Q6

3.1 Rewrite systems, rewriting of terms, definition.

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)~$_1$false

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

réécriture

on utilise a=b ou b=a (a qui se reécrit en b

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 $T_{\Sigma,X}$

DEF -> ARS

Décomposé en steps I~r∈->
On évite la symétrie et la réflexivité sinon on risque d'avoir qqch qui ne se termine pas

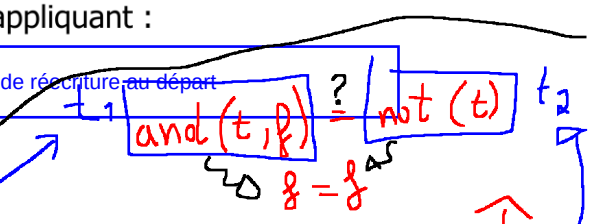
réécrire

forme normale -> irreductible form

Closure : système obtenu à partir des règles basiques en appliquant :
- ✓ La substitution
- ✓ La substitutivité
- ✓ La transitivité (Si (t,t') et (t',t'')∈->* => (t,t'')∈->*)

Closure(A)=($T_{\Sigma,X}$,->*)

système de réécriture au départ



$Rew_{\Sigma,X}$ ⊆ $T_\Sigma(X)$ x $T_\Sigma(X)$ ensemble des rewrite rules

résultat de la réécriture des deux termes t1 et t2qui r

donc ils sont égaux

Déduction de théorèmes depuis le ARS :
2 termes sont égaux dans cette théorie si :
Si t1=t2∈ $Th_{->}$(Spec) <=> ∃t∈T_Σ(X) tel que t1->*t AND t2->*t
(Théorème valide dans ARS => valide dans la théorie équationelle $Th_{->}(Spec) \subseteq Th_{eq}(Spec)$)

Processus pour le rewriting :
- filtering : choix de la règle : not(not(true))
- substitution : On met la partie droite de la règle correspondante : ~not(false)

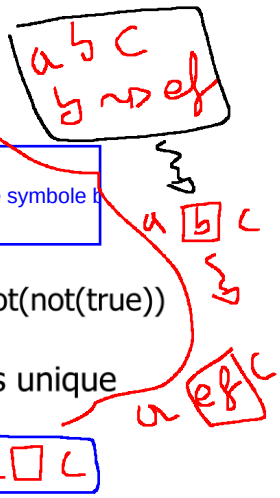
Rewriting terms : steps

on remplace l par r

Soit l~r une rewrite rule avec l,r∈T_Σ(X)
- filter(t,l) =<σ,c><=> ∃σ∈X_s->($T_{\Sigma,X}$)$_s$, ∃c tel que t=c[σl] (t = contexte appliqué à l) et t'=c[σr]
- <t,t'>∈$Rew_{l\sim r}$ un rewrite step

le contexte est un terme ou on a rajoutéle symbole □

contexte



contexte c = terme avec un trou dedans où on peut remplacer le sous-terme. Ex: not(not(true))
l = not(true), σ = une substitution, t=not(not(true)), t'=not(false), r=false
La procédure n'est pas unique (non déterministe) car le choix du contexte pas unique
Ex: and(not(true), not(false))

Exemple : S={Nat, Bool}, X_{Nat} = {x,y,z}, X_{Bool} = {a,b}
OP={+:nat,nat->nat, 0:->nat, 1:->nat, true:->Bool, false:->Bool, >:nat,nat->Bool}
Axiomes : (0)x+0=x (1)x+Succ(y)=Succ(x+y) (2)1=Succ(0) (3)(Succ(x)>0)=true
-> Rewrite rules : x+0~$_0$x, x+Succ(y)~$_1$Succ(x+y), 1~$_2$Succ(0), (Succ(x)>0)~$_3$true
Rewriting terms :
- 1>0~$_2$Succ(0)>0~$_3$true
- 1+1~$_2$Succ(0)+1~$_2$Succ(0)+Succ(0)~$_{1,\ x=Succ(0),\ y=0}$Succ(Succ(0)+0)~$_{0,x=Succ(0)}$Succ(Succ(0))