Programmation avancée 20 novembre 2008

Mini-projet 5 : Flots et contraintes

NB. Respectez les noms et les signatures indiquées dans les canevas et les données des exercices !

Première partie : Flots

Exercice 1

$1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \frac{1}{5} - \cdots$ est une suite qui converge vers $\ln(2)$.

1. Définissez un flot formé des éléments de cette suite, en utilisant le même principe que celui vu au cours pour la suite convergeant vers π.
2. Appliquez la technique d'accélération d'Euler pour obtenir un flot qui converge plus rapidement que le précédent.
3. Appliquez ensuite la technique des tableaux pour obtenir une série convergeant encore plus rapidement.

Au bout de combien d'itérations arrivez-vous à un résultat jusqu'à la 10^{e} décimale, sachant que $\ln(2) \approx 0.693147180559945309$? Estimez le nombre d'itérations qui auraient été nécessaires pour obtenir cette précision avec la série non-accélérée, et estimez ainsi l'accélération totale fournie par l'utilisation des tableaux et de la technique d'Euler.

Exercice 2

Nous nous intéressons maintenant au flot des pairs d'entiers (i, j) avec $i, j \geq 0$.

1. Définissez `fromTo` qui retourne le flot de tous les entiers n pour $l \leq n \leq u$.
   ```
   def fromTo(l: Int, u: Int): Stream[Int] = ...
   ```
2. Définissez `intPairs` qui retourne le flot de toutes les pairs d'entiers (i, j) pour $i, j \geq 0$ *en utilisant des compréhensions for*.
   ```
   def intPairs: Stream[(Int, Int)] = ...
   ```
3. Définissez une fonction équivalente à la fonction précédente *sans utiliser de compréhension for*

Exercice 3

Nous souhaitons construire des flot d'entiers qui appartiennent à la séquence suivante :

```
1
1 1
2 1
1 2 1 1
1 1 1 2 2 1
3 1 2 2 1 1
```

Quel est le prochain élément de cette séquence de flot ? Indication : une nouvelle séquence intercale les éléments de la séquence précédente avec des compteurs qui comptent les éléments respectifs.

1. Ecrivez une fonction `countStream` qui prend un flot en argument et calcule le prochain élément dans la séquence ci-dessus. Cela signifie,

   ```
   countStream(Stream.cons(2, Stream.cons(1, Stream.empty)))
   = Stream.cons(1, Stream.cons(2,
       Stream.cons(1, Stream.cons(1, Stream.empty))))
   ```

   La fonction a la signature suivante :

   ```
   def countStream(s: Stream[Int]): Stream[Int] = ...
   ```

2. Ecrivez une fonction `allCountStreams` qui calcule un flot de tous les flots obtenus avec `countStream`. La fonction a la signature suivante :

   ```
   def allCountStreams: Stream[Stream[Int]] = ...
   ```

Seconde partie : résolution de contraintes

Vous devez améliorer le système de résolution de contraintes présenté au cours. Ce système est composé de *quantités* et de *contraintes*.

- Une quantité contient une valeur, qui peut être définie ou non, et relie un certain nombre de contraintes. Les quantités sont les arêtes d'un réseau de contraintes.
- Une contrainte lie plusieurs quantités par une équation simple. Les contraintes sont les nœuds d'un réseau de contraintes.

Une contrainte d'addition, par exemple, modélise une équation de la forme $a + b = c$ où a, b et c sont des quantités.

Exercice 1

Vous trouverez sur la page Web du cours une implantation partielle du système de contraintes que vous compléterez.

- Définissez une contrainte de multiplication modélisant une équation de la forme $a \times b = c$; cette contrainte doit *savoir* que $0 \times x = x \times 0 = 0$ quel que soit x, donc même si x est indéfini,
- Définissez une contrainte d'égalité modélisant une équation du type $a = b$.

Pour chacune de ces nouvelles contraintes, ajoutez un opérateur à la classe `Quantity` qui permette l'écriture agréable de contraintes, en vous inspirant de l'opérateur `+` déjà défini. Nommez l'opérateur pour les contraintes de multiplication $\star$, et celui pour les contraintes d'égalité `===`.

Exercice 2

Est-il possible d'implanter une équation d'élévation au carré de la forme $a^2 = b$ au moyen de votre contrainte de multiplication ? Si oui, implantez-la, si non, expliquez pourquoi et donnez une solution directe qui n'utilise pas d'autres contraintes.