Chapitre 3. Récursion et Théorème de Rice

- Problème de l’arrêt h(p, x)=1 si p(x) s’arrête et 0 sinon. On a vu une preuve qu’il n’existe pas de procédure hp qui calcule h.
- L’usage de la récursion donne une autre preuve. Supposons que h soit calculable par une procédure hp.

Récursion

- Soit gamma la procédure suivante :
- int gamma (int x)

```
if (hp (gamma, x)) while (1) ;
else return 0;}
```

- Que vaut gamma(x) ?

Récursion

- Soit gamma la procédure suivante :
- int gamma (int x)

  {if (hp (gamma, x)) while (1) ; (else return 0;}
- Que vaut gamma(x) ?
- gamma(x) s'arrête => hp(gamma,x) =1

  => gamma(x) boucle (while(1);)
- gamma(x) boucle => hp(gamma,x)=0

  => gamma(x) donne 0 et est défini

Récursion

- Dans tous les cas on arrive à une contradiction donc hp n’existe pas

Théorème de Rice

- Pourquoi limiter la méthode précédente au problème de l'arrêt ?
- Prenons par exemple le prédicat δ tel que :

  $\delta(p)=1$ si p calcule l'identité et 0 sinon

- p calcule l'identité si $\forall x, p(x)=x$
- Supposons qu'il existe une procédure C delta qui calcule δ

Théorème de Rice

- Soit gamma la procédure suivante :
- int gamma (int x)

```
    if (delta(gamma)) return 0 ;
    else return x;}
```

- Si $\forall x$ gamma(x)= x => delta(gamma)=1

  => $\forall x$ gamma(x)=0

- Si $\exists x$ gamma(x) $\neq$ x => delta(gamma)=0

  => $\forall x$ gamma(x)=x

Théorème de Rice

- On peut donc généraliser.
- Définition : Un prédicat P est une fonction totale dans {0,1}. Un prédicat est trivial s'il vaut toujours la même valeur quelque soit ses paramètres

  (i.e. $\forall x\ P(x)=1$ ou $\forall x\ P(x)=0$)

- On considère dans la suite des prédicats définis sur l'ensemble des procédures

Théorème de Rice

- Théorème de Rice.
- Soit P un prédicat tel que

$$\forall x\ p(x)=q(x) \Rightarrow P(p)=P(q)$$

Alors P est indécidable (il n’y a pas de procédure qui calcule P) si P n’est pas trivial.

Preuve.

- P non trivial donc il existe une procédure p_0 telle que $P(p_0)=0$ et une procédure p_1 telle que $P(p_1)=1$.
- Supposons que la procédure procP calcule P.

Preuve.

- P non trivial donc il existe une procédure p_0 telle que $P(p_0)=0$ et une procédure p_1 telle que $P(p_1)=1$.
- Supposons que la procédure procP calcule P.
- Donc on peut écrire la procédure gamma suivante :

```
int gamma(x) (int x) {
      if (procP(gamma)) return p0(x) ;
                        else  return p1(x) ;
}
```

- Que vaut procP(gamma) ? Toutes les hypothèses mènent à une contradiction.

Preuve.

- int gamma(x) (int x) {

  if (procP(gamma)) return p_0(x) ;

  else return p_1(x) ;}

- En effet si procP(gamma)=1 alors gamma donne toujours le même résultat que p_0 ($\forall$x) or P(p_0)=0 donc on devrait avoir P(gamma)=P(p_0)=0.

Preuve.

- int gamma(x) (int x) {

  if (procP(gamma)) return $p_0(x)$;

  else return $p_1(x)$;}

- En effet si procP(gamma)=1 alors gamma donne toujours le même résultat que p_0 ($\forall x$) or $P(p_0)=0$ donc on devrait avoir $P(gamma)=P(p_0)=0$.

- Si procP(gamma)=0 alors gamma donne toujours le même résultat que p_1 ($\forall x$) or $P(p_1)=1$ donc on devrait avoir $P(gamma)=P(p_1)=1$.

Chapitre 4. Ensemble Récursivement Enumable

- La fonction caractéristique X d'un ensemble d'entiers E est la fonction définie par :

  X(x)=1 si x $\in$ E et X(x)=0 si x $\notin$ E (X est donc une fonction totale).

  E est dit décidable si sa fonction caractéristique est calculable.

Chapitre 4. Ensemble Récursivement Enumable

- E est récursivement énumérable si

  E={f(0), f(1), f(2),….,f(i),…} où f désigne une fonction totale calculable.
- On dit que f énumère les éléments de E (à l'aide de f on peut afficher tous les éléments de E)

Chapitre 4. Ensemble Récursivement Enumable

- Théorème : tout ensemble fini est décidable

En effet on peut représenter un ensemble fini par sa liste d'éléments le parcours de la liste donne la valeur de la fonction caractéristique pour un entier fixé.

Chapitre 4. Ensemble Récursivement Enumable

- Théorème : tout ensemble E décidable est récursivement énumérable (re)

- Preuve soit fc la procédure que calcule la fonction caractéristique. On peut programmer la fonction f qui énumère les éléments de E.

Chapitre 4. Ensemble Récursivement Enumable

- Théorème : tout ensemble E décidable est récursivement énumérable (re)

- int f (int n){

```
int f (int n){
    int x, cpt=-1 ;
    for (x=0;;x++) {
        if (fc(x)) cpt++ ;
        if (cpt==n) return x ; }
}
```

Chapitre 4. Ensemble Récursivement Enumable

- f énumère bien les éléments de E : f(i) est le (i+1)^{ième} élément de E. On peut donc afficher les éléments de E :

  Pour chaque n afficher f(n)

- On peut aussi afficher les éléments de E de la façon suivante :

  Pour chaque x si fc(x) alors afficher(x) ;

Chapitre 4. Ensemble Récursivement Enumable

- Réciproquement avec la fonction f qui énumère E peut-on écrire la fonction caractéristique ?
- Essayons !

Chapitre 4. Ensemble Récursivement Enumable

- Réciproquement avec la fonction f qui énumère E peut-on écrire la fonction caractéristique ?
- int fc (int x) {

  //si on trouve x dans l'énumération de E on a gagné

  for (int n=0;;n++) if (f(n)==x) return 1 ;

  }

Chapitre 4. Ensemble Récursivement Enumable

- Réciproquement avec la fonction f qui énumère E peut-on écrire la fonction caractéristique ?
- int fc (int x) {

  //si on trouve x dans l'énumération de E on a gagné

  for (int n=0;;n++) if (f(n)==x) return 1 ;

  }
- Peut-on détecter quand il faut retourner 0 ?
- Cette fonction donne toujours 1 si x est dans E et est indéfinie sinon. Elle est appelée la fonction semi-caractéristique de E.

Chapitre 4. Ensemble Récursivement Enumable

- On soupçonne donc qu'il existe des ensembles re et non décidables.
- Pour en construire un on va faire intervenir un autre concept **le temps**.

Chapitre 4. Le rôle du temps

- h(p, x)=1 est si p(x) est défini et 0 sinon. On a vu que h n'est pas calculable.
- Surchargeons l'opérateur h
- h(p, x, t)=1 si p(x) est fini après t étapes et 0 sinon

  Le calcul du temps (seconde, nombre d'instructions, nombre de transitions ...) dépend du modèle mais on ne rentre pas dans ces détails.

Chapitre 4. Le rôle du temps

- $h(p, x, t)=1$ si $p(x)$ est fini après t étapes et 0 sinon

- On exige 2 axiomes :

  - la fonction h à 3 variables (p, x, t) est calculable

  - $h(p, x)=1$ si et seulement si $\exists t$ avec $h(p, x, t)$

Chapitre 4. Le rôle du temps

- Théorème H={(p, x)| tel que p(x) est défini} est récursivement énumérable (et indécidable)
- Preuve :

(i) Il est indécidable car le problème de l'arrêt n'est pas calculable

(ii) Il est re car on peut afficher les éléments de H :

pour chaque triplet (p, x, t)

si h(p, x, t)=1 alors afficher (p, x)

Chapitre 4. Un ensemble non re

- Théorème :

  Si E et son complémentaire sont récursivement énumérable alors E est décidable

- Preuve : Si E et son complémentaire sont re alors il existe une fonction calculable et totale f qui énumère E et une fonction calculable et totale g qui énumère $\overline{E}$.

  En utilisant f et g on peut écrire la fonction caractéristique de E.

Chapitre 4. Un ensemble non re

- En utilisant f et g on peut écrire la fonction caractéristique de E.
- int fc (int x) {

```
    for (int n=0;;n++) {
        if (f(n)==x) return 1 ;
        if (g(n)==x) return 0 ;
    }
```

- Donc E est décidable.

Chapitre 4. Un ensemble non re

- On a vu que H={(p,x) | p(x) est défini} est re et indécidable.
- Soit $\overline{H}$ son complémentaire.

  $\overline{H}$ = {(p,x)| p(x) est non défini}
- $\overline{H}$ ne peut pas être re car sinon d'après le théorème précédent on déduirait que H est décidable.