Séminaire du DEG **spécial** du temps des fêtes

Turing pour tous

Xavier Provençal

11 décembre 2025

Défi : Présenter le théorème de Turing *et une preuve* compréhensible pour tous, en moins de 15 minutes !

L'indécidabilité du problème de l'arrêt

Définition : Programme

Un **programme** est un suite de caractères respectant une *syntaxe* définie par un *langage de programmation* et qui peut être exécutée par un ordinateur.

L'indécidabilité du problème de l'arrêt

Définition : Programme

Un **programme** est un suite de caractères respectant une *syntaxe* définie par un *langage de programmation* et qui peut être exécutée par un ordinateur.

- Il reçoit des données en **entrée**.

L'indécidabilité du problème de l'arrêt

Définition : Programme

Un **programme** est un suite de caractères respectant une *syntaxe* définie par un *langage de programmation* et qui peut être exécutée par un ordinateur.

- Il reçoit des données en **entrée**.
- À l'exécution, trois cas sont possibles :

L'indécidabilité du problème de l'arrêt

Définition : Programme

Un **programme** est un suite de caractères respectant une *syntaxe* définie par un *langage de programmation* et qui peut être exécutée par un ordinateur.

- Il reçoit des données en **entrée**.
- À l'exécution, trois cas sont possibles :
  1. Le programme **accepte** l'entrée : produit **1** en sortie et s'arrête.
  2. Le programme **refuse** l'entrée : produit **0** en sortie et s'arrête.

L'indécidabilité du problème de l'arrêt

Définition : Programme

Un **programme** est un suite de caractères respectant une *syntaxe* définie par un *langage de programmation* et qui peut être exécutée par un ordinateur.

- Il reçoit des données en **entrée**.
- À l'exécution, trois cas sont possibles :
  1. Le programme **accepte** l'entrée : produit **1** en sortie et s'arrête.
  2. Le programme **refuse** l'entrée : produit **0** en sortie et s'arrête.
  3. Le programme **ne termine pas** : il boucle indéfiniment.

L'indécidabilité du problème de l'arrêt

Définition : Programme

Un **programme** est un suite de caractères respectant une *syntaxe* définie par un *langage de programmation* et qui peut être exécutée par un ordinateur.

- Il reçoit des données en **entrée**.
- À l'exécution, trois cas sont possibles :
  1. Le programme **accepte** l'entrée : produit **1** en sortie et s'arrête.
  2. Le programme **refuse** l'entrée : produit **0** en sortie et s'arrête.
  3. Le programme **ne termine pas** : il boucle indéfiniment.

Étant donné un problème P, existe-t-il des données d telles que l'exécute de P ne termine pas ?

L'indécidabilité du problème de l'arrêt

Définition : Programme

Un **programme** est un suite de caractères respectant une *syntaxe* définie par un *langage de programmation* et qui peut être exécutée par un ordinateur.

- Il reçoit des données en **entrée**.
- À l'exécution, trois cas sont possibles :
  1. Le programme **accepte** l'entrée : produit **1** en sortie et s'arrête.
  2. Le programme **refuse** l'entrée : produit **0** en sortie et s'arrête.
  3. Le programme **ne termine pas** : il boucle indéfiniment.

Étant donné un problème P, existe-t-il des données d telles que l'exécute de P ne termine pas ?

Définition : Le problème de l'arrêt

Étant donné un programme P et des données d, est-ce que l'exécution de P avec d en entrée termine ?

L'indécidabilité du problème de l'arrêt

Définition : Programme

Un **programme** est un suite de caractères respectant une *syntaxe* définie par un *langage de programmation* et qui peut être exécutée par un ordinateur.

- Il reçoit des données en **entrée**.
- À l'exécution, trois cas sont possibles :
  1. Le programme **accepte** l'entrée : produit **1** en sortie et s'arrête.
  2. Le programme **refuse** l'entrée : produit **0** en sortie et s'arrête.
  3. Le programme **ne termine pas** : il boucle indéfiniment.

Étant donné un problème P, existe-t-il des données d telles que l'exécute de P ne termine pas ?

Définition : Le problème de l'arrêt

Étant donné un programme P et des données d, est-ce que l'exécution de P avec d en entrée termine ?

Théorème (Turing, 1936)

Le problème de l'arrêt est *indécidable*.

Preuve en trois arguments

Argument formel	**Présentation**
1. Réduction de problème	Exemple de la pile infinie
2. Paradoxe de Russell	Le barbier menteur
3. Construction d'un programme impossible	Construction d'un programme impossible

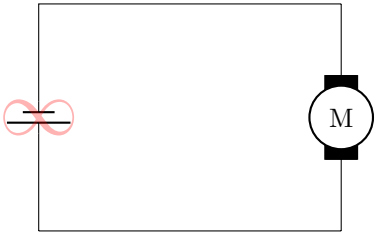
Argument 1 : Réduction de problème (la pile infinie)

Un peu de thermodynamique

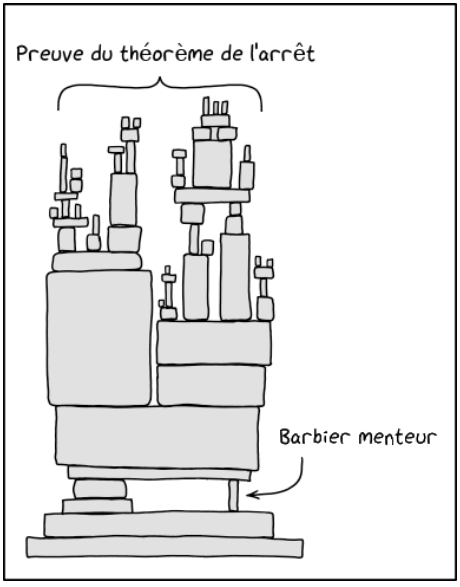
- On accepte le fait que la **machine à mouvement perpétuel** n'existe pas.

Un peu de thermodynamique

- On accepte le fait que la **machine à mouvement perpétuel** n'existe pas.
- Ceci implique que la **pile infinie** ne peut pas exister.



Argument 2 : Le paradoxe de Russell (le barbier menteur)


Preuve du théorème de l'arrêt
Barbier menteur

Le paradoxe de Russell

Le paradoxe de Russell

Le paradoxe de Russell

Le paradoxe de Russell

C'est un menteur !

Argument 3 : Construction d'un programme impossible

Un programme impossible

Soit `X` le programme qui **accepte** (retourne 1) toutes les entrées, sauf s'il s'agit du code d'un programme qui s'accepte lui-même.

Un programme impossible

Soit `X` le programme qui **accepte** (retourne 1) toutes les entrées, sauf s'il s'agit du code d'un programme qui s'accepte lui-même.

Soit d des données fournies en entrée au programme `X`,

- `X` **accepte** l'entrée d (retourne 1) si
  - d n'est pas un programme,
  - d est un programme que **ne termine pas** lorsqu'exécuté avec l'entrée d,
  - d est un programme qui **refuse** (retourne 0) l'entrée d.

- `X` **refuse** l'entrée d (retourne 0) si
  - d est un programme qui **accepte** (retourne 1) l'entrée d.

Un programme impossible

Soit `X` le programme qui **accepte** (retourne 1) toutes les entrées, sauf s'il s'agit du code d'un programme qui s'accepte lui-même.

Soit `X` des données fournies en entrée au programme `X`,

- `X` **accepte** l'entrée `X` (retourne 1) si
  - `X` n'est pas un programme,
  - `X` est un programme que **ne termine pas** lorsqu'exécuté avec l'entrée `X`,
  - `X` est un programme qui **refuse** (retourne 0) l'entrée `X`.

- `X` **refuse** l'entrée `X` (retourne 0) si
  - `X` est un programme qui **accepte** (retourne 1) l'entrée `X`.

Un programme impossible

Soit `X` le programme qui **accepte** (retourne 1) toutes les entrées, sauf s'il s'agit du code d'un programme qui s'accepte lui-même.

Soit `X` des données fournies en entrée au programme `X`,

- `X` **accepte** l'entrée `X` (retourne 1) si
  - `X` n'est pas un programme,
  - `X` est un programme que **ne termine pas** lorsqu'exécuté avec l'entrée `X`,
  - `X` est un programme qui **refuse** (retourne 0) l'entrée `X`.

- `X` **refuse** l'entrée `X` (retourne 0) si
  - `X` est un programme qui **accepte** (retourne 1) l'entrée `X`.

Conclusion : le programme `X` n'existe pas !

L'indécidabilité du problème de l'arrêt

```
1  programme X(d)
2      si d n'est pas un programme valide alors
3          retourne 1
4      si le programme d ne termine pas sur l'entrée d alors
5          retourne 1
6      Simuler l'exécution du programme d sur l'entrée d
7      si la simulation a retourné 0 alors
8          retourne 1
9      sinon
10         retourne 0
```

L'indécidabilité du problème de l'arrêt

```
1  programme X(d)
2      si d n'est pas un programme valide alors
3          retourne 1
4      si le programme d ne termine pas sur l'entrée d alors
5          retourne 1
6      Simuler l'exécution du programme d sur l'entrée d
7      si la simulation a retourné 0 alors
8          retourne 1
9      sinon
10         retourne 0
```

C'est un **interpréteur**

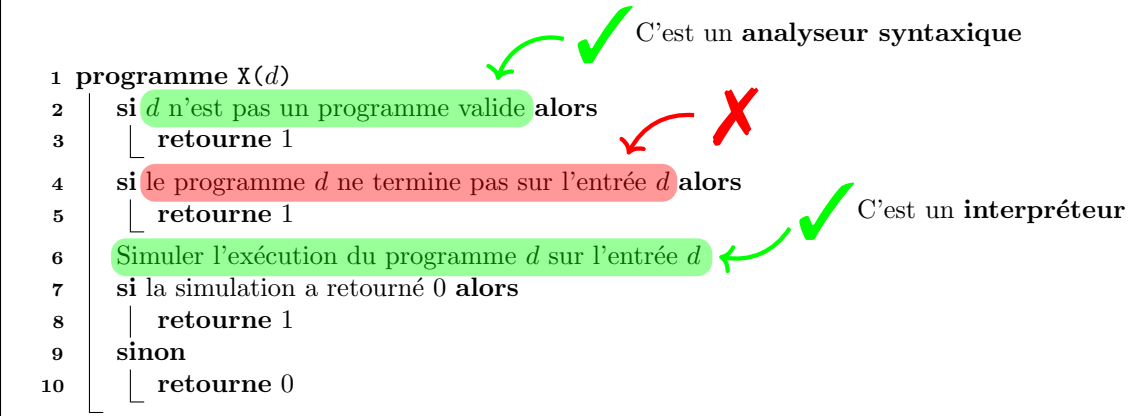
L'indécidabilité du problème de l'arrêt

C'est un **analyseur syntaxique**

```
1  programme X(d)
2      si d n'est pas un programme valide alors
3          retourne 1
4      si le programme d ne termine pas sur l'entrée d alors
5          retourne 1
6      Simuler l'exécution du programme d sur l'entrée d
7      si la simulation a retourné 0 alors
8          retourne 1
9      sinon
10         retourne 0
```

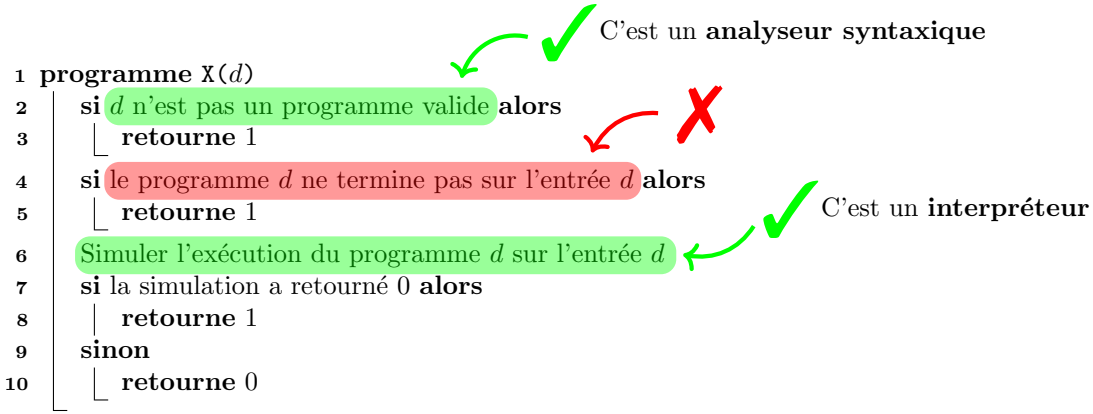
C'est un **interpréteur**

L'indécidabilité du problème de l'arrêt




```
1  programme X(d)
2      si d n'est pas un programme valide alors
3          retourne 1
4      si le programme d ne termine pas sur l'entrée d alors
5          retourne 1
6      Simuler l'exécution du programme d sur l'entrée d
7      si la simulation a retourné 0 alors
8          retourne 1
9      sinon
10         retourne 0
```

L'indécidabilité du problème de l'arrêt



Comme `X` n'existe pas, le problème de l'arrêt est indécidable.