Q8

3.3 Operational view, definition of strategies.

But : Implémenter opérationnellement les rewrite systems

Operational view : (On donne une sémantique opérationnelle à un language.)

rewritting =

rules + méchanisme d'application (rewrite rules) + stratégie

Stratégie = La manière d'implémenter les règles et les méchanismes d'application
Ex : left-right-inner-most ou left-right-outer-most ex: t(f(x,y),g(x,y))
Il n'y a pas de stratégie optimale.

Comment choisir les règles à appliquer lors du rewriting?
Rewriting à la TOM: language pour les stratégies
Appliquer les règles de l'Abstract rewrite system avec un contrôle : par ex. choix entre les termes
Les rewrite rules sont basées sur les règles de base (données par les axiomes l->r)
Application d'une étape de rewriting sur les termes :
$Rew_{Ax}[t]: T_\Sigma \to (T_\Sigma \cup \{fail\})$, ensemble des rewrite rules. rewrite terme t en utlisant l'axiome
$\exists$ substitution σ telle que $(\sigma(l)=t) \Rightarrow Rew_{Ax \cup \{<l,r>\}}[t]=\sigma(r)$
sinon $Rew_{Ax}[t]=fail$

lors de la réecriture, soit on retourne un termesoit on retourne fail

fail : aucune règle ne peut être appliquée
Seulement les termes à l'origine peuvent être réécris.
Ex: not(true)->false OK mais not(not(true))->fail car pas d'axiome existant.

Definition Stratégie :
$Strat(S): (T_\Sigma \cup \{fail\}) \to (T_\Sigma \cup \{fail\})$
Application d'une stratégie sur le rewriting (basic rules) et sur un terme : $Strat(Rew_{Ax})[t]$
Toute stratégie fail sur le terme fail : (S)[fail]=fail

si s1() fail directement, alors la sequence entière

Opérations basiques pour les stratégies :
(Identitiy)[t]=t
(Fail)[t]=fail
(s1)[t]=fail=>(Sequence(s1,s2))[t] = fail // sequence = composition de stratégies
(s1)[t]=t'=>(Sequence(s1,s2))[t] = (s2)[t']

sequence: appliquer s1() puis s2()

(s1)[t]=t' => (Choice(s1,s2))[t]=t' // choice : choisir entre 2 stratégies
(s1)[t]=fail => (Choice(s1,s2))[t]=t' //La première si elle ne fail pas, la deuxième sinon

choice s1() et s2(), on tente d applique s1()si échéc, on applique plutot s2()

All : appliquer la même stratégie sur tous les sous-termes :
(s)[t1]=t1', ..., (s)[tn]=tn'=> (All(s))[f(t1,...,tn)]=f(t1',...,tn') (substitutivité)
Si un ti'=fail->all=fail // all(constante)=constante

il faut que tout ti marche (non f

One : Appliquer la stratégie sur un seul sous-terme : Non déterministe -> pas fonctionnel
(s)[ti]=ti'=>(One(s))[f(t1,...,tn)]=f(t1,...,ti',...,tn)
Si tous fail, One fail, One(constante)=fail

il suffit que 1 ti soit bon

Implémenter une stratégie :
Try(s)=Choice(s,Identity) (si t->t': t', sinon t)
μ = opérateur de récursion
Repeat(s)=μx.Choice(Sequence(s,x),Identity) // x peut être remplacé à l'infini par choice(Seq...)
OnceBottomUp(s)=μx.Choice(One(x),s) //Appliquer s au plus profond des sous-termes
BottomUp(s)=μx.Sequence(All(x),s) //Appliquer s partout du plus profond au moins profond
TopDown(s)=μx.Sequence(s,All(x))
Innermost(s)=μx.Sequence(All(Innermost(x)),Try(Sequence(s,x)))
-> Gérer les erreurs, avoir un terme même si un sous-terme a fail

Rewrite System t~*t' <=> Innermost(Rew)[t]=t' (aller aux sous-termes et essayer toutes les règles)
Rew = ensemble de rewrite rules, rewrite relation ~*