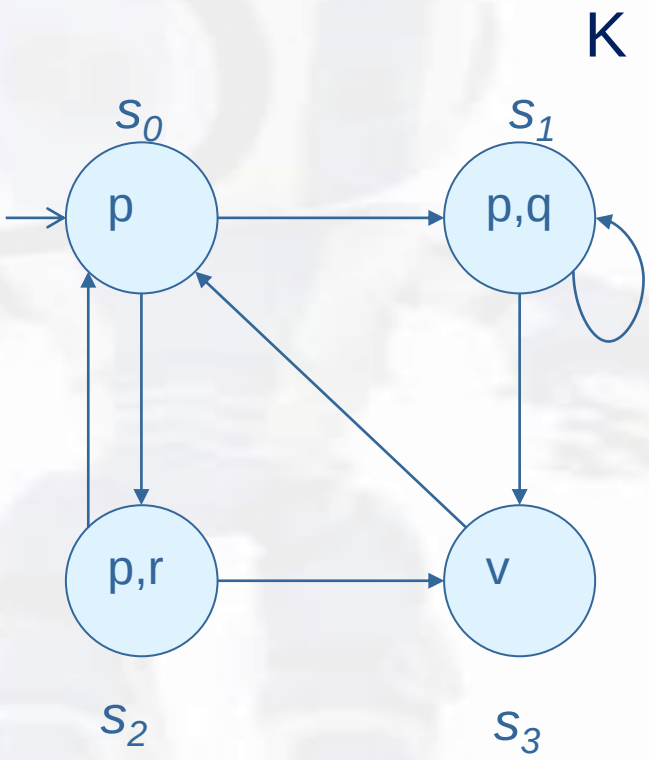
Exemples

- $K, s_0 \models \mathbf{AG}(p \vee v)$



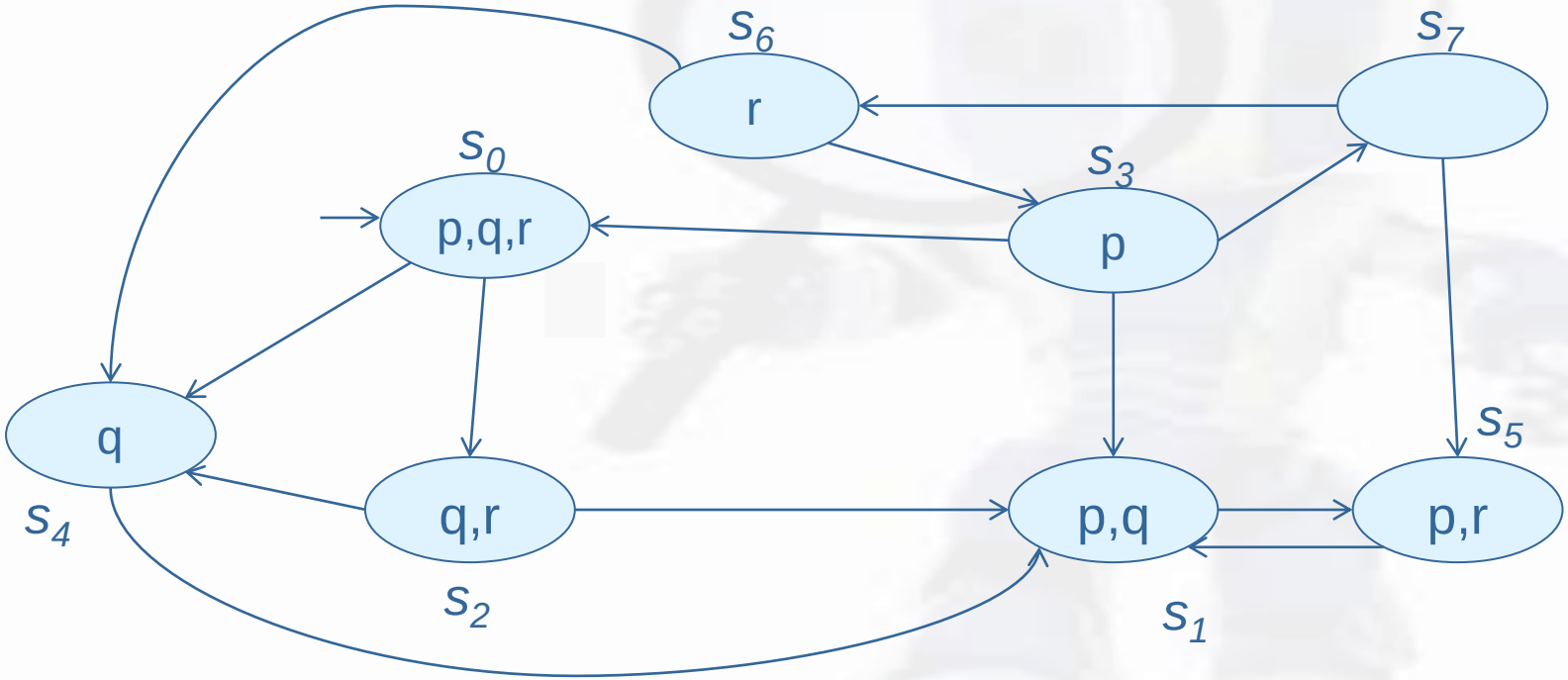
Exemples

- $K, s_0 \models p\mathbf{EU}v$



Application

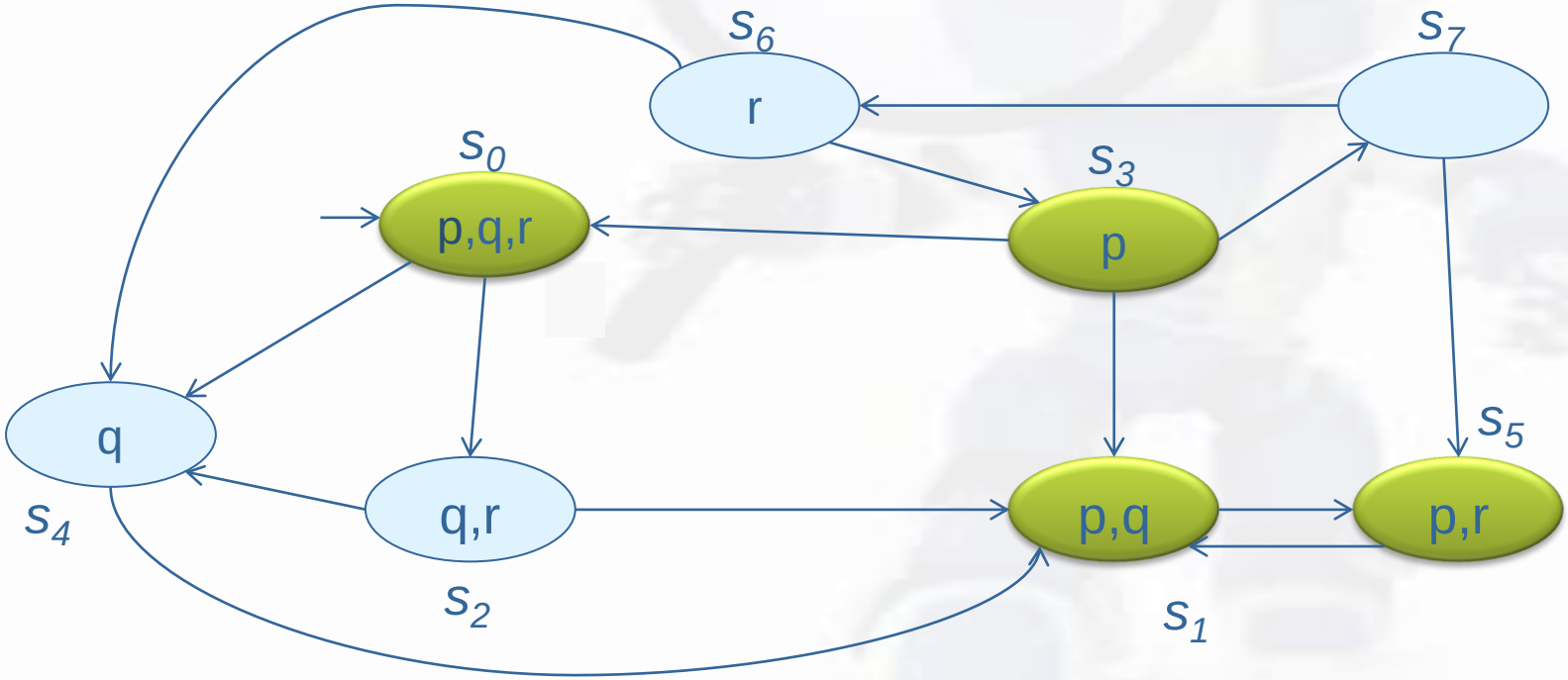
- Soit la structure de Kripke K :



- A-t-on K,s_0 ⊨ **AF AG** p ?

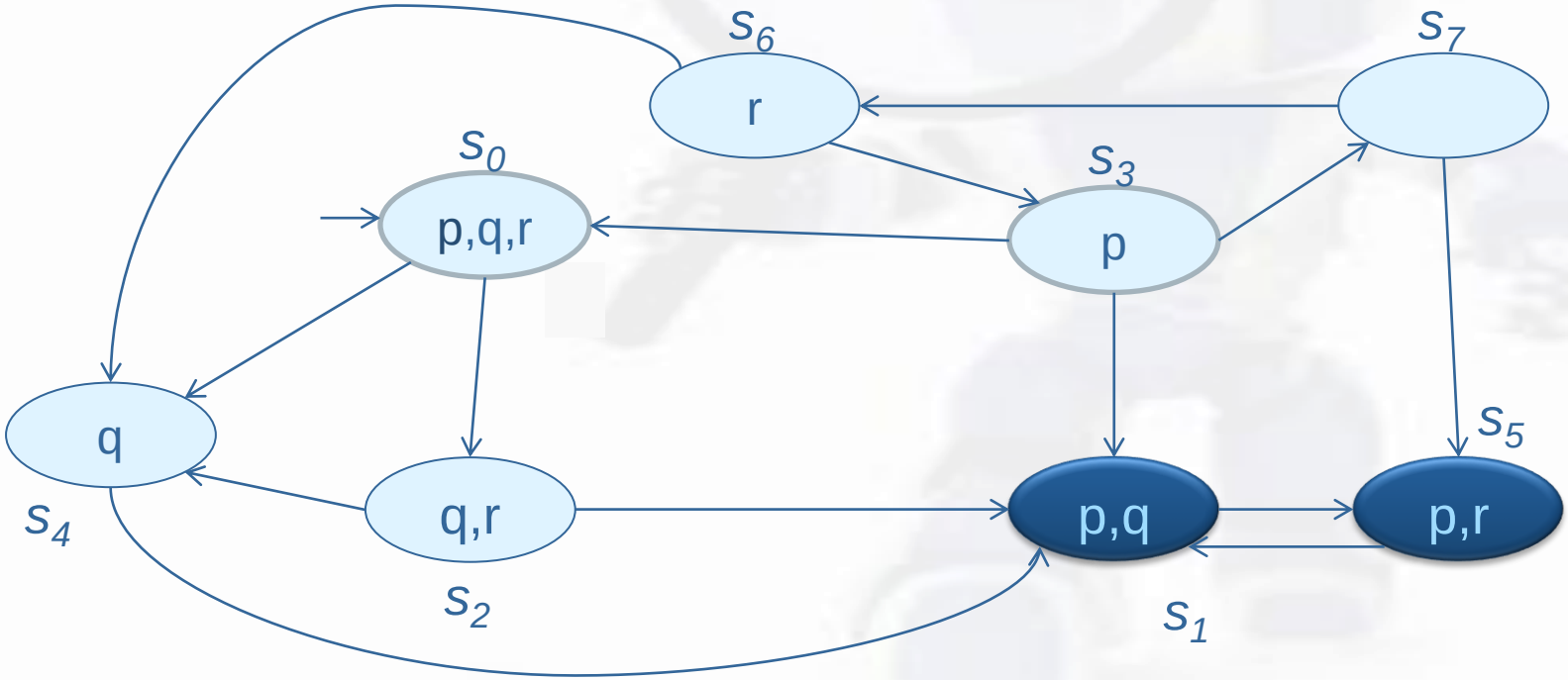
Application

- A-t-on K,s0 ⊨**AF AG** p ?
- Commençons par trouver les états s_i tel que K,s_i ⊨p.



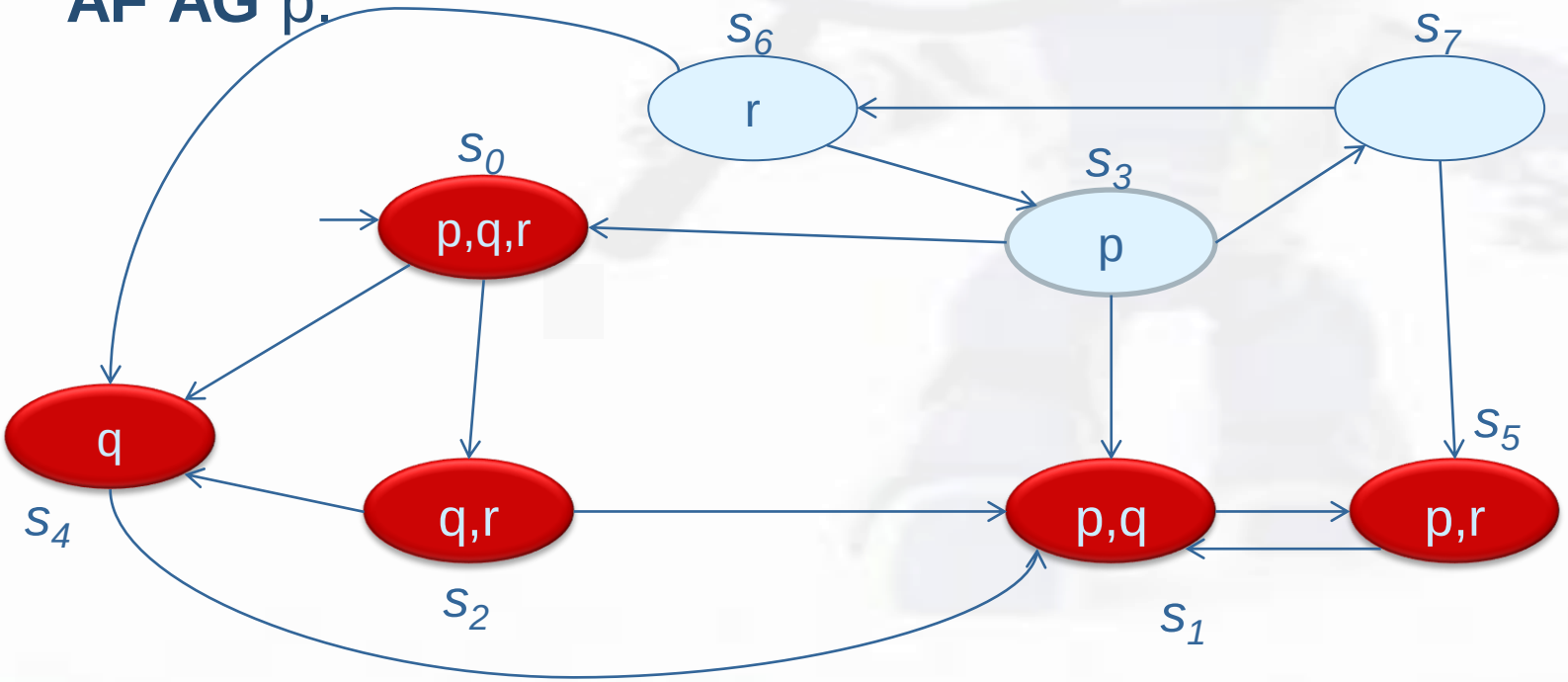
Application

- A-t-on K,s0 ⊨ **AF AG** p ?
- Ensuite trouvons les états s_i tel que K,s_i ⊨ **AG**p.



Application

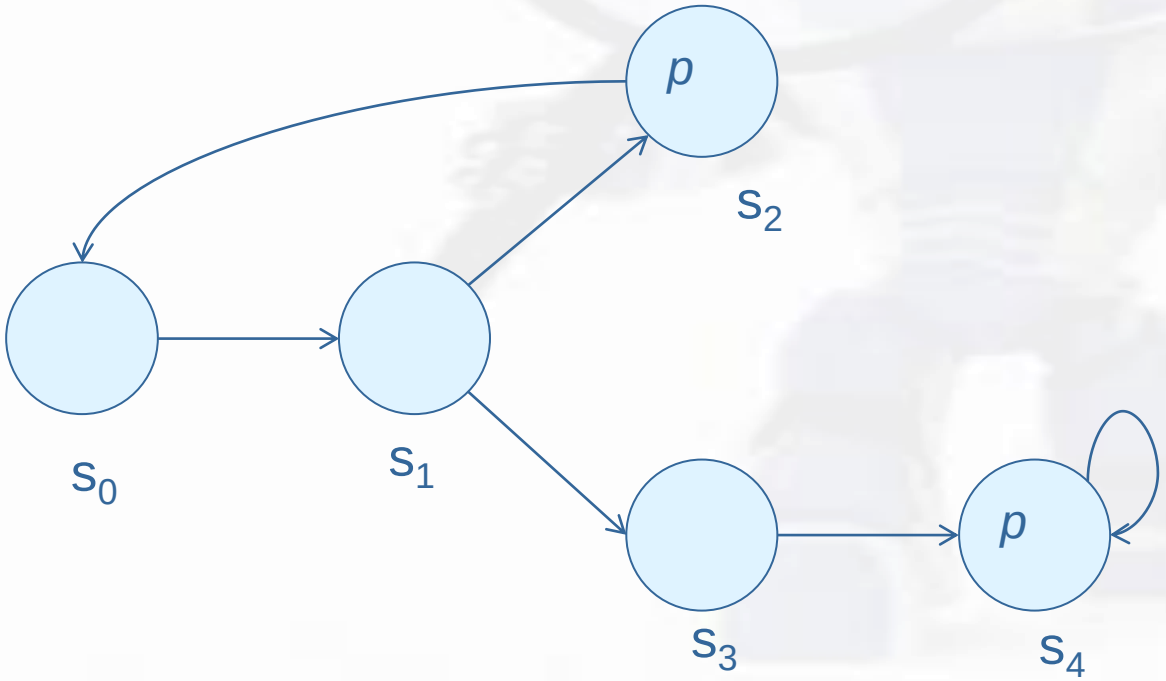
- A-t-on $K,s_0 \models$ **AF AG** p ?
- Enfin trouvons les états s_i tel que $K,s_i \models$ **AF AG** p.
- s_0 est dans l'ensemble satisfaisant **AF AG** p,**donc** $K,s_0 \models$ **AF AG** p.



Exercice

- A-t-on $K,s_0 \models$ **AXAF**p ?

K



Expression en CTL

- « *Les feux vert et rouge ne sont jamais allumés en même temps* » :
  - **AG** ¬(feurouge ∧ feuvert)
- « *Il est possible que l'intrus attrape un message* » :
  - **EF**(ackVole ∨ msgVole).
- «*Un état de blocage n'est jamais atteint* » :
  - **AG**(¬blocage).
- «*Un processus entrera infiniment souvent dans se section crtitique* » :
  - **AGAF**(sec_crit).

LTL vs CTL

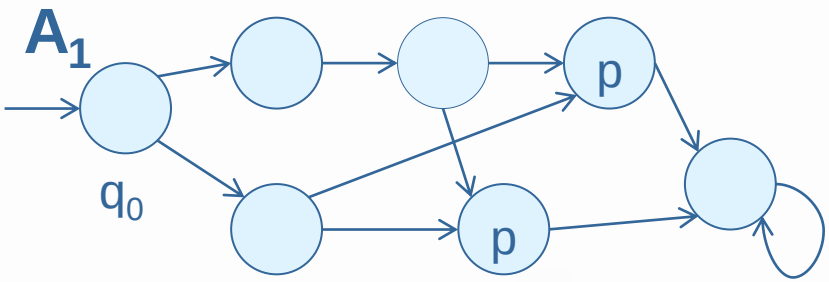
- Il existe des propriétés exprimables aussi bien en CTL qu'en LTL :
  - Invariants («*p n'est jamais vraie* »)
    - **AG** ¬p ou **G**¬p
  - Réactivité (« *si p devient vraie, q peut devenir vraie* »
    - **AG**(p → **AF** q) ou **G**(p → **F** q)

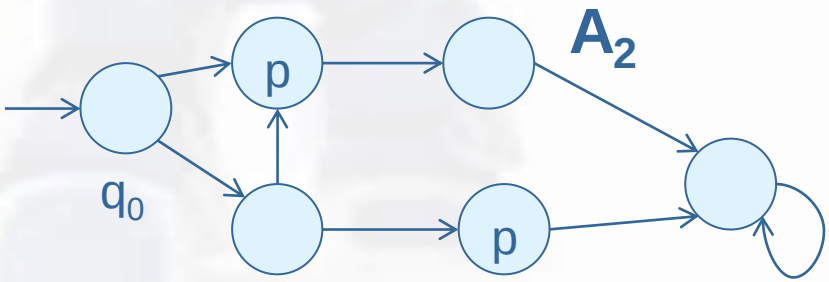
LTL vs CTL

- CTL permet de raisonner sur les comportements de branchement alors que LTL ne le permet pas puisqu'il considère le modèle comme des exécutions individuelles.
- Exemples:
  - La propriété CTL **AG EF** *p* n'est pas exprimable en LTL.
  - La propriété CTL **AF AX** *p* permet de distinguer entre ces deux modèles alors que la propriété LTL **F X** *p* ne le permet pas.

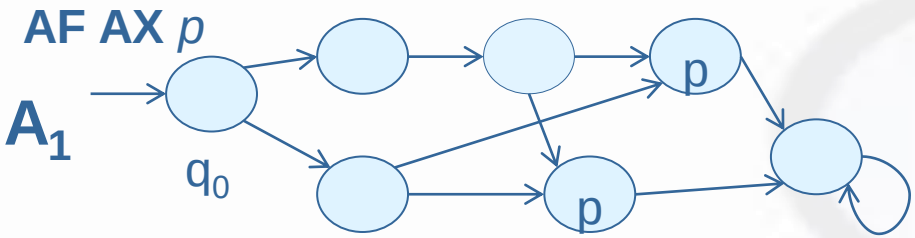
AF AX *p* (CTL) et FX *p* (LTL)

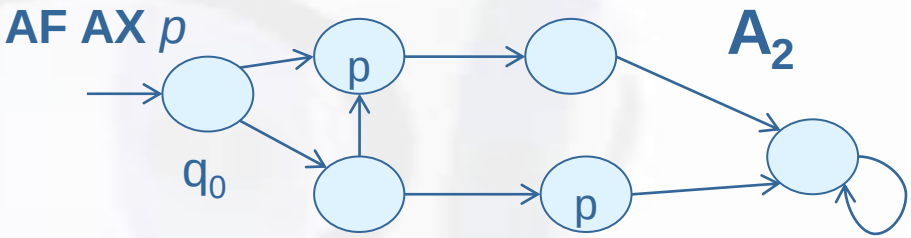
- **AF AX** *p* (CTL) et **FX** *p* (LTL) :
  - $A_1, q_0 \models$ **AF AX** *p* ?
  - $A_2, q_0 \models$ **AF AX** *p* ?
  - $A_1, q_0 \models$ **FX** *p* ?
  - $A_2, q_0 \models$ **FX** *p* ?



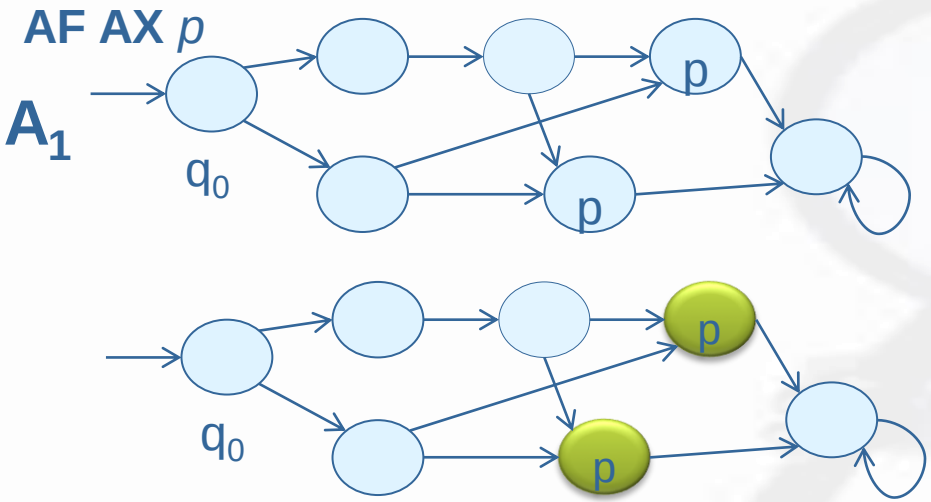


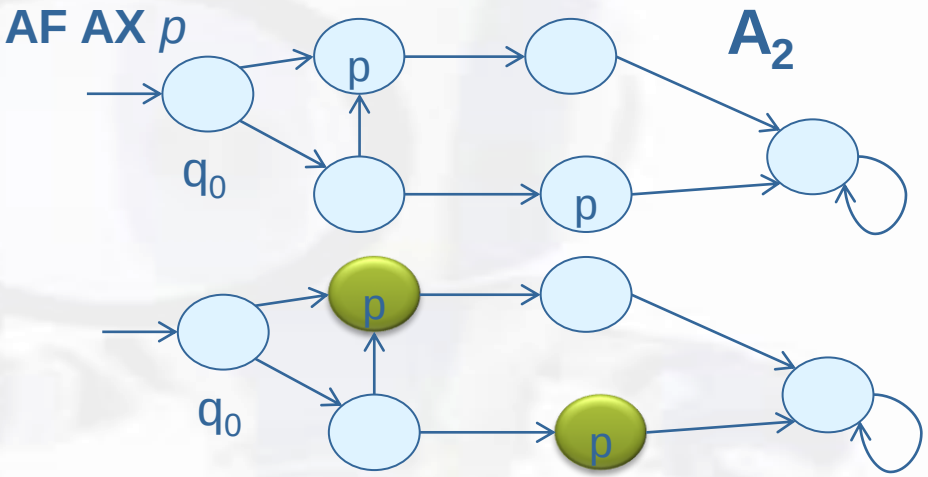
AF AX *p*



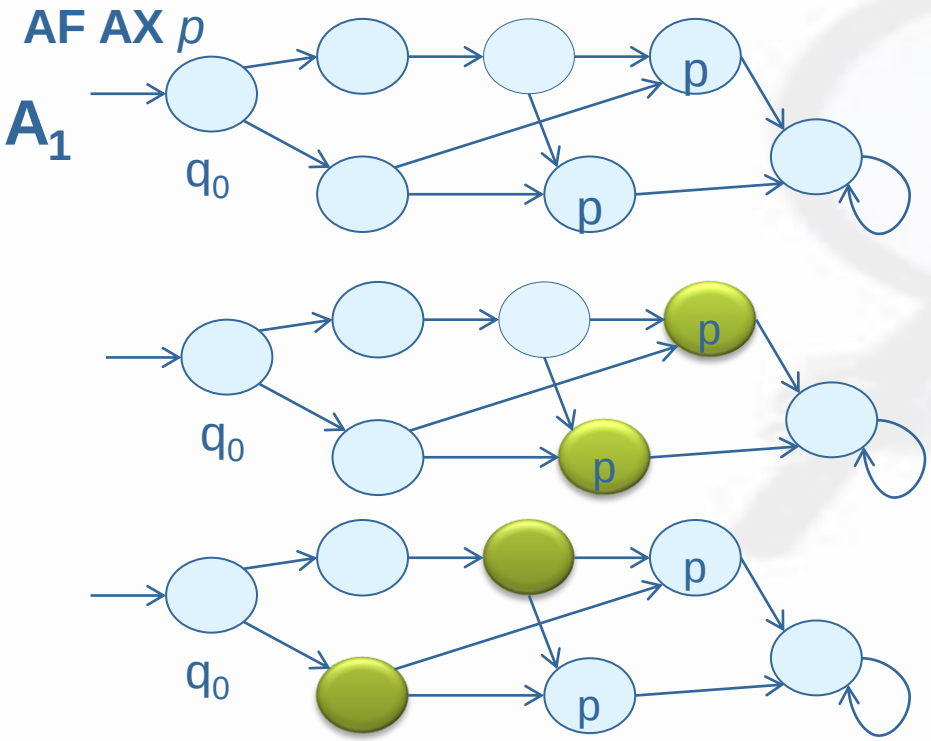


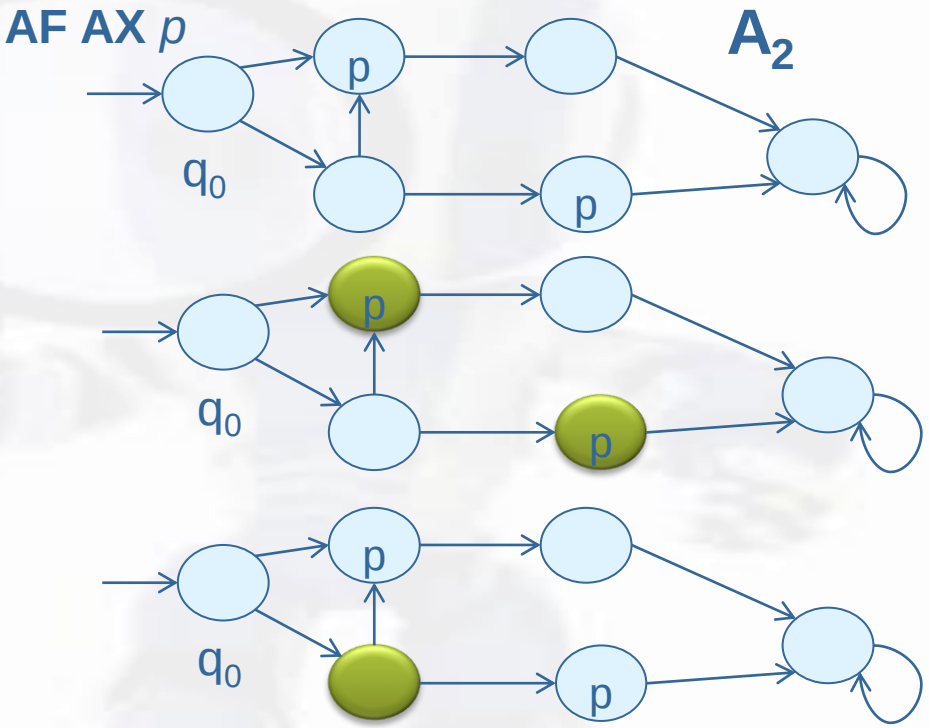
AF AX *p*



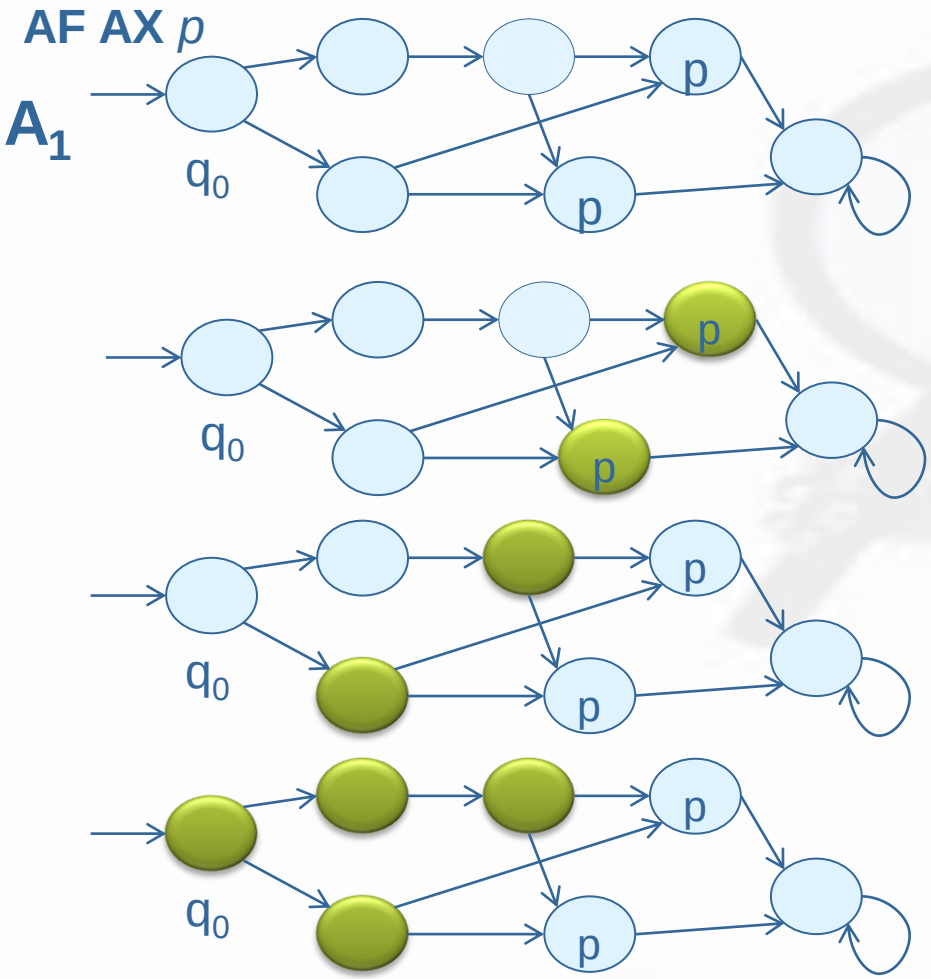


AF AX *p*

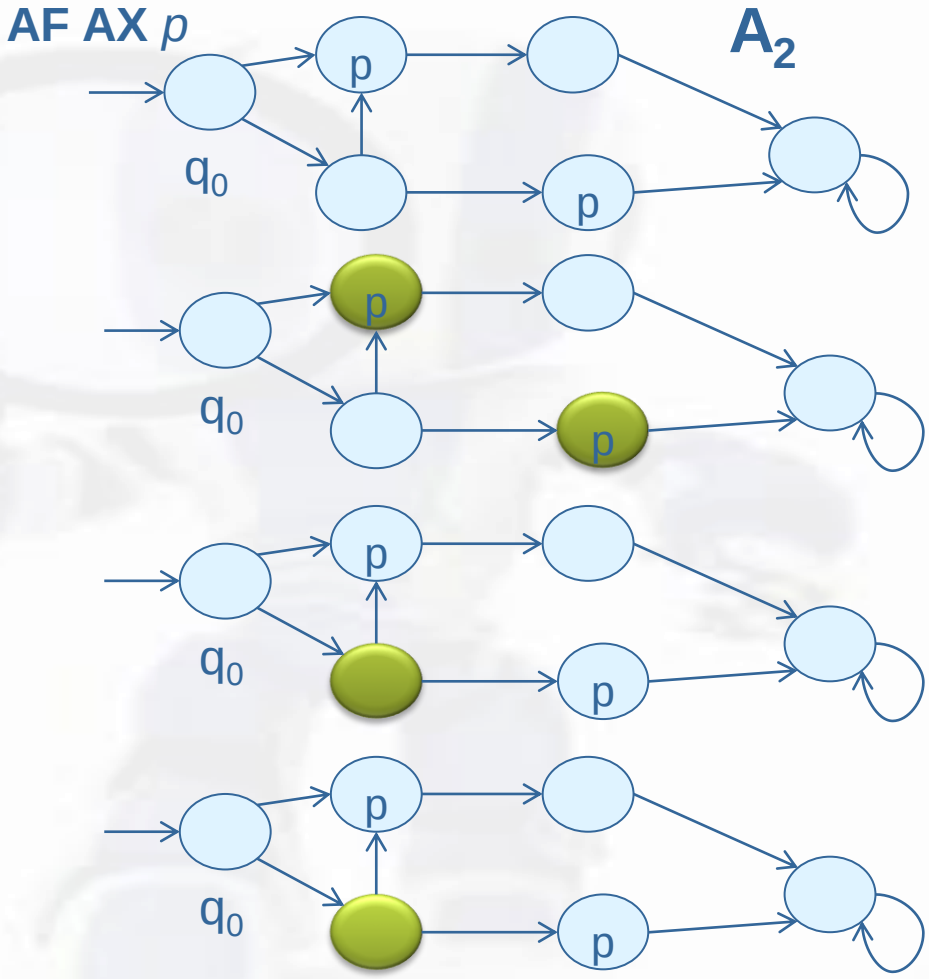




AF AX *p*

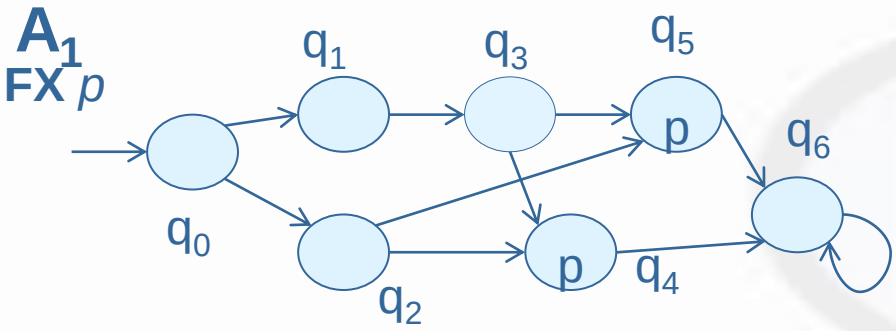


$A_1, q_0 \vdash$ **AF AX** *p*

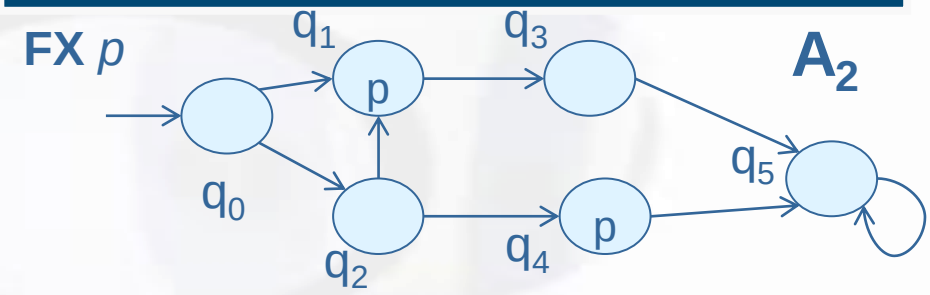


$A_2, q_0 \nvdash$ **AF AX** *p*

FX *p*



- Les exécutions de A_1 :
  - $\sigma_1 = q_0 q_1 q_3 q_5 q_6 q_6 q_6 \ldots$
  - $\sigma_2 = q_0 q_1 q_3 q_4 q_6 q_6 q_6 \ldots$
  - $\sigma_3 = q_0 q_2 q_5 q_6 q_6 q_6 \ldots$
  - $\sigma_4 = q_0 q_2 q_4 q_6 q_6 q_6 \ldots$
- Dans toutes les exécutions, il est possible d'atteindre un état dans lequel le prochain état satisfait p.



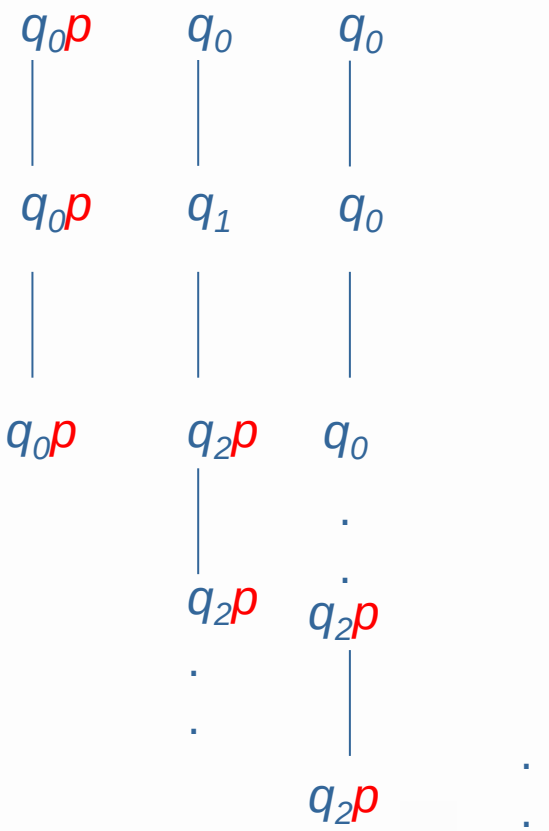
- Les exécutions de A_2 :
  - $\sigma_1 = q_0 q_1 q_3 q_5 q_5 q_5 q_5 \ldots$
  - $\sigma_2 = q_0 q_2 q_1 q_3 q_5 q_5 q_5 \ldots$
  - $\sigma_3 = q_0 q_2 q_4 q_5 q_5 q_5 \ldots$
- Dans toutes les exécutions, il est possible d'atteindre un état dans lequel le prochain état satisfait p.

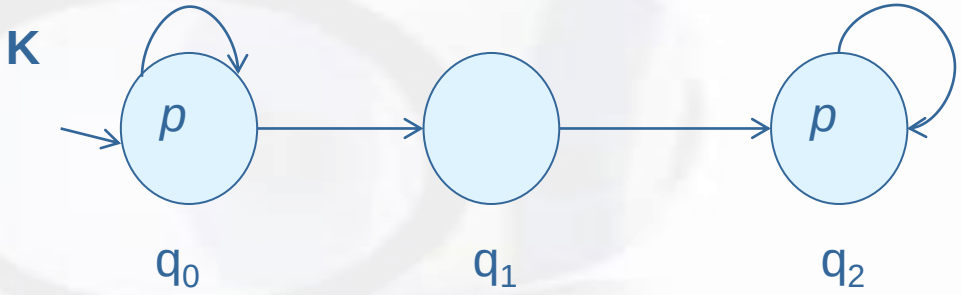
LTL vs CTL

- Il existe également des propriétés exprimables en LTL mais qui ne le sont pas en CTL comme **FG** p :
  - *dans le futur, p deviendra toujours vraie tout au long de l'exécution* (et ce pour toutes les exécutions).
- Notez que **AFAG** *p* n'exprime pas la même propriété :
  - *toutes les exécution possibles vérifient qu'il est possible d'atteindre des états qui vérifient que p est toujours vérifiées dans toutes les exécutions*.

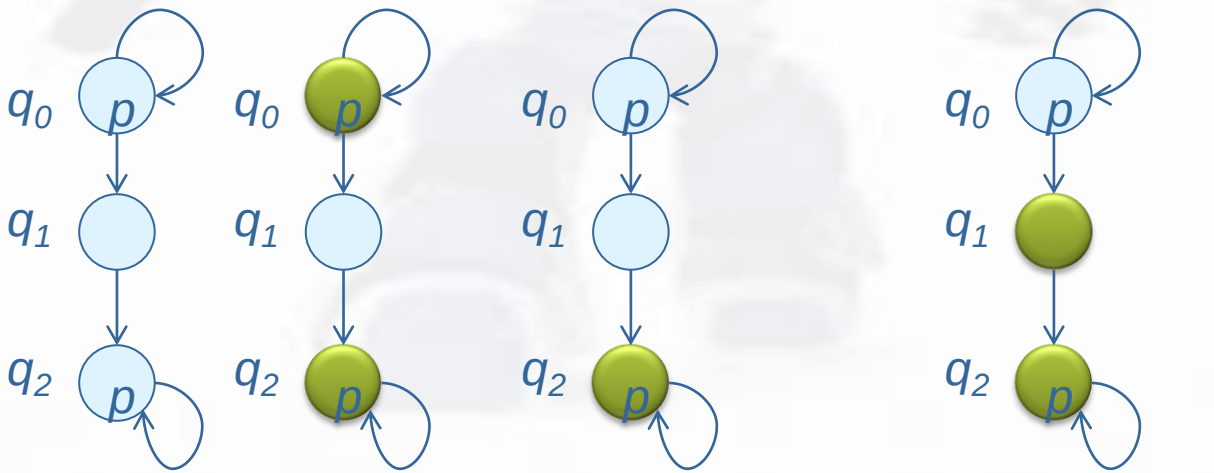
FGp vs AFAG*p*

$K \models FG\ p$





$K \not\models AFAG\ p$



Limites de CTL

- Ce sont des propriétés qui portent sur le comportement du modèle le long d'une exécutions comme :
  - *dans le futur, φ deviendra toujours vraie tout au long de l'exécution* ***FG*** *φ.*
- *Ce sont des propriétés nécessitant l'imbrication des modalités de chemin :*
  - *Il existe un chemin le long duquel une formule est satisfiable infiniment souvent* ***EGF*** *φ.*

CTL*

- CTL* est une logique plus expressive que CTL et LTL.
- Syntaxe :

$\phi ::= true | flase | a \in AP |$ *propositions atomiques*

$\neg\varphi \; | \varphi_1 \wedge \varphi_2 |$ *combinateurs booléens*

$\varphi_1 \vee \varphi_2 | \varphi_1 \Rightarrow \varphi_2 |$

$\mathbf{X}\varphi | \mathbf{F}\varphi | \mathbf{G}\varphi | \varphi \mathbf{U} \psi |$ *opérateurs temporels*

$\mathbf{A}\varphi \mid \mathbf{E}\varphi$ *quantificateurs de chemin*

Sémantique de CTL*

$\sigma, i \vDash true$

$\sigma, i \nvDash false$

$\sigma, i \vDash a \; ssi \; a \in l(\sigma(i))$

$\sigma, i \vDash \neg\phi \; ssi \; \sigma, i \vDash \not\phi$

$\sigma, i \vDash \phi \vee \phi' ssi \; \sigma, i \vdash \phi \; ou \; \sigma, i \vdash \phi'$

$\sigma, i \vDash \phi \wedge \phi' ssi \; \sigma, i \vdash \phi \; et \; \sigma, i \vdash \phi'$

$\sigma, i \vDash \mathbf{X}\phi \; ssi \; i < |\sigma| \; et \; \sigma, i+1 \vDash \phi$

$\sigma, i \vDash \mathbf{F}\phi \; ssi \; \exists j | i \leq j \leq |\sigma| \; et \; \sigma, j \vDash \phi$

$\sigma, i \vDash \mathbf{G}\phi \; ssi \; \forall j | i \leq j \leq |\sigma| \; on \; a \; \sigma, j \vDash \phi$

$\sigma, i \vDash \phi \mathbf{U} \phi' \; ssi \; \exists j | i \leq j \leq |\sigma| \; et \; \sigma, j \vDash \phi'$

$et \; \forall k | i \leq k \leq j \; on \; a \; \sigma, k \vDash \phi'$

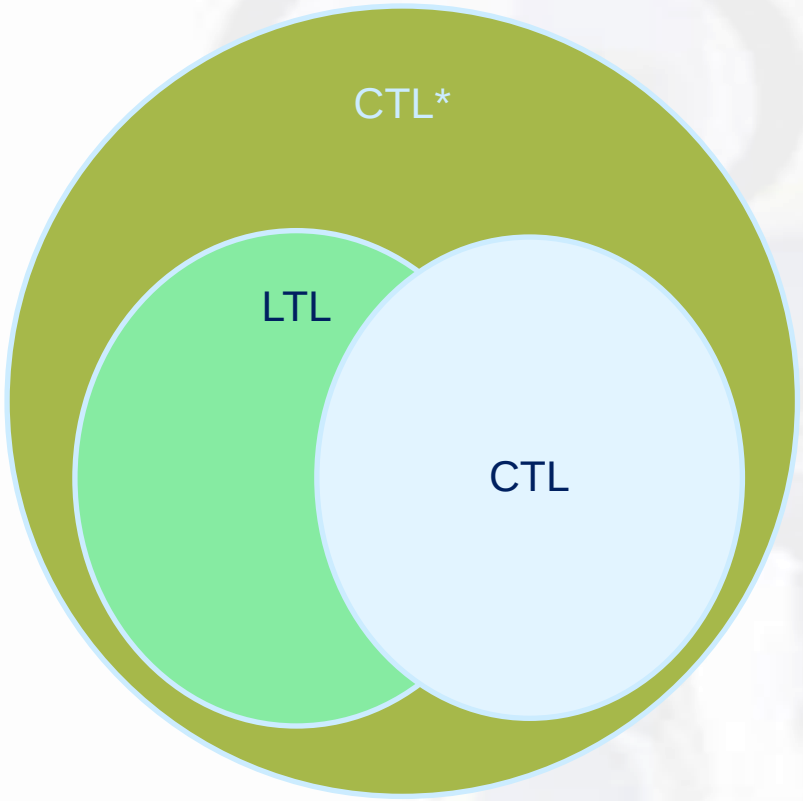
$\sigma, i \vDash \mathbf{E}\phi \; ssi \; \exists \sigma' | \sigma'(0) \ldots \sigma'(i) = \sigma(0) \ldots \sigma(i) \; et \; \sigma', i \vDash \phi$

$\sigma, i \vDash \mathbf{A}\phi \; ssi \; \forall \sigma' | \sigma'(0) \ldots \sigma'(i) = \sigma(0) \ldots \sigma(i) \; on \; a \; \sigma', i \vDash \phi$

CTL vs CTL*

- Les formule CTL* suivantes ne sont pas exprimables en CTL :
  - **AFG***p*
  - **EFG***p*
  - **AGF***p*
  - **E**(**G**p ∧ **X**q)

Comparaison des logiques temporelles



Comparaison

- LTL
  - Intuitif
  - Expressif sur un chemin (équité)
  - Pas d'expressivité sur les futurs possibles
  - Model checking coûteux
- CTL
  - Manque d'expressivité pour les chemins
  - Model checking efficace
  - Peu intuitif

Comparaison

- CTL*
  - Expressif
  - Model checking moins coûteux que LTL (mais il reste coûteux)
  - Peu intuitif