GL02 TD#5 : Spécifications algébriques

A. Type abstrait - Ensemble

Donner la spécification algébrique de la sorte « E<I> », un ensemble générique pouvant contenir des items « I ». Vous vous doterez des opérations élémentaires d'union, d'intersection et d'ajout d'un élément à l'ensemble. Vous définirez les axiomes associés aux opérations d'inspection pour la cardinalité et le test d'appartenance d'un élément à l'ensemble. (remarque : un ensemble ne contient qu'un exemplaire d'un même élément.)

1) Définissez un titre pour votre sorte, les autres sortes au(x)quelle(s) elle fait référence, ainsi que sa description.

2) Spécifiez la signature de l'ensemble des opérations.

3) Complétez les axiomes de la sorte E<I> en ajoutant ceux relatifs à l'opération d'inspection Card. Prenez garde à ne pas définir d'axiomes contradictoires (consistance) et à être le plus complet possible (les axiomes d'une opération sont à définir pour chaque opération de construction).

   Axiomes :
   Appartient(Créer, it) = Faux
   Appartient(Ajouter(ens, it1), it2) = Vrai ssi it1 = it2 ∨ Appartient(ens, it2)
   Appartient(Union(ens1, ens2), it) = Appartient(ens1, it) ∨ Appartient(ens2, it)
   Appartient(Intersection(ens1, ens2), it) = Appartient(ens1, it) ∧ Appartient(ens2, it)
   Card([…]
   Union(Ajouter(ens, it), Créer) = Ajouter(ens, it)
   Intersection(Ajouter(ens, it), Créer) = Créer

4) Déroulez la définition récursive de l'opération de cardinal sur un ensemble quelconque (par exemple, Card({3,4,8}) = …)

B. Type abstrait – Point d'intérêt (POI)

Définissez une sorte de point d'intérêt (en somme un point sur une carte). Pour l'exercice, un point d'intérêt se construit avec une latitude et une longitude. Cette spécification s'inscrivant dans un cadre de référence géographique existant, un certain nombre de préconditions sont à définir sur les valeurs de types Float que peuvent prendre latitude ([-90 ; 90]) et longitude ([-180 ; 180]) (en degré décimaux ici).

1) Définissez un nom pour votre sorte, les autres sortes auquel elle fait référence, ainsi qu'une description synthétique informelle qui précisera les contraintes sur les valeurs de latitude et longitude.

2) Définissez la signature de trois opérations (en plus d'une opération de construction) : Lat (qui permet d'obtenir la latitude d'un POI), Lng (qui permet d'obtenir la longitude d'un POI), DistanceApprox (qui calcul la distance Euclidienne[1] entre deux POI)

3) Définissez les axiomes pour les opérations définies précédemment.

4) Etant donnée une sorte d'ensemble de POI similaire à celle du premier exercice, définissez la signature et un axiome pour l'opération zonePOI qui délimite : un ensemble des POI défini entre des bornes de latitudes (min, max) et de longitudes (min, max). Utilisez une notation ensembliste pour exprimer la définition de l'opération (ex. pour un ensemble de POI, ensPoi, l'ensemble des éléments de latitude > 0 peut se définir : { el ∈ ensPoi | Latitude(el) > 0 })

---

[1] Uniquement pour simplifier l'exercice, cette approximation ne vaut que pour de courtes distances (cf. http://fr.wikipedia.org/wiki/Orthodromie)

Correction :

A.Type abstrait Ensemble

Chaque question donne à rédiger une partie de la spécification algébrique.

1) On s'occupe du titre, du nom de la sorte, de sa description et des types auxquels elle fait référence.

Titre : ENSEMBLE

Sorte : E<I>

Description : Spécifie un ensemble générique vide ou contenant des éléments de type I non spécifié. Permet l'ajout et les opérations ensemblistes d'union et intersection. L'appartenance d'un item à un ensemble peut être vérifié et le nombre de ses éléments compter. Un ensemble ne contient qu'un exemplaire d'un même élément.

Référence(s) : Entier, Booléen

2) On s'occupe de la signature des opérations qui à part la notation ne fait pas grande différence avec la définition d'une fonction dans un langage de programmation avec typage fort. (ie, l'opération Créer ne prend pas de paramètre et retourne un E<I>, Ajouter prend comme paramètre un ensemble E<I> où ajouter les élément et un élément de type indéfini/libre I et l'opération renvoi un nouvel ensemble).

Signature :

(opérations de construction primitives – les opérations qui construisent le type, on le voit au type de leur valeur de retour)
Créer : → E<I>
Ajouter : E<I> x I → E<I>
(opérations de construction secondaire)
Union : E<I> x E<I> → E<I>
Intersection : E<I> x E<I> → E<I>

(opérations d'inspection – les opérations qui donnent des informations sur le type)
Card : E<I> → Entier
Appartient : E<I> x I → Booléen

3) Pour chaque opération d'inspection et de construction secondaire, on précise le résultat ou les contraintes sur le résultat en utilisant les opérations de construction primaire comme arguments (c'est une bonne pratique afin de définir un ensemble d'axiome qui soit complet). Dans l'exercice il est juste demander de compléter les axiomes en définissant ceux de l'opération d'inspection Card() (mis en italique).

Axiomes :

Appartient(Créer, it) = Faux
Appartient(Ajouter(ens, it1), it2) = Vrai ssi it1 = it2 V Appartient(ens, it2)
Appartient(Union(ens1, ens2), it) = Appartient(ens1, it) V Appartient(ens2, it)
Appartient(Intersection(ens1, ens2), it) = Appartient(ens1, it) Λ Appartient(ens2, it)

Card(Créer) = 0
Card(Ajouter(ens, it)) = Card(ens) + 1 ssi Appartient(ens, it) = Faux
Card(Union(ens1, ens2)) = Card(ens1) + Card(ens2) - Card(Intersection(ens1,ens2))
// ou bien : Card(Intersection(ens1, ens2)) = Card(ens1) + Card(ens2) - Card(Union(ens1,ens2)), comme il s'agit bien ici du signe égale dans un contexte arithmétique (type Entier) l'une ou l'autre des formulations suffit.

Union(Ajouter(ens, it), Créer) = Ajouter(ens, it)
Intersection(Ajouter(ens, it), Créer) = Créer

Axiomes complémentaires déductibles (donc non nécessaire de les mettre dans la liste)
Appartient(Intersection(ens, Créer), it) = Faux
Appartient(Union(ens, Créer), it) = Appartient(ens, it)
Card(Intersection(ens, Créer)) = 0
Card(Union(ens, Créer)) = Card(ens)

4) Dans un ensemble il n'y pas d'ordre dans les éléments donc on les prend comme on veut
Card({3,4,8}) = Card(Ajouter({3,4}, 8)) = Card({3,4}) + 1 = Card(Ajouter({4}, 3)) + 1 = Card({4}) + 1 + 1 = Card(Ajouter({}, 4)) + 1 + 1 = Card(Créer) + 1 + 1 + 1 = 0 + 1 + 1 + 1 = 3

B. Type abstrait POI

Titre : POI

Sorte : Poi

Description : Spécifie un type de point géoréférencé en latitude et longitude (système décimal). La valeur de chaque coordonnée peut être évaluée indépendamment par les opérations Lat et Lng. La distance entre deux points peut-être calculée de façon approximative à l'aide de l'opération DistanceApprox.

Préconditions : On définit les types dérivés Lat et Lng qui sont des sous-ensembles de Float et répondant aux contraintes suivantes sur leurs valeurs possibles.

Lat = { lt ∈ Float | -90.0 ≤ lt ≤ 90.0 }

Lng = { lg ∈ Float | -180.0 ≤ lg ≤ 180.0 }

Référence(s) : Float

Signature :

CréerPoi : Lat x Lng → Poi
Latitude : Poi → Lat
Longitude : Poi → Lng
DistanceApprox : Poi x Poi → Float

Axiomes :

Latitude(CréerPoi (lt, lg)) = lt
Longitude(CréerPoi (lt, lg)) = lg
DistanceApprox(CréerPoi (lt1, lg1), CréerPoi (lt2, lg2)) = sqrt([Latitude(CréerPoi (lt1, lg1)) - Latitude(CréerPoi lt2, lg2)]^2 + [Longitude(CréerPoi (lt1, lg1)) - Longitude(CréerPoi (lt2, lg2)]^2)

Titre : ENSEMBLE(POI)

Sorte : Ens(Poi)

Références : E<I>, Poi, Booléen, Entier

Description : Spécifie un ensemble de point d'intérêt. Dispose des mêmes opérations que la sorte Ens mais ajoute la contrainte que tous les éléments de l'ensemble sont des POI. ((pour tout) ∀ el ∈ Ens(Poi) → el ∈ Poi).

Signature pour zonePoi :

zonePoi : Ens(Poi) x Poi x Poi → Ens(Poi)

Axiomes pour zonePoi :

// Pour ltmin < ltmax et lgmin < lgmax

zonePoi(Ajouter(ens, p), CréerPoi(ltmin, lgmin), CréerPoi(ltmax, lgmax)) = { el ∈ ens U {p} | ltmin < Latitude(el) < ltmax Λ lgmin < Longitude(el) < lgmax }

Pour être plus précis :

Dans l'expression précédente on présuppose l'ordre des arguments avec min et max implicite. On peut être encore plus précis en mentionnant explicitement la sorte Float dans les références et en posant qu'elle offre les opérations minFloat et maxFloat (de signature, Float x Float → Float) comparant respectivement de nombre et retournant le plus petit ou le plus grand (i.e. Axiome : minFloat(CréerFloat(a), CréerFloat(b)) → si a >= b alors b sinon a)

zonePoi(Ajouter(ens, p), CréerPoi(lt1, lg1), CréerPoi(lt2, lg2)) = { el ∈ ens U {p} | minFloat(lt1,lt2) < Latitude(el) < maxFloat(lt1,lt2) Λ minFloat(lg1, lg2) < Longitude(el) < maxFloat(lg1,lg2) }

Pour être complet et comme la sorte Ens(Poi) dispose des opérations de la sorte E<I>, il faudrait également traiter le cas où le premier argument est l'opération de création de l'ensemble.

zonePoi(Créer, CréerPoi(ltmin, lgmin), CréerPoi(ltmax, lgmax)) = Créer