Informatique théorique : feuille numéro 2
Ensembles (1)

1 Plan du cours

1.1 Définitions de base

- Utilisation en informatique : exemples : utilisateurs d'un système, réponses possibles à une requête, sommets d'un graphe, voisins d'un sommet, etc.
- Comparaison avec les notions de suite et de multi-ensembles : exemple du tri pour les multi-ensembles, de file pour les suites. En programmation, le comportement des itérateurs diffère en fonction de la structure choisie.
- Définition par compréhension (sur un type ou en ensemble *existant*) et appartenance. Inclusion, union, intersection, différence, et leur définition logique. La définition par extension se traduit facilement en termes de compréhension.
- union, intersection généralisées
- produit cartésien, partitions

1.2 Exemples élaborés

- Utilisateurs d'un systèmes, droits d'accès en fonction d'une arborescence
- Une spécification décrit un ensemble de réponses possibles. Comparaisons entre spécifs, réalisations.
- Le produit cartésien permet de décrire un système à plusieurs variables. Ensembles d'états intiaux, d'états satisfiant une contrainte.

1.3 Plus sur les types et les ensembles

- Le paradoxe de Russel.
- La notion admise de type permet d'éviter les définitions paradoxales
- rapport avec les types en informatique.

1.4 Notation

Soit n et p deux nombres entiers (éléments de $\mathbb{Z}$). On notera $n..p$ l'ensemble $\{i : \mathbb{Z} \mid n \leq i \leq p\}$.

Combien d'éléments ont les ensembles $0..7$, $7..7$, $8..7$?

2 Exercices

2.1

Soient A, B, C trois ensembles sur un même type.

- Montrer que les deux propositions suivantes sont équivalentes :
  $(A \cup B) \subseteq C$ et $A \subseteq C \wedge B \subseteq C$
- Montrer que $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$.
- A-t-on $A \cap B = A \cup B \Rightarrow A = B$?
- A-t-on $A \cup B = A \cup C \Rightarrow B = C$?
- Montrer que $A \cap C = A \cup B$ si, et seulement si, $B \subseteq A \subseteq C$.
- Montrer que si $A \cup B \subseteq A \cup C$ et $A \cap B \subseteq A \cap C$, alors $B \subseteq C$.

2.2

Dans le cadre d'un tri par sélection d'un tableau t de taille n, on considère un indice k et les propositions

- $\forall i.\, i \in 0..k-2 \Rightarrow t[i] \leq t[i+1]$
- $\forall i, j.\, i \in 0..k-1 \wedge j \in k..n-1 \Rightarrow t[i] \leq t[j]$

Pour quelles valeurs de k ces propositions ont-elles un sens ? Existe-t-il une valeur de k rendant ces propositions toujours vraies ?

2.3

- Soit E un ensemble d'entiers. Soit P la propriété

$$\forall x \in E.x^2 \geq n$$

. Quelle est la négation de P ? On suppose maintenant que $E = \emptyset$. La négation de P est-elle vraie ou fausse ? P est-elle vraie ou fausse ?

2.4

Soient A,B,C,D quatre ensembles.

Comparer pour la relation d'inclusion et/ou d'égalité les paires d'ensembles ci-dessous (on considère que l'opérateur $\times$ est prioritaire sur $\cup$ et $\cap$.)

On s'appuiera sur des preuves ou des contre-exemples pour justifier ses réponses.

- $A \times (C \cup D)$ et $A \times C \cup A \times D$
- $A \times (C \cap D)$ et $A \times C \cap A \times D$
- $A \times C \cup B \times D$ et $(A \cup B) \times (C \cup D)$

Sur quels types les ensembles A, B, C et D peuvent-ils être définis pour que les questions ci-dessus aient un sens ?

2.5

On considère le problème de signalisation vu en cours.
Un *état* du système peut se décomposer en plusieurs informations :
- w_A : nombre de trains attendant en gare A pour aller vers B
- w_B : nombre de trains attendant en gare B pour aller vers A
- c_A : couleur du feu en gare A (rouge ou vert)
- c_B : couleur du feu en gare B (rouge ou vert)
- n_{AB} : nombre de trains sur la voie unique allant de A vers B
- n_{BA} : nombre de trains sur la voie unique allant de B vers A

En utilisant la notion de produit cartésien, exprimer les ensembles suivants :
- Ensemble de toutes les situations (états) possibles
- Ensemble de tous les états à éviter
- Le complémentaire du précédent
- Ensemble des états non dangereux, mais qu'on ne souhaite pas se voir prolonger indéfiniment

2.6

1. Soit le chemin absolu d'un fichier $fich$ dans une arborescence Unix : `/d1/d2/.../dn/fich`. Pour tout i entre 1 et n, les ensembles R_i, W_i, X_i contiennent les utilisateurs qui ont respectivement le droit de lecture, écriture et exécution sur le répertoire d_i. De même R, W, X sont les ensembles des utilisateurs qui ont respectivement le droit de lecture, écriture et exécution sur le fichier `fich`.
   Exprimer formellement l'ensemble des utilisateurs qui peuvent modifier ou effacer le fichier `fich` à partir de la racine de l'arborescence.

2. Un système de fichiers a trois groupes d'utilisateurs : les dirigeants, les employés, les clients. Le système de fichiers gère quels utilisateurs ont accès à quels fichiers. Les ensembles R_d , R_e et R_c contiennent respectivement les fichiers que les dirigeants, les employés et les clients ont le droit de lire. De même W_d , W_e et W_c contiennent les ensembles de fichiers que les dirigeants, les employés et les clients peuvent modifier. Tous les objets dans ces ensembles sont du type FILE.
   Exprimer les énoncés suivants sous forme de prédicats :
   - le fichier `passwd` peut seulement être modifié par les dirigeants,
   - les dirigeants n'ont pas moins de droits que les employés : si les employés peuvent lire un fichier, alors les dirigeants le peuvent ; si les employés peuvent modifier un fichier, alors les dirigeants le peuvent ;
   - tous les fichiers peuvent être lus par au moins un groupe d'utilisateurs,
   - aucun fichier ne peut être modifié sans être lu par un groupe d'utilisateurs,
   - tout fichier que les clients peuvent à la fois lire et modifier doit être lisible par les employés.

2.7

Dans un graphe G représenté par deux ensembles V et $E \subseteq V \times V$, traduire en termes ensemblistes :

- G est non orienté,
- L'ensemble des voisins d'un sommet
- L'ensemble des sommets à distance ≤ 2 d'un sommet
- G est complet
- G est un anneau

2.8

Étudier les représentations possibles d'ensembles dans un langage de programmation, avec plusieurs cas :

- ensembles sur un type à n éléments (n connu)
- ensembles finis de nombres
- ensembles finis sur des types plus complexes
- ensembles pouvant être infinis

Pour chaque cas, discuter quelles fonctionnalités sont permises parmi :

- tester l'appartenance
- représenter l'union, l'intersection, la différence
- tester si un ensemble est inclus dans un autre
- programmation d'un itérateur :

```
pour tout élément x de E
faire
      ...
fait
```