Q12

[redacted] 5.1 CTL model checking - Recursive definition

vérifier si une formule est respecté pour um model

Modèle M, état s -> satisfaction : M,s|=ϕ vérifie si la formule ϕ est valide sur s
Les algorithmes de model checking fixent un modèle Krypke M = <S,->,AP,v>
et une formule ϕ et calculent $[\phi]_M=\{s\in S | M,s|=\phi\}$: dénotation de ϕ
AP = propositions atomiques, v indique quels états vérfient chaque AP
$[\phi]_M$ = l'ensemble d'états satisfaisant la formule

ceci est essentiel

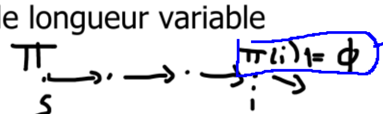
$[\phi]_M =$ ensemble d etaitsqui satisf

But : trouver un algorithme calculant $[\phi]_M$

Définition de [ϕ] récursivement à partir de ϕ, ψ et p∈AP :
[t] = S (constante true est satisfaite dans tous les états)
[f] = Ø (false jamais satisfait)
[p]=v(p), p∈AP
opérateurs logiques :
[¬ϕ] = S-[ϕ] (complément sur l'ensemble des états)
[ϕ OR ψ] = [ϕ]∪[ψ]
[ϕ AND ψ] = [ϕ] ∩[ψ]
Calculables car [ϕ] est fini
Plus complexe pour les autres formules ex : [EXϕ], [AXϕ], [EFϕ], [EGϕ], ...

Pour [EXϕ] : Calculer les prédessesseurs de [ϕ] : pre([ϕ])
Donc $\boxed{[EX\phi] = pre_\exists([\phi])}$ et $\boxed{[AX\phi] = pre_\forall([\phi])}$ avec
$pre_\exists([Y]) =\{s\in S| \boxed{\exists} s'\in S, s->s' \boxed{et}\ s'\in Y\}$ il existe 1 seul (ca suff
$pre_\forall([Y]) =\{s\in S| \boxed{\forall} s'\in S, s->s' \boxed{=>}\ s'\in Y\}$ tous les etats preceden
Y = ensemble d'états ϕ
qui respecte la propriété

Pour EF et EG le chemin n'est pas connu, il peut être de longueur variable
EF :
$[EF\phi] = \{s\in S|\ _\exists path\ \pi$ tel que $\pi(0)=s,\ _\exists i\ \pi(i)|=\phi\}$



etat qui verifie la proprie

Besoin d'une définition récursive : l'état satisfait ϕ ou le prochain
$EF_i\phi$ = il existe un chemin de longueur i dans lequel à un moment ϕ est satisfait
(S1) $EF_0\phi$ = f (false)
(S2) $EF_{i+1}\phi = \phi$ OR EX $EF_i\phi$ soit c est vrai pour l etat actuelsc
Donc :
(S3) $EF_1\phi = \phi$
(S4) $EF_2\phi = \phi$ OR EX ϕ
(S5) $EF_3\phi = \phi$ OR EX(ϕ OR EX ϕ)

vrai mtnvrai mtn ou vrai ou suivantvrai mt

Or, si |S|=n et k>n alors $[EF_k\phi] = [EF_n\phi]$
Donc, $[EF\phi]=[EF_n\phi]$ et
$[EF_0\phi] = Ø$
$[EF_{i+1}\phi] = [\phi]\cup pre_\exists([EF_i\phi])$
->Processus pour calculer [EFϕ]
Par contre nous devrions nous arrêter si nous atteignons un point fixe et ne pas calculer les n étapes (prendre le prédécesseur ne change rien)

EG : Un chemin, ϕ est satisfait sur tous le chemin
$EG_0\phi$ = t (true, chemin inexistant)
$EG_{i+1}\phi = \phi$ AND EX $EG_i\phi$ (conjonction)
Donc :
$EG_1\phi = \phi$
$EG_2\phi = \phi$ AND EX ϕ

and -> car globalement

$EG_3\phi = \phi$ AND EX(ϕ AND EX ϕ)
Si k>n:$[EG_k\phi]=[EG_n\phi]=[EG\phi]$
Les autres peuvent être trouvés par équivalence avec les opérateurs d'existence.