▶ **MP & MP* – Option Informatique**
Année 2002–2003, Troisième TP Caml
Vincent Simonet (http://cristal.inria.fr/~simonet/)

Commutation d'interrupteurs

D'après l'épreuve du concours 2000 de l'École polytechnique

Nous considérons un système commandé par un tableau de bord comportant N interrupteurs, chacun pouvant être baissé ou levé. On désire tester ce système (pour le valider ou pour effectuer une opération de maintenance) en essayant mécaniquement chacune des 2^N configurations possibles pour l'ensemble des interrupteurs. Le coût de cette opération, qu'elle soit réalisée par un opérateur humain ou par un robot, sera le nombre total de mouvements d'interrupteurs nécessaires. Nous supposons que chaque fois qu'un interrupteur est commuté le système effectue un diagnostic automatiquement et instantanément. Les interrupteurs sont indexés de 0 à $N-1$.

1 Parties d'un ensemble

Nous appelons *partie* un sous-ensemble fini de l'ensemble $\mathbb{N}$ des entiers naturels. Un élément d'une partie est appelé *indice*. La *différence symétrique* de deux parties P et Q est définie par :

$$P \vartriangle Q = (P \backslash Q) \cup (Q \backslash P) = (P \cup Q) \backslash (P \cap Q)$$

On vérifie facilement que la différence symétrique est commutative et associative, ce que l'on ne demande pas de démontrer. Pour tout entier n positif ou nul, nous notons $\mathcal{I}_n = \{0, \ldots, n-1\}$ l'ensemble des entiers inférieurs strictement à n et $\mathcal{P}_n$ l'ensemble des parties de $\mathcal{I}_n$.
Une partie sera représentée par une liste chaînée d'indices distincts **apparaissant dans l'ordre croissant des entiers**. On utilisera les types suivants :

```
type indice == int;;
type partie == indice list;;
```

▶ **Question 1** *Écrivez la fonction* card *qui retourne le nombre d'éléments d'une partie. Vous ne ferez bien entendu pas appel à la fonction de bibliothèque* list_length .

card: partie −> int

▶ **Question 2** *Écrivez la fonction* delta *qui réalise la différence symétrique de deux parties. Le nombre d'opérations ne devra pas excéder $O(m + n)$ où n et m sont les cardinaux des éléments. Nous rappelons que dans toute liste chaînée représentant une partie les indices sont distincts et doivent apparaître dans l'ordre croissant des entiers.*

delta : partie −> partie −> partie

Nous représentons une configuration des interrupteurs par la partie formée des indices des interrupteurs baissés.

▶ **Question 3** *Écrivez une fonction* test *qui imprime la liste des indices d'interrupteurs à commuter pour passer d'une configuration à une autre.*

test: partie −> partie −> unit

Pour imprimer un entier i, pour imprimer un espace ou pour imprimer un saut de ligne, vous pourrez utiliser respectivement :

```
print_int i;
print_string " ";
print_newline ();
```

2 Énumération des parties par incrément

À toute partie P, nous associons l'entier $e(P) = \sum_{i \in P} 2^i$ (avec la convention $e(\emptyset) = 0$). Nous définissons le successeur de P comme l'unique partie Q telle que $e(Q) = e(P) + 1$.

▶ **Question 4** *Écrivez une fonction* succ *qui retourne le successeur d'une partie. Le nombre d'opérations ne devra pas excéder $O(\ell)$ où ℓ est le plus petit indice absent dans la partie donnée en argument.*

succ: partie −> partie

En application de ce mode d'énumération des parties, nous voulons réaliser le test de toutes les configurations d'interrupteurs. **Au début et à la fin du test tous les interrupteurs seront levés**.

▶ **Question 5** *Écrivez un programme qui imprime la liste des indices des interrupteurs à commuter pour réaliser la totalité du test pour N interrupteurs et qui examine les configurations dans l'ordre défini par le successeur. L'argument de cette fonction sera l'entier N.*

test_incr: int −> unit

▶ **Question 6** *Exprimez, en fonction de N, le nombre total de commutations d'interrupteurs à effecuer pour réaliser le test de cette manière.*

3 Énumération des parties par un code de Gray

Nous notons $\langle u_0, \ldots, u_{\ell-1} \rangle$ une suite finie de ℓ entiers. La concaténation de deux suites finies de longueur ℓ et ℓ' respectivement est une suite finie de longueur $\ell + \ell'$ définie par

$$\langle u_0, \ldots, u_{\ell-1} \rangle \odot \langle u'_0, \ldots, u_{\ell'-1} \rangle = \langle u_0, \ldots, u_{\ell-1}, u'_0, \ldots, u_{\ell'-1} \rangle$$

La suite vide, notée $\langle\rangle$ est de longueur 0. Une suite finie U est *préfixe* d'une autre suite finie V s'il existe une suite finie W telle que $V = U \odot W$ (autrement dit U est le début de V).

Pour tout entier n positif ou nul, nous considérons la suite finie $T(n)$ de longueur $2^n - 1$ définie par

$$T(0) = \langle\rangle$$
$$T(n+1) = T(n) \odot \langle n \rangle \odot T(n)$$

Pour tout entier i positif ou nul, nous notons t_i le $(i+1)$-ème élément de $T(n)$ s'il existe. Puisque $T(n)$ est préfixe de $T(n+1)$, la suite $(t_i)_{i \geq 0}$ est définie sans ambiguïté. Enfin, nous posons $S_0 = \emptyset$ et pour tout entier i positif ou nul nous définissons l'ensemble $S_{i+1} = S_i \vartriangle \{t_i\}$.

▶ **Question 7** *Donnez les valeurs de $T(1)$, $T(2)$, $T(3)$ et $T(4)$. Donnez la valeur de S_i pour tout i inférieur ou égal à 15.*

▶ **Question 8** *Nous voulons montrer que les S_i peuvent être utilisés pour énumérer les parties de $\mathcal{I}_n$, et ainsi résoudre notre problème d'interrupteurs.*

1. *Donnez la valeur de S_{2^n-1} pour tout $n \geq 0$.*
2. *Montrez que pour tout $n > 0$ et tout $i < 2^n$, on a $S_{2^n+i} = S_i \vartriangle \{n-1, n\}$.*
3. *En déduire que pour tout $n \geq 0$ l'ensemble $\mathcal{P}_n = \{S_0, S_1, \ldots, S_{2^n-1}\}$.*

Nous allons utiliser ce résultat pour résoudre le problème des interrupteurs. Comme dans la deuxième partie, nous imposons que les interrupteurs soient levés au début et à la fin du test.

▶ **Question 9** *Écrivez une fonction* test_gray *s'inspirant des résultats de cette partie, qui imprime une liste d'indices d'interrupteurs à commuter pour réaliser la totalité du test. L'argument de cette fonction sera le nombre d'interrupteurs N.*

test_gray: int −> unit

▶ **Question 10** *Quel est le coût du test avec cette méthode (c'est-à-dire le nombre total d'interrupteurs à commuter) ? Peut-on réaliser le test à un coût moindre ?*

On s'intéresse enfin à la notion de *successeur de Gray* d'une partie.

▶ **Question 11** *Donnez une expression de t_i en fonction de S_i pour i impair. Déduisez-en la fonction* gray *qui prend en argument une partie et retourne celle qui la suit immédiatement dans l'ordre défini par la suite $(S_i)_{i \geq 0}$.*

gray: partie −> partie

Commutation d'interrupteurs

Un corrigé

▶ Question 1

```
let rec card = function
    [] -> 0
  | _ :: q -> 1 + card q
;;
```

▶ Question 2

```
let rec delta p1 p2 =
  match p1, p2 with
    [], _ -> p2
  | _, [] -> p1
  | t1 :: q1, t2 :: _ when t1 < t2 ->
      t1 :: (delta q1 p2)
  | t1 :: _, t2 :: q2 when t1 > t2 ->
      t2 :: (delta p1 q2)
  | _ :: q1, _ :: q2 ->
      delta q1 q2
;;
```

▶ Question 3

```
let test p1 p2 =
let rec print = function
    [] -> ()
  | t :: q ->
      print_int t;
      print_string " ";
      print q
  in
  print (delta p1 p2);
  print_newline ()
;;
```

▶ Question 4

```
let succ p =
    let rec aux i = function
        [] -> [i]
      | j :: q when i = j ->
          aux (i + 1) q
      | j :: q -> i :: j :: q
  in
  aux 0 p
  ;;
```

▶ Question 5

```
let test_incr n =
  let rec aux liste1 liste2 =
    if liste2 = [n] then
      test liste1 []
    else begin
      test liste1 liste2;
      aux liste2 (succ liste2)
    end
  in
  aux [] [0]
;;
```

▶ Question 6 Dans les situations initiale et finale, les interrupteurs sont tous levés. On en déduit donc que, lors d'une exploration complète, on effectue autant de commutations dans les deux sens. Or, chaque fois que l'on considère une nouvelle partie, on effectue une commutation vers le bas. On en déduit que l'on fait en tout $2^n - 1$ commutations vers le bas et donc $2^{n+1} - 2$ commutations au total.

▶ Question 7 On calcule successivement les différents $T(i)$ en utilisant directement la définition récursive de la suite :

$$\begin{aligned}
T(1) &= \langle 0 \rangle \\
T(2) &= \langle 0, 1, 0 \rangle \\
T(3) &= \langle 0, 1, 0, 2, 0, 1, 0 \rangle \\
T(4) &= \langle 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0 \rangle
\end{aligned}$$

On déduit de l'expression de $T(4)$ les valeurs des S_i :

$$\begin{array}{lll}
S_1 = \{0\} & S_6 = \{0, 2\} & S_{11} = \{1, 2, 3\} \\
S_2 = \{0, 1\} & S_7 = \{2\} & S_{12} = \{1, 3\} \\
S_3 = \{1\} & S_8 = \{2, 3\} & S_{13} = \{0, 1, 3\} \\
S_4 = \{1, 2\} & S_9 = \{0, 2, 3\} & S_{14} = \{0, 3\} \\
S_5 = \{0, 1, 2\} & S_{10} = \{0, 1, 2, 3\} & S_{15} = \{3\}
\end{array}$$

▶ Question 8

1. On a $S_{2^n-1} = \{n-1\}$. La différence symétrique étant associative, on peut écrire

$$\begin{aligned} S_{2^n-1} = \{t_0\} \vartriangle \ldots \vartriangle \{t_{2^{n-1}-2}\} \vartriangle \{t_{2^{n-1}-1}\} \\ \vartriangle \{t_{2^{n-1}}\} \vartriangle \ldots \vartriangle \{t_{2^n-2}\} \end{aligned}$$

En utilisant les deux relations conséquences de la définition de la suite T

$$\begin{array}{ll} t_{2^n-1} = n & \text{pour } n > 0 \\ t_{2^n+k} = t_k & \text{pour } 0 \leq k < 2^n - 1 \end{array}$$

on en déduit

$$\begin{aligned} S_{2^n-1} = \{t_0\} \vartriangle \ldots \vartriangle \{t_{2^{n-1}-2}\} \vartriangle \{n-1\} \\ \vartriangle \{t_0\} \vartriangle \ldots \vartriangle \{t_{2^{n-1}-2}\} \end{aligned}$$

En utilisant la commutativité de la différence symétrique et le fait que $X \vartriangle X = \emptyset$ pour toute partie X, on en déduit que $S_{2^n-1} = \{n-1\}$.

2. On utilise simplement le résultat précédent.

$$\begin{aligned} S_{2^n+i} &= S_{2^n-1} \vartriangle \{t_{2^n-1}\} \vartriangle \{t_{2^n}\} \ldots \vartriangle \{t_{2^n+i-1}\} \\ &= \{n-1\} \vartriangle \{t_{2^n-1}\} \vartriangle \{t_{2^n}\} \ldots \vartriangle \{t_{2^n+i-1}\} \\ &= \{n-1\} \vartriangle \{n\} \vartriangle \{t_0\} \ldots \vartriangle \{t_{i-1}\} \\ &= \{n-1, n\} \vartriangle S_i \\ &= S_i \vartriangle \{n-1, n\} \end{aligned}$$

3. Notons Π_n l'ensemble des parties de $\mathcal{I}_n$. On cherche à montrer $\Pi_n = \mathcal{P}_n$, par récurrence sur n. Pour $n = 0$, on a $\mathcal{P}_0 = \{S_0\} = \{\emptyset\} = \Pi_0$. Supposons désormais que $\Pi_n = \mathcal{P}_n$. On a $\Pi_{n+1} = \Pi_n \cup \{X \cup \{n\} \mid X \in \Pi_n\}$. Or

$$\begin{aligned} \mathcal{P}_{n+1} &= \{S_0, \ldots, S_{2^n-1}, S_{2^n}, \ldots S_{2^{n+2}-1}\} \\ &= \mathcal{P}_n \cup \{S \vartriangle \{n-1, n\} \mid S \in \mathcal{P}_n\} \\ &= \mathcal{P}_n \cup \{(S \vartriangle \{n-1\}) \vartriangle \{n\} \mid S \in \mathcal{P}_n\} \end{aligned}$$

or $S \mapsto S \vartriangle \{n-1\}$ est une bijection de $\mathcal{P}_n$ dans lui-même donc

$$\begin{aligned} \mathcal{P}_{n+1} &= \mathcal{P}_n \cup \{S \vartriangle \{n\} \mid S \in \mathcal{P}_n\} \\ &= \mathcal{P}_n \cup \{S \cup \{n\} \mid S \in \mathcal{P}_n\} \end{aligned}$$

puisque pour $S \in \mathcal{P}_n$, $n \notin S$. Par hypothèse de récurrence, il vient

$$\begin{aligned} \mathcal{P}_{n+1} &= \Pi_n \cup \{S \cup \{n\} \mid S \in \Pi_n\} \\ &= \Pi_{n+1} \end{aligned}$$

▶ Question 9

```
let test_gray n =
  let rec t = function
      0 -> ()
    | n ->
        t (n - 1);
        print_int (n - 1);
        print_newline ();
        t (n - 1)
  in
  t n;
  print_int (n - 1);
  print_newline ()
;;
```

▶ Question 11

On a $t_i = \min(S_i) + 1$ pour i impair et $t_i = 0$ pour i pair. En remarquant que la parité de i et du cardinal de S_i sont identiques, on en déduit une implantation de la fonction gray.

▶ Question 11

```
let rec min_list = function
    [] -> raise (Invalid_argument "min_list")
  | x :: [] -> x
  | x :: y :: q -> min_list ((min x y) :: q)
;;

let gray partie =
  if (card partie) mod 2 = 0 then
    delta partie [0]
  else
    delta partie [min_list partie + 1]
;;
```