ÉCOLE POLYTECHNIQUE — FILIÈRE MP

CONCOURS D'ADMISSION 1999

COMPOSITION D'INFORMATIQUE

(Durée : 3 heures)

L'utilisation des calculatrices n'est pas autorisée pour cette épreuve.

★★★

AVERTISSEMENT. On attachera une grande importance à la clarté, à la précision et à la concision de la rédaction. En particulier, le candidat doit systématiquement justifier ses résultats.

Partie I. Fonctions booléennes

Soit $\mathcal{B} = \{0, 1\}$ l'ensemble des booléens. Étant donné un entier n supérieur ou égal à 1, on note $\mathcal{B}^n \to \mathcal{B}$ l'ensemble des fonctions booléennes d'ordre n, c'est à dire l'ensemble des applications de $\mathcal{B}^n$ dans $\mathcal{B}$.

Dans le problème, on adopte généralement la convention qu'une application g de $\mathcal{B}^n \to \mathcal{B}$ est une application des n variables $x_0, x_1, \ldots, x_{n-1}$. Selon cette convention, on notera donc x_i la projection qui à tout n-uplet de booléens $(x_0, x_1, \ldots, x_{n-1})$ associe le booléen x_i. On notera également 0 et 1 les applications constantes qui à tout n-uplet de booléens associent respectivement les booléens 0 et 1.

L'ensemble $\mathcal{B} \to \mathcal{B}$ contient les deux applications constantes qui à tout booléen x_0 associent respectivement 0 et 1, la fonction identité et la fonction « négation » notée $\neg$ et définie ainsi :

x_0	$(\neg\, x_0)$
0	1
1	0

L'ensemble $\mathcal{B}^2 \to \mathcal{B}$ contient en particulier les fonctions « ou », « et », et « implique », notées respectivement par les opérateurs infixes $\vee$, $\wedge$ et $\Rightarrow$, et définies sur les couples de booléens (x_0, x_1) par les tables suivantes :

x_0	x_1	$(x_0 \vee x_1)$	$(x_0 \wedge x_1)$	$(x_0 \Rightarrow x_1)$
0	0	0	0	1
0	1	1	0	1
1	0	1	0	0
1	1	1	1	1

Les symboles $\vee$, $\wedge$ et $\Rightarrow$ sont appelés *connecteurs binaires*. Les parenthèses introduites ici permettent d'éviter les ambiguïtés dans les expressions complexes, les candidats doivent les utiliser.

Question 1. En machine, l'ensemble $\mathcal{B}$ est codé par le type des booléens (`bool` en Caml, `boolean` en Pascal), le booléen machine `false` étant 0 et le booléen machine `true` étant 1. Écrire les trois fonctions `ou`, `et` et `implique`, qui réalisent respectivement « et », « ou » et « implique ». On utilisera uniquement les constantes `false` et `true`, les variables et l'instruction conditionnelle `if... then... else...` On se conformera aux déclarations suivantes :

(* *Caml* *)

```
value et : bool -> bool -> bool
and ou : bool -> bool -> bool
and implique : bool -> bool -> bool
```

{ *Pascal* }

```
function et (x0, x1 : boolean) : boolean ;
function ou (x0, x1 : boolean) : boolean ;
function implique (x0, x1 : boolean) : boolean ;
```

Étant donnée une application g de $\mathcal{B}^n \to \mathcal{B}$ on note $(\neg g)$, l'application qui à tout n-uplet de booléens $(x_0, x_1, \ldots, x_{n-1})$ associe le booléen $(\neg\, g(x_0, x_1, \ldots, x_{n-1}))$.

De même, étant donné un connecteur binaire « op », ainsi que deux applications g et g' de $\mathcal{B}^n \to \mathcal{B}$, on note $(g \text{ op } g')$ l'application de $\mathcal{B}^n \to \mathcal{B}$ définie par :

$$(g \text{ op } g')(x_0, x_1, \ldots, x_{n-1}) = (g(x_0, x_1, \ldots, x_{n-1}) \text{ op } g'(x_0, x_1, \ldots, x_{n-1}))$$

On peut donc, n étant fixé, noter une application de $\mathcal{B}^n \to \mathcal{B}$ à l'aide de 0, de 1, de x_i, du symbole $\neg$ et des connecteurs binaires. Ainsi, $(((x_1 \Rightarrow 0) \vee (\neg\, x_2)) \wedge x_0)$, représente une application de $\mathcal{B}^n \to \mathcal{B}$ pour tout n supérieur ou égal à 3.

Les fonctions « et » et « ou » étant associatives, on pourra omettre certaines parenthèses. Par exemple, $((f \wedge g) \wedge h)$ et $(f \wedge (g \wedge h))$ dénotent la même fonction qui sera simplement notée $(f \wedge g \wedge h)$.

Une *tautologie* est une fonction booléenne égale à l'application constante 1. Une *contradiction* est une fonction booléenne égale à 0. Une fonction booléenne qui n'est pas une contradiction est dite *satisfiable*. Étant donnée g, fonction satisfiable de $\mathcal{B}^n \to \mathcal{B}$, une *solution* de g est un n-uplet de booléens $(b_0, b_1, \ldots, b_{n-1})$, tel que $g(b_0, b_1, \ldots, b_{n-1})$ vaut 1.

Question 2.

a) Voici un certain nombre d'affirmations :

1. $(x_0 \Rightarrow x_0)$ est une tautologie.
2. $(\neg(x_0 \Rightarrow x_1))$ et $(x_0 \wedge (\neg\, x_1))$ sont égales.
3. Il y a des tautologies qui ne sont pas satisfiables.
4. Si $(\neg\, g)$ est une tautologie, alors g est satisfiable.

Déterminer les affirmations exactes et les affirmations inexactes, en vous justifiant brièvement.

b) Donner une solution de $(x_0 \Rightarrow (x_1 \Rightarrow x_2))$.

Question 3. La direction d'une crèche décide de contrôler les jouets qu'apportent les enfants. Elle définit donc une politique d'admission des jouets, qui doivent se conformer à l'ensemble des cinq règles suivantes :

1. Les jouets doivent être de petite taille, sauf les peluches.
2. Un jouet est soit vert, soit grand, soit grand et vert.
3. Les jouets électriques sont obligatoirement accompagnés de leurs piles.
4. Un enfant ne peut à la fois apporter des piles et une peluche.
5. Tous les jouets verts sont des peluches et sont électriques.

Un jouet est caractérisé par cinq variables booléennes, à savoir `peluche`, `petit`, `vert`, `electrique`, et `piles`. Ainsi, si Oscar se présente avec un train électrique rouge, muni de piles et que ce jouet est jugé grand, `peluche` vaut 0, `petit` vaut 0, `vert` vaut 0, `electrique` vaut 1 et `piles` vaut 1.

a) Oscar peut-il rentrer dans la crèche avec son train ?

b) Expliciter, a l'aide de $\neg$ et des connecteurs binaires, une fonction booléenne p de $\mathcal{B}^5 \to \mathcal{B}$ qui vaut 1 si et seulement si le jouet auquel elle s'applique est admis. On pourra écrire p comme une fonction des variables `peluche`, `petit`, `vert`, `electrique`, et `piles`.

Partie II. Arbres de décision

Étant donné un ensemble ordonné de n variables notées $x_0, x_1, \ldots, x_{n-1}$, on définit les *arbres de décision* sur ces variables de la façon suivante :

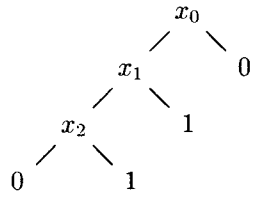
1. Les symboles 0 et 1 sont des arbres de décision.
2. $\text{test}_{x_i}(v, f)$ est un arbre de décision, si v et f sont des arbres de décision tels que :

(a) les arbres v et f sont différents,

(b) toutes les variables de v et f sont des x_j avec $j > i$.

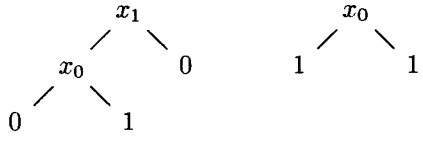
Dans un nœud interne $\text{test}_{x_i}(v, f)$, x_i est la *variable testée*, v est le *fils positif* et f le *fils négatif*.

Un arbre de décision est donc d'abord un arbre dont les feuilles sont 0 ou 1 et dont les nœuds internes sont binaires et étiquetés par une variable. En outre, les deux fils d'un nœud interne sont distincts et tous les chemins de la racine aux feuilles passent par des nœuds d'étiquettes strictement croissantes. Voici par exemple, un arbre de décision :



qui peut également s'écrire « $\text{test}_{x_0}(\text{test}_{x_1}(\text{test}_{x_2}(0, 1), 1), 0)$ ». On remarquera que dans un nœud $\text{test}_{x_i}(v, f)$, le fils positif v est systématiquement le fils gauche, et le fils négatif f le fils droit.

En revanche, les deux arbres ci-dessous ne sont pas des arbres de décision. Le premier arbre viole la contrainte 2.(b) sur l'ordre des variables, tandis que le second viole la contrainte 2.(a) sur les fils distincts.



Étant données trois applications g, g' et g'' de $\mathcal{B}^n \to \mathcal{B}$, $\text{test}(g, g', g'')$ est une application de $\mathcal{B}^n \to \mathcal{B}$ définie de la manière suivante :

1. Si $g(x_0, x_1, \ldots, x_{n-1})$ vaut 1, alors $\text{test}(g, g', g'')(x_0, x_1, \ldots, x_{n-1})$ vaut $g'(x_0, x_1, \ldots, x_{n-1})$.
2. Sinon, $g(x_0, x_1, \ldots, x_{n-1})$ vaut 0 et $\text{test}(g, g', g'')(x_0, x_1, \ldots, x_{n-1})$ vaut $g''(x_0, x_1, \ldots, x_{n-1})$.

Un arbre de décision représente une application de $\mathcal{B}^n \to \mathcal{B}$ de la manière suivante :

1. L'arbre 0 représente l'application constante 0, l'arbre 1 représente l'application constante 1.
2. L'arbre $\text{test}_{x_i}(v, f)$ représente l'application $\text{test}(x_i, g', g'')$, où les arbres v et f représentent respectivement les applications g' et g''.

Question 4.

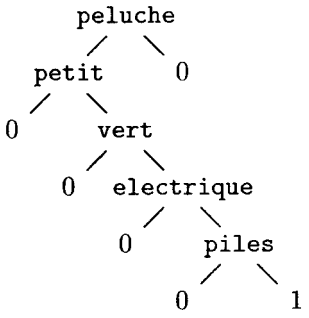
a) Donner des arbres de décision qui représentent les cinq fonctions x_0, $(\neg x_1)$, $(x_0 \wedge (\neg x_1))$, $((\neg x_1) \wedge x_0)$ et $(\neg(x_0 \Rightarrow x_1))$.

b) Prouver l'affirmation : « Si l'arbre a représente une tautologie, alors a est réduit à la feuille 1 ».

c) Montrer que toute fonction booléenne est représentable par un arbre de décision. Dans leur démonstration les candidats pourront utiliser les *applications partielles* $g_{i\leftarrow 0}$ et $g_{i\leftarrow 1}$ de $\mathcal{B}^n \to \mathcal{B}$ définies pour toute application g de $\mathcal{B}^n \to \mathcal{B}$, comme suit :

$$\begin{aligned} g_{i\leftarrow 0}(x_0, \ldots, x_i, \ldots, x_{n-1}) &= g(x_0, \ldots, 0, \ldots, x_{n-1}) \\ g_{i\leftarrow 1}(x_0, \ldots, x_i, \ldots, x_{n-1}) &= g(x_0, \ldots, 1, \ldots, x_{n-1}) \end{aligned}$$

Question 5. La direction de la crèche décide de représenter la fonction d'admission des jouets de la question 3 par un arbre de décision. Après avoir choisi d'ordonner les variables dans l'ordre `peluche`, `petit`, `vert`, `electrique` et `piles`, elle obtient l'arbre suivant :



a) Reformuler en français la politique p d'une façon plus simple que les cinq règles de la question 3.

b) La direction n'est finalement pas satisfaite de sa politique. Elle décide d'étudier deux autres politiques. Une nouvelle politique comprend les cinq règles de la politique p de la question 3, plus une au choix des règles suivantes :

6-a. Toutes les peluches sont vertes.

6-b. Indépendamment de toutes les autres règles, les jouets verts sont autorisés.

Donner les arbres des nouvelles politiques p_a et p_b.

c) Devant les protestations des parents, la direction décide que tout jouet refusé une fois est autorisé les jours suivants. Elle se donne donc une nouvelle variable booléenne `deja_vu`, ainsi que la nouvelle sixième règle :

6-c. Indépendamment des autres règles, tous les jouets déjà vus sont admis.

Préciser un nouvel ordre sur les caractéristiques des jouets et l'arbre de décision de la nouvelle politique p_c selon cet ordre.

Question 6. Soit g une application de $\mathcal{B}^n \to \mathcal{B}$.

a) Exprimer g et $(\neg g)$ à l'aide de la projection x_i, et des applications partielles $g_{i\leftarrow 0}$ et $g_{i\leftarrow 1}$.

b) Soient en outre g' et g'' dans $B^n \to \mathcal{B}$, exprimer $\text{test}(g, g', g'')$ à l'aide de la projection x_i, et des applications partielles $g_{i\leftarrow 0}$, $g_{i\leftarrow 1}$, $g'_{i\leftarrow 0}$, $g'_{i\leftarrow 1}$, $g''_{i\leftarrow 0}$ et $g''_{i\leftarrow 1}$.

Partie III. Programmation des arbres de décision

Les arbres de décision seront représentés en machine par des objets de type `abd` (pour *arbre binaire de décision*), dont voici les définitions :

(* *Caml* *)

```
type variable == int
;;

type abd =
  Bool of bool
| Test of variable * abd * abd
;;
```

{ *Pascal* }

```
type
 variable = integer ;
 nature = (Test, Bool) ;
 abd = ^cellule ;
 cellule = record
   case nature : nature of
     Bool : (val : boolean);
     Test:
      (testee : variable ;
       pos, neg : abd)
   end ;
```

NOTE. Les candidats qui composent en Pascal n'utiliseront pas la primitive `new` mais les fonctions de construction `fait_bool` et `fait_test` :

```
function fait_bool(val : boolean) : abd ;
function fait_test(testee : variable ; pos, neg : abd) : abd ;
```

Ces deux fonctions renvoient un pointeur vers respectivement une nouvelle feuille et un nouveau nœud interne dont les champs sont initialisés aux valeurs des variables homonymes passées en arguments. À titre d'information, voici le source de la fonction `fait_test` :

```
function fait_test (testee : variable ; pos, neg : abd) : abd ;
var
   r : abd;
begin
   new(r);
   r^.nature := Test ;
   r^.testee := testee ;
   r^.pos := pos ;
   r^.neg := neg ;
   fait_test := r
end;
```

Question 7.

a) Programmer la fonction `abd_proj` qui prend en argument un indice de variable et renvoie un arbre de décision représentant la projection selon cet indice. On se conformera à :

(* *Caml* *)	{ *Pascal* }
`value abd_proj : variable -> abd`	`function abd_proj (i : variable) : abd ;`

b) Programmer la fonction `abd_neg` qui prend en argument un arbre de décision représentant une fonction booléenne g et renvoie un arbre de décision représentant la fonction $(\neg g)$. On se conformera aux déclarations suivantes :

(* *Caml* *)	{ *Pascal* }
`value abd_neg : abd -> abd`	`function abd_neg (g : abd) : abd ;`

Justifier que votre fonction `abd_neg` rend un arbre de décision bien formé si son argument est un arbre de décision bien formé.

Question 8.

a) Programmer la fonction `abd_egal` qui prend en argument deux arbres de décision et renvoie `true` si ces arbres sont structurellement égaux, et `false` autrement. On se conformera à :

(* *Caml* *)	{ *Pascal* }
`value abd_egal : abd -> abd -> bool`	`function abd_egal (a0, a1 : abd) : boolean ;`

NOTE. Les candidats qui composent en Caml ne peuvent pas définir `abd_egal` comme suit :

```
let abd_egal a0 a1 = a0 = a1
;;
```

Toutes les définitions d'esprit similaire sont bien évidemment également interdites.

b) Programmer la fonction `abd_partiel` qui prend en arguments, un indice de variable i, un booléen b, ainsi qu'une fonction booléenne g représentée par un arbre de décision ; et renvoie un arbre de décision qui représente $g_{i \leftarrow b}$. On se conformera à :

(* *Caml* *)	{ *Pascal* }
`value abd_partiel :` `  variable -> bool -> abd -> abd`	`function abd_partiel` `  (i : variable ; b : boolean ; g : abd) : abd ;`

c) Programmer la fonction `abd_test` qui prend en arguments trois fonctions booléennes c, v et f représentées par des arbres de décision, et qui renvoie l'arbre de décision qui représente $\text{test}(c, v, f)$. On se conformera aux déclarations :

(* *Caml* *)	{ *Pascal* }
`value abd_test : abd -> abd -> abd -> abd`	`function abd_test (c, v, f : abd) : abd ;`

d) Donner les fonctions `abd_et`, `abd_ou`, et `abd_implique`, qui réalisent les opérations $(f \wedge g)$, $(f \vee g)$ et $(f \Rightarrow g)$, où f et g sont des fonctions booléennes représentées en machine par des arbres de décision. On se conformera à :

(* *Caml* *)

```
value abd_et : abd -> abd -> abd
and abd_ou : abd -> abd -> abd
and abd_implique : abd -> abd -> abd
```

{ *Pascal* }

```
function abd_et (f, g : abd) : abd ;
function abd_ou (f, g : abd) : abd ;
function abd_implique (f, g : abd) : abd ;
```

Question 9. Les trois politiques p, p_a et p_b sont représentées en machine par les variables p, pa et pb, de type abd. Donner les instructions qui calculent ces arbres de décision à partir des définition des politiques de l'énoncé, c'est à dire des règles de la question 3. pour p, assorties des règles supplémentaires de la question 5.b) pour p_a et p_b. Les candidats peuvent définir et utiliser les variables intermédiaires de leur choix.

Question 10. Dans cette question on se limite à des jouets caractérisés par les cinq variables `peluche`, `petit`, `vert`, `electrique` et `piles`, prises dans cet ordre. En outre on dispose de `nom`, un tableau de cinq chaînes qui contient les noms de ces cinq variables, indicés par 0, 1, 2, 3 et 4. De sorte que, par exemple, `nom.(2)` (`nom[2]` en Pascal) est la chaîne `"vert"` (`'vert'` en Pascal).

La direction de la crèche décide d'afficher ses politiques sur la porte de la crèche. Les affiches sont composées de lignes, une caractéristique donnée pouvant apparaître au plus une fois par ligne, soit pour être acceptée, soit pour être rejetée.

a) Écrire la fonction (procédure en Pascal) `abd_solutions` qui prend une politique p représentée par son arbre de décision en argument et compose l'affiche des jouets autorisés.

(* *Caml* *)

```
value abd_solutions : abd -> unit
```

{ *Pascal* }

```
procedure abd_solutions (p : abd);
```

RAPPEL. On affiche une chaîne à l'écran par `print_string` en Caml et `write` en Pascal. Un saut à la ligne est ajouté à l'affichage si on utilise `print_endline` en Caml et `writeln` en Pascal.

b) Quel est le nombre de lignes des affiches produites par votre fonction pour les politiques p, p_a et p_b, ainsi que pour la politique laxiste qui autorise tous les jouets ?