1 Les ensembles

1.1 Base

Contrairement aux structures séquentielles comme les listes où l'ordre des éléments est fondamental, pour les ensembles c'est une notion sans importance. Ce qui importe, c'est qu'un élément soit présent ou non dans la structure.

TYPES
ensemble

UTILISE
élément, booléen

OPÉRATIONS

ensemblevide	: → ensemble
_ ∈ _	: élément × ensemble → booléen

AXIOMES
x ∈ *ensemblevide* = faux

1.2 Opérations supplémentaires

En plus de tester l'appartenance d'un élément à un ensemble, on doit pouvoir ajouter et supprimer un élément. On considère ici les opérations comme sans effet en cas d'erreur : lorsque l'élément à ajouter est déjà présent, ou l'élément à supprimer est absent.

OPÉRATIONS

ajouter	: élément × ensemble → ensemble
supprimer	: élément × ensemble → ensemble

AXIOMES
x = y ⇒ (x ∈ *ajouter*(y, e)) = vrai
x ≠ y ⇒ (x ∈ *ajouter*(y, e)) = (x ∈ e)
x = y ⇒ (x ∈ *supprimer*(y, e)) = faux
x ≠ y ⇒ (x ∈ *supprimer*(y, e)) = (x ∈ e)

Il faut y ajouter l'opération qui retourne le nombre d'éléments de l'ensemble, l'opération *card* définie partout.

OPÉRATIONS

card	: ensemble → entier

AXIOMES
card(*ensemblevide*) = 0 x ∈ e = vrai ⇒ *card*(*ajouter*(x, e)) = *card*(e)
x ∈ e = faux ⇒ *card*(*ajouter*(x, e)) = *card*(e) + 1
x ∈ e = vrai ⇒ *card*(*supprimer*(x, e)) = *card*(e) - 1
x ∈ e = faux ⇒ *card*(*supprimer*(x, e)) = *card*(e)

Il serait tout aussi envisageable d'ajouter les opérations classiques sur les ensembles telles que union, intersection, complémentaire et sous-ensemble.

1.3 Énumération

Pour appliquer un même traitement à tous les éléments, il faut choisir un élément, lui appliquer le traitement et recommencer sur l'ensemble obtenu en supprimant l'élément traité. On ajoute alors une opération de choix à la signature du type ensemble.

OPÉRATIONS

choisir	:	ensemble → élément

PRÉCONDITIONS

choisir(e) **est-défini-ssi** e ≠ *ensemblevide*

AXIOMES

choisir(e) ∈ e = vrai

2 Multi-ensembles

Les muti-ensembles (ou familles) sont des ensembles qui acceptent les occurrences multiples d'éléments.

Dans ces cas là, on conserve les opérations précédentes, mais leur comportement diffère. En effet, l'ajout d'un élément existant augmente le nombre d'éléments de l'ensemble. De même la suppression d'un élément n'implique pas qu'il disparaisse de l'ensemble (s'il en existe plusieurs occurrences). On ajoutera alors une nouvelle opération (*nboccurrences*) qui permet de connaître le nombre d'occurrences d'un élément dans le multi-ensemble.

Ce qui donne le type algébrique abstrait suivant :

TYPES

multiensemble

UTILISE

élément, booléen, entier

OPÉRATIONS

multiensemblevide	:	→ multiensemble
ajouter	:	élément × multiensemble → multiensemble
supprimer	:	élément × multiensemble → multiensemble
_ ∈ _	:	élément × multiensemble → booléen
card	:	multiensemble → entier
nboccurrences	:	élément × multiensemble → entier

AXIOMES

(x ∈ *multiensemblevide*) = faux
x = y ⇒ (x ∈ *ajouter*(y, me)) = vrai
x ≠ y ⇒ (x ∈ *ajouter*(y, me)) = (x ∈ me)
nboccurrences(x, me) <= 1 ⇒ (x ∈ *supprimer*(x, me)) = faux
x ≠ y ⇒ (x ∈ *supprimer*(y, me)) = (x ∈ me)

card(*multiensemblevide*) = 0
card(*ajouter*(x, me)) = *card*(me) + 1
x ∈ me = vrai ⇒ *card*(*supprimer*(x, me)) = *card*(me) - 1
x ∈ me = faux ⇒ *card*(*supprimer*(x, me)) = *card*(me)

nboccurrences(x, *multiensemblevide*) = 0
x = y ⇒ *nboccurrences*(x, *ajouter*(y, me)) = *nboccurrences*(x, me) + 1
x ≠ y ⇒ *nboccurrences*(x, *ajouter*(y, me)) = *nboccurrences*(x, me)
nboccurrences(x, me) = 0 ⇒ *nboccurrences*(x, *supprimer*(x, me)) = 0
nboccurrences(x, me) ≥ 1 ⇒
nboccurrences(x, *supprimer*(x, me)) = *nboccurrences*(x, me) - 1
x ≠ y ⇒ *nboccurrences*(x, *supprimer*(y, me)) = *nboccurrences*(x, me)