Q7

3.2 Properties of rewrite systems : proof of equalities, terminaison, confluence

But :
- automatiser les preuves
- évaluer les expressions (en connaissant les propriétés(termes) de cette expression)
(- prototyping. On a la définition du programme voulu, et on vérifie que c'est bien ce qu'on veut faire)

->transformer les équations (axiomes) en rewrite rules a=b-> a est réécrit en b
Ex : not true = false -> not(true)$\sim_1$false

Problèmes :
- Quelle direction? (réécrire le terme en un terme deplus faible complexité)
- Ensemble des rewriting rules est complet?
- terminaision (nombre fini d'étapes?)
- confluence (une solution finale (normale) unique pour chaque réécriture)

Abstract rewriting system : adaptation de la théorie équationelle :
Avant : théorèmes déduits des axiomes
Ici : théorèmes obtenus en transformant un terme en un autre
Definition :
Soit Σ = <S,F> une signature, X un S-sorted set de variables
-> un ARS est A=($T_{\Sigma,X}$,->), avec -> ⊆ $T_{\Sigma,X}$ x $T_{\Sigma,X}$

Propriétés :
Procédure non déterministe (choix du contexte)->Problème pour automatiser
Ex: and(not(true), not(false))

Proof of equalities :

Définition Rewrite theories:

On réécrit un terme pour prouver une propriété
Soit une specifcation Spec=<Σ,X,AX> et un ensemble de rewrite rules définies par la relation ~ Alors
Pour tout t1,t2∈$T_{\Sigma,X}$, t1 = t2∈ $Th_{\sim}$(Spec) <=> ∃t∈T_{Σ}(X) tel que t1~*t AND t2~*t

Propriétés (hypothèse nécessaire): (1) (2)
On atteint le t en un nombre finis d'étapes (terminaison) et t est unique (confluence)
(Les théorèmes qui peuvent être prouvés par l'abstract rewrite system sont les mêmes que ceux qui peuvent être prouvés par le concrete rewrite system. Et abstract rule = operational rule:
$Th_{\to}$(Spec) <=> $Th_{\sim}$(Spec)
Equational theory -> ARS. On peut utiliser la version concrète à la place.

Utiliser une graceful presentation aide pour trouver un bon rewrite system. Car :
- on examine la sémantique cas par cas sur le générateur (complétude) (2)
- On doit vérifier qu'on réduit la complexité de gauche à droite (terminaison)) (1)

Exemple de problème de terminaison : Il faudrait vérifier qu'on réduit la complexité de gauche à droite
Pour tous les termes il doit exister une forme normale.
Rewrite system :
(1) f(a,b,x)~>f(x,x,x) (2)g(x,y)~>x (3)g(x,y)~>y avec a,b cte f,g fcts, x,y variables
-> f(g(a,b),g(a,b),g(a,b))~>$_2$f(a,g(a,b),g(a,b))~>$_3$f(a,b,g(a,b))~>$_1$f(g(a,b),g(a,b),g(a,b)) -> pas de terminais

Exemple de problème de confluence :
Si le rewrite system converge, c'est sur une seule valeur (unique):
(1)f(x,x)~>a (2)f(x,g(x))~>b (3)c~>g(c)
->f(c,c)~>$_1$a et f(c,c)~>$_3$f(c,g(c))~>$_2$b -> pas de confluence