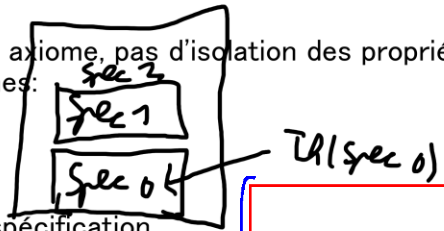
Q5 2.2

~~2.1~~ Hierarchies in Algebraic Abstract Data Types
Sufficient completeness - Hierarchical consistency

Modèle = implémentations qui satisfont les axiomes.
Problème : Avec nos modèles, il est toujours possible d'ajouter d'autres propriétés qui marchent pour le modèle sans qu'il n'y ait de contradiction. −> + de propriétés que celles qu'on peut dériver
(ex: changer les naturels par les modulos, satisfont encore = et +, cependant ils ont d'autres propriétés (3=0))
− Pas de contradictions logiques possibles (car uniquement des égalités)
− Définitions incomplètes font des objets ratachés à aucun autre (générateurs).

Les théorèmes constructibles sur M : Th(M)<=>{t1=t2|t1,t2∈ $T_{\Sigma,X}$ et M|−t1=t2}
On peut ordonner les modèles : M1≤M2 <=> Th(M1) ⊆ Th(M2)
(Ensemble minimal : modèle initial (un seul : clauses de Horn), celui avec le plus d'égalités : modèle final (Z_0))

Donc nous hiérarchisons :
specification plate (non hiérarchique) −> Pas de limite sur l'effet d'un axiome, pas d'isolation des propriétés
Besoin d'isoler les propriétés : Développement progressif des systèmes:
1. Créer Spec0 (ensemble de propriétés)
2. Créer Spec1 tel que les propriétés de spec0 ne sont pas modifiées (pas de perturbation)



Contraintes hierarchiques : reflète la décomposition du processus de spécification
Les différents modules de la spec sont implémentés séparément

imaginons qu'on spécifie tout sur les

ce qu on veut éviter

Perturbations possibles en ajoutant des propriétés:
− junk values : ajouter des éléments/valeurs non définies auparavant
− confusion : des valeurs peuvent se confondrent ou être réduites ex: avoir true = false

Modèle hierarchique : → union
Spec = ΔSpec⊕Spec0
Modèles tels que la partie du modèle (sous−modèle) respectant Spec0 préserve la sémantique de Spec0:
Si on a une implémentation des naturels, on ne change pas cette implémentation si on ajoute quelque chose

THM(Spec) = THM(Spec0 + Δ Spec) = THM(Spec) ∪ THM(Δ Spec)

Exemples of problems:

(non) Sufficient completeness :
On a 2 types(specs) : Nat, Bool
On ajoute l'opération f : Nat−>Bool
On ajoute un seul axiome : f(succ(x))=false;
−> Erreur car par déf de f on sait que f(0)∈Bool mais nous ne savons pas si c'est true ou false
(-> Nombre infinies de nouvelles valeurs->perturbe la définition des booléens)
-> Pas assez d'axiomes. Ex: f(0)=true (insuficient completeness)

Pas suff complet	
Nat	Bool

Over defined : (non) hierarchical consistency :
Même situation
Axiomes :
1 f(succ(x))=false
2 f(0)=true
3 f(succ(succ(x)))=true;
−> déduit true=false(transitivité)
Trop d'axiomes
-> Contradiction (vis à vis des booléens, pas en terme de logique)
->Perturbe le type booléen

Pas Consistant	
Nat	Bool

1 x = S(z)
⇒ f(S(S(z)) = false
3 → f(S(S(x)) = true
!

t = f