Décidabilité - Introduction

Maria Malek

7 novembre 2010

Avant Propos

- L'objectif de cette introduction est de faire le lien entre :
  - *d'un côté*, l'ensemble des problèmes à résoudre, les instances des problème ainsi que les programme proposés pour résoudre ces problèmes,
  - *et de l'autre côté*, les langages et les automates.

Problèmes & Programmes - 1

Le problème posé est complètement indépendant du programme proposé pour le résoudre.

- Notion de problème.
- Notion de Programme exécuté sur ordinateur.
- *Exemples d'un problème*
  1. Déterminer si un nombre naturel est pair ou impair.
     - Question générique sur un ensemble d'éléments : nombres naturels.
     - Une instance de problème a une réponse : 37 est pair ?
     - Les instances de ce problèmes peuvent être représentées à l'aide de la *notation binaire*.

Problèmes & Programmes - 2

- Le problème posé est complètement indépendant du programme proposé pour le résoudre.
- Exemples de problème :
- Déterminer si un nombre naturel est pair ou impair.
  - Les instances de ce problèmes peuvent être représentées à l'aide de la notation binaire.
  - Un programme possible serait :
    1. Examiner le dernier chiffre de la représentation
    2. Répondre *nombre pair* si ce chiffre est 0.
    3. Répondre *nombre impair* si ce chiffre est 1.

Problèmes & Programmes - 3

- Le problème posé est complètement indépendant du programme proposé pour le résoudre.
- Exemples de problème :
  1. Déterminer si un nombre naturel est pair ou impair.
  2. Trier un tableau de nombre.
  3. Déterminer si un programme écrit en un langage donné s'arrête quelles que soient les valeurs des données (problème d'arrêt).
- Les deux premiers problèmes sont solubles par un programme, le troisième : non.
- Nous étudions dans ce cours une classe limitée de problèmes : *les problèmes dont la réponse est binaire non ou non.*

Solution à un problème & Programme

- Un programme est une procédure effective.
- Nous formalisations la procédure effective par des automates.
- Ceci nécessite une formalisation des instances du problème : collection d'entiers, chaînes de caractères, etc.
- Simplification : les instances du problèmes peuvent être représentés par un chaîne finie de symboles.
  - Exemple : {0,..,9}, {a,...,z}.

Alphabets & Mots

- Un alphabet est un ensemble fini de symboles
  - Exemples : $\Sigma_1 = \{a, b, c\}$, $\Sigma_2 = \{1, 2, 3, 4\}$
- Un mot défini sur un alphabet est une séquence finie d'éléments de cet alphabet.
  - Un mot a une longueur finie.
  - Exemples : $w_1 = abccba$, $w_2 = 2314$
  - $|w_1| = 6, |w_2| = 4,$

Représentation des problèmes

- Nous pouvons représenter *les instances d'un problème* par des mots ! !
- Soit un problème binaire dont les instances sont encodés sur un alphabet Σ, L'ensemble de mots défini sur Σ est partitionné en 3 sous ensembles :
  1. Les mots pour lesquels la réponse est oui : les instances positives.
  2. Les mots pour lesquels la réponse est non : les instances négatives.
  3. Les mots qui ne représentent pas des instances de problème.
- Nous pouvons regrouper les deux dernières classes.

Langages

- Un langage est un ensemble de mots défini sur le même alphabet.
- Résoudre un problème =
  - Reconnaître le langage décrivant les instances positives.
- Exemple : $\{aab, \epsilon, bbbbba\}$ est un langage défini sur l'alphabet $\{a, b\}$.

Les langages réguliers - 1

- Soient L_1 et L_2 deux langages :
  - Les opérations possibles : $L_1 \cup L_2$, $L_1.L_2$, L_1^*
  - L'ensemble R des langages réguliers sur un alphabet Σ est le plus petit ensemble tel que :
    1. $\Phi \in R$; $\{\epsilon\} \in R$.
    2. $\{a\} \in R$ pour tout $\{a\} \in \Sigma$.
    3. si $A, B \in R$ alors $A \cup B, A.B, A* \in R$
  - Les langages réguliers sont décrits par les expressions régulières.

Les langages réguliers - 2

- Caractéristiques des langages régulières :
  1. Les expressions régulières.
  2. Les automates finis déterministes.
  3. Les automates finis non déterministes.
  4. les grammaires régulières (de type 3).

Grammaires, Langages & Automates

Une grammaire G=(V,Σ,R,S) avec V le vocabulaire, $\Sigma \subset V$ est l'ensemble des terminaux, R étant l'ensemble de règles, S est le symbole de départ appartenant à l'ensemble des symboles non terminaux $(V - \Sigma)$.

- Type 0 : pas de restriction sur les règles : machine de Turing
- Type 1 Grammaires sensible au contexte : $\alpha \rightarrow \beta$ avec $|\alpha| \leq |\beta|$
- Type 2 Grammaires hors contexte : $A \rightarrow \beta$ avec A non terminal : automates à pile
- Type 3 Grammaires régulières : $A \rightarrow wB$ et $A \rightarrow w$ avec A,B non terminaux et $w \in \Sigma^*$: automates finis