

## Sixième partie

CTL – logique temporelle arborescente



# Plan

- 1 CTL
  - Syntaxe
  - Sémantique
- 2 Expressivité



# Computational Tree Logic

## Modèles

Une formule CTL se rapporte toujours à un **état** donné  $s$  d'un système, duquel partent des traces  $Traces(s)$ .

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 <b>toute</b> trace partant de $s$
$\forall \bigcirc P$	$P$ est vrai à l'instant suivant
$\forall \square P$	$P$ est toujours vrai à chaque état
$\forall \diamond P$	$P$ finit par être vrai (dans le futur)
$P \forall U Q$	$Q$ finit par être vrai, et en attendant $P$ reste vrai

## Quantification existentielle

Opérateurs duaux construits à partir du quantificateur  $\exists$ .

formule	interprétation (pour $s$ un état) pour <b>au moins une</b> trace partant de $s$
$\exists \square P$	$P$ est toujours vrai à chaque état
...	...

# Opérateurs minimaux

Les opérateurs minimaux sont  $\forall \circ P$ ,  $P \forall U Q$  et  $P \exists U Q$  :

- $\exists \circ P \triangleq \neg \forall \circ \neg P$
- $\forall \diamond P \triangleq \text{True} \forall U P$
- $\exists \diamond P \triangleq \text{True} \exists U P$
- $\forall \square P \triangleq \neg \exists \diamond \neg P$
- $\exists \square P \triangleq \neg \forall \diamond \neg P$
- Opérateur waiting-for
  - $P \exists W Q \triangleq \exists \square P \vee P \exists U Q$
  - $P \forall W Q \triangleq \forall \square P \vee P \forall U Q$
  - $\triangleq \neg (\neg Q \exists U (\neg P \wedge \neg Q))$

# Syntaxe alternative

On trouve fréquemment une autre syntaxe :

$\forall \leftrightarrow A$  (all)

$\exists \leftrightarrow E$  (exists)

$\square \leftrightarrow G$  (globally)

$\diamond \leftrightarrow F$  (finally)

$\circ \leftrightarrow X$  (next)

$\forall \square \exists \diamond P \leftrightarrow AG EF P$

$f \forall U g \leftrightarrow A(f U g)$



# Sémantique (système)

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

## Vérification par un système

Un système  $\mathcal{S}$  vérifie (valide) la formule  $F$  ssi tous les états initiaux de  $\mathcal{S}$  la valident :

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

Contrairement à LTL, pour toute propriété CTL, on a :

soit  $\mathcal{S} \models F$ , soit  $\mathcal{S} \models \neg F$ ,

et  $\mathcal{S} \not\models F \equiv \mathcal{S} \models \neg F$ .

# Sémantique (opérateurs logiques)

$$\overline{s \models s}$$

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

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



## Sémantique (opérateurs temporels)

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

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

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

## Sémantique (opérateurs temporels dérivés)

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

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

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

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

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

27

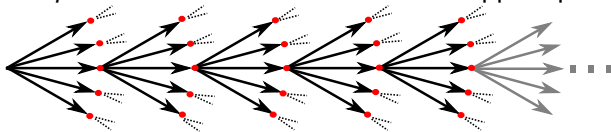
# Plan

- 1 CTL
  - Syntaxe
  - Sémantique
  
- 2 Expressivité



## Exemples amusants

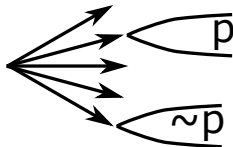
- $\exists \square \forall \circ p$  : une exécution avec une “enveloppe” qui vérifie  $p$



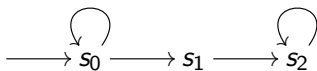
- $\exists \circ \forall \square p \wedge \exists \circ \forall \square \neg p$

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

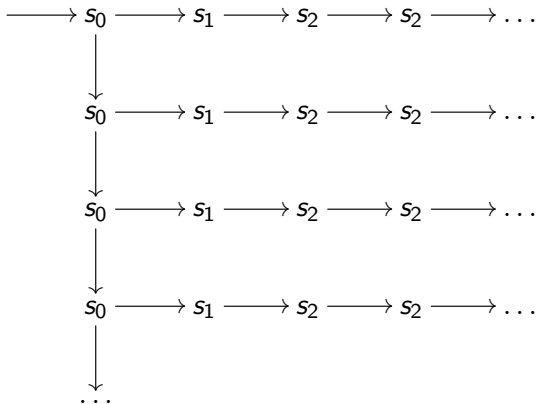


## Exemple

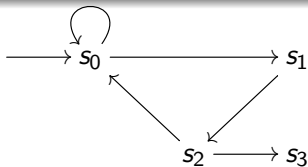


	pas d'équité	équité faible ( $s_0, s_1$ )
$s_0 \wedge \forall \bigcirc s_0$		
$s_0 \wedge \exists \bigcirc s_0$		
$\forall \square (s_0 \Rightarrow \exists \bigcirc s_0)$		
$\forall \square (s_0 \Rightarrow \exists \diamond s_2)$		
$\forall \square (s_0 \Rightarrow \forall \diamond s_2)$		
$\exists \diamond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

## Exemple - l'arbre d'exécutions



## Exemple 2



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

# Invariance, Possibilité

## Invariance

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

$$S \models \forall \square P$$

## Stabilité

Spécifier la stabilité d'une situation si elle survient :

$$S \models \forall \square (P \Rightarrow \forall \square P)$$

## Possibilité

Spécifier qu'il est possible d'atteindre un certain état vérifiant  $P$  dans une certaine exécution :

$$S \models \exists \diamond P$$



## Possibilité complexe

## Séquence

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

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

## Réinitialisabilité

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

$$\mathcal{S} \models \forall \square \exists \diamond I$$

## Possibilité arbitraire

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

$$\mathcal{S} \models \forall \square (P \Rightarrow \exists \diamond Q)$$

## Client/serveur

## Réponse

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

$$S \models \forall \square (P \Rightarrow \forall \diamond Q)$$

## 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  $Q$  :

$$S \models \forall \square (P \Rightarrow P \vee W Q)$$

# Combinaisons

## Infiniment souvent

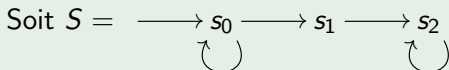
Spécifier que  $P$  est infiniment souvent vrai dans toute exécution :

$$S \models \forall \square \forall \diamond P$$

## Finalement toujours

Spécifier que  $P$  finit par rester définitivement vrai :

**impossible !**  $S \models \forall \diamond \forall \square P$  ne convient pas (trop fort)



en LTL :  $S \models \diamond \square (s_0 \vee s_2)$

mais CTL :  $S \not\models \forall \diamond \forall \square (s_0 \vee s_2)$  (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  $\mathcal{X} \in \{\forall \square, \exists \square, \forall \diamond, \exists \diamond\}$



# 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  $\langle S, I, R \rangle$  :

$$S \models I \wedge \forall\bigcirc R$$

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
si $ I =1$	$S \models \neg P \vee P$	<del><math>S \models \neg P \vee P</math></del>
l'un de $P$ ou $Q$ inévitable	<del><math>S \models \forall \diamond P \vee \forall \diamond Q</math></del> $S \models \forall \diamond (P \vee Q)$	$S \models \diamond P \vee \diamond Q$
l'un de $P$ ou $Q$ continu	<del><math>S \models \forall \square (P \vee Q)</math></del> <del><math>S \models \forall \square P \vee \forall \square Q</math></del>	$S \models \square P \vee \square Q$
$\neg P$ transitoire	<del><math>S \models \forall \diamond \forall \square P</math></del>	$S \models \diamond \square P$
possibilité	$S \models \exists \diamond P \wedge \exists \diamond \neg P$	<del><math>S \models \diamond P \wedge \diamond \neg P</math></del>
négation	$S \models \neg P \equiv S \not\models P$	$S \models \neg P \not\equiv S \not\models P$

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 CTL vs. LTL

## Linear Time Logic

- + Intuitive
- ... sauf la négation
- + Suffisante pour décrire un système de transition
- + y compris l'équité exprimable
- 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

## CTL\*

CTL\* autorise tout mélange des quantificateurs de traces  $\forall, \exists$  et d'états  $\square, \diamond, \circ, \mathcal{U}$ .

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

CTL\* est strictement plus expressif que CTL et LTL.

