Leçon 914 : Décidabilité et indécidabilité. Exemples.

Julie Parreaux

2018 - 2019

Références pour la leçon

[1] Carton, *Langages formels, calculabilité et complexité.*
[2] Huth et Ryan, *Logic in computer science : Modelling and reasonning about systems.*
[3] Wolper, *Introduction à la calculabilité.*

Développements de la leçon

Caractérisation RE	Indécidabilité du problème de validité dans la logique FO

Plan de la leçon

Motivations

Ce qu'en dit le jury

Le programme de l'option offre de très nombreuses possibilités d'exemples. Si les exemples classiques de problèmes sur les machines de Turing figurent naturellement dans la leçon, le jury apprécie des exemples issus d'autres parties du programme : théorie des langages, logique,...

Le jury portera une attention particulière à une formalisation propre des réductions, qui sont parfois très approximatives.

1 La théorie de la décidabilité

1.1 Des modèles de calcul

Des modèles de calculs qui donnent un cadre à la théorie.

— *Définition* : MT / langages décidables / fonction calculables
— *Théorèmes* : équivalence de ces modèles
— Thèse de Church

1.2 Des problèmes de décision

Ce sont les problèmes sur lesquels on décide l'indécidabilité ;)

— Définition d'un problème de décision
— Correspondance avec les langages et les fonctions

1.3 Les classes R et RE

Ces classes permettent de définir la frontière entre le décidable et l'indécidable.

— *Définition* : Classes R et RE + exemples
— *Remarque* : RE est non vide (argument diagonal)
— *Définition* : énumérateur + exemples
— *Définition* : co-RE + exemples (et contre-exemples)
— *Théorème* : Caractérisation de RE DEV

2 Prouver l'indécidabilité

Comment savoir si un problème est indécidable ? Et surtout comment le montrer ?

2.1 Un premier problème indécidable

— *Problème* : Arrêt
— *Théorème* : Arrêt est RE et indécidable

2.2 La technique de la réduction

Pour montrer qu'un problème est indécidable, nous utilisons une technique de preuve : la réduction. Pour appliquer le principe de réduction il nous faut connaître un premier problème indécidable.

— *Définition* : Réduction
— *Propriété* : Implications du à la réduction
— *Exemples* : réductions

3 Expressivité et décidabilité : un compromis

Plus un modèle de calcul, une machine, un objet est expressif moins il est décidable.

3.1 Une frontière a atteindre

Illustration de cette expressivité contre la décidabilité

Hiérarchie de Chomsky

— Décidabilité des langages rationnels : il est difficile de trouver un problème indécidable (exemple)
— Décidabilité et indécidabilité des langages algébriques (exemples)
— Machines de Turing : Rice
— *Application* : langage de programmation (sémantique : correction et terminaison)

Logique

— Décidabilité de la validité et de la satisfiabilité dans la logique propositionnelle.
— Indécidabilité de validité dans la logique du premier ordre DEV
— Application aux bases de données
— Décidabilité de Presburger et indécidabilité de Peano

3.2 Indécidabilité : rien n'est perdu

3.3 Décidabilité : rien n'est gagné

Pas de limite théorique mais des limites "pratiques"

— Théorie de la complexité
— Déjà difficile pour NP alors pour EXPTIME

Quelques notions importantes

Problèmes de décisions en logique

Nous allons maintenant présenter trois problème de décision légitime lorsque nous étudions une logique : le problème de la satisfiabilité, de la validité et du model checking. Nous rappelons également leur complexité (si elle existe).

Définition. Une formule (close) φ est satisfiable s'il existe un modèle $\mathcal{M}$ tel que $\mathcal{M}, v \models \varphi$.

Définition. Une formule (close) φ est valide (ou tautologique) si pour tout modèle $\mathcal{M}$, $\mathcal{M}, v \models \varphi$.

Définition. On définit le problème SAT sur les formules d'une logique.
Problème : SAT
entrée : ϕ une formule (close)
sortie : Oui si ϕ est satisfiable ; non sinon

Définition. On définit le problème VALIDE sur les formules d'une logique.
Problème : VALIDE
entrée : ϕ une formule (close)
sortie : Oui si ϕ est valide ; non sinon

Définition. On définit le problème MODEL-CHECKING sur les formules d'une logique.

Problème : MODEL-CHECKING
entrée : ϕ une formule (close), $\mathcal{M}$ un modèle fini
sortie : Oui si $\mathcal{M} \models \phi$; non sinon

Présentons quelques résultats sur ces problèmes de décisions.

Problème	Propositionnelle	FO
SAT	NP-complet (Cook)	Indécidable
VALIDE	NP-complet	Indécidable
MODEL-CHECKING	P	PSPACE-complet

Technique de preuve : la réduction

Pour montrer qu'un problème apparaît dans une certaine classe de complexité (et même sa dureté) ou qu'il est indécidable, nous utilisons une technique de preuve : la réduction. Pour appliquer le principe de réduction il nous faut connaître un premier problème possédant les propriétés que l'on souhaite montrer sur le deuxième.

Nous présentons ici le principe de la réduction dans sa généralité puis nous verrons comment le spécialiser pour en faire ce que nous souhaitons.

Définition. Une réduction d'un problème A à un problème B (Figure 1) est une fonction tr calculable telle que pour tout w instance de A, w est une instance positive de A si et seulement si $tr(w)$ est une instance positive de B. On note $A \leq B$.

On dit que A se réduit à B s'il existe une réduction de A à B (intuitivement, A est plus facile que B).

Remarque : En fonction des propriétés sur la fonction tr, on obtient différentes réductions qui vont nous permettre de spécialiser la réduction au résultat que nous souhaitons montrer.

Théorème (Principe de la réduction)**.** *Si A se réduit à B, alors si P est une propriété sur B alors P est une propriété sur A (dans notre cas, P peut être l'appartenance à une classe de complexité ou être le caractère indécidable d'un problème, ...).*

Démonstration. On raisonne par l'absurde et grâce à la fonction de traduction, on obtient une contradiction. □

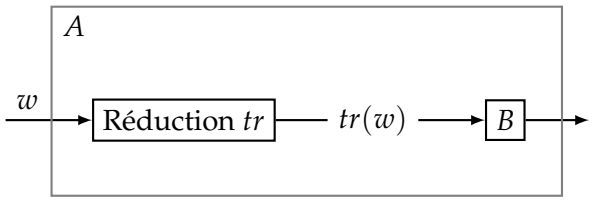

FIGURE 1 – Schéma du principe de réduction du problème A au problème B.

Nous allons donner quelques réductions de A à B et leurs propriétés. On note C une classe de complexité telle que $P \subseteq C$.

Réduction	Propriétés	Conséquences
Calculable	tr est calculable par une machine de Turing (variante de MT : équivalence)	A indécidable $\Rightarrow$ B indécidable B décidable $\Rightarrow$ A décidable
Temps polynomial	tr est calculable par une machine de Turing déterministe en temps polynomial	A est C-dur $\Rightarrow$ B est C-dur $(C \neq P)$ $B \in C \Rightarrow A \in C$
Espace logarithmique	tr est calculable par une machine de Turing déterministe en espace logarithmique (la MT a trois rubans)	A est D-dur $\Rightarrow$ B est D-dur $B \in D \Rightarrow A \in D$ $(D \neq P)$ avec $D \in \{L, NL, co-NL, P\}$

Remarque : La réduction en espace logarithmique est une réduction très spéciale car une machine qui travail en espace logarithmique a au moins deux rubans (il ne faut pas que l'entrée rentre dans la calcul). Mais pour la réduction, la sortie n'est pas non plus dans la borne de la mémoire utilisé (notons qu'elle est polynomiale (car

une réduction en espace logarithmique s'exécute en temps polynomial)), il nous faut donc un troisième ruban. La machine de Turing effectuant la réduction a donc trois rubans : un ruban contenant l'entrée en lecture seule et en une seule passe ; un ruban de travail logarithmique et un ruban de sortie polynomial en écriture seule et en une seule passe.

Proposition. *Une réduction en espace logarithmique est une réduction en temps polynomial.*

Indécidabilité sur les machines de Turing

Nous allons présenter un résumé sur les problèmes de décisions pour les machines de Turing. La majorité (sinon la totalité) de ces problèmes sont indécidables. Nous donnerons alors quelques éléments de leur preuve d'indécidabilité et de leurs applications.

Définition. Le *problème de l'*ARRÊT sur une machine de Turing déterministe.

Problème ARRÊT
entrée : Une machine de Turing déterministe M ; un mot w
sortie Oui si $M(w)$ s'arrête ; non sinon

Théorème (Un premier problème indécidable [3])**.** *Le problème de l'*ARRÊT *est indécidable et dans RE.*

Idée de la preuve. On donne une machine de Turing universelle $\mathcal{U}$ qui accepte l'entrée $\mathcal{M}$,w si et seulement si $\mathcal{M}$ s'arrête sur w. Ceci montre que le problème de l'arrêt est récursivement énumérable.

Pour montrer que le problème de l'ARRÊT est indécidable, on raisonne par l'absurde. Il existe alors une machine de Turing $\mathcal{A}$ telle que elle s'arrête pour tout entrée $\mathcal{M}$, w et accepte une telle entrée si et seulement si $\mathcal{M}(w)$ termine. On construit une machine PARADOXE (algorithme 1). On remarque que PARADOXE(PARADOXE) ne termine pas si et seulement si $\mathcal{A}$ accepte (PARADOXE, ⟨PARADOXE⟩) où ⟨PARADOXE⟩ est le codage de la machine de Turing PARADOXE. Soit PARADOXE(PARADOXE) ne termine pas si et seulement si PARADOXE(PARADOXE) termine. Contradiction. □

Algorithm 1 La procédure PARADOXE de la preuve de l'indécidabilité du problème de l'ARRÊT.

```
1: procedure PARADOXE(M) X
2:     if A accepte (M, ⟨M⟩) then
3:         Boucler
4:     else
5:         Accepter
6:     end if
7: end procedure
```

Algorithm 2 La procédure $\mathcal{N}_{\mathcal{M},w}$ de la preuve du théorème de Rice.

```
1: procedure N_{M,w}(x)
2:     M(w).
3:     if G accepte x then
4:         accepter
5:     else
6:         rejeter
7:     end if
8: end procedure
```

Applications :

— Le problème de l'ARRÊT est le premier problème que nous avons montrer qu'il était indécidable, nous allons donc l'utiliser pour des réductions afin de montrer qu'il n'est pas le seul problème qui est indécidable.

— Le théorème de Rice et ses applications.

— Si on identifie une machine de Turing à un langage de programmation (qui contient une boucle while, par exemple IMP), on vient de montrer que déterminer si un programme d'un tel langage est termine indécidable.

Théorème (Rice [3])**.** *Pour toute propriété non triviale $\mathcal{P}$ sur les langages récursivement énumérables, le problème de savoir si le langage $L(\mathcal{M})$ d'une machine de Turing $\mathcal{M}$ vérifie $\mathcal{P}$ est indécidable.*

Démonstration. Sans perte de généralité, on suppose que $\emptyset \in \mathcal{P}$. On définit le problème $P_{\mathcal{P}}$.

Problème $P_{\mathcal{P}}$
entrée : Une machine de Turing $\mathcal{M}$
sortie : Oui si $L(\mathcal{M}) \in \mathcal{P}$; non sinon

Réduisons ARRÊT à $P_{\mathcal{P}}$ (voir Figure **??**). Soit $G \in \mathcal{P}$. Comme $G \in RE$, il existe une machine G qui accepte G. La réduction tr est définie par $tr(\mathcal{M}, w) = \mathcal{N}_{\mathcal{M},w}$ où $\mathcal{N}_{\mathcal{M},w}$ est la machine décrite dans l'algorithme 2.

1. tr est une fonction calculable : on construit effectivement $\mathcal{N}_{\mathcal{M},w}$ à partir de $\mathcal{M}$ et w ;
2. $(\mathcal{M}, w)$ instance positive de ARRÊT si et seulement si $\mathcal{M}$ s'arrête sur w. Comme

$$L(\mathcal{N}_{\mathcal{M},w}) = \begin{cases} G & \text{si } \mathcal{M} \text{ s'arrête sur } w \\ & \text{sinon} \end{cases}$$

$(\mathcal{M}, w)$ instance positive de ARRÊT si et seulement si $L(\mathcal{N}_{\mathcal{M},w}) \in \mathcal{P}$. Donc, $(\mathcal{M}, w)$ instance positive de ARRÊT si et seulement si $tr(\mathcal{M}, w) = \mathcal{N}_{\mathcal{M},w}$ est instance positive de $\mathcal{P}$.

□

Applications :

— Si on considère $\mathcal{P} = \emptyset$, le problème LANGAGEVIDE est indécidable.
Problème LANGAGEVIDE
entrée : Une machine de Turing $\mathcal{M}$
sortie : Oui si $L(\mathcal{M}) = \emptyset$; non sinon

— Si on identifie une machine de Turing à un langage de programmation, on vient de montrer que la correction d'un programme est un problème indécidable.

Définition. Le *problème de l'*ACCEPTATION sur une machine de Turing déterministe.
Problème ACCEPTATION
entrée : Une machine de Turing déterministe M ; un mot w
sortie Oui si M accepte w ; non sinon

Théorème. *Le problème de l'*ACCEPTATION *est indécidable.*

Idée de la démonstration. On réduit le problème de l'ARRÊT au problème de l'ACCEPTATION : on construit une machine de Turing qui accepte w si et seulement si $\mathcal{M}$ s'arrête sur w où $\mathcal{M}, w$ est une instance de ARRÊT. □

Application : Nous permet de montrer que le problème POST est indécidable.

Définition. Le *problème de* POST sur une famille de tuile.
Problème POST
entrée : Un alphabet fini, Σ et une famille finie de couples de mots $((h_i, bi)_{i=1\ldots n})$ sur Σ
sortie Oui s'il existe $i_1, \ldots i_k \in \{1, \ldots n\}$ tels que $h_{i_1} \ldots h_{i_k} = b_{i_1} \ldots b_{i_k}$; non sinon

Théorème. *Le problème de* POST *est indécidable.*

Idée de la démonstration. On réduit le problème de POST MARQUÉ au problème de POST où POST MARQUÉ est un problème analogue à POST mais dont la première tuile est définie. On montre l'indécidabilité de POST MARQUÉ par réduction du problème de l'ACCEPTATION dans POST MARQUÉ. □

Application : En utilisant le même principe de réduction, on montre par exemple que le problème de validité de la logique du premier ordre est indécidable.

Références

[1] O. Carton. *Langages formels, calculabilité et complexité*. Vuibert, 2008.

[2] M. Huth and M. Ryan. *Logic in Computer Science : Modelling and Reasoning About Systems*. Cambridge University Press, New York, NY, USA, 2004.

[3] P. Wolper. *Introduction à la calculabilité*. Dunod, 2006.