Calculabilité

TP2

Y. Deville
C-H. Bertrand Van Ouytsel - V. Coppé - A. Gerniers
N. Golenvaux - M. Parmentier

Février 2021

Questions du test

1. La fonction prenant en arguments 6 nombres entiers compris entre 1 et 45 et produisant 1 s'ils correspondent à la combinaison gagnante pour le prochain Lotto belge et 0 sinon est-elle calculable ?

   Réponse : Oui
   Pour le justifier, on peut générer autant de programmes que de combinaisons possibles pour le Lotto. Pour chaque combinaison possible C, le programme sera :

   if $\underline{input == C}$ **then**
   ⌊ return(1)

   else
   ⌊ return(0)

   Au moins un de ces programmes calcule la fonction décrite.

Questions du test

2. L'ensemble des nombres premiers est-il récursif ?

   Réponse : Oui
   Voici un programme qui permet de le justifier. Celui-ci prend en entrée un nombre naturel n et renvoie 1 si n est premier, 0 sinon.

```
if n == 0  or  n == 1 then
    return(0)
else
    for i from 2 to n − 1 do
        if n%i == 0 then
            return(0)
return(1)
```

Questions du test

3. Est-il vrai qu'un ensemble X est récursif si et seulement si son complémentaire (dans $\mathbb{N}$), $\overline{X}$, est lui-même récursif ?

   Réponse : Oui
   Soit $P_X(n)$ le programme qui permet de justifier que X est récursif. Alors le programme $P_{\overline{X}}(n)$ ci-dessous permet de justifier que $\overline{X}$ est récursif.

$$P_{\overline{X}}(n) \equiv [\texttt{return } 1 - P_X(n)$$

   Puisque $\overline{\overline{X}} = X$, l'implication dans l'autre sens découle de celle-ci.

Questions du test

4. Est-il vrai qu'un ensemble X est récursif si et seulement si X est récursivement énumérable et $\overline{X}$ est récursivement énumérable ?

   Réponse : Oui
   Si X est récursif, alors il est certainement récusivement énumérable (le programme qui permet de justifier que X est récursif permet également de justifier que X est récursivement énumérable). Par la question précédente, on sait alors que le complémentaire de X est alors lui aussi récursif et donc récursivement énumérable.

   Pour ce qui est de l'implication dans l'autre sens : supposons que X et $\overline{X}$ sont récursivement énumérables. Soient les deux programmes $\mathbb{P}_X(n)$ et $\mathbb{P}_{\overline{X}}(n)$ qui correspondent.

Questions du test

Deux réponses possibles : la réponse « moderne » et la réponse « classique ». La réponse « moderne » consiste à dire qu'il suffit de créer un programme P_X qui exécute en parallèle $\mathbb{P}_X(n)$ et $\mathbb{P}_{\overline{X}}(n)$ (multithreading). Aussi tôt que $\mathbb{P}_X(n)$ ou $\mathbb{P}_{\overline{X}}(n)$ se termine pour un certain $n \in \mathbb{N}$ (et nécessairement au moins un des deux doit se terminer), on peut décider si n appartient ou non à X.

Si cette réponse moderne vous convient, fantastique (elle sera acceptée à l'examen) ! Mais dans le cas contraire, il vous faut la réponse « classique », qui explicite ce qu'on cache dérrière les mots tels que « multithreading » ou « exécuter en parallèle ».

Questions du test

Pour la réponse « classique », introduisons la notion d'opérations élémentaires d'un programme. Par opérations élémentaires, on entend ici les étapes individuelles des instructions des programmes [1]. On construit alors le programme suivant (voir slide suivante). Pout tout $n \in \mathbb{N}$, ce programme produit 1 si $n \in X$ et 0 si $n \notin X$. X est donc bien récursif.

1. Ces opérations élémentaires dépendent du formalisme que vous utilisez. Dans le cas du formalisme des machines de Turing, le modèle traditionnel de la calculabilité (dont nous parlerons en détail plus tard dans le cours), ces opérations élémentaires sont explicites. En attendant, pour nourrir votre intuition, dites-vous par exemple qu'une instruction telle que « for i from 1 to 10 : k=k+1 » correspond à 10 opérations élémentaires.

Questions du test

$P_X(n) \equiv$

```
k = 0
while TRUE :
  do the k first elementary operations of P_X(n)
  if P_X(n) terminates :
    if P_X(n) == 1 :
      return 1
    else :
      return 0
  do the k first elementary operations of P_X̄(n)
  if P_X̄(n) terminates :
    if P_X(n) == 1 :
      return 0
    else :
      return 1
  k = k + 1
```

Questions du test

Remarque : il est nécessaire de parler des opérations élémentaires des deux programmes $\mathbb{P}_X$ et $\mathbb{P}_{\overline{X}}$ et non des instructions de ces deux programmes parce qu'une unique instruction peut mener un programme à boucler indéfiniment. Alterner les instructions (et non les opérations élémentaires) de ces deux programmes pourrait donc par exemple mener à une situation où le programme construit entrerait dans une boucle infinie du programme $\mathbb{P}_{\overline{X}}$ avant d'avoir atteint la dernière instruction du programme $\mathbb{P}_X$.

Questions du test

5. Si deux ensembles X et Y sont récursifs, alors il est toujours vrai que $X \cup Y$, $X \cap Y$ et $\overline{X}$ sont aussi des ensembles récursifs.

   Réponse : Oui
   Si X et Y sont récursifs, on a deux programmes $P_X(n)$ et $P_Y(n)$ qui permettent repectivement de décider si un élément $n \in \mathbb{N}$ est dans X ou Y. Pour justifier le fait que $X \cup Y$, $X \cap Y$ et $\overline{X}$, il suffit de considérer les programmes suivants.

$$P_{X\cup Y}(n) \equiv \left[\begin{array}{l} \texttt{if } P_X(n) == 1 \texttt{ or } P_Y(n) == 1 : \\ \quad \texttt{return } 1 \\ \texttt{else:} \\ \quad \texttt{return } 0 \end{array}\right.$$

Questions du test

$$P_{X\cap Y}(n) \equiv \left[\begin{array}{l} \texttt{if } P_X(n) == 1 \texttt{ and } P_Y(n) == 1: \\ \quad \texttt{return } 1 \\ \texttt{else:} \\ \quad \texttt{return } 0 \end{array}\right.$$

$$P_{\overline{X}}(n) \equiv [\texttt{return } 1 - P_X(n)$$

Questions du test

6. Si deux ensembles X et Y sont récursivement énumérables, alors il est toujours vrai que $X \cup Y$, $X \cap Y$ et $\overline{X}$ sont aussi des ensembles récursivement énumérables.

   Réponse : Non
   $X \cup Y$ et $X \cap Y$ seront bien toujours également récursivement énumérables, mais $\overline{X}$ pas nécessairement.
   Contre-exemple : $X = K = \{n \in \mathbb{N} \mid (n, n) \in \text{HALT}\}$.
   Comme K est récursivement énumérable mais n'est pas récursif, $\overline{K}$ ne peut pas être récursivement énumérable.

Questions du test

7. Est-il vrai que la somme de deux fonctions calculables à une variable est toujours une fonction calculable ?
   Remarque : la somme de deux fonctions partielles n'est définie qu'aux points pour lesquels les deux fonctions de départ sont bien définies.

   Réponse : Oui
   Soient deux fonctions calculables à une variable f_1, f_2 et soient P_{f_1}, P_{f_2} deux programmes qui calculent ces fonctions. Alors la fonction $f = f_1 + f_2$ définie sur l'intersection des domaines de f_1 et f_2 est calculée par le programme suivant.

$$P_f(n) \equiv [\texttt{return}\ \ P_{f_1}(n) + P_{f_2}(n)$$

Questions du test

8. Est-il vrai qu'un ensemble X est récursivement énumérable si et seulement si il existe une fonction calculable partielle f telle que $X = \mathrm{dom}(f)$?

   Réponse : Oui
   Démontrons d'abord que l'existence d'une fonction calculable partielle f telle que $X = \mathrm{dom}(f)$ est une condition suffisante.
   Soit P_f un programme qui permet de calculer la fonction f.
   Alors le programme suivant permet de justifier que X est récursivement énumérable.

$$P_X(n) \equiv \left[\begin{array}{l} P_f(n) \\ \texttt{return } 1 \end{array} \right.$$

Questions du test

À présent, démontrons que l'existence d'une fonction calculable partielle f telle que $X = \mathrm{dom}(f)$ est une condition nécessaire. Soit P_X un programme qui permet de justifier que X est récursivement énumérable. Alors le programme suivant calcule une fonction f telle que $X = \mathrm{dom}(f)$.

$$P_f(n) \equiv \left[\begin{array}{l} \texttt{if } P_X(n) == 1 : \\ \quad \texttt{return } n \\ \texttt{else:} \\ \quad \texttt{while TRUE :} \\ \quad\quad \texttt{pass} \end{array}\right.$$

Questions du test

9. Est-il vrai qu'un ensemble X est récursivement énumérable si et seulement si il existe un programme Python qui énumère les éléments de X ?

   Réponse : Oui
   Démontrons d'abord que l'existence d'un tel programme est une condition suffisante. Soit $P_{\text{énum}X}$ un programme qui énumère les éléments de X. Alors le programme suivant permet de justifier que X est récursivement énumérable.

$$P_X(n) \equiv \left[\begin{array}{l} k = 0 \\ \texttt{while TRUE}: \\ \quad \texttt{if } P_{\text{énum}X}(k) == n: \\ \quad\quad \texttt{return } 1 \\ \quad k = k + 1 \end{array}\right.$$

Questions du test

À présent, démontrons que l'existence d'un tel programme est une condition nécessaire. Soit P_X un programme qui permet de justifier que X est récursivement énumérable. Alors on peut créer un programme $P_{\text{énum}X}(n)$ qui énumère les éléments de X.

On lance en parallèle le programme P_X pour chaque donnée k. $P_{\text{énum}X}(n)$ renvoie ensuite la donnée k du n-ième thread qui termine.

Plus formellement, on peut construire une table avec à la position i, j les j premières opérations élémentaires du programme $P_X(i)$. Cela permet d'exécuter les programmes en « parallèle » en suivant les diagonales descendantes de cette table.

	Opérations élémentaires du programme
$P_X(0)$	• • • • • • ⋯
$P_X(1)$	• • • • • • ⋯
$P_X(2)$	• • • • • • ⋯
⋮	⋮ ⋮ ⋮ ⋮ ⋮ ⋮

Questions du test

$$P_{\text{énum}X}(n) \equiv$$

```
c = 0
i = 0
while TRUE :
  j = 0
  while j < i :
    k = i − j
    do the k first elementary operations of P_X(j)
    if P_X(j) terminates and did not terminate with
    k − 1 elementary operations :
      if P_X(j) == 1 :
        if c == n :
          return j
        c = c + 1
    j = j + 1
  i = i + 1
```

Questions du test

10. Est-il vrai qu'un ensemble X est récursivement énumérable si et seulement si X est l'ensemble vide ou il existe une fonction calculable totale f telle que $X = \text{image}(f)$?

    Réponse : Oui
    Démontrons d'abord qu'il s'agit d'une condition suffisante. Si X est l'ensemble vide, X est évidemment récursivement énumérable. Supposons que X est non vide. Soit une fonction calculable totale f telle que $X = \text{image}(f)$. P_f un programme qui calcule cette fonction. Alors le programme suivant permet de justifier que X est récursivement énumérable.

$$P_X(n) \equiv \left[\begin{array}{l} k = 0 \\ \texttt{while TRUE}: \\ \quad \texttt{if } P_f(k) == n: \\ \quad\quad \texttt{return } 1 \\ \quad k = k + 1 \end{array}\right.$$

Questions du test

À présent, démontrons qu'il s'agit d'une condition nécessaire. Soit P_X un programme qui permet de justifier que X est récursivement énumérable.

- Si X est l'ensemble vide, alors X est l'ensemble vide.
- Si X est non-vide et fini, alors $X = \{x_0, x_1, \ldots, x_{n-1}\}$ et on peut alors facilement trouver une fonction totale calculable telle que $X = \text{image}(f) : f(k) = x_{k\%n}$ ($\%$ est l'opérateur modulo).
- Si X est non-vide et infini, alors le programme $P_{\text{énum}X}$ (voir exercice 9) calcule une fonction totale f telle que $X = \text{image}(f)$.

Question 1 du TP

Let $X \subseteq \mathbb{N}$ and $f(n)$ a total function defined by

$$f(n) = \#\{x \in X | x < n\}$$

where $\#A$ is the cardinal of A. Show that X is recursive iff f is computable.

Question 1 du TP

Réponse : Commençons par supposer que X est récursif. Alors il existe un programme P_X qui décide X. Alors le programme suivant calcule la fonction f :

$$P_f(n) \equiv \left[\begin{array}{l} k = 0 \\ \text{result} = 0 \\ \texttt{while } k < n : \\ \quad \texttt{if } P_X(k) == 1 : \\ \quad\quad \text{result} = \text{result} + 1 \\ \quad k = k + 1 \\ \texttt{return result} \end{array}\right.$$

Supposons maintenant que la fonction f est calculable. Alors il existe un programme P_f qui calcule f. Alors le programme suivant décide l'ensemble X :

$$P_X(n) \equiv [\texttt{return } P_f(n+1) - P_f(n)$$

Question 2 du TP

True or false ?

1. If the domain of a function is finite, then the function is computable.
2. If a function is computable, then its domain is recursive.
3. A function whose table can be defined in a finite way is necessarily computable.

Question 2 du TP

Réponse :

1. Vrai

   Soit f une fonction dont le domaine est fini. Soient n_1, n_2, ..., n_k tous les éléments de son domaine. Soient respectivement m_1, m_2, ..., m_k les images de ces nombres par la fonction f. Alors le programme ci-dessous calcule la fonction f.

$$P_f(n) \equiv \left\lceil \begin{array}{l} \texttt{if } n == n_1 : \\ \quad \texttt{return } m_1 \\ \texttt{else if } n == n_2 : \\ \quad \texttt{return } m_2 \\ \qquad \vdots \\ \texttt{else if } n == n_k : \\ \quad \texttt{return } m_k \\ \texttt{else} : \\ \quad \texttt{while TRUE} : \\ \qquad \texttt{pass} \end{array} \right.$$

Question 2 du TP

Réponse :

2. Faux
   Contre-exemple : le programme ci-dessous calcule une fonction dont le domaine est l'ensemble K (qui n'est pas récursif).

$$P_f(n) \equiv \left[\begin{array}{l} P_n(n) \\ \texttt{return } 1 \end{array} \right.$$

Question 2 du TP

Réponse :

3. Faux
   Contre-exemple : la fonction dont la table est décrite à l'aide de la phrase finie suivante n'est pas calculable.

   « La fonction f définie sur $\mathbb{N}$ telle que pour tout $n \in \mathbb{N}$,
   $f(n) = 1$ si $n \in K$ et 0 si $n \notin K$. »

Question 3 du TP

The Matiyasevich theorem is a famous theorem in number theory which gives an answer to Hilbert's tenth problem. The formulation of the theorem is quite simple :

A subset of $\mathbb{N}$ is diophantine iff it is recursively enumerable.

(Any subset of $\mathbb{N}$ is called diophantine iff it is of the form $E_D = \{a \in \mathbb{N} \mid \exists x_1, ..., x_k \in \mathbb{N} : D(a, x_1, ..., x_k) = 0\}$ where $D(a, x_1, ..., x_k)$ is a fixed polynomial with one parameter a and with k variables $x_1, ..., x_k$.)

Prove half of the theorem : show that if a set X is diophantine, then it is recursively enumerable.
Hint : write a program which proves that the set of perfect squares, which is a diophantine set ($\{a \in \mathbb{N} \mid \exists x \in \mathbb{N} : x^2 - a = 0\}$), is recursively enumerable but does not prove that it is recursive (even though it is). Then do the same for the set of sums of two perfect squares.

Question 3 du TP

Réponse : Soit E_D un ensemble diophantien et soit $D(a, x_1, ..., x_k)$ le polynôme associé. Le programme ci-dessous permet de justifier que E_D est récursivement énumérable.

$P_{E_D}(n) \equiv$

```
i_1 = 0
i_2 = 0
  ⋮
i_k = 0
while TRUE :
  while i_2 < i_1 :
    while i_3 < i_2 :
      ⋱
        while i_k < i_{k-1} :
          if D(a, i_1, ..., i_k) == 0 :
            return 1
          i_k = i_k + 1
        i_k = 0
        i_{k-1} = i_{k-1} + 1
      ⋰
    i_3 = 0
    i_2 = i_2 + 1
  i_2 = 0
  i_1 = i_1 + 1
```

Question 4 du TP

Show that all monotonically decreasing total functions $f : \mathbb{N} \to \mathbb{N}$ are computable.

Challenge : Construct a monotonically increasing total function that is not computable. (Hint : use HALT)

Question 4 du TP

Réponse : Soit une fonction $f : \mathbb{N} \to \mathbb{N}$ qui est décroissante. Puisque que $\mathbb{N}$ est discret et est borné inférieurement par 0, f a nécessairement une imagine finie. Soient m_1, m_2, ..., m_k les éléments de cet image. Puisque que f est décroissante, pour tout $i \in \mathbb{N}$ avec $1 \leq i \leq k-1$, il existe $a_i, b_i \in \mathbb{N}$ avec $a_i \leq b_i$ tels que tout élément de $f^{-1}(m_i)$ est compris entre a_i et b_i, et pour $i = k$ il existe a_k tel que tout élément de $f^{-1}(m_k)$ est plus grand ou égal à a_k. Dès lors, le programme suivant calcule la fonction f.

$$P_f(n) \equiv \left[\begin{array}{l} \texttt{if } a_1 \leq n \leq b_1 : \\ \quad \texttt{return } m_1 \\ \texttt{else if } a_2 \leq n \leq b_2 : \\ \quad \texttt{return } m_2 \\ \qquad \vdots \\ \texttt{else if } a_k \leq n : \\ \quad \texttt{return } m_k \end{array}\right.$$