19.2 Décidabilité

NSI terminale - JB Duthoit

19.2.1 Décidable et indécidable

✔ Dans le cadre de la spécialité NSI, nous utilisons le langage Python, même si nous savons qu'il existe bien d'autres langages de programmation, comme JavaScript, C, Perl, Java, Fortran, etc.
Un même algorithme peut être traduit en un programme dans l'un ou l'autre des langages de notre choix.
La thèse de **Church** postule que tous ces modèles de calcul sont équivalents : une fonction calculable pour un modèle l'est pour un autre. Cela nous permet d'utiliser le modèle des fonctions programmables en Python sans perdre de généralité

Définition

Une propriété mathématique est dite **décidable** s'il existe un algorithme, un procédure mécanique qui détermine en un nombre fini d'étapes, si elle est vraie ou fausse.
S'il n'existe pas de tels algorithmes, le problème est dit indécidable.

Exercice 19.1
Construire une fonction python `est_pair` qui prend en argument un entier positif x et qui renvoie `True` si le nombre est pair, `False` sinon.

Exercice 19.2
Construire une fonction python `est_premier` qui prend en argument un entier positif x et qui renvoie `True` si le nombre est premier, `False` sinon.

Exemples

- L'ensemble des nombres premiers est décidable.
  ☛ Soit x un entier. Est-il possible de savoir en un nombre fini d'étapes si x appartient à l'ensemble des nombres premiers ?
  La réponse sera soit 'vrai', soit 'faux' et un algorithme simple est de diviser ce nombre par les entiers inférieurs à lui même. Il y a donc un nombre fini d'étapes et une réponse qui est soit vrai soit faux.

- On peut construire un algorithme pour décider si un entier naturel est pair ou non (on fait la division par deux, si le reste est zéro, le nombre est pair, si le reste est un, il ne l'est pas), donc l'ensemble des entiers naturels pairs est décidable

19.2.2 Un exemple important de l'indécidabilité : le problème de l'arrêt

Le "cauchemar" d'un programmeur est que son programme, dans certains cas, "tombe" dans une boucle infinie et ne s'arrête jamais. Dans ce cas, le logiciel est incapable de fournir la réponse attendue par l'utilisateur.

Un programme qui permettrait de tester si un autre programme va finir par s'arrêter, quel que soit le cas traité, serait d'une grande aide pour tous les développeurs du monde !

Existe-t-il un programme capable de répondre à la question "ce programme va-t-il s'arrêter" ?

Pourtant, depuis 1937 et les travaux d'Alonso Church et d'Alan Turing, on sait qu'un tel programme ne peut pas exister.

☞ Le problème de l'arrêt est un problème classique en décidabilité. Il fait partie des problème qui n'ont pas de solution : il est impossible d'écrire un programme qui résout ce problème.

19.2.3 Preuve de l'indécidabilité de l'arrêt

Un peu de logique pour commencer !

Amusons nous avec un peu de logique. Si je vous dis "je mens"...vous avez un soucis :

- Si je suis effectivement un menteur, alors l'affirmation "je mens" est fausse...donc je ne mens pas ?
- Si je ne suis pas un menteur, alors je mens en disant que je suis un menteur, je suis donc un menteur ?

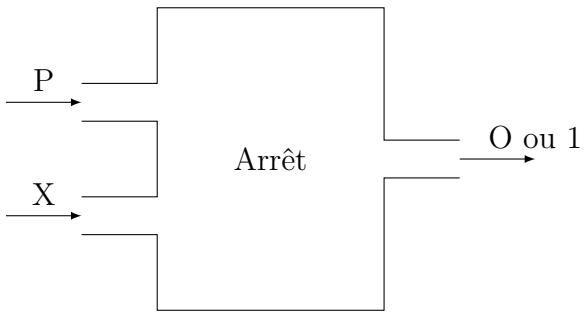
Dans les deux cas...l'affirmation est absurde, affirmer "je suis un menteur", n'est pas une assertion logique.

Une démonstration

On suppose l'existence d'un programme, appelé "arrêt" , ayant deux entrées :

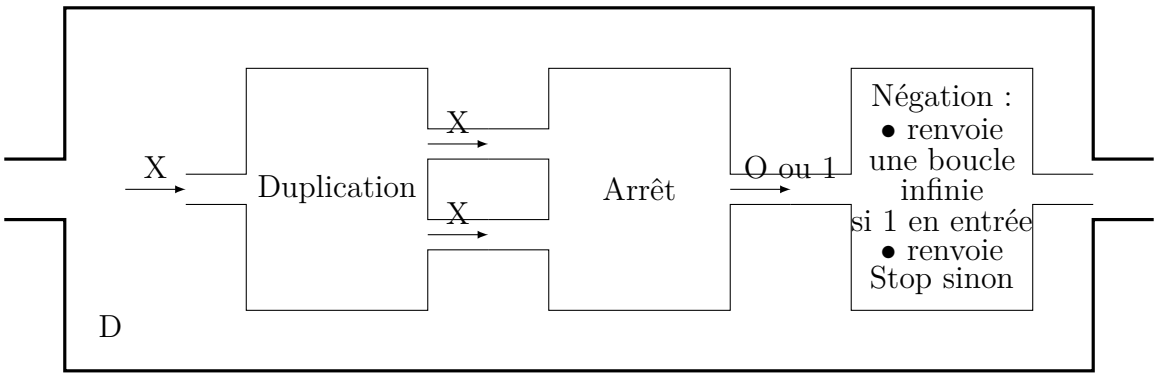
- La première est un programme P
- La seconde une donnée X

Ce programme détermine sans jamais se tromper si l'exécution du programme P se termine ou non ; il renvoie 0 si le programme s'arrête et 1 s'il ne s'arrête pas .

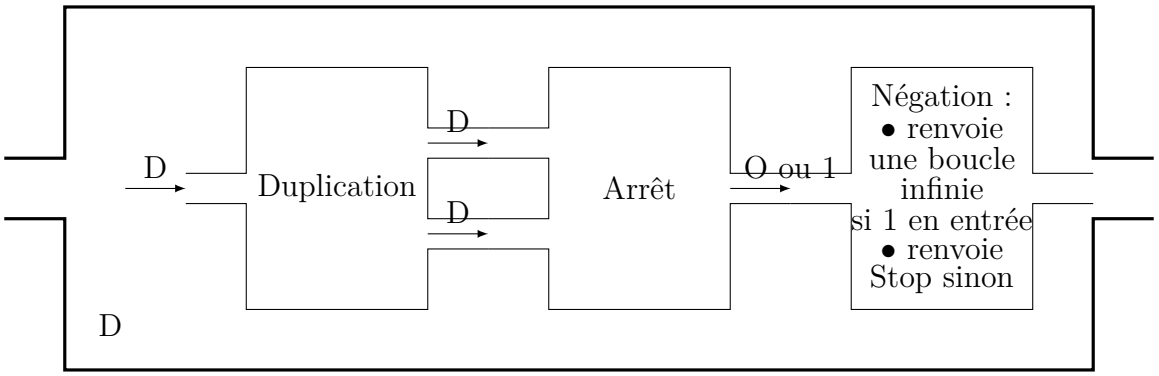


Nous construisons ensuite le programme, appelé D, qui est composé de 3 programmes successifs, comme le montre la figure suivante :

Le programme Négation a pour effet de tomber dans une boucle infinie si le programme en entrée se termine en un nombre fini d'étapes, et de renvoyer un programme qui se termine dans le cas contraire.

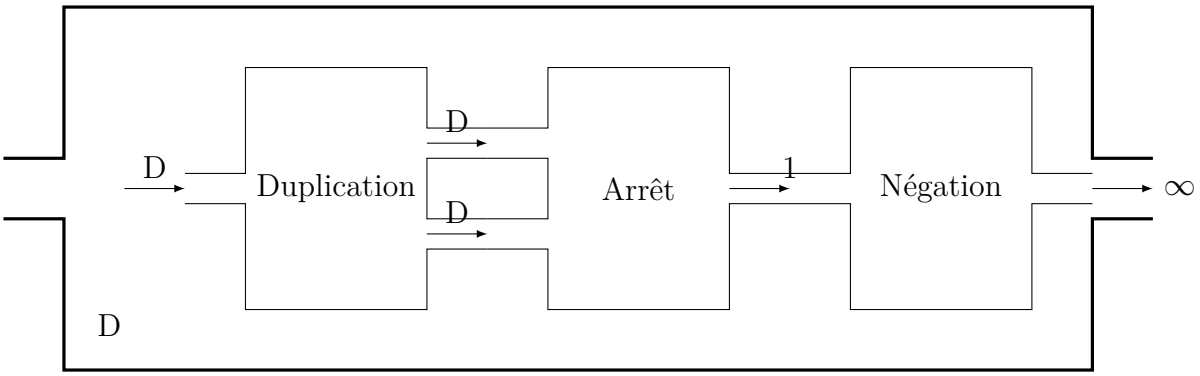


Plaçons maintenant D en entrée ! On obtient :



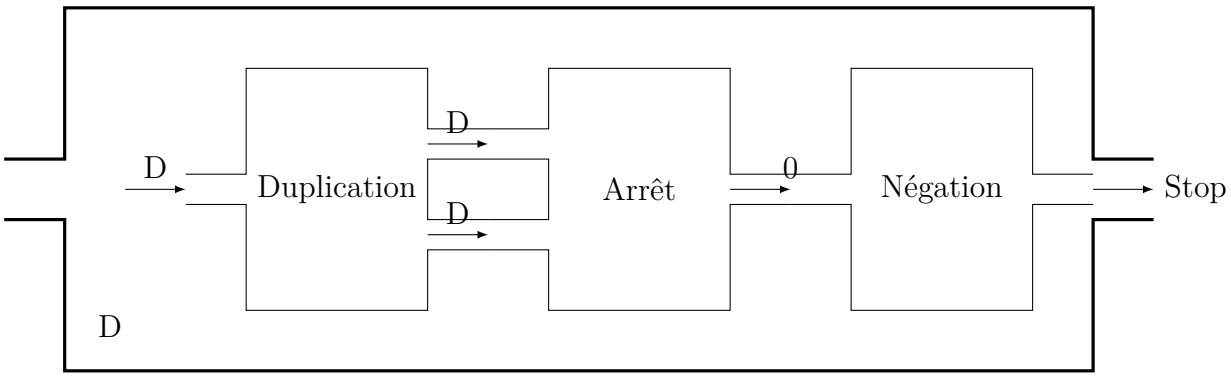
Deux cas peuvent se présenter :

- Supposons que le programme D s'arrête en un nombre fini d'étapes. La fonction "Arrêt" va donc retourner le nombre 1.



  Mais le programme D renvoie donc le nombre 0, ce qui signifie que l'on a une boucle infinie ; ce qui signifie que D s'arrête.

- Supposons que le programme D ne s'arrête pas en un nombre fini d'étapes. La fonction "Arrêt" va donc retourner le nombre 0.



Mais le programme D renvoie donc le nombre 1, ce qui signifie que D s'arrête.

Dans les deux cas, on arrive à une contradiction, ce qui montre l'impossibilité de l'existence d'un programme "Arrêt".

Il n'existe donc pas de programme qui détermine de façon générale si un programme donné se termine sur une entrée donnée.

19.2.4 Problème de l'arrêt reformulé pour des fonctions Python

Soit `arret(f,e)` une fonction Python prenant en paramètre une fonction Python `f` et une entrée `e` pour cette fonction `f`. Cette fonction `arret(f,e)` renvoie `True` si l'exécution de `f` sur `e` finit, et `False` sinon.

Considérons que cette fonction `arret(f,e)` existe et renvoie en un temps fini soit `True`, soit `False`, selon que l'exécution de `f` sur `e` s'arrête ou pas.

Considérons la fonction `paradoxe(n)` suivante :

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

Listing 19.1 – paradoxe

Deux cas se présentent :

- l