Les logiques CTL et CTL*

Introduction à la Vérification - SIN6U22

Jean-Marc Talbot

Université d'Aix-Marseille

2020-21

Rappel LTL

LTL (Linear-time logic) :

- décrit des propriétés d'exécutions maximales (individuelles) d'un système (logique linéaire)
  - un système satisfait/vérifie une formule φ si **toutes ses exécutions maximales** satisfont φ

Rappel LTL

LTL (Linear-time logic) :

- décrit des propriétés d'exécutions maximales (individuelles) d'un système (logique linéaire)
  - un système satisfait/vérifie une formule φ si **toutes ses exécutions maximales** satisfont φ
- sémantique d'une formule est donnée comme un ensemble d'exécutions :

$$[\![\varphi]\!]_{\mathcal{A}} = \{\rho \text{ une exécution maximale de } \mathcal{A} \mid \rho \models \varphi\}$$

Rappel LTL

LTL (Linear-time logic) :

- décrit des propriétés d'exécutions maximales (individuelles) d'un système (logique linéaire)
  - un système satisfait/vérifie une formule φ si **toutes ses exécutions maximales** satisfont φ
- sémantique d'une formule est donnée comme un ensemble d'exécutions :

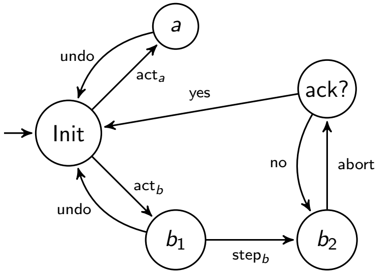
$$[\![\varphi]\!]_{\mathcal{A}} = \{\rho \text{ une exécution maximale de } \mathcal{A} \mid \rho \models \varphi\}$$

- utilisation de connecteurs temporels :
  - $\mathsf{X}\varphi$: l'exécution (maximale) à partir de l'état suivant vérifie φ
  - $\mathsf{F}\varphi$: il existe une position à partir de laquelle l'exécution maximale vérifie φ
  - $\mathsf{G}\varphi$: à partir de toute position, les exécutions maximales vérifient φ

Nouvel exemple de propriétés

[Propriété de réinitialisation] (reset property) :

- Depuis tout état du système, une suite d'actions permet de revenir à l'état initial



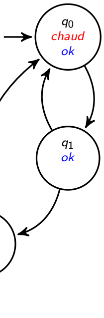
La logique CTL : préliminaires

CTL (Computation-tree logic) [Clarke & Emerson 81]

- décrit des propriétés d'un arbre de calcul (computation tree) : les formules raisonnent sur l'ensemble de toutes les exécutions regroupées au sein de l'arbre de calcul du système
- logique à temps branchant (branching-time logic)
- la sémantique d'une formule est donnée comme un ensemble d'états.

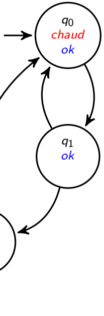
Arbre de calcul

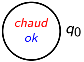
Intuitivement, dépliage potentiellement infini sans cycle d'un système (éventuellement annoté par des propositions)



Arbre de calcul

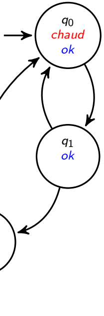
Intuitivement, dépliage potentiellement infini sans cycle d'un système (éventuellement annoté par des propositions)

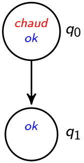




Arbre de calcul

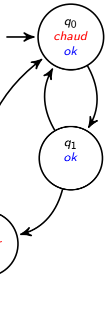
Intuitivement, dépliage potentiellement infini sans cycle d'un système (éventuellement annoté par des propositions)

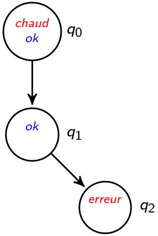




Arbre de calcul

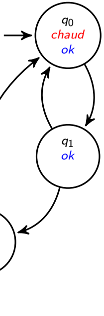
Intuitivement, dépliage potentiellement infini sans cycle d'un système (éventuellement annoté par des propositions)

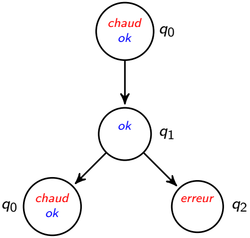




Arbre de calcul

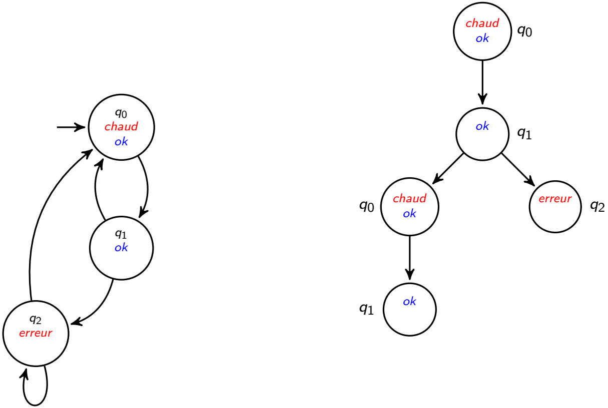
Intuitivement, dépliage potentiellement infini sans cycle d'un système (éventuellement annoté par des propositions)





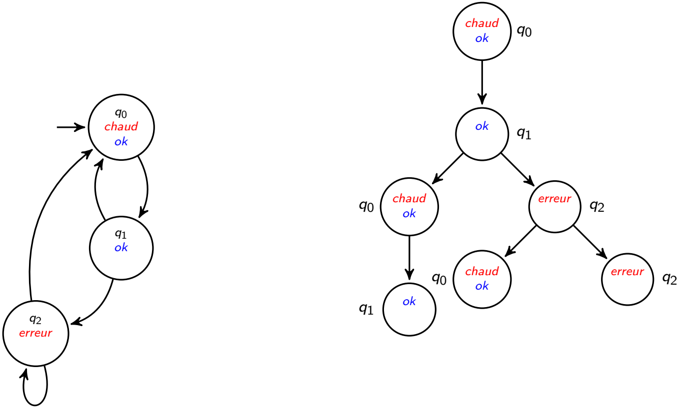
Arbre de calcul

Intuitivement, dépliage potentiellement infini sans cycle d'un système (éventuellement annoté par des propositions)



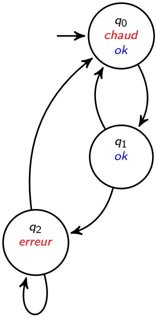
Arbre de calcul

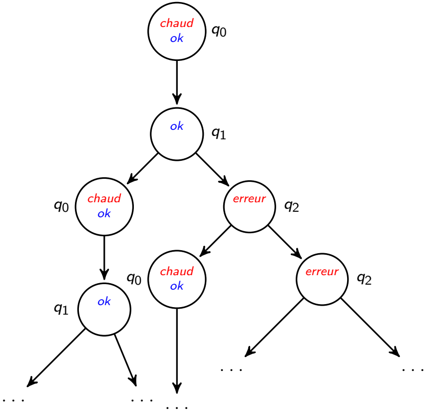
Intuitivement, dépliage potentiellement infini sans cycle d'un système (éventuellement annoté par des propositions)



Arbre de calcul

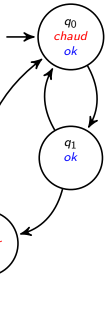
Intuitivement, dépliage potentiellement infini sans cycle d'un système (éventuellement annoté par des propositions)

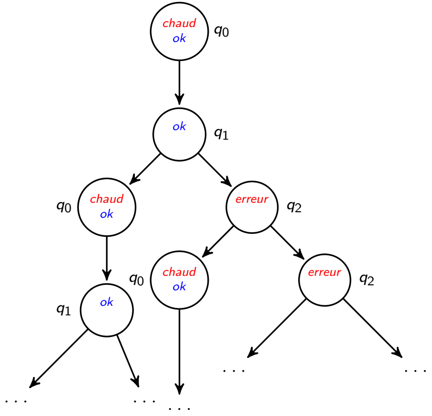




Arbre de calcul

Intuitivement, dépliage potentiellement infini sans cycle d'un système (éventuellement annoté par des propositions)





La construction de l'arbre de calcul n'est pas nécessaire pour la vérification par model-checking du système mais aide à la compréhension de CTL.

Arbre de calcul : formellement

Soit $(S, \rightarrow, s_0)$ un système fini.

Son arbre de calcul est le système (potentiellement infini) $(U, \rightsquigarrow, u_0)$ tel que

- il existe une **application** $h : U \mapsto S$ avec
- $h(u_0) = s_0$
- si $u \in U$, $h(u) = s$ et $s \rightarrow s'$ alors il existe u' dans U tel que $u \rightsquigarrow u'$ et $h(u') = s'$
- tous les éléments de u de U ont un unique antécédent par $\rightsquigarrow$ sauf u_0 qui n'en a aucun

Remarque : si $h(u_1) = h(u_2)$ alors les sous-arbres enracinés en u_1 et u_2 sont identiques.

Arbre de calcul vs ensemble d'exécutions

Chaque branche de l'arbre de calcul représente une exécution maximale

Arbre de calcul vs ensemble d'exécutions

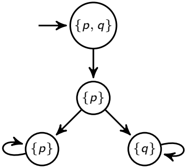
Chaque branche de l'arbre de calcul représente une exécution maximale

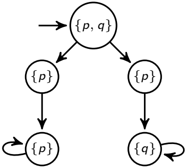
Mais l'arbre de calcul est plus précis que l'ensemble des exécutions maximales

Arbre de calcul vs ensemble d'exécutions

Chaque branche de l'arbre de calcul représente une exécution maximale

Mais l'arbre de calcul est plus précis que l'ensemble des exécutions maximales

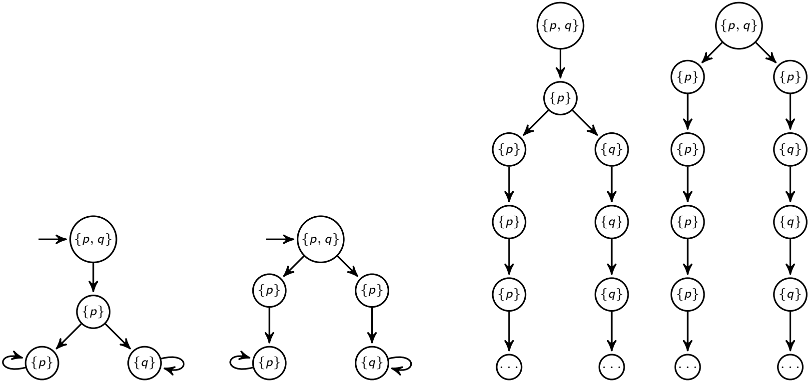




Arbre de calcul vs ensemble d'exécutions

Chaque branche de l'arbre de calcul représente une exécution maximale

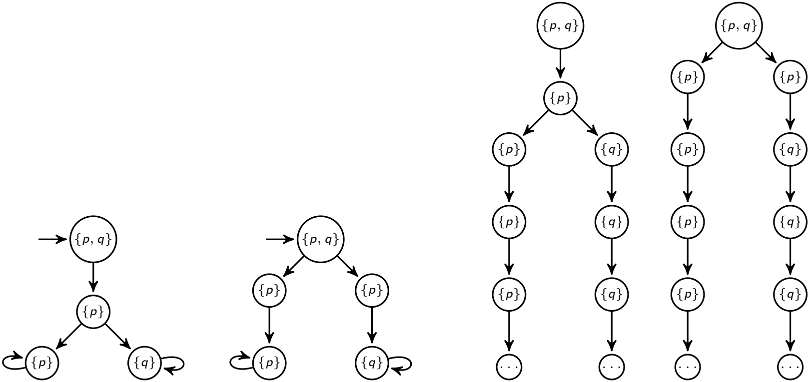
Mais l'arbre de calcul est plus précis que l'ensemble des exécutions maximales



Arbre de calcul vs ensemble d'exécutions

Chaque branche de l'arbre de calcul représente une exécution maximale

Mais l'arbre de calcul est plus précis que l'ensemble des exécutions maximales



Exécutions max. : $\{\{p,q\}\{p\}\{p\}\{p\}\{p\}\ldots,\ \{p,q\}\{p\}\{q\}\{q\}\{q\}\ldots\}$

CTL : nouveaux connecteurs temporels

Combine une quantification sur les chemins et un opérateur temporel de chemin (à la LTL)

Les opérateurs ont la forme suivante :

QT

E : il existe un chemin
A : pour tout chemin

X : NeXt
F : Finally
G : Globally
U : Until

CTL : syntaxe et sémantique

Formules de CTL :

Soit AP un ensemble de propositions et $p \in AP$

$$\begin{array}{rcl} \varphi & ::= & p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \\ & & EX\varphi \mid AX\varphi \mid EF\varphi \mid AF\varphi \\ & & EG\varphi \mid AG\varphi \mid \varphi EU\varphi \mid \varphi AU\varphi \end{array}$$

CTL : syntaxe et sémantique

Formules de CTL :

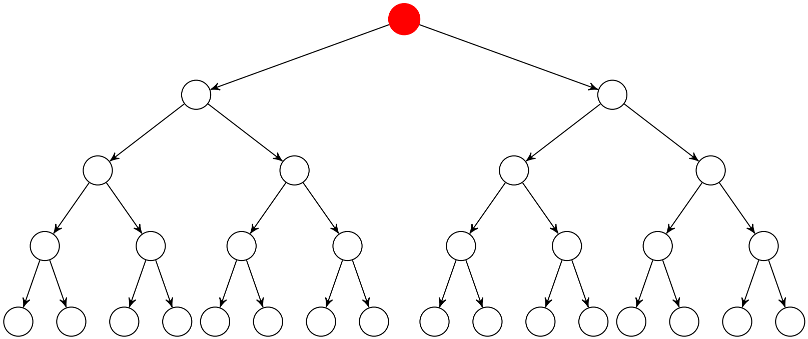
Soit AP un ensemble de propositions et $p \in AP$

$$
\begin{array}{rcl}
\varphi & ::= & p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \\
 & & EX\varphi \mid AX\varphi \mid EF\varphi \mid AF\varphi \\
 & & EG\varphi \mid AG\varphi \mid \varphi EU\varphi \mid \varphi AU\varphi
\end{array}
$$

Le système $\mathcal{S}$ vérifie la formule φ noté $\mathcal{S} \models \varphi$ si l'état initial s_0 (la racine de l'arbre de calcul) de $\mathcal{S}$ vérifie la formule φ ($s_0 \models \varphi$).

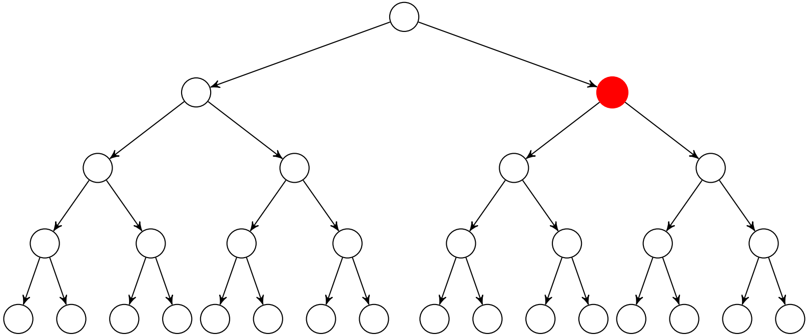
CTL : sémantique des propositions

$s_0 \models$ ●

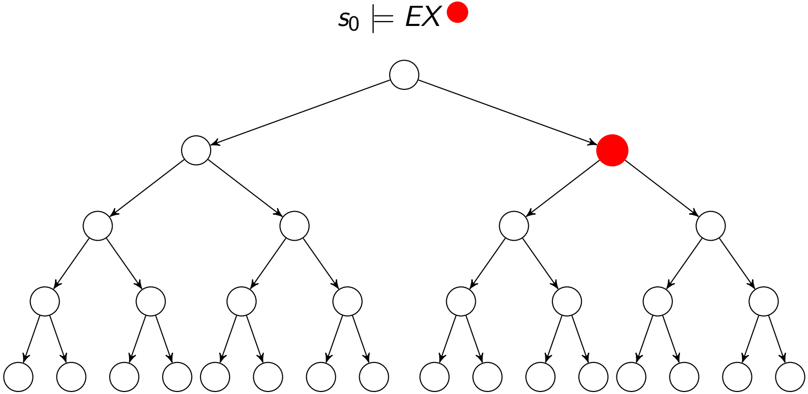


CTL : sémantique des connecteurs (I)

$s_0 \models EX \bullet$



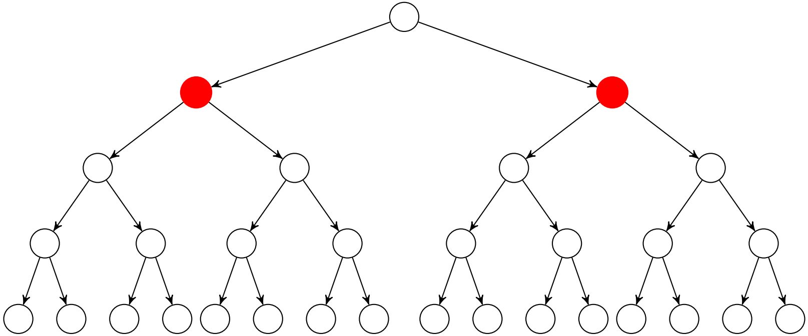
CTL : sémantique des connecteurs (I)



$s_0 \models EX\,\varphi$ si il existe un successeur s_1 de s_0 ($s_0 \to s_1$) tel que $s_1 \models \varphi$

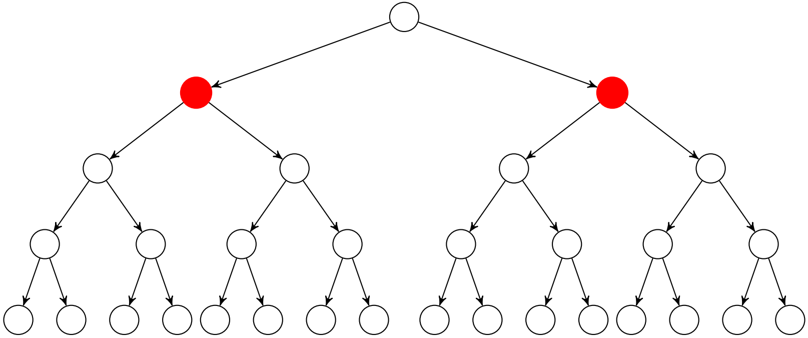
CTL : sémantique des connecteurs (II)

$s_0 \models AX\,\bullet$



CTL : sémantique des connecteurs (II)

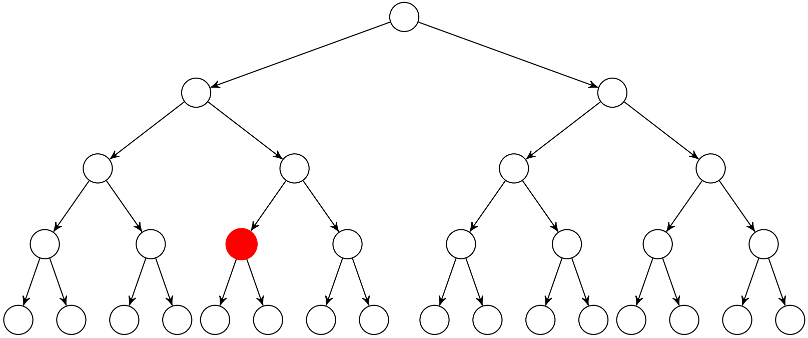
$s_0 \models AX$ ●



$s_0 \models AX\,\varphi$ si pour tout successeur s_1 de s_0 ($s_0 \to s_1$), $s_1 \models \varphi$

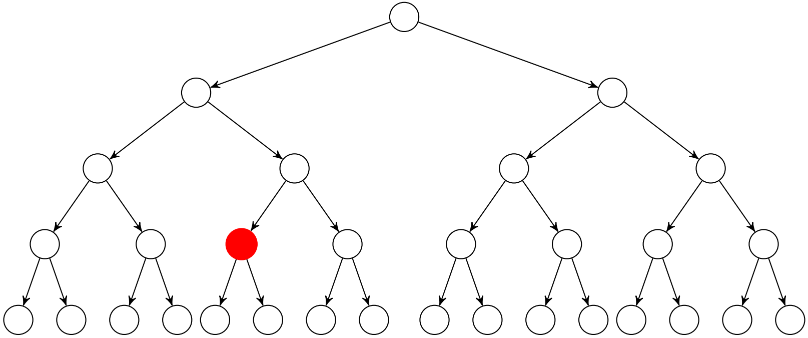
CTL : sémantique des connecteurs (III)

$$s_0 \models EF\,\bullet$$



CTL : sémantique des connecteurs (III)

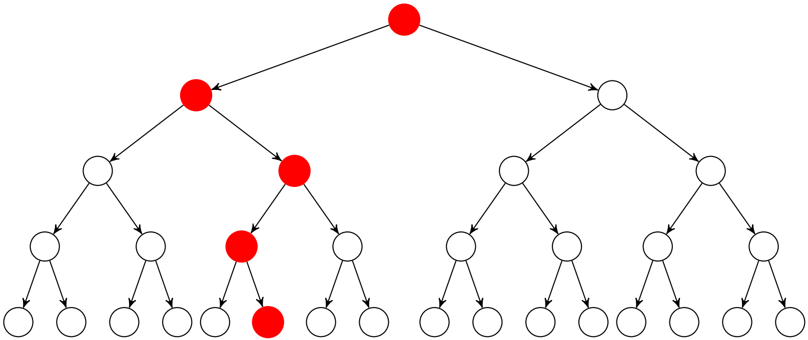
$s_0 \models EF$ ●



$s_0 \models EF\,\varphi$ s'il existe un chemin $s_0 s_1 s_2 \ldots$ et un entier k tel que $s_k \models \varphi$

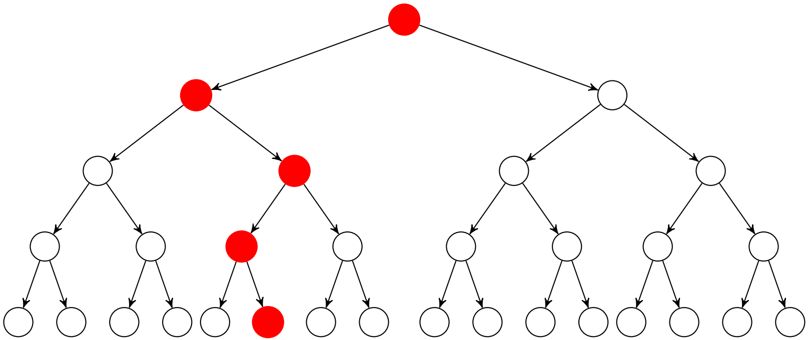
CTL : sémantique des connecteurs (IV)

$s_0 \models EG$ ●



CTL : sémantique des connecteurs (IV)

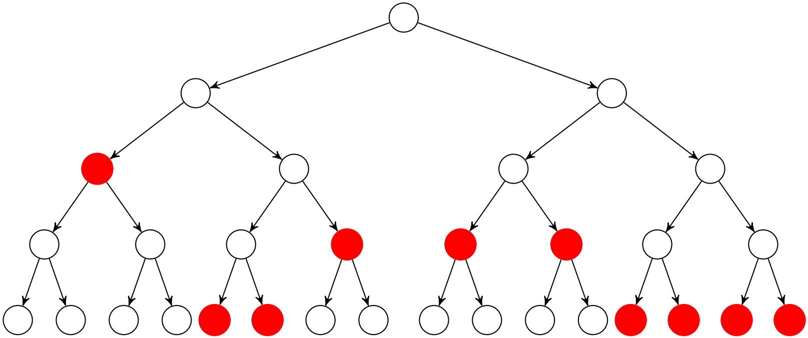
$s_0 \models EG$ ●



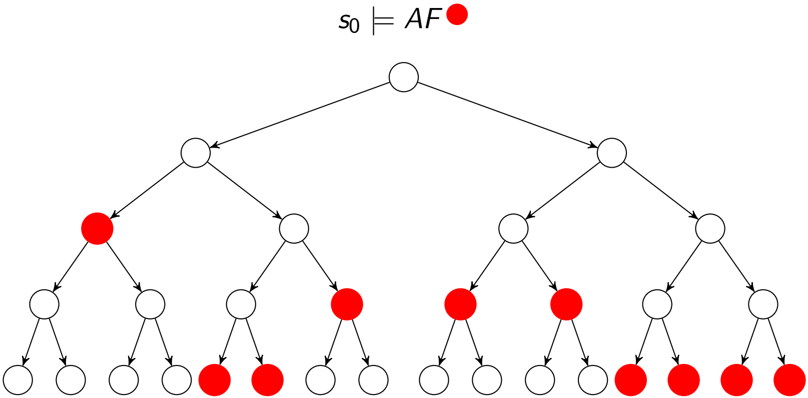
$s_0 \models EG\,\varphi$ s'il existe un chemin $s_0 s_1 s_2 \ldots$ et pour tout entier k, $s_k \models \varphi$

CTL : sémantique des connecteurs (V)

$s_0 \models AF$ ●



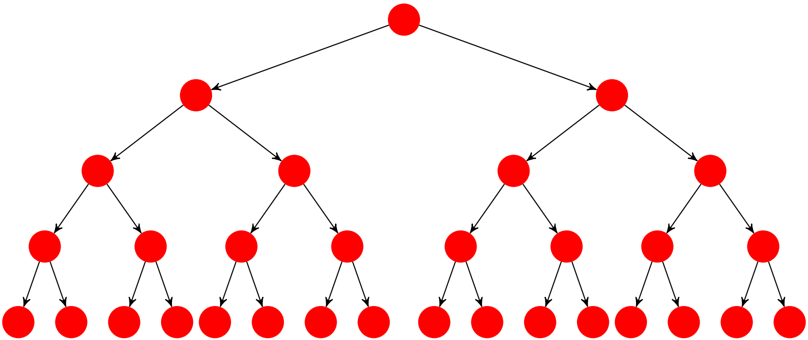
CTL : sémantique des connecteurs (V)



$s_0 \models AF\,\varphi$ si pour tout chemin $s_0 s_1 s_2 \ldots$, il existe un entier k tel que $s_k \models \varphi$

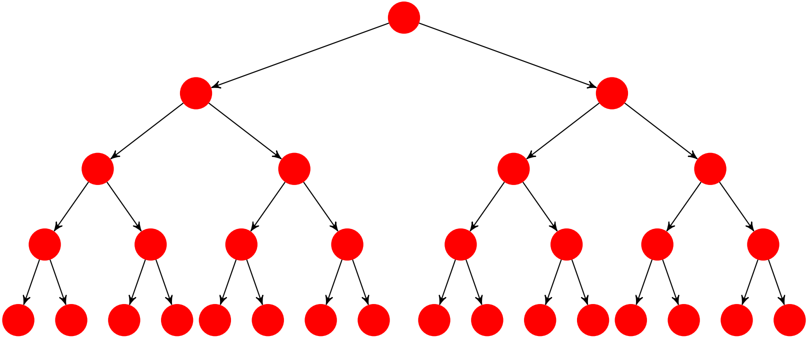
CTL : sémantique des connecteurs (VI)

$s_0 \models AG$●



CTL : sémantique des connecteurs (VI)

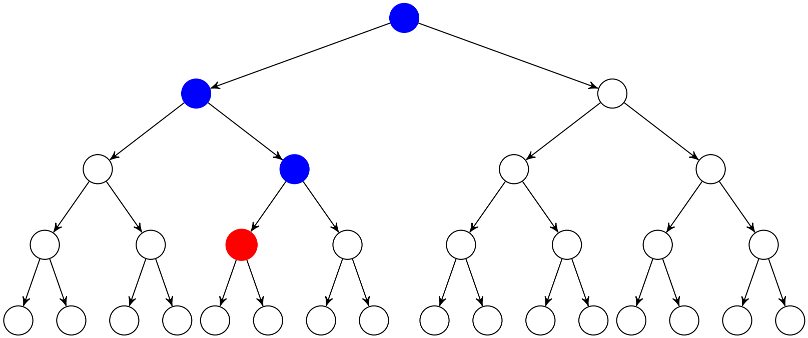
$s_0 \models AG$



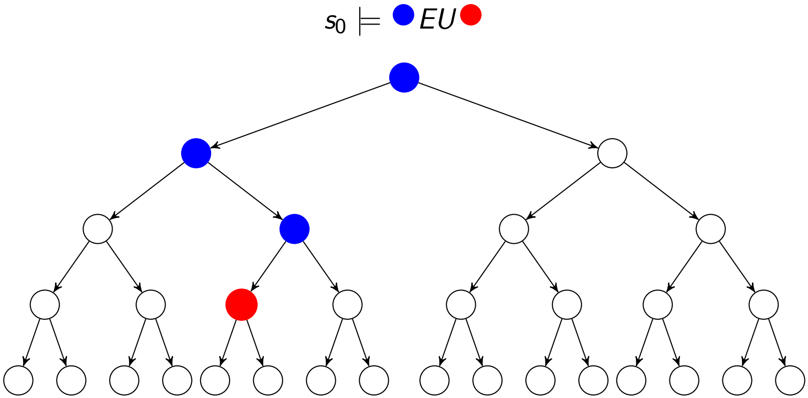
$s_0 \models AG\,\varphi$ si pour tout chemin $s_0 s_1 s_2 \ldots$, pour tout entier k, $s_k \models \varphi$

CTL : sémantique des connecteurs (VII)

$$s_0 \models \bullet\, EU \,\bullet$$



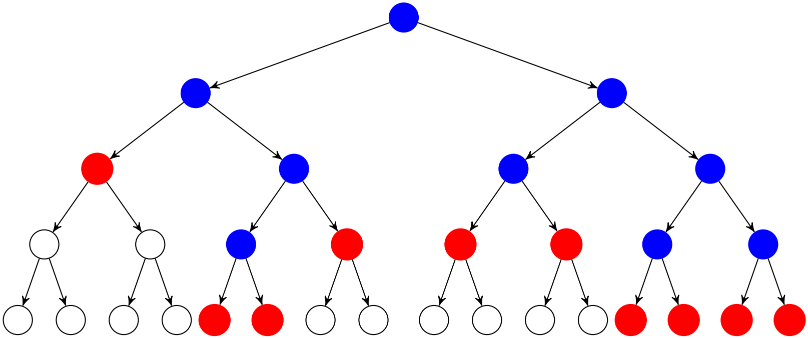
CTL : sémantique des connecteurs (VII)



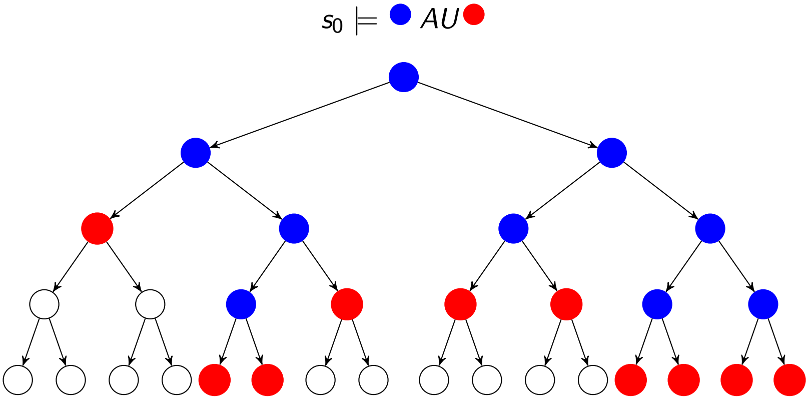
$s_0 \models \varphi_1 \, EU \, \varphi_2$ s'il existe un chemin $s_0 s_1 s_2 \ldots$ et un entier k tel que $s_k \models \varphi_2$ et pour tout $0 \leq l < k$, $s_l \models \varphi_1$

CTL : sémantique des connecteurs (VIII)

$s_0 \models$ ● AU ●



CTL : sémantique des connecteurs (VIII)



$s_0 \models \varphi_1 \, AU \, \varphi_2$ si pour tout chemin $s_0 s_1 s_2 \ldots$, il existe un entier k tel que $s_k \models \varphi_2$ et pour tout $0 \leq l < k$, $s_l \models \varphi_1$

CTL : équivalence de formules

$$AX\,\varphi \equiv \neg EX\,\neg\varphi$$

$$EG\,\varphi \equiv \neg AF\,\neg\varphi \qquad AG\,\varphi \equiv \neg EF\,\neg\varphi$$

$$EF\,\varphi \equiv true\;EU\,\varphi \qquad AF\,\varphi \equiv true\;AU\,\varphi$$

CTL : équivalence de formules

$$AX\,\varphi \equiv \neg EX\,\neg\varphi$$

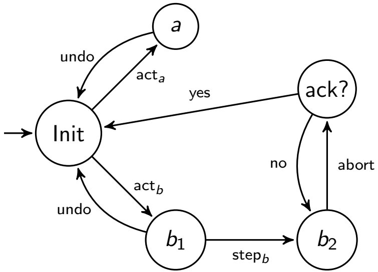
$$EG\,\varphi \equiv \neg AF\,\neg\varphi \qquad AG\,\varphi \equiv \neg EF\,\neg\varphi$$

$$EF\,\varphi \equiv \mathit{true}\;EU\,\varphi \qquad AF\,\varphi \equiv \mathit{true}\;AU\,\varphi$$

$$\varphi_1\,EU\varphi_2 \equiv \varphi_2 \vee (\varphi_1 \wedge EX\,(\varphi_1\,EU\varphi_2))$$
$$\varphi_1\,AU\varphi_2 \equiv \varphi_2 \vee (\varphi_1 \wedge AX\,(\varphi_1\,AU\varphi_2))$$

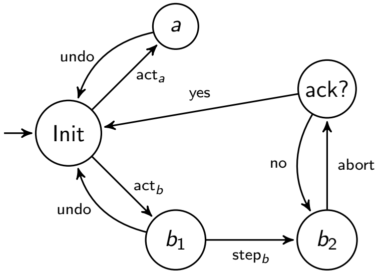
$$AG\,\varphi \equiv \varphi \wedge AX\,AG\,\varphi$$
$$AF\,\varphi \equiv \varphi \vee AX\,AF\,\varphi$$

La propriété de réinitialisation



AG EF Init

La propriété de réinitialisation



AG EF Init

La propriété de réinitialisation n'est pas exprimable en LTL

Model-checking de CTL

Pour $(\mathcal{S}, \rightarrow, s_0)$ dont les états sont annotés par des propositions de AP et une formule CTL φ, déterminer si $\mathcal{S} \models \varphi$.

Algorithme par étiquetage (model-checking global)

- on considère toutes les sous-formules de φ
- on étiquète les états de $\mathcal{S}$ par les sous-formules de φ (s porte l'étiquette φ' si $s \models \varphi'$).
- le calcul de l'étiquetage d'un état pour une formule se fait en fonction de l'étiquetage de tous les états pour les sous-formules.

Model-checking de CTL

Pour $(\mathcal{S}, \rightarrow, s_0)$ dont les états sont annotés par des propositions de AP et une formule CTL φ, déterminer si $\mathcal{S} \models \varphi$.

Algorithme par étiquetage (model-checking global)

- on considère toutes les sous-formules de φ
- on étiquète les états de $\mathcal{S}$ par les sous-formules de φ (s porte l'étiquette φ' si $s \models \varphi'$).
- le calcul de l'étiquetage d'un état pour une formule se fait en fonction de l'étiquetage de tous les états pour les sous-formules.

Pour simplifier, on considère les connecteurs $\neg, \wedge, EX, AX, EU, AU$

Model-checking de CTL : formules de base

Pour chaque état s,

- s est déjà étiqueté par la proposition p si p y est vraie.
- si $\neg\varphi$ est une sous-formule et que s n'est pas étiqueté par φ alors on étiquète s par $\neg\varphi$
- si $\varphi_1 \wedge \varphi_2$ est une sous-formule et que s est étiqueté par φ_1 et φ_2 alors on étiquète s par $\varphi_1 \wedge \varphi_2$
- si $EX\,\varphi$ est une sous-formule, que s possède un successeur par $\rightarrow$ étiqueté par φ alors on étiquète s par $EX\,\varphi$
- si $AX\,\varphi$ est une sous-formule, que tous les successeurs de s par $\rightarrow$ sont étiquetés par φ alors on étiquète s par $AX\,\varphi$

Model-checking de CTL : formule EU

- $\varphi_1 \, EU \, \varphi_2$:

  $Visited = \varnothing$
  $L = \varnothing$

  Pour tous les états s faire
  si φ_2 étiquète s alors $L = L \cup \{s\}$

  Tant que L est non vide faire
  Choisir un élément s dans L
  $L = L \setminus \{s\}$
  on étiquète s avec $\varphi_1 \, EU \, \varphi_2$

  Pour tous les prédécesseurs s' de s par $\rightarrow$ faire
  si s' n'appartient pas à *Visited* alors
  $Visited = Visited \cup \{s'\}$
  si s' est étiqueté par φ_1 alors $L = L \cup \{s'\}$

Model-checking de CTL : formule AU

- $\varphi_1 \, AU \, \varphi_2$:

  N : tableau [etat] d'entiers
  $L = \varnothing$

  Pour tous les états s faire
  $N[s] = degre(s)$
  si φ_2 étiquète s alors $L = L \cup \{s\}$

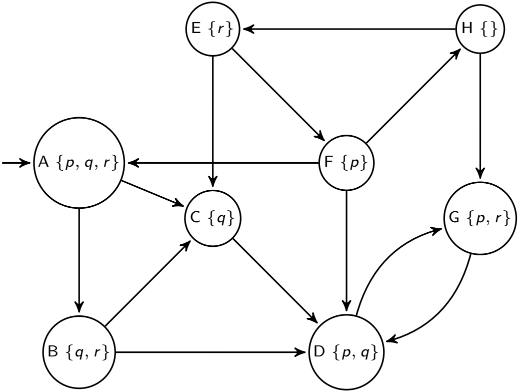
  Tant que L est non vide faire
  Choisir un élément s dans L
  $L = L \setminus \{s\}$
  on étiquète s avec $\varphi_1 \, AU \, \varphi_2$

  Pour tous les prédécesseurs s' de s par $\rightarrow$ faire
  $N[s'] = N[s'] - 1$
  si $(N[s'] == 0)$ et s' est étiqueté par φ_1
  et s' n'est pas déjà étiqueté par $\varphi_1 \, AU \, \varphi_2$ alors $L = L \cup \{s'\}$

Model-checking de CTL : Exemple

$$\varphi = AF\,AGp \;\wedge\; EXq$$

,

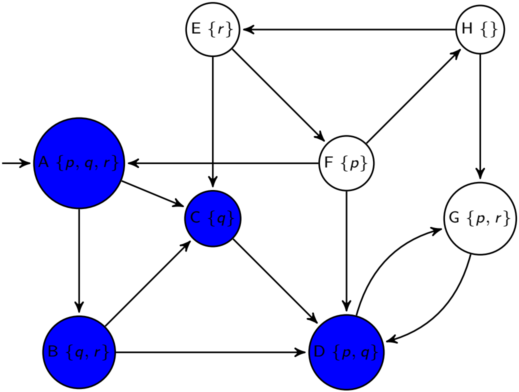


Model-checking de CTL : Exemple

$$\varphi = AF\,AGp \ \wedge \ EXq$$

,

q

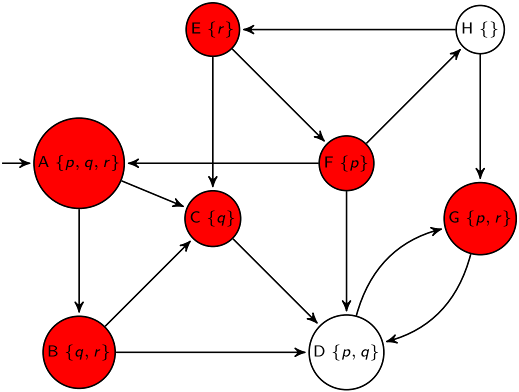


Model-checking de CTL : Exemple

$$\varphi = AF\,AGp \;\wedge\; EXq$$

,

EX q

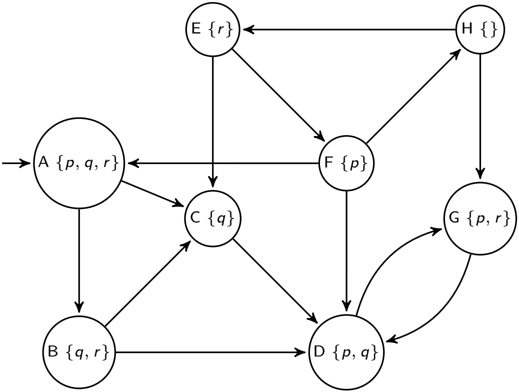


Model-checking de CTL : Exemple

$$\varphi = AF\,AGp \ \wedge \ EXq$$

,

$$AGp \equiv \neg EF \neg p$$

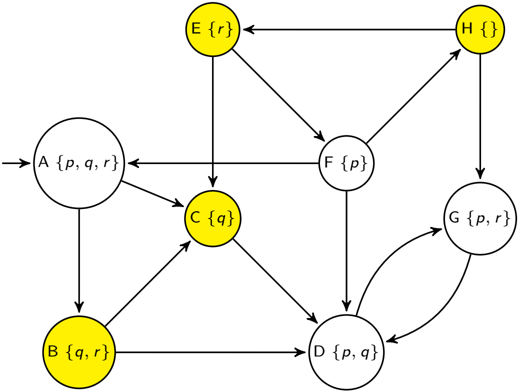


Model-checking de CTL : Exemple

$$\varphi = AF\,AGp \;\wedge\; EXq$$

,

$\neg p$

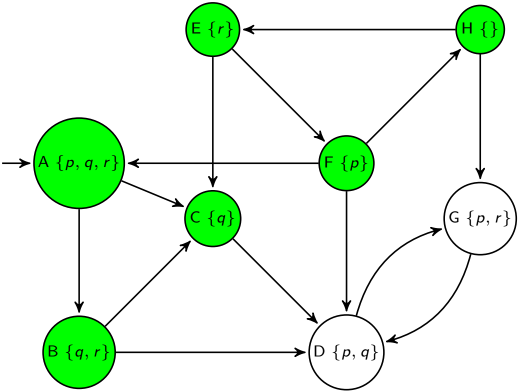


Model-checking de CTL : Exemple

$$\varphi = AF\, AGp \;\wedge\; EXq$$

,

$$EF\, \neg p \equiv true\; EU \neg p$$

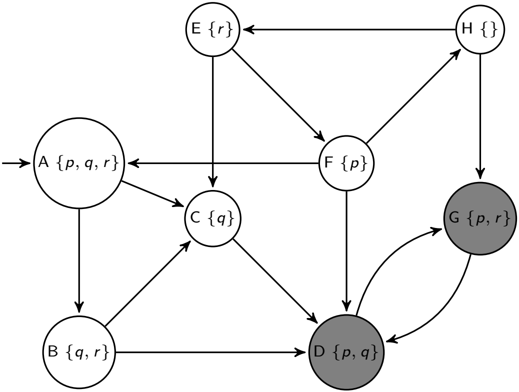


Model-checking de CTL : Exemple

$$\varphi = AF\,AGp \;\wedge\; EXq$$

,

$AG\,p$

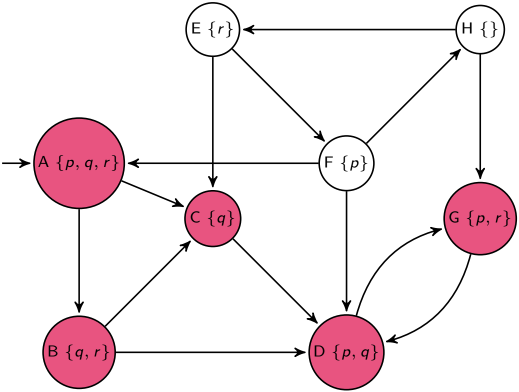


Model-checking de CTL : Exemple

$$\varphi = AF\,AGp \;\wedge\; EXq$$

,

$$AF\,AG\,p \equiv true\,AU\,AG\,p$$

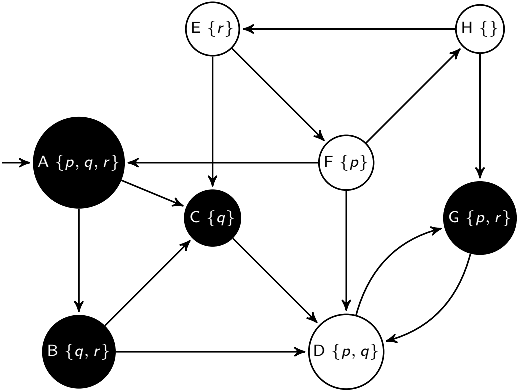


Model-checking de CTL : Exemple

$$\varphi = AF\,AGp \;\wedge\; EXq$$

,

φ



Model-checking de CTL : Complexité

$$\mathcal{S} \models \varphi$$

- n_φ le nombre de sous-formules de φ est linéaire dans la taille de φ
- une fois marqué un état le reste jusqu'à la fin ($|\mathcal{S}| * n_\varphi$ étapes d'étiquetage)
- Une étape d'étiquetage
  - $O(1)$ pour les connecteurs booléens
  - $O(|\mathcal{S}|)$ pour EX, AX
  - Pour EU, AU, chaque état n'est au plus qu'une fois dans L, polynomial en $|\mathcal{S}|$

Model-checking de CTL : Complexité

$$\mathcal{S} \models \varphi$$

- n_φ le nombre de sous-formules de φ est linéaire dans la taille de φ
- une fois marqué un état le reste jusqu'à la fin ($|\mathcal{S}| * n_\varphi$ étapes d'étiquetage)
- Une étape d'étiquetage
  - $O(1)$ pour les connecteurs booléens
  - $O(|\mathcal{S}|)$ pour EX, AX
  - Pour EU, AU, chaque état n'est au plus qu'une fois dans L, polynomial en $|\mathcal{S}|$

Le model-checking de CTL est polynomial dans la taille du système et de la formule.

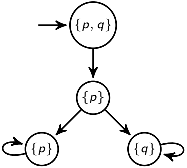
Expressivité : LTL vs CTL

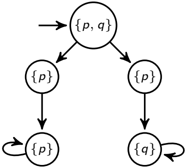
Nombreuses propriétés exprimables dans les deux logiques

- $X\varphi \Leftrightarrow AX\varphi$
- Invariance : $G\varphi \Leftrightarrow AG\varphi$
- vivacité : $G(req \rightarrow F\, grant) \Leftrightarrow AG(req \rightarrow AF\, grant)$

Expressivité : LTL vs CTL

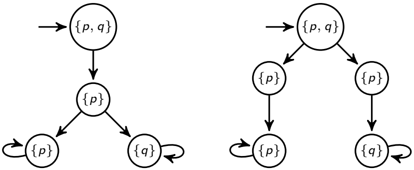
Il existe des propriétés exprimables en CTL qui ne le sont pas en LTL.





Expressivité : LTL vs CTL

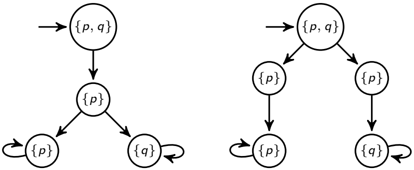
Il existe des propriétés exprimables en CTL qui ne le sont pas en LTL.



Les deux systèmes ont les mêmes exécutions maximales
Ils vérifient donc les mêmes formules LTL

Expressivité : LTL vs CTL

Il existe des propriétés exprimables en CTL qui ne le sont pas en LTL.

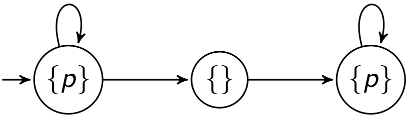


Les deux systèmes ont les mêmes exécutions maximales
Ils vérifient donc les mêmes formules LTL

Mais $AX\ EF\ \neg p$ est vraie pour le premier système mais faux pour le second

Expressivité : LTL vs CTL

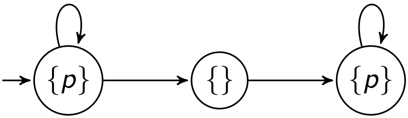
Il existe des propriétés exprimables en LTL qui ne le sont pas en CTL.



$$\mathcal{S} \models FG\, p \text{ mais } \mathcal{S} \not\models AF\, AG\, p$$

Expressivité : LTL vs CTL

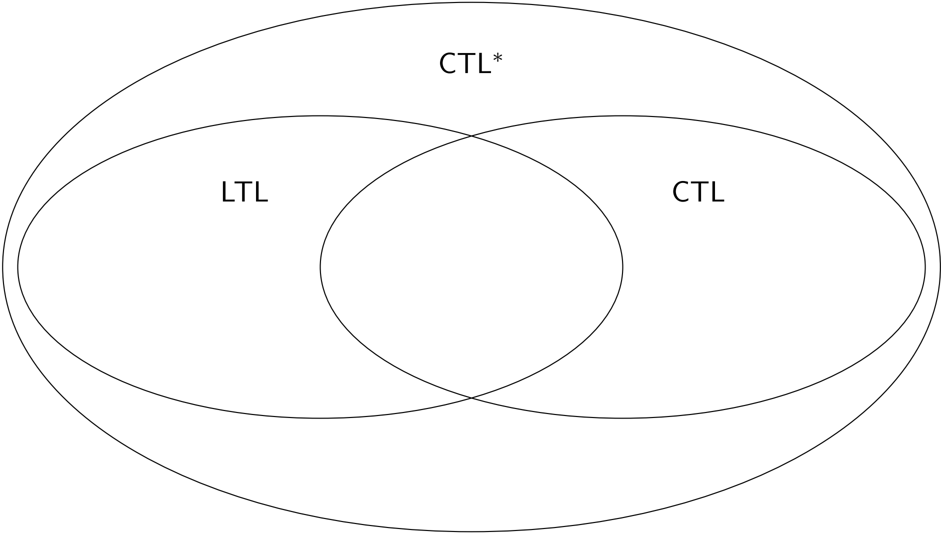
Il existe des propriétés exprimables en LTL qui ne le sont pas en CTL.



$$\mathcal{S} \models FG\,p \text{ mais } \mathcal{S} \not\models AF\,AG\,p$$

La formule $FG\,\varphi$ n'a pas d'équivalent en CTL.

La logique CTL*



La logique CTL* : définition

Idée : découpler la quantification de chemins et les connecteurs temporels

Formules d'états : $\Phi = p \mid \neg\Phi \mid \Phi \wedge \Phi \mid E\varphi \mid A\varphi$
Formules de chemins : $\varphi = \Phi \mid \neg\varphi \mid \varphi \wedge \varphi \mid X\,\varphi \mid \varphi\, U\, \varphi$

La logique CTL* : définition

Idée : découpler la quantification de chemins et les connecteurs temporels

Formules d'états : $\Phi = p \mid \neg\Phi \mid \Phi \wedge \Phi \mid E\varphi \mid A\varphi$
Formules de chemins : $\varphi = \Phi \mid \neg\varphi \mid \varphi \wedge \varphi \mid X\,\varphi \mid \varphi\, U\, \varphi$

$$\varphi_1\, EU\, \varphi_2 \text{ est en fait } E(\varphi_1\, U\, \varphi_2)$$

La logique CTL* : sémantique

$s \models p$ si p étiquète s
$s \models \Phi \wedge \Phi'$ si $s \models \Phi$ et $s \models \Phi'$
$s \models E\varphi$ s'il existe un chemin maximal w débutant en s tq $w, 0 \models \varphi$
...

$w, i \models \Phi$ si $w[i] \models \Phi$
$w, i \models \varphi_1 \wedge \varphi_2$ si $w, i \models \varphi_1$ et $w, i \models \varphi_2$
$w, i \models \varphi_1 \, U \, \varphi_2$ s'il existe $j \geq i$ tel que $w, j \models \varphi_2$
et pour tout $i \leq k < j$, $w, k \models \varphi_1$
...

La logique CTL* : expressivité

- toute formule CTL est également une formule CTL*
- Si φ une formule LTL alors $A\varphi$ est une formule CTL* équivalente.

La logique CTL* : expressivité

- toute formule CTL est également une formule CTL*
- Si φ une formule LTL alors $A\varphi$ est une formule CTL* équivalente.

Certaines propriétés exprimables par CTL* ne le sont ni par LTL, ni par CTL :

$$EX\ p \wedge AFG\ p$$

La logique CTL* : model-checking (I)

Les formules CTL* sont constituées d'une alternance de strates de formules de chemins et de strates de formules d'états.

Ainsi,

- les formules de chemins peuvent être vues comme des formules de LTL avec les formules de la strate CTL inférieure comme atomes
- les formules d'états peuvent être vues comme des formules de CTL avec des formules $E\varphi$ et $A\varphi$ comme formules atomiques (φ étant une formule de chemins appartement à la strate inférieure).

La logique CTL* : model-checking (II)

Le model-checking annote les états du système avec les formules d'états qui y sont vraies (comme CTL) et opère selon les strates croissantes :

- Pour une formule de chemin φ et pour un état donné, on considère cet état comme l'état initial et la formule de chemin comme une formule LTL avec les formules de la strate CTL inférieure comme atomes. On utilise l'algorithme de model-checking de LTL pour obtenir une réponse.
- Pour les formules d'états :
  - pour la formule $A\varphi$, on vérifie pour chaque état si tous les chemins débutant dans cet état vérifie la formule de chemin φ. Si tel est le cas, on étiquète l'état par $A\varphi$
  - pour la formule $E\varphi$, on vérifie pour chaque état si tous les chemins débutant dans cet état vérifie la formule de chemin $\neg\varphi$. Si tel n'est pas le cas, on étiquète l'état par $E\varphi$
  - Pour toutes les autres constructions de formule d'états, on procède comme pour CTL