Républiquealgérienne démocratique et populaire
Ministre de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mohamed Khider – Biskra
Faculté des Sc.Exactes, et SNV.
Département d'Informatique

CALCULABILITÉ

Niveau: 1er année Master
Option: Génie Logiciel et Systèmes Distribués

2023/2024

Fiche Contact

- **Établissement:** Université Mohamed Khider Biskra
- **Faculté:** Sciences Exactes et Sciences de la Nature et de la Vie
- **Département:** Informatique
- **Public cible:** 1ére année Master, spécialité GLSD
- **Intitulé de l'UE : Modélisation et Vérification**
- **Intitulé de la matière : Calculabilité**
- **Crédit** : 05 | **Coefficient** : 03
- **Durée :** 14 semaines
- **Enseignant:** *Dr. Mohamed RAMDANI*
- **Contact par mail:** mohamed.ramdani@univ-biskra.dz

Objectifs du Cours

Permet à l'étudiant d'apprendre les notions de base du calcul informatique. Les problèmes les plus importants liés à la décidabilité et à la terminaison.

Définir, indépendamment de la technologie :

Ce qui est calculable ou pas (**théorie de la calculabilité**) ;

Connaissances préalables

Vous devrez avoir assimilé les concepts de base mathématique avant de commencer ce module, Il est recommandé aux apprenants de connaître:

- Théorie de langages,
- Automates,
- Logiques Mathématiques.

RÉFÉRENCES

- Pierre Wolper, « Introduction à la calculabilité », 3eme édition, Dunod, 2006
- Olivier Carton, « *Langages formels, calculabilité et complexité* »,Vuibert, 2008.

INTRODUCTION

- L'informatique se repose sur deux points essentiels :

1. **Mathématiques** : Qu'est ce qu'on peut calculé ? Et qu'est ce qu'on peut pas calculé ?

2. **Génie électrique et électronique** : Comment construire un ordinateur . (pas dans ce cours)

INTRODUCTION

Il existe des problèmes qui n'ont pas une solution algorithmique (qui ne sont pas soluble; indécidables), quelque soit la technologie et la capacité des machines (ordinateurs).

Objectifs:

- Connaître les notions liées à la décidabilité pour identifier ce type de problèmes et
- Savoir les limites de l'automatisation.

L'objectif de ce cours est d'introduire et formaliser ces notions fondamentales.

INTRODUCTION

- La question à laquelle nous allons essayer de répondre dans ce cours est:

  - Que peut-on calculer ?
  - Qu'est-ce qui ne peut pas être calculé ?
  - et où est la ligne entre les deux ?

DÉFINITION DE CONCEPTS

Quelle est l'objectif de l'informatique ?

La solution **automatique** des **problèmes**

il y a deux notions fondamentales :

- Problème
- programme

NOTION DE PROBLÈME

Commençons par examiner un exemple.

Exemple 1 : **Déterminer si un nombre naturel est pair ou impair est un problème.**

Cet exemple nous permet déjà de mettre en évidence plusieurs caractéristiques de la notion de problème.

- Un problème est une question générique, c'est-à-dire qui s'applique à un ensemble d'éléments (dans le cas de l'exemple, les nombres naturels).
- Chaque instance du problème, c'est-à-dire la question posée pour un élément particulier de l'ensemble (35 est-il pair ?) a une réponse (non).
- La notion de problème est indépendante de la notion de programme.

On peut écrire un programme qui résout un problème, mais le problème n'est pas défini par le programme. Par ailleurs, il existe le plus souvent plusieurs programmes permettant de résoudre un même problème.

NOTION DE PROBLÈME

Pour résoudre un problème par un programme, il est nécessaire de se fixer une représentation des instances du problème. C'est sur cette représentation qu'opérera le programme.

NOTION DE PROBLÈME

Exemple 2 Les instances du problème de l'exemple 1 (les nombres naturels) peuvent être représentées à l'aide de la notation binaire. Un exemple de programme qui résout le problème est alors celui qui examine le dernier chiffre de la représentation, répond nombre pair si ce chiffre est 0 et répond nombre impair si ce chiffre est 1.

Ce problème pourrait aussi être résolu par le programme (un peu idiot, soit) qui convertit la représentation binaire en représentation décimale et répond nombre pair si le dernier chiffre de cette représentation est dans l'ensemble {0,2,4,6,8} et nombre impair dans les autres cas.

NOTION DE PROBLÈME

Exemple 3

- Trier un tableau de nombres est un problème.
- Déterminer si un programme écrit dans un langage tel que C ou C++ s'arrête quelles que soient les valeurs des données qui lui sont fournies est un problème (problème de l'arrêt).

Le premier de ces problèmes est soluble par un programme exécuté sur un ordinateur (méthodes de tri connues). La théorie que nous allons développer dans cet ouvrage montrera que le 2eme ne l'est pas.

NOTION DE PROBLÈME

Dans ce cours, nous considérerons seulement une classe limitée de problèmes :

Les problèmes dont la réponse est **binaire** (oui/non, 0/1). Ce type de problème est appelé Problème de décision.

Un **problème de décision** est une question **générique**, c'est-à-dire qui s'applique à un ensemble d'éléments, dont la réponse peut être **oui** (instance positive) ou **non** (instance négative).

PROBLÈME DE DÉCISION

- La définition d'un problème de décision est la donnée
  - d'un ensemble ***E*** d'instances et
  - d'un sous-ensemble ***P*** inclus dans ***E*** des instances dites positives (dont la réponse est oui).

- Exemple: Problème des nombres paires
- ***E :*** l'ensemble des nombre naturels N.
- ***P*** = { n de N| n est pair}
- **12** est une instance positive.
- **35** est une instance négative.

DÉFINITION DE PROGRAMME

- Programme: Séquence d'instruction destinée à être exécutée par un ordinateur.
- Plus formellement, on parle de **Procédure effective (PE)**.
- Une solution est appelée formellement une ***procédure effective*** pour un problème donné si:
  - Elle est constitué d'un nombre fini, d'instructions finies.
  - Quand appliquée à une instance de son problème:
    - Elle se termine toujours après un nombre fini d'étapes.
    - Elle donne toujours la bonne réponse.
  - Elle ne nécessite aucune intervention pour être appliquée.

PROCÉDURE EFFECTIVE?

- **Déterminer si le programme n'a pas de boucle ou de séquence d'appels récursifs infinis.**

```
while (x > 1)
 {
        if (x mod 2 = 0) then
        x := x/2;
        else
        x := 3x + 1;
}
```

PROCÉDURE EFFECTIVE?

- **Déterminer si le programme n'a pas de boucle ou de séquence d'appels récursifs infinis.**

```
while (x > 1)
 {
        if (x mod 2 = 0) then
        x := x/2;
        else
        x := 3x + 1;
}
```

  - Il n'y a pas de ***procédure effective*** qui résout ce problème.
  - *Problème de l'arrêt*.

On verra qu'il est indécidable!!!

FORMALISATION DE PE

- Une procédure effective peut être représenté de différentes manière (programmation, fonctions, …)
- Nous allons utiliserun formalisme très simple et intuitif: **Automates et Machine de Turing.**

FORMALISATION DE PROBLÈMES

Si un problème doit être résolu par une procédure effective,

1. Ces instances doivent être représentées d'une façon accessible à cette procédure effective.
2. Les machines de Turing travaillent sur des **mots** d'un alphabet fixé.

Donc:
On va coder les instances de nos problèmes par des mots.

CODAGE DE PROBLÈME

Une ***instance*** d'un problème est représentée par un mot sur un alphabet prédéfini.

a) l'alphabet est défini selon la nature du problème.

   Exp: problème de nombres pairs, la représentation binaire est la plus appropriée.

b) Un problème est donc un langage composés des mots représentant ses instances positives.

c) Un codage est une fonction injective de ***E*** dans Σ^*.

   - Le codage d'une instance x est le mot sur Σ^* noté par $<x>$.
   - Le langage associé à un problème P est l'ensemble

$$\mathbf{Lp} = \{ < \boldsymbol{x} > \quad | \ \boldsymbol{x} \ \mathbf{de} \ \boldsymbol{P} \}.$$

CODAGE DE PROBLÈME

EXEMPLE 1

Un entier peut être codé par sa représentation décimale, binaire, ou unaire.

1. Codage décimal: $\sum = \{ 0,..,9\}$, par exemple <5> = 5.
2. Codage binaire: $\sum = \{ 0,1 \}$, par exemple <5> = 101.
3. Codage unaire: $\sum = \{ 1 \}$, par exemple <5> = 11111.

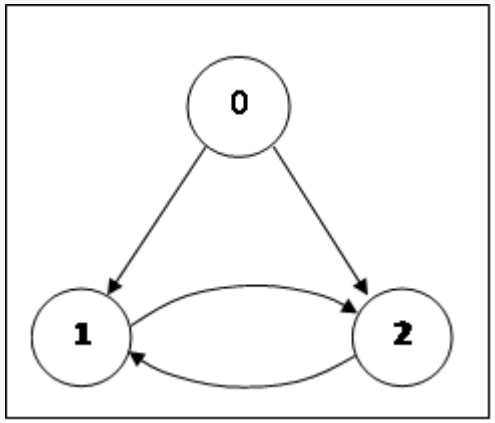
CODAGE DE PROBLÈME

Pour coder un graphe G=(V,E), on peut adopter le codage suivant:

1) ∑= { 0,1,(,),',',} .
2) Un sommet *v* est codé par son numéro en représentation binaire.
3) Le codage <V> de l'ensemble des sommets *V* est la suite des codages des sommets séparés par des virgules et entourés par des parenthèses.
4) Chaque arc partant du sommet *u* vers *v* est représenté par (< u >, <v>).
5) Le codage <E> de l'ensemble des arc *E* est formé par la suite des codage des arcs séparés par des virgules et entourés par des parenthèses.
6) Le codage <G> d'un graphe G est le mot (<V> ,<E>).

EXEMPLE 2 (SUITE)

QUEL EST LE CODAGE DU GRAPHE CI- DESSOUS?



Formalisation des problème

ALPHABET ET MOT

- **Un alphabet** (noté $\sum$) est un ensemble fini de symboles (caractères).
- Les symboles importent peu. La chose la plus importante est la taille.
- Exemple: un alphabet de taille 2 , peut être: {0, 1}, {a,b}, {*, +}, …
- **Un mot** (chaîne de caractère) défini sur un alphabet est une suite fini de symboles de cet alphabet.
- **La longueur** d'un mot w est le nombre de symboles qui le composent, dénotée par |w|

LANGAGE

- Un langage est un ensemble de mots définis sur le même alphabet.
- Un problème est caractérisé par le langage composé des encodages de ses instances positives.
- Résoudre un problème = reconnaître le langagedes encodages des instances positives de ce problème.

Exemples:

- *{aa, abb, aaaaabbbb, ε, bbbbbb} est un langage sur l'alphabet {a, b}.*
- *{ε, 0,1,00,01,10,11,000,001,010,011,…} est le langage de tous les mots sur l'alphabet {0,1}.*
- *Le langage contenant les mots représentant des programmes en langage C qui se terminent toujours.*

DESCRIPTION DE LANGAGES

- Il n'existe pas une notation qui permet de décrire tous les langages.
- Si le langage est fini, on peut le définir en énumérant ces mots.
- On peut définir certains langages en fonction d'autres langages plus simples par les opérations ci-dessous.
- Soit L1 et L2 deux langages:

  - Union: $L_1 \cup L_2 = \{ w \mid w \in L_1 \text{ ou } w \in L_2 \}$
  - Concaténation
  - Fermeture itérative
  - Complément

EXPRESSIONS RÉGULIÈRES

Les langages réguliers sont décrits par les expressions régulières.

Définition (Langages réguliers): Soit R l'ensemble des langages réguliers, R est défini commesuit:

(1) $\varnothing \in \mathcal{R}$; $\{\varepsilon\} \in \mathcal{R}$;

(2) $\{a\} \in \mathcal{R}$ pour tout $a \in \Sigma$;

(3) si $A, B \in \mathcal{R}$, alors $A \cup B$, $A \cdot B$ et $A^* \in \mathcal{R}$.

EXPRESSIONS RÉGULIÈRES

Définition (Expressions régulières)

1. *Ø , ε, et les élément de $\sum$ sont des expressions régulières.*
2. *Si α et β sont des expressions régulières, alors (α U β), (αβ) et (α)* sont des expressions régulières.*

IL EXISTE DES LANGAGES NON RÉGULIERS

- Il n'ya pas assez d'expression régulières pour représenter tous les langages.
- Il faut comparer le nombre d'expressions et le nombre de langages.
- Les deux ensembles sont infinis!

COMPARAISONS ENTRE ENSEMBLES

- Si deux ensembles ont le même nombre d'éléments, on peut mettre leurs éléments en correspondance (un à un).
  - Exp: {0,1,2,3} et {a,b,c,d} ont la même taille.
    Ils peuvent être mis en bijection {(0,a),(1,b),(2,c),(3,d)}

- Si deux ensembles peuvent être mis en bijection
  ➔ Ils ont la même cardinalité.

- Le premier et le plus petit ensemble est l'ensemble des entiers naturels.

- Un ensemble infini est dénombrable, s'il existe une bijection entre cet ensemble et l'ensemble des entiers naturels.

EXEMPLES:

- Tout sous-ensemble de N est dénombrable.
- L'ensemble des mots sur l'alphabet {a,b} est dénombrable.
- L' ensemble des nombres rationnels est dénombrable.
- L' ensemble des expressions régulières est dénombrable.
- L'ensemble des sous-ensemble d'un ensemble dénombrable n'est pas dénombrable.

DIAGONALISATION (1/3)

Théorème **I**: L'ensemble des sous-ensembles d'un ensemble dénombrable n'est pas dénombrable.

Démonstration: (Technique de diagonalisation)

- Soit un ensemble dénombrable $A=\{a_0,a_1,\ldots\}$
- Soit **S** l'ensemble des sous-ensembles de **A**
- **Hypothèse**: Supposant que **S** est dénombrable, $S=\{s_0,s_1,\ldots\}$
- Puisque les deux ensembles sont **dénombrables**, selon l'hypothèse, alors il existe une bijection entre eux.
- Nous pouvons schématiser cette bijection par le tableau suivant:

DIAGONALISATION (2/3)

	a_0	a_1	a_2	a_3	a_4	$\cdots$
s_0	×	×		×		
s_1	×	□		×		
s_2		×	×		×	
s_3	×		×	□		
s_4		×		×	□	
$\vdots$						

- le symbole x: signifie que l'élément a_j de la colonne appartient à l'ensemble s_i de la ligne.
- Le symbole carré: signifie que l'élément a_i n'appartient pas à l'ensemble s_i du même rang. (absence de croix sur la diagonale)

DIAGONALISATION (3/3)

Considérons l'ensemble $D=\{a_i \mid a_i$ *n'appartient pas à* $s_i\}$

- L'ensemble ***D*** est défini par les symboles carrés de la diagonale du tableau.
- ***D*** est un élément de ***S*** (puisqu'il est un sous-ensemble de A).
- ***Mais*** l'ensemble **D** ne peut être un des élément s_i car:
  - Si D est par exemple s_k ,
  - a_k appartient à ***D*** ssi a_k n'appartient pas à ***D***.
  - Alors, nous avons une contradiction (qui contredit l'hypothèse)

  ☐ l'ensemble S n'est pas dénombrable.

CONCLUSION

Le ***théorème I*** implique que tous les langages ne peuvent être réguliers :

- L'ensemble des mots est un ensemble dénombrable.
- L'ensemble des langages (ensemble de sous-ensemble de mots) n'est pas dénombrable.
- l'ensemble des expressions régulières est dénombrable.
- Donc l'ensembles des langages réguliers est dénombrable.

Alors:

Il y a plus de langages que de langages réguliers.

CONCLUSION

- Ensemble des langages non dénombrable
  - ➔ Ensemble des problèmes non dénombrable
- Ensemble des chaînes de caractères (mots) dénombrable
  - ➔ Ensemble des procédure effectives dénombrable
- Ce résultat nous permettra d'établir que:

il existe des problèmes insolubles par des procédures effectives