NSI · TERMINALE / COURS

Calculabilité et décidabilité

Astrolab Cours · astrolabcours.fr

01 Le problème

Il serait formidable d'avoir un outil qui lit n'importe quel programme et annonce : « attention, il va tourner en boucle à l'infini ». On le chercherait en vain : on peut prouver qu'il n'existera jamais.

01 L'idée : le détecteur interrogé sur lui-même

Imagine un détecteur qui prédit toujours ce que tu vas faire. Tu décides de faire exactement le contraire de sa prédiction. S'il dit « tu vas t'arrêter », tu continues ; s'il dit « tu vas continuer », tu t'arrêtes. Quelle que soit sa réponse, il a tort. Un tel détecteur infaillible ne peut donc pas exister.

01 Un programme est une donnée

Un programme est un texte : on peut le donner en entrée à un autre programme. C'est ce que font un compilateur, un interpréteur Python ou un antivirus, qui lisent du code.

01 Le problème de l'arrêt

Existe-t-il une fonction `arret(prog, x)` qui, pour **tout** programme `prog` et toute entrée `x`, renvoie `True` si `prog(x)` s'arrête et `False` s'il boucle indéfiniment ?

Réponse : non. Supposons qu'elle existe, et écrivons :

```
def paradoxe(prog):
    if arret(prog, prog):
        while True:
            pass
    else:
        return
```

Que fait `paradoxe(paradoxe)` ?

- Si `arret(paradoxe, paradoxe)` vaut `True`, `paradoxe` entre dans une boucle infinie : il ne s'arrête pas. Contradiction.
- Si elle vaut `False`, `paradoxe` s'arrête aussitôt. Contradiction encore.

Dans les deux cas, c'est absurde : la fonction `arret` ne peut pas exister.

© Astrolab Cours · astrolabcours.fr/nsi/terminale/calculabilite/ · Tous droits réservés : usage personnel uniquement, republication et usage commercial interdits

Décidable, indécidable

Un problème est **décidable** s'il existe un algorithme qui répond correctement oui ou non pour toute entrée, en un temps fini. « Ce nombre est-il premier ? » est décidable. Le problème de l'arrêt est **indécidable**.

Ce qu'est un calcul

En 1936, Alan Turing décrit une machine abstraite très simple : un ruban de cases, une tête de lecture, une table de règles. La **thèse de Church-Turing** affirme que tout ce qui est calculable par un algorithme l'est par une machine de Turing. Ton ordinateur, aussi puissant soit-il, ne calcule rien de plus : il calcule plus vite.

Exemple guidé

Ce programme s'arrête-t-il pour tout entier $n \geq 1$?

```
def syracuse(n):
    while n != 1:
        if n % 2 == 0:
            n = n // 2
        else:
            n = 3 * n + 1
```

1. **Pour** $n = 6$: 6, 3, 10, 5, 16, 8, 4, 2, 1. Il s'arrête après 8 étapes.
2. **Pour tous les entiers testés**, jusqu'à des milliards de milliards, il s'arrête aussi.
3. **Mais personne n'a démontré** qu'il s'arrête toujours : c'est la conjecture de Syracuse, ouverte depuis les années 1930.

Ce n'est pas une preuve d'indécidabilité : pour ce programme précis, la réponse existe peut-être. Mais l'exemple montre qu'aucune méthode simple ne permet de lire un programme de quatre lignes et d'en déduire s'il termine.

Retour au problème

On peut prouver l'arrêt de certains programmes, un par un, mais aucun outil ne le fait pour tous. Les analyseurs de code existent quand même : ils signalent des boucles suspectes, avec des fausses alertes ou des oublis.

Les pièges

- Indécidable ne signifie pas difficile ou lent : c'est impossible en général, par principe.
- On peut savoir si **certains** programmes s'arrêtent ; c'est pour **tous** les programmes que c'est impossible.
- Un problème indécidable n'a rien à voir avec un problème dont la réponse est inconnue aujourd'hui.

© Astrolab Cours · astrolabcours.fr/nsi/terminale/calculabilite/ · Tous droits réservés : usage personnel uniquement, republication et usage commercial interdits