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

Preuve de l'indécidabilité et problème de l'arrêt

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

2023/2024

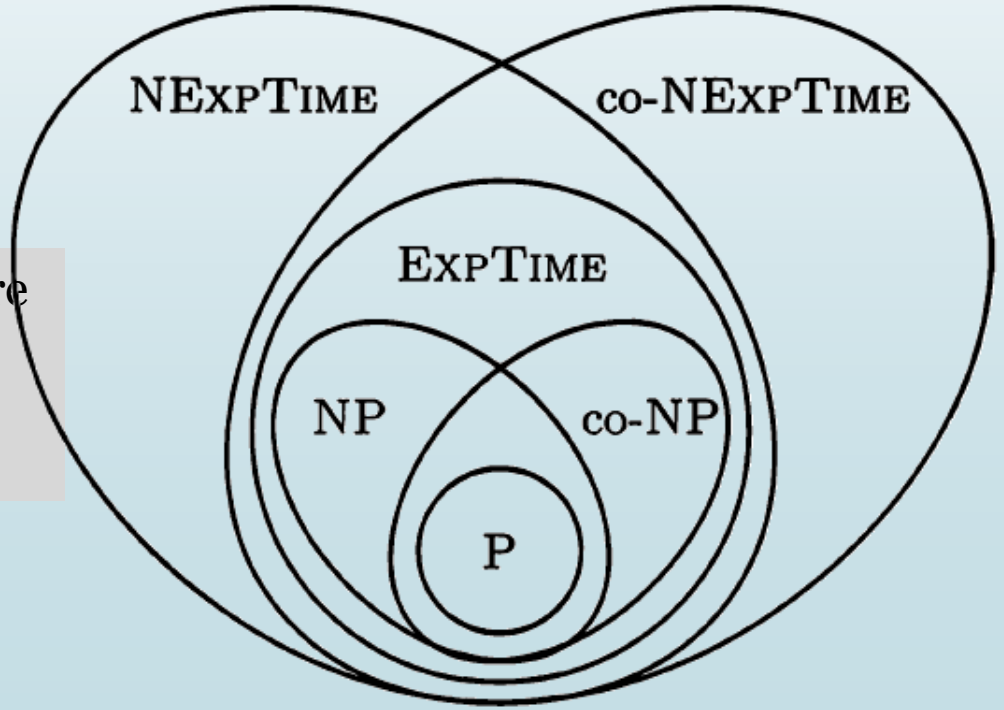
La décidabilité

- La décidabilité se réfère à la capacité d'un problème à être résolu de manière algorithmique.
- Un problème est dit décidable s'il existe un algorithme qui, pour toute instance du problème, peut déterminer de manière certaine et en un temps fini si l'instance donnée appartient à la solution ou non.

Classes de Complexité

- La classe P (Problèmes polynomiaux)
- La classe NP (Nondeterministic Polynomial)
- La classe NP-complet (Nondeterministic Polynomial-Complete)
- La classe NP-difficile (Nondeterministic Polynomial-Hard)

Les classes de la décidabilité nous permettent de comprendre les limites de la résolubilité algorithmique,
c'est-à-dire de déterminer si un problème donné peut être résolu avec un PE ou non.



Classes de décidabilité

- **La classe RE (Reconnaissables) :** Un langage est dit reconnaissable (ou récursivement énumérable) s'il existe une machine de Turing qui l'accepte en s'arrêtant sur les mots du langage et qui peut éventuellement boucler sur les mots qui n'appartiennent pas au langage. En d'autres termes, la machine de Turing peut reconnaître tous les mots du langage, mais peut ne pas s'arrêter sur les mots qui ne lui appartiennent pas.

- **La classe co-RE (Reconnaissables complémentaires) :** Le complément d'un langage reconnaissable est également reconnaissable. Un langage est dit co-reconnaissable s'il existe une machine de Turing qui accepte tous les mots qui n'appartiennent pas au langage, mais qui peut boucler sur les mots qui appartiennent au langage.

Classes de décidabilité vs les machines de Turing et les langages formels

- **La classe R (Récursifs) :** Un langage est dit récursif (ou décidable) s'il existe une machine de Turing qui l'accepte en s'arrêtant sur tous les mots du langage et qui peut également s'arrêter sur les mots qui n'appartiennent pas au langage. En d'autres termes, la machine de Turing peut décider de manière certaine si un mot appartient ou non au langage.

- **La classe co-R (Récursifs complémentaires) :** Le complément d'un langage récursif est également récursif. Un langage est dit co-récursif s'il existe une machine de Turing qui accepte tous les mots qui n'appartiennent pas au langage, mais qui peut également s'arrêter sur les mots qui appartiennent au langage.

Classes de décidabilité

Définition:

La classe de décidabilité R est l ensemble des langages décidables par une machine de Turing,

- Les classes de décidabilité en relation avec les machines de Turing et les langages formels nous permettent de classifier les problèmes selon leur résolubilité.

Langages indécidables :

- Un langage est dit indécidable s'il n'existe aucune machine de Turing qui puisse décider de manière générale si un mot donné appartient ou non à ce langage.
- En d'autres termes, il n'y a pas d'algorithme qui puisse toujours fournir une réponse correcte pour tous les mots du langage.
- *La classe* R *est contenue dans la classe* RE ($R \subseteq RE$).

Ce fait est une conséquence directe des définitions : si un langage est décidé par une machine de Turing (est dans la classe R), il est aussi accepté par cette machine de Turing (il est dans la classe RE).

Un premier langage indécidable

Théorème 1: Le langage L_o défini par :

$$L_0 = \{w | w = w_i \wedge M_i \; n'accepte \; pas \; w_i\}$$

est indécidable et n'appartient pas à RE

Un deuxième langage (moins) indécidable

Théorème 2: le langage

$$\overline{L_0} = \{w | w = w_i \wedge M_i \textit{ accepte } w_i\}$$

est indécidable mais appartient à RE (semi-décidable)

Technique de la réduction

- Permet de démontrer l'indécidabilité d'un langage *L2* connaissant l'indécidabilité d'un langage *L1* $(L_1 \notin \mathbf{R})$

1. On démontre que s'il existe un algorithme qui décide le langage *L2*, alors il existe un qui décide *L1*.
   ➔ On donne un algorithme (MT) qui décide L1 en se servant (comme d'un sous-programme) d'un algorithme qui décide L2.
2. Ceci mène à une contradiction car L1 est indécidable
   ➔ donc L2 est indécidable.

Le langage Universel (problème d'acceptation)

- Le langage LU défini par:

$$\mathrm{LU} = \{< M, w > \mid M \text{ accepte } w\}$$

- LU appartient à RE ➔ La MT universelle (vu dans le chapitre précédent) accepte ce langage.

- LU est indécidable (n'appartient pas à R) ➔ par la technique de réduction.

Réduction de LU

- Une réduction de $\mathrm{LU} = \{< M, w > \mid M \text{ accepte } w\}$

à partir de $\overline{L_0} = \{w \mid w = w_i \wedge M_i \text{ accepte } w_i\}$

permet de démontrer que $\mathrm{LU} \notin \mathrm{R}$

- Algorithme de réduction $\boldsymbol{LU}$ à partir de $\overline{L_0}$:

1. Déterminer l'indice i tel que $w=w_i$.
2. Déterminer la machine M_i.
3. Appliquer la procédure de décision de LU pour $< M_i, w_i>$. Si le résultat est positif, accepter, sinon rejeter.

Qu'en est-il de $\overline{\mathrm{LU}}$?

Le problème de l'arrêt (Halting Problem)

- **Le problème de l'arrêt (Halting Problem)** est un exemple célèbre de langage indécidable. Il consiste à déterminer, étant donné un programme informatique et une entrée, si le programme s'arrête ou s'il entre dans une boucle infinie pour cette entrée donnée.

- Alan Turing a démontré en 1936 que le problème de l'arrêt est indécidable, ce qui signifie qu'il n'existe pas de programme ou d'algorithme qui puisse résoudre ce problème pour tous les cas possibles.

- La preuve de l'indécidabilité du problème de l'arrêt est basée sur un raisonnement par l'absurde. On suppose qu'il existe un algorithme qui peut décider si un programme s'arrête ou non. En utilisant cette hypothèse, on peut construire une machine de Turing qui aboutit à une contradiction logique, prouvant ainsi que l'algorithme supposé n'existe pas.

Problème de l'arrêt

- Le problème indécidable le plus connu.
- Il s'agit de déterminer si une MT (algorithme) s'arrête pour un mot d'entrée donné.

$$\mathrm{H} = \{< M, w > | \ M \text{ s'arrête sur } w\}$$

- Réduction à partir de LU:

Décider une entrée <M,w> de LU en utilisant une procédure de décision de H.

1. Appliquer l'algorithme décidant H sur $<M,w>$.
2. Si la réponse est non, alors rejeter $< M, w > \notin \mathrm{LU}$
3. Sinon (M s'arrête sur w), alors simuler M sur w et retourner la réponse (oui ou non)

Problème du vide

- Si l'ensemble des mots acceptés par une machine de Turing M est vide

$$L = \{ \langle M \rangle \mid L(M) = \emptyset \}$$

- On montre que le problème d'acceptation (LU) se réduit au problème du vide.

Théorème de Rice

- Beaucoup de problèmes sur les machines de Turing sont indécidables.

- Rice: toutes les questions non triviales portant sur les langages des machines de Turing sont indécidables.

- On dit qu'une propriété P sur les langages est *non triviale* si:
  - il existe au moins un langage récursivement énumérable L qui vérifie P
  - et au moins un langage récursivement énumérable L' qui ne vérifie pas P.

Théorème de Rice

- Si une propriété P est triviale, tous les langages récursivement énumérables vérifient ou ne vérifient pas la propriété P.

Théorème (Rice). *Pour toute propriété non triviale P sur les langages, le problème de savoir si le langage L(M) d'une machine de Turing M vérifie P est indécidable.*

Conclusion

- Les langage accepter par une machine Turing sont des langages Récursivement Enumérable .
- L'incomplétude et la non-calculabilité sont des concepts fondamentaux en informatique théorique qui mettent en évidence les limites de ce qui peut être résolu par des algorithmes.
- Cela peut être interprété de plusieurs manières :
  1. Limite des capacités de calcul
  2. Barrières théoriques
  3. Implications philosophiques
  4. Importance de l'heuristique