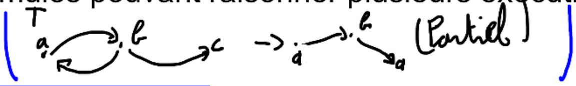
Q 9

# [redacted] 4.1 CTL syntax of the operators

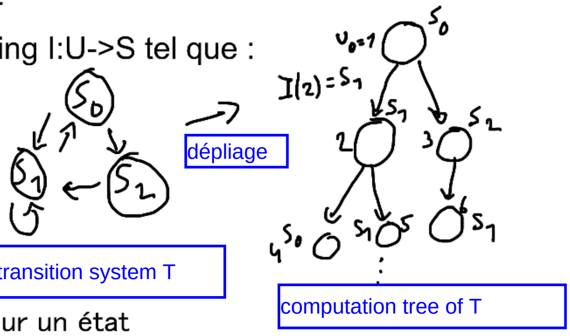
## Correspondance of CTL operators and equivalence

la logique classique ne serait pas suffisante

CTL = Computation Tree Logic
But : ~~pour~~voir de nouveaux opérateurs à la logique classique par rapport à l'exécution d'un processus.
CTL décrit les propriétés d'un arbre de calcul : formules pouvant raisonner plusieurs exécutions à la fois.
(-> Propriétés valides pour un arbre d'exécution)




Computation tree : Arbre décrivant l'exécution acyclique
Transition system T: ensemble d'états avec relations entre les états et état initial : T=<S,->,$s^0$>
Le computation tree de T : déploiement(unfolding) acyclique de T
Unfolding : Plus petit système de transition <U,->',$u_0$> avec labelling l:U->S tel que :
- $u^0 \in U$ et $l(u^0)=s^0$
- Si $u \in U$, $l(u)=s$ et $s \to s'$ alors $\exists u' \in U$ avec $u \to u'$ et $l(u')=s'$
- Tous les états de U ont un unique prédessesseur sauf $u^0$




CTL Syntax
C'est une logique donc il y a :
– operateurs classiques (AND, OR, NOT, ...)
– propositions atomiques AP : propriétés sur les états. Ex: On/Off sur un état
Opérateurs composé de 2 parties :
– quantification Q (il existe/pour tout)
– opérateurs temporels T (next/finally/globally/until)
(Exprimés sur les exécutions du système)
Opérateur : QT
Q:
E: Il existe une exécution
A: Pour toutes les exécutions
T:
X: next : le prochain état vérifie une propriété?
AX: demande que tous les prochains états possibles vérifient la propriété
EX: Au moins un des prochains états doit vérifier la propriété
G: globally: la propriété est vérifiée jusqu'à la fin de l'exécution
AG: Dans toutes les exécutions
EG: Il existe une exécution
F: finally: La propriété est vérifiée (une fois) au fil de l'exécution
AF: Pour toute exécution, on va trouver la propriété
EF: Au moins une exécution
U:until: prop p jusqu'à. ex: p vraie jusqu'à q vrai
AU: Pour toutes les exécutions
EU: Pour au moins une exécution

Syntaxe minimale :
Soit AP l'ensemble des propositions atomiques (on/off).
L'ensemble des formules CTL sur AP est le suivant:
– $a \in AP$ -> a est une formulle CTL
- Si $\phi_1,\phi_2$ sont des formules CTL alors $\boxed{NOT\phi_1}$, $\boxed{\phi_1 OR \phi_2}$, $\boxed{EX\phi_1}$, $\boxed{EG\phi_1}$, $\boxed{\phi_1 EU \phi_2}$ sont des formules CTL
Exemples d'expression :
NOT(EX On) ->Il n'existe pas "On" sur le prochain état
↳ ¬(EX on) → pas de 'ON' sur le prochain état
(EX On) OR (On EU Off)

opérateurs temporels

Kripke structure $K=(S,\rightarrow,s^0,AP,v)$:
- système de transition (états, relation entre les états, état initial)
- AP : atomic propositions
- v: fonction donnant les états valides pour toute proposition atomique

EXTENDED SYNTAX

## Équivalence :

$[\phi]_K$ : ensemble d'états satisfaisants la formule CTL $\phi$ sur AP vis à vis de K

$\phi_1$ est équivalent à $\phi_2$ si pour toute Kripke structure K : $[\phi_1]_K=[\phi_2]_K$

$\phi_1$ AND $\phi_2 \equiv \neg(\neg\phi_1$ OR $\neg\phi_2)$ (règle de Morgan)

AX $\phi \equiv \neg$EX $\neg\phi$ ($\neg$(un des nexts est $\neg\phi$))

true $\equiv$ a OR $\neg$a

AG $\phi \equiv \neg$EF $\neg\phi$ ($\neg$(à un moment on a $\neg\phi$ sur une exécution))

false $\equiv \neg$true

AF $\phi \equiv \neg$EG $\neg\phi$ ($\neg$(une exécution est globalement $\neg\phi$))

$\phi_1$EW $\phi_2 \equiv$ EG $\phi_1$ OR ($\phi_1$ EU $\phi_2$)

$\phi_1$ AW $\phi_2 \equiv \neg(\neg\phi_2$ EU $\neg(\phi_1 \cup \phi_2))$ (Aucune exécution a $\neg\phi_2$ jusqu'à avoir $\neg(\phi_1 \cup \phi_2)$)

EF $\phi \equiv$ true EU $\phi$ (un $\phi$ apparaît à un moment sur une exécution)

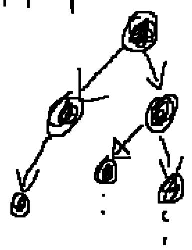
$\phi_1$ AU$\phi_2 \equiv$ AF $\phi_2$ AND ($\phi_1$ AW $\phi_2$)

W = weak until (soit $\phi_1$ vrai tous le longou $\phi_1$ vrai jusqu'à $\phi_2$ vrai)

---

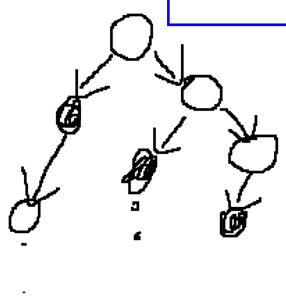
AGp

tous les noeudsont p



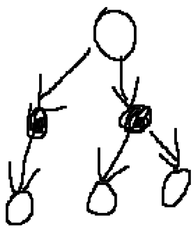
AFp

dans toute branch y aev



AXp

le suivant dans toute executionest p



PAUQ

pour tous les chemins, p est vraijusqu a ce que