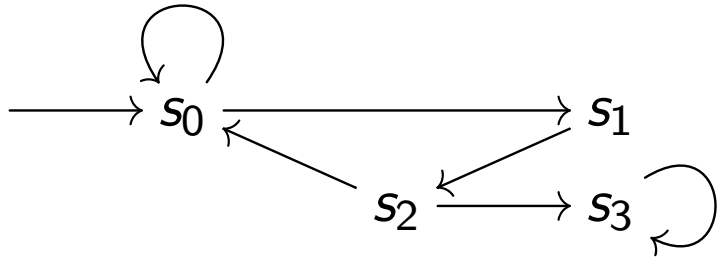
Sixième partie

CTL – logique temporelle arborescente

Plan

Ensemble des exécutions vs arbre des exécutions

Soit le système de transitions :

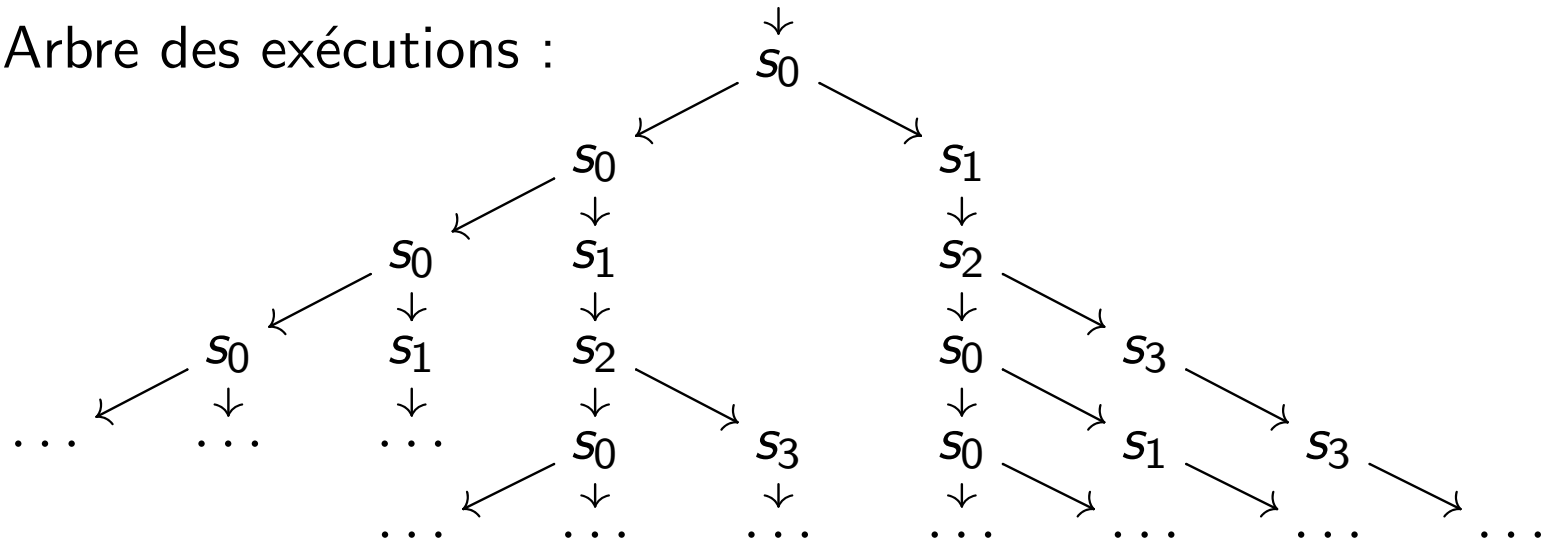


Ensemble des exécutions :

$$\langle (s_0{}^+ \to s_1 \to s_2)^* \to s_0{}^\omega \rangle, \langle (s_0{}^+ \to s_1 \to s_2)^\omega \rangle, \langle (s_0{}^+ \to s_1 \to s_2)^+ \to s_3{}^\omega \rangle$$

$$\text{ou} \left\{ \begin{array}{l} s_0 \to s_0 \to \cdots, s_0 \to s_1 \to s_2 \to s_0 \to s_0 \to \cdots, \\ s_0 \to s_1 \to s_2 \to s_0 \to \cdots, s_0 \to s_0 \to s_1 \to s_2 \to s_0 \to \cdots, \ldots \\ s_0 \to s_1 \to s_2 \to s_3 \to s_3 \to \cdots, \ldots \end{array} \right\}$$

Arbre des exécutions :



Computational Tree Logic – logique temporelle arborescente

Modèles

Une formule CTL se rapporte toujours à un état donné s d'un système, duquel partent des traces $\mathit{Traces}(s)$.
Les états de S constituent les modèles de cette logique.

La différence (syntaxiquement parlant) avec LTL réside dans l'apparition dans les opérateurs temporels de quantificateurs de traces.

Syntaxe de la CTL

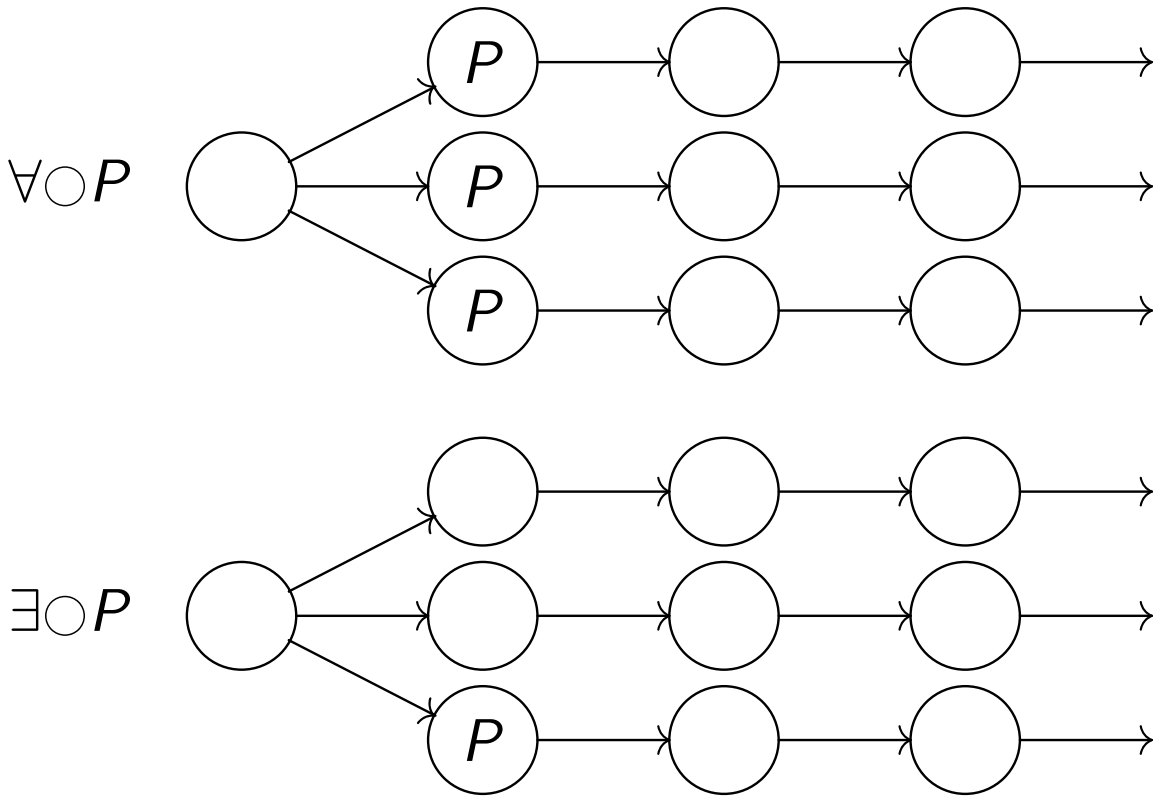
Quantification universelle

formule	interprétation (pour s un état) pour toute trace partant de s
$\forall\bigcirc P$	P est vrai à l'instant suivant
$\forall\Box P$	P est toujours vrai à chaque état
$\forall\Diamond P$	P finit par être vrai (dans le futur)
$P \forall\mathcal{U}\ Q$	Q finit par être vrai, et en attendant P reste vrai

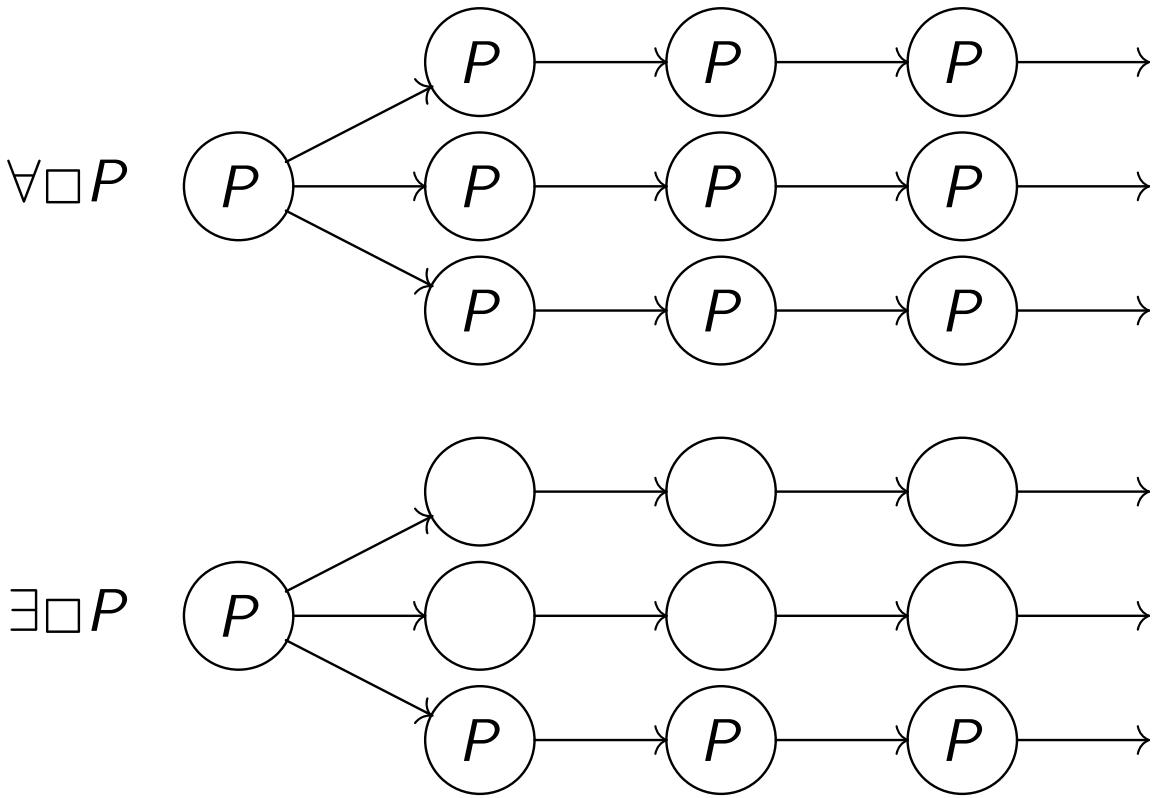
Quantification existentielle

formule	interprétation (pour s un état) pour au moins une trace partant de s
$\exists\bigcirc P$	P est vrai à l'instant suivant
$\exists\Box P$	P est toujours vrai à chaque état
$\exists\Diamond P$	P finit par être vrai (dans le futur)
$P \exists\mathcal{U}\ Q$	Q finit par être vrai, et en attendant P reste vrai

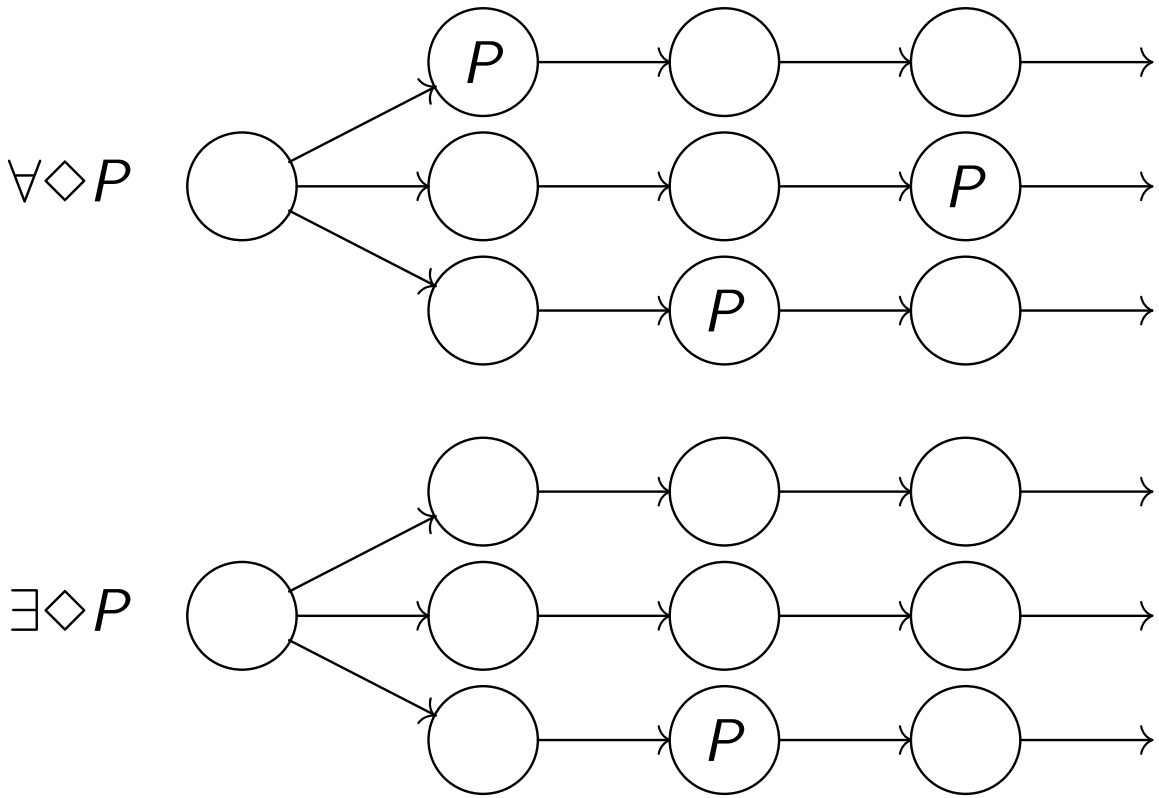
Intuition sémantique $\forall\bigcirc$, $\exists\bigcirc$



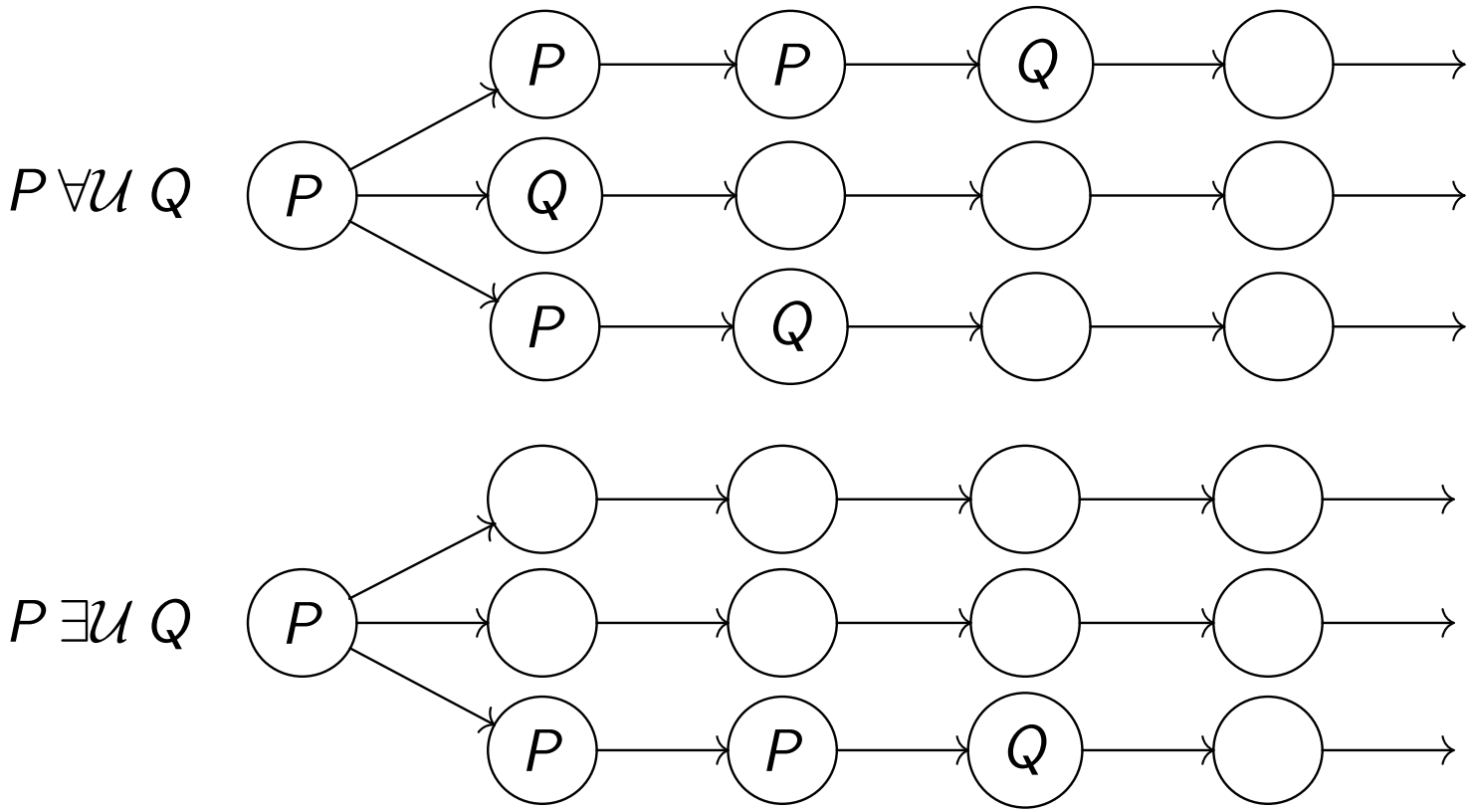
Intuition sémantique $\forall\Box$, $\exists\Box$



Intuition sémantique $\forall\Diamond$, $\exists\Diamond$



Intuition sémantique $\forall\mathcal{U}$, $\exists\mathcal{U}$



Opérateurs minimaux

Les opérateurs minimaux sont $\forall\bigcirc P$, $P \forall\mathcal{U}\ Q$ et $P \exists\mathcal{U}\ Q$:

- $\exists\bigcirc P \triangleq \neg\forall\bigcirc\neg P$
- $\forall\Diamond P \triangleq \mathit{True} \forall\mathcal{U}\ P$
- $\exists\Diamond P \triangleq \mathit{True} \exists\mathcal{U}\ P$
- $\forall\Box P \triangleq \neg\exists\Diamond\neg P$
- $\exists\Box P \triangleq \neg\forall\Diamond\neg P$

Syntaxe alternative

Syntaxe alternative

On trouve très fréquemment une autre syntaxe :

$\forall \quad \leftrightarrow$ A (all)
$\exists \quad \leftrightarrow$ E (exists)
$\Box \quad \leftrightarrow$ G (globally)
$\Diamond \quad \leftrightarrow$ F (finally)
$\bigcirc \quad \leftrightarrow$ X (next)

Par exemple :

$\forall \Box \exists \Diamond P \leftrightarrow$ AG EF P
$f \, \forall \mathcal{U} \, g \leftrightarrow$ A(f U g)

Opérateur complémentaire waiting-for ♪♪♪

$P \, \exists \mathcal{W} \, Q \triangleq \exists \Box P \vee P \, \exists \mathcal{U} \, Q$
$P \, \forall \mathcal{W} \, Q \not\triangleq \forall \Box P \vee P \, \forall \mathcal{U} \, Q$ – *trop fort*
$\triangleq \neg(\neg Q \, \exists \mathcal{U}(\neg P \wedge \neg Q))$

Sémantique (système)

La relation de validation sémantique ne fait intervenir que l'état courant.

Vérification par un système

Un système $\mathcal{S} = \langle S, I, R \rangle$ vérifie (valide) la formule F ssi tous les états initiaux de $\mathcal{S}$ la valident :

$$\frac{\forall s \in I : s \models F}{\mathcal{S} \models F}$$

Sémantique (opérateurs logiques)

$$\frac{}{s \models s}$$

$$\frac{s \models P \quad s \models Q}{s \models P \wedge Q}$$

$$\frac{s \models P}{s \models P \vee Q} \qquad \frac{s \models Q}{s \models P \vee Q}$$

$$\frac{s \models P}{s \not\models \neg P}$$

Sémantique (opérateurs temporels)

(rappel : pour une trace σ, σ_i est le i-ième élément de σ en commençant à 0, et pour un état s, $\mathit{Traces}(s)$ est l'ensemble des traces issues de s)

$$\frac{\forall \sigma \in \mathit{Traces}(s) : \sigma_1 \models P}{s \models \forall \bigcirc P}$$

$$\frac{\forall \sigma \in \mathit{Traces}(s) : \exists j \geq 0 : \sigma_j \models Q \wedge \forall i < j : \sigma_i \models P}{s \models P \, \forall \mathcal{U} \, Q}$$

$$\frac{\exists \sigma \in \mathit{Traces}(s) : \exists j \geq 0 : \sigma_j \models Q \wedge \forall i < j : \sigma_i \models P}{s \models P \, \exists \mathcal{U} \, Q}$$

Sémantique (opérateurs temporels dérivés)

$$\frac{\exists \sigma \in \mathit{Traces}(s) : \sigma_1 \models P}{s \models \exists \bigcirc P}$$

$$\frac{\forall \sigma \in \mathit{Traces}(s) : \forall i \geq 0 : \sigma_i \models P}{s \models \forall \Box P}$$

$$\frac{\exists \sigma \in \mathit{Traces}(s) : \forall i \geq 0 : \sigma_i \models P}{s \models \exists \Box P}$$

$$\frac{\forall \sigma \in \mathit{Traces}(s) : \exists i \geq 0 : \sigma_i \models P}{s \models \forall \Diamond P}$$

$$\frac{\exists \sigma \in \mathit{Traces}(s) : \exists i \geq 0 : \sigma_i \models P}{s \models \exists \Diamond P}$$

Négation

♪♪♪

Négation

Contrairement à LTL, pour toute propriété CTL, on a :
soit $\mathcal{S} \models F$, soit $\mathcal{S} \models \neg F$,
et $\mathcal{S} \not\models F \equiv \mathcal{S} \models \neg F$.

Négation des formules $\forall, \exists, \Box, \Diamond$

La négation d'une formule à base de $\forall, \exists, \Box, \Diamond$ se fait simplement en inversant chaque opérateur pour son dual.

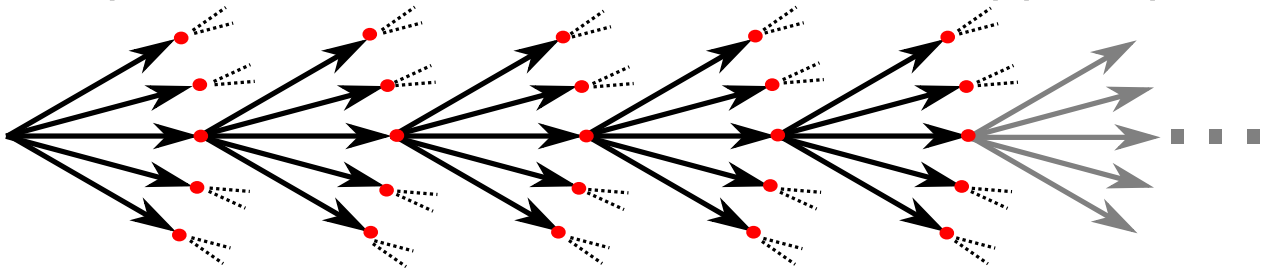
exemples :
$\neg(\forall\Diamond\exists\Box p) = \exists\Box\forall\Diamond\neg p$
$(\forall\Diamond\neg s_0 \Rightarrow \forall\Diamond s_3) = (\exists\Box s_0 \vee \forall\Diamond s_3)$ car $(p \Rightarrow q) = (\neg p \vee q)$

Plan

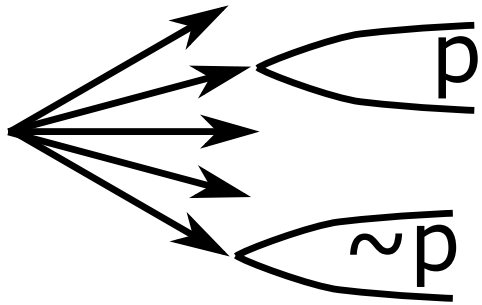
Exemples amusants

♪♪♪

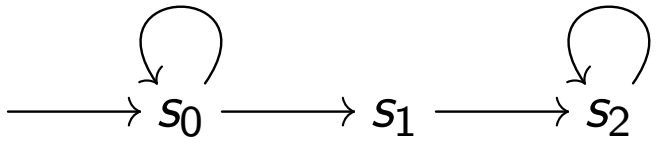
- $\exists\Box\ \forall\bigcirc\ p$: une exécution avec une “enveloppe” qui vérifie p



- $\exists\bigcirc\ \forall\Box p\ \wedge\ \exists\bigcirc\ \forall\Box\neg p$
  un état successeur à partir duquel p est toujours et partout vrai,
  et un état successeur à partir duquel $\neg p$ est toujours et partout vrai

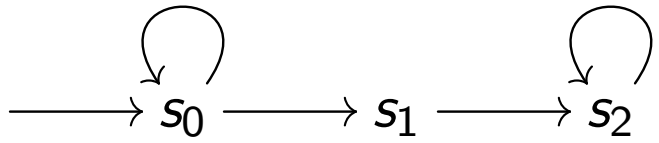


Exemple



	pas d'équité	équité faible (s_0, s_1)
$s_0 \wedge \forall\bigcirc s_0$		
$s_0 \wedge \exists\bigcirc s_0$		
$\forall\Box(s_0 \Rightarrow \exists\bigcirc s_0)$		
$\forall\Box(s_0 \Rightarrow \exists\Diamond s_2)$		
$\forall\Box(s_0 \Rightarrow \forall\Diamond s_2)$		
$\exists\Diamond\neg s_0$		
$\forall\Diamond\neg s_0$		
$\forall\Box\exists\Diamond s_2$		
$\forall\Box\forall\Diamond s_2$		
$\forall\Diamond\exists\Diamond s_1$		
$\forall\Box\exists\Diamond s_1$		

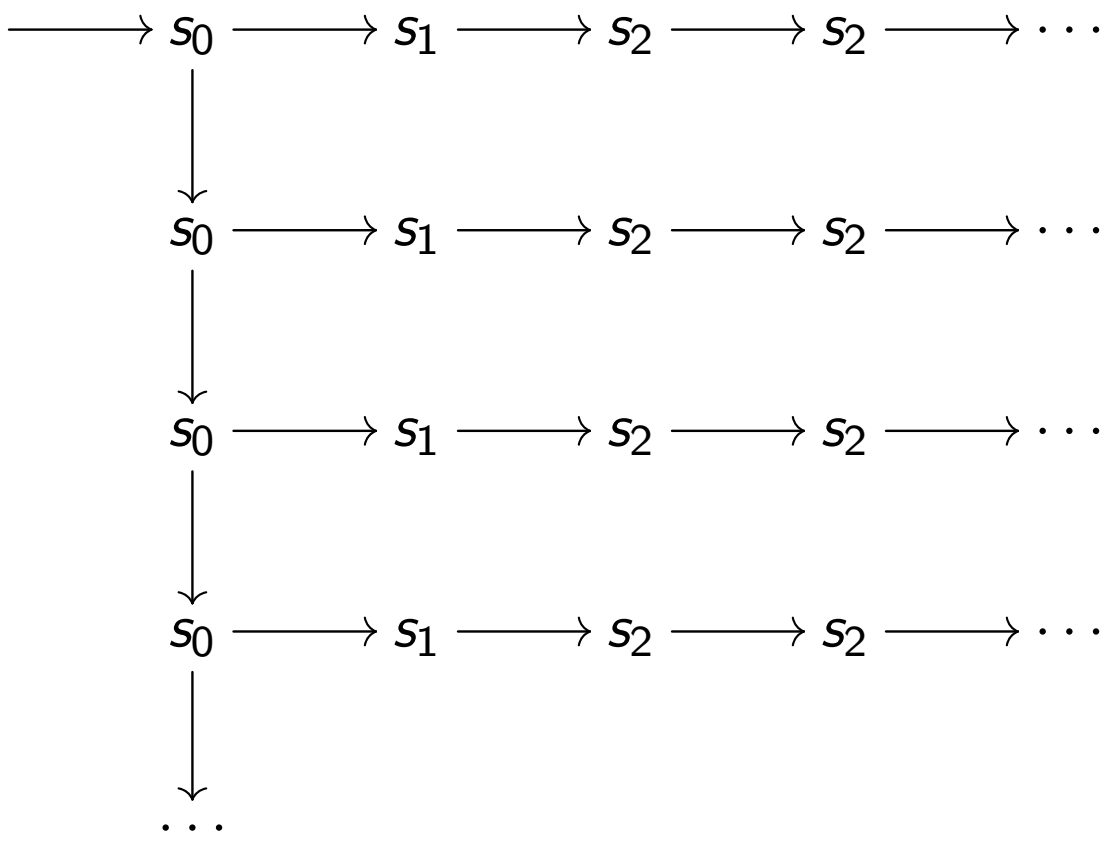
Exemple



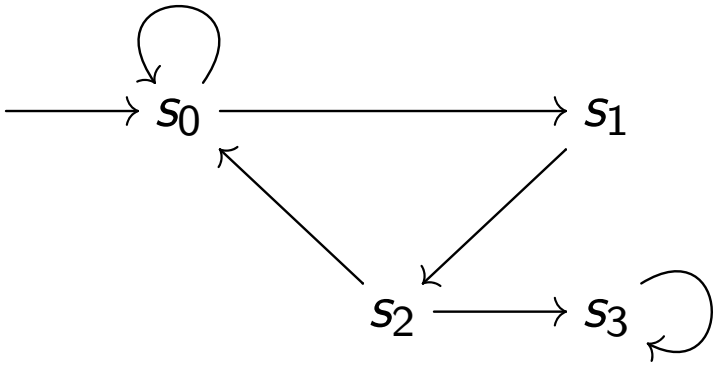
	pas d'équité	équité faible (s_0, s_1)	
$s_0 \wedge \forall\bigcirc s_0$	n	n	
$s_0 \wedge \exists\bigcirc s_0$	o	o	
$\forall\Box(s_0 \Rightarrow \exists\bigcirc s_0)$	o	o	♪♪♪
$\forall\Box(s_0 \Rightarrow \exists\Diamond s_2)$	o	o	♪♪♪
$\forall\Box(s_0 \Rightarrow \forall\Diamond s_2)$	n	o	♪♪♪
$\exists\Diamond\neg s_0$	o	o	
$\forall\Diamond\neg s_0$	n	o	
$\forall\Box\exists\Diamond s_2$	o	o	♪♪♪
$\forall\Box\forall\Diamond s_2$	n	o	♪♪♪
$\forall\Diamond\exists\Diamond s_1$	o	o	♪♪♪
$\forall\Box\exists\Diamond s_1$	n	n	♪♪♪

Exemple - Arbre des exécutions

Arbre des exécutions du système de transition précédent



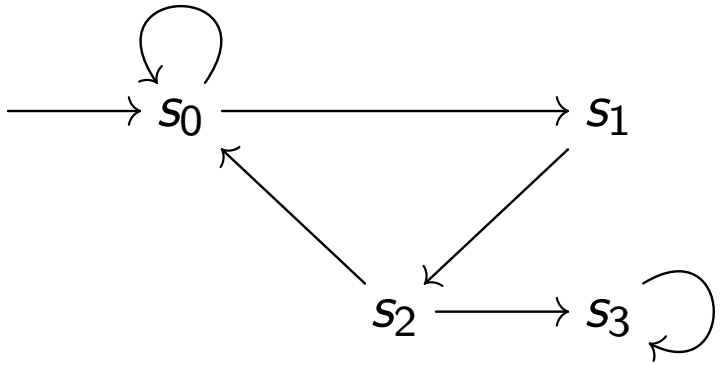
Exemple 2



	pas d'équité	faible (s_0, s_1)	forte (s_2, s_3)	forte (s_2, s_3) faible (s_0, s_1)
$\exists\Box s_0$				
$\forall\Box\exists\Diamond s_3$				
$\forall\Box\forall\Diamond s_3$				
$\forall\Diamond\forall\Box s_3$				
$\exists\Box s_0 \vee \forall\Diamond s_3$				
$\forall\Diamond\neg s_0 \Rightarrow \forall\Diamond s_3$				

Exemple 2

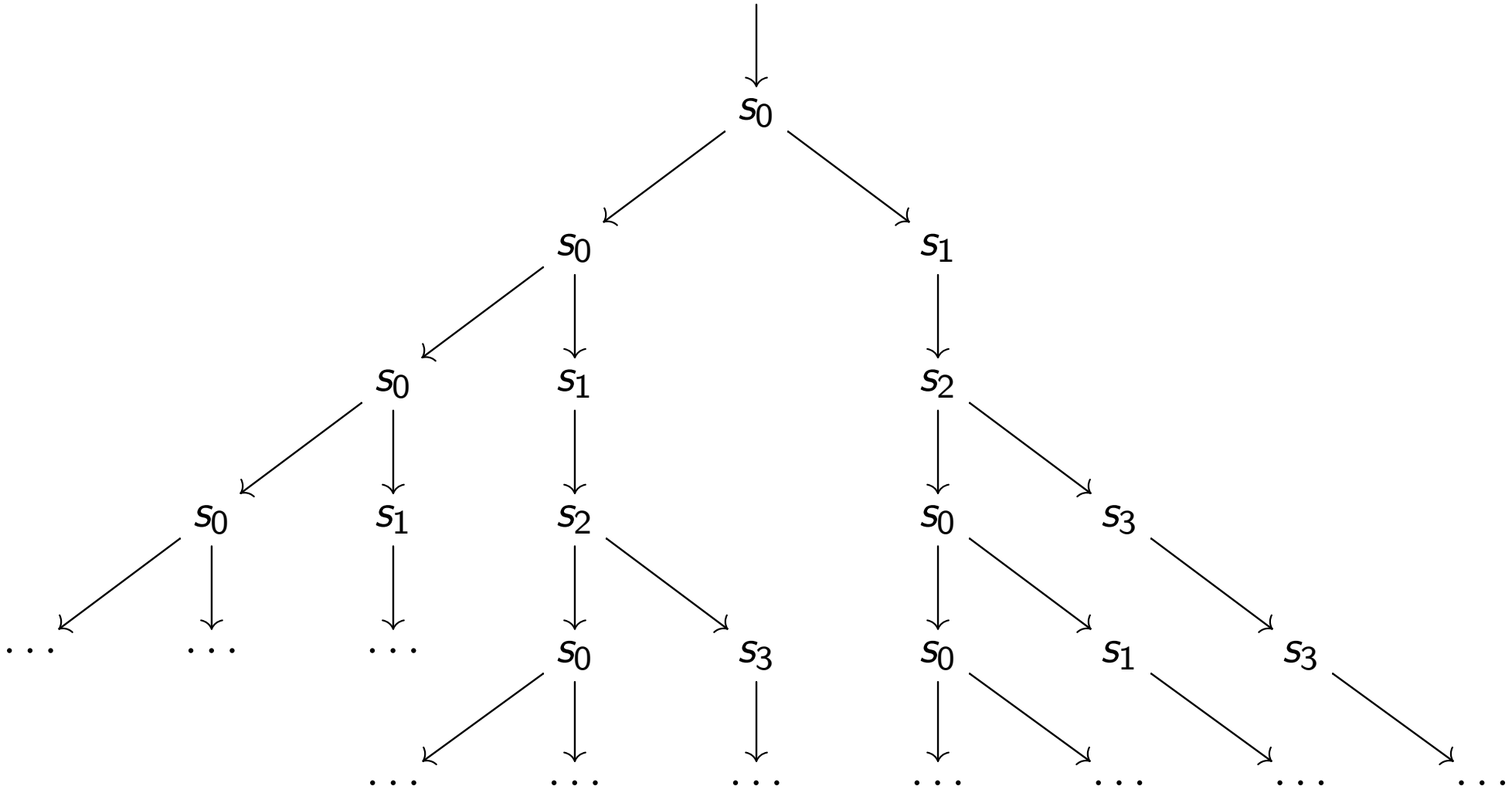
♪♪♪



	pas d'équité	faible (s_0, s_1)	forte (s_2, s_3)	forte (s_2, s_3) faible (s_0, s_1)	
$\exists\Box s_0$	o	n	o	n	
$\forall\Box\exists\Diamond s_3$	o	o	o	o	♪♪♪
$\forall\Box\forall\Diamond s_3$	n	n	n	o	♪♪♪
$\forall\Diamond\forall\Box s_3$	n	n	n	o	
$\exists\Box s_0 \vee \forall\Diamond s_3$	o	n	o	o	♪♪♪
$\forall\Diamond\neg s_0 \Rightarrow \forall\Diamond s_3$	o	n	o	o	♪♪♪

Contrairement à LTL, avec l'équité, on peut à la fois rendre vraie une propriété (p.e. contenant $\Diamond$ ou $\mathcal{U}$) et rendre faux une propriété (p.e. contenant $\exists\cdot$).

Exemple 2 - Arbre des exécutions



Invariance, Possibilité

Invariance

Spécifier un sur-ensemble des états accessibles d'un système :

$$\mathcal{S} \models \forall\Box P$$

où P est un prédicat d'état.

Stabilité

Spécifier la stabilité d'une situation si elle survient :

$$\mathcal{S} \models \forall\Box(P \Rightarrow \forall\Box P)$$

où P est un prédicat d'état.

Possibilité

Spécifier qu'il est possible d'atteindre un état vérifiant P :

$$\mathcal{S} \models \exists\Diamond P$$

Possibilité complexe

Séquence

Spécifier qu'un scénario d'exécution $\langle s_1 \rightarrow s_2 \rightarrow \ldots \rightarrow s_n \rangle$ est possible.

$$\mathcal{S} \models s_1 \wedge \exists\bigcirc(s_2 \wedge \ldots \wedge \exists\bigcirc(s_{n-1} \wedge \exists\bigcirc s_n)\ldots)$$

Réinitialisabilité

Spécifier que quelque soit l'état courant, il est possible de revenir dans un des états initiaux (définis par le prédicat I).

$$\mathcal{S} \models \forall\Box\exists\Diamond I$$

Possibilité arbitraire

Spécifier que si P devient vrai, il est toujours possible (mais pas nécessaire) que Q le devienne après.

$$\mathcal{S} \models \forall\Box(P \Rightarrow \exists\Diamond Q)$$

Client/serveur

Réponse

Spécifier qu'un système (jouant le rôle d'un serveur) répond toujours (Q) à un requête donnée (P) :

$$\mathcal{S} \models \forall\Box(P \Rightarrow \forall\Diamond Q)$$

Stabilité d'une requête

Spécifier que la requête P d'un système (jouant le rôle d'un client) est stable tant qu'il n'y a pas de réponse favorable Q :

$$\mathcal{S} \models \forall\Box(P \Rightarrow P\,\forall\mathcal{W}\,Q)$$

Combinaisons

♪♪♪

Infiniment souvent

Spécifier que P est infiniment souvent vrai dans toute exécution :

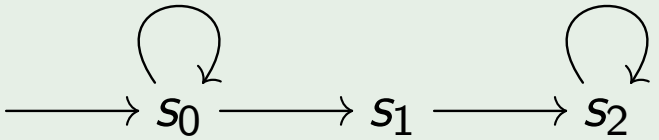
$$\mathcal{S} \models \forall\Box\forall\Diamond P$$

Finalement toujours

Spécifier que P finit par rester définitivement vrai :

impossible ! $\mathcal{S} \models \forall\Diamond\forall\Box P$ ne convient pas (trop fort)

Soit $S =$



en LTL : $S \models \Diamond\Box(s_0 \vee s_2)$

mais CTL : $S \not\models \forall\Diamond\forall\Box(s_0 \vee s_2)$

(tant qu'on est en s_0, on *peut* passer en s_1 : $S \models \forall\Diamond\exists\Diamond s_1$)

Note : $\mathcal{X}\mathcal{X}P = \mathcal{X}P$ pour $\mathcal{X} \in \{\forall\Box, \exists\Box, \forall\Diamond, \exists\Diamond\}$

Spécification d'un ST

Si on utilise une description en intention, et si l'on remplace l'utilisation de l'opérateur $\forall\bigcirc$ par les variables primées, alors on peut spécifier toutes les exécutions permises par un système $\langle S, I, R\rangle$:

$$\mathcal{S} \models I \wedge \forall\Box R$$

L'utilisation de variables primées n'est pas nécessaire mais simplifie les formules.
Par exemple $P(x, x')$ est équivalent à la formule :
$\forall v : x = v \Rightarrow \forall\bigcirc P(v, x)$
qui nécessite une quantification sur une variable.

Comparaison CTL vs. LTL

Contrairement à CTL, les opérateurs temporels LTL parlent tous de la même trace. Les combinaisons de connecteurs temporels ont parfois des sens (subtilement) différents.

	CTL	LTL
$\forall P$, nécessairement P ou $\neg P$	$\mathcal{S} \models P \vee \mathcal{S} \models \neg P$	~~$\mathcal{S} \models P \vee \mathcal{S} \models \neg P$~~
négation	$S \models \neg P \equiv S \not\models P$	~~$S \models \neg P \equiv S \not\models P$~~
l'un de P ou Q inévitable	~~$\mathcal{S} \models \forall\Diamond P \vee \forall\Diamond Q$~~ $\mathcal{S} \models \forall\Diamond(P \vee Q)$	$\mathcal{S} \models \Diamond P \vee \Diamond Q$
l'un de P ou Q continu	~~$\mathcal{S} \models \forall\Box(P \vee Q)$~~ ~~$\mathcal{S} \models \forall\Box P \vee \forall\Box Q$~~	$\mathcal{S} \models \Box P \vee \Box Q$
$\neg P$ transitoire	~~$\mathcal{S} \models \forall\Diamond\forall\Box P$~~	$\mathcal{S} \models \Diamond\Box P$
possibilité	$S \models \exists\Diamond P$	~~$S \models \Diamond P$~~

Conséquence : l'équité n'est pas exprimable en CTL.

Néanmoins, on peut vérifier des propriétés CTL sur un ST comportant des contraintes d'équité.

Comparaison LTL vs. CTL

Linear Time Logic

+ Intuitive
− . . . sauf la négation
+ Suffisante pour décrire un système de transition
+ y compris l'équité
− Vérification exponentielle en le nombre d'opérateurs temporels

Computational Tree Logic

− Expressivité parfois déroutante
+ Propriétés de possibilité (p.e. réinitialisabilité)
+ Suffisante pour décrire un système de transition
− . . . sauf l'équité non exprimable (mais utilisable)
+ Vérification linéaire en le nombre d'opérateurs temporels

Au-delà : CTL*

CTL* autorise tout mélange des quantificateurs de traces $\forall, \exists$ et d'états $\Box, \Diamond, \bigcirc, \mathcal{U}$.

Exemple : $\exists((\Box\Diamond P) \wedge (\Diamond Q)) =$ il existe une exécution où P est infiniment souvent vrai, et où Q sera vrai.

CTL* est strictement plus expressif que CTL et LTL. L'usage pratique est rare (hors les fragments correspondant à CTL et LTL).