U.F.R SCIENCES ET TECHNIQUES
Département d'Informatique
B.P. 1155
64013 PAU CEDEX
Téléphone secrétariat : 05.59.40.79.64
Télécopie : 05.59.40.76.54

I- Calculabilité et algorithme : Machine de Turing
Partie II

IV-Calculabilité et Décidabilité

Rappel:

La machine de Turing a été introduite pour formaliser les notions :

- d'**algorithme**
- et de **calculabilité**.

Calculabilité au sens de Turing

Un problème est dit **calculable** s'il peut résolu par un **algorithme** exécutable par une **machine de Turing**.

Tout **problème calculable** au sens de la machine de Turing est **calculable**.

Calcul au sens de la machine de Turing

On appelle **calcul** de Σ sur un mot **w** $\in \Sigma^*$, une suite (finie ou infinie) de **configurations**:

$$C_i \vdash C_{i+1} \qquad i=0,1,2\ldots.$$

-**C0** : configuration **initiale**

-une suite de configurations se termine:

- soit par une configuration **acceptante**
- soit par une configuration **refusante**.

Problème de calculabilité

La machine de Turing a pu mettre en évidence deux **classes de problèmes**:

-ceux qui sont **calculables** (programmables): ceux dont le **calcul se termine**,

-et ceux qui ne le sont pas : ceux dont le **calcul ne se termine pas**.

Décidabilité/Calculabilité

De manière générale, on parle de:

-**décidabilité** quand il s'agit des problèmes où on cherche à déterminer si une certaine **propriété** est vraie ou fausse.

-**calculabilité** quand il s'agit des problèmes de **calcul** au sens large.

Décidabilité

Un **problème de la décision** est la recherche d'une procédure qui indique:

- dans chaque **contexte**
- et au bout d'un **temps fini**,

si une **propriété** **P** est vraie ou fausse,

$$\mathbf{P} \quad \text{ou} \quad \neg\mathbf{P} \ \ (\text{non } P)$$

Calculabilité

Un **problème de calcul** est la construction d'un **algorithme** qui:

- pour chaque fonction **F**
- et chaque argument x de **F**,

calcule la valeur y telle que :

$$y = F(x).$$

Equivalence décidabilité/calculabilité

Les deux notions sont **équivalentes** dans la mesure où :

-la première peut se ramener à la seconde

-et vice et versa.

En effet à toute **propriété** mathématique, il est toujours possible d'associer une **fonction** numérique.

Il suffit de remarquer qu'une **propriété** mathématique **P** est une **fonction** **F** vers un domaine à deux valeurs {vrai, faux}. :

$$\mathbf{F} : \mathbf{D} \rightarrow \{\mathbf{vrai}, \mathbf{faux}\}.$$

Réciproquement, à toute **fonction** numérique, on peut associer une **propriété** mathématique.

En effet, à toute fonction **F** :

$$y = F(x)$$

on peut associer une propriété **P**(x, y) :

- **vraie** si y = **F**(x)
- et **fausse** sinon.

Problème de décision

Considérons la formulation suivante d'un problème P de **décision** :

Etant donné :

- un ensemble **U** de toutes les données possibles
- un ensemble **B** de toutes les données dites “bien” :

$$\mathbf{B} \subseteq \mathbf{U}$$

Pour chaque élément $x \in$ **U**, répondre :

- "OUI" si $x \in$ **B**
- et "NON" si $x \notin$ **B**.

On note : **P** = **P**(**U**,**B**)

Problème décidable

Un problème $P = (U,B)$ est **décidable**

$$\Leftrightarrow$$

Il existe un **algorithme** pour P.

Il existe un **algorithme** pour **P** qui s'arrête et répond :

- "OUI" si la donnée $x \in \mathbf{B}$
- "NON" si la donnée $x \notin \mathbf{B}$.

Problème indécidable

Le problème P est **indécidable**

$\Leftrightarrow$

Il n’existe **pas d’algorithme** pour P.

Problème semi-décidable

Le problème P est **semi-décidable**

$$\Leftrightarrow$$

Il existe un **semi-algorithme** pour P.

Qu'est-ce qu'un **semi-algorithme** ?

Il existe une procédure pour P telle que:

- si $x \in B$ elle retourne "OUI"

- si $x \notin B$ elle retourne "NON"

ou "**ne retourne pas de réponse**".

Constats

1) Il est clair qu'un problème **décidable** est aussi **semi-décidable**.

2) Pour montrer qu'un problème est **décidable**, il suffit de trouver un algorithme: **un seul suffit** !

Problème indécidable

Par contre, pour montrer qu'un problème **P** est **indé-cidable**, il faut :

-considérer **tous** les algorithmes **possibles**

-et montrer qu'**aucun** d'eux **ne résout** **P.**

Problème incalculable

Il existe des problèmes qui ne relèvent pas du domaine du calcul.

Aucun algorithme n'existe pour les résoudre.

Cette affirmation va très loin :

-elle ne signifie pas que nous ne connaissons pas encore, d'**algorithme** pour résoudre ces problèmes.

-elle dit plus précisément que **nous ne connaîtrons jamais de tels algorithmes**: il a été **démontré** que ces algorithme n'**existent pas** !.

Conclusion:

La **calculabilité** est une notion qui marque des **«limites dures»** de l'informatique.

Inconsistance et incomplétude

Supposons que le problème consiste à concevoir un **programme** **P**:

-auquel on fournit les **axiomes** d'un système formel

-en lui demandant d'**appliquer** successivement tous ces axiomes.

Hypothèse:

Le programme **P** est alors capable de lister **toutes** les formules qu'il **peut déduire** des axiomes considérés.

Problème:

Nous voulons savoir si une certaine formule **F** peut être déduite de ces axiomes.

Quatre cas peuvent se présenter pour le programme:

-cas 1: P listera la formule F **et pas** la formule $\neg$ F,

-cas 2: P listera la formule $\neg$ F **et pas** la formule F,

-cas 3: P listera la formule F **et** la formule $\neg$F,

-cas 4: P ne listera **ni** F, **ni** $\neg$F.

Dans le cas 1:

Le programme **P** signifie que:

« la formule **F** est **vraie**. »

Dans le cas 2:

Le programme **P** signifie que:

«la formule **non F** est **vraie »**

Dans le cas 3 :

On conclut à l'**inconsistance**.

P permet, à la fois, de déduire que:

« la formule **F** est **vraie**. »

et son contraire :

«la formule **¬F** est **vraie »**

Dans le cas 4:

On conclut à l’**incomplétude**.

P ne peut conclure ni **F** ni $\neg$**F** .