*Documents et calculatrices non autorisés* Jeudi 12 mai 2022
*Durée totale de l'épreuve (calculabilité et complexité) : 2h*

# Examen de Calculabilité

Master 1 Informatique

## Échauffement (6 points)

**1.** Le langage des nombres premiers codés en binaire est-il récursif ? récursivement énumérable ? Justifier.

**2.** Le langage des grilles de Sudoku classique, de taille $9 \times 9$, qui possèdent une solution est-il récursivement énumérable ? récursif ? Justifier.

**3⋆.** Existe-t-il un programme qui à partir d'un programme fourni en entrée retourne le programme le plus court qui calcule la même fonction ? Justifier.

## Réductions (6 points)

INFINI est le langage des machines de Turing qui s'arrêtent sur un nombre infini d'entrées. $\overline{\text{INFINI}}$ est le complémentaire de INFINI. ARRET est le problème de l'arrêt vu en cours.

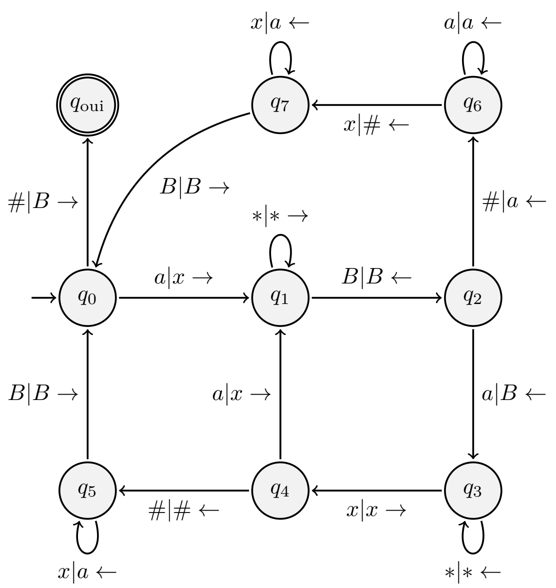
**4.** Montrer que ARRET $\leqslant_m$ INFINI.

**5.** Montrer que ARRET $\leqslant_m$ $\overline{\text{INFINI}}$.

**6.** Que déduire des deux questions précédentes ?

## Analyse de machine de Turing (6 points)

La machine de Turing ci-dessous travaille sur l'alphabet $\{B, a, x, \#\}$ où $B$ est le blanc. Elle démarre avec un ruban partout blanc sauf un mot constitué d'un certain nombre de lettres $a$ séparées en deux morceaux par la lettre $\#$. La tête est positionnée sur la première lettre du mot, dans l'état $q_0$. La machine s'arrête dans l'état $q_{\text{oui}}$ pour accepter un mot. Lorsqu'aucune transition n'est définie, elle rejette le mot. Une transition $*|*$ s'applique quand aucune autre transition n'est définie pour cet état.




**7.** Simuler l'exécution de la machine sur les entrées $aaaa\#aa$ et $aaa\#aaaaa$.

**8⋆.** Caractériser la fonction calculée par la machine.

*Documents et calculatrices non autorisés* Jeudi 12 mai 2022
*Durée totale de l'épreuve (calculabilité et complexité) : 2h*

# Examen de Complexité

Master 1 Informatique

## Échauffement (6 points)

**1.** Rappeler les définitions des classes de complexité P, NP, coNP et préciser les inclusions.

**2.** Rappeler la définition du problème SAT. Soit $\varphi(x_1, \ldots, x_n)$ une instance de SAT. On note $\varphi_0(x_2, \ldots, x_n) = \varphi(0, x_2, \ldots, x_n)$ et $\varphi_1(x_2, \ldots, x_n) = \varphi(1, x_2, \ldots, x_n)$ les formules à $n-1$ variables obtenues en fixant la valeur de la première variable de $\varphi$. Montrer que $\varphi \in$ SAT si et seulement si $\varphi_0 \in$ SAT ou $\varphi_1 \in$ SAT.

**3.** Un langage $L$ est dans la classe DP s'il existe deux langages $L_1 \in$ NP et $L_2 \in$ coNP tels que $L = L_1 \cap L_2$. Montrer que si NP = coNP alors DP = NP.

## Autour de SAT (6 points)

**4.** DNF-SAT est la variante du problème SAT dans laquelle une formule est donnée en forme normale disjonctive (inverser ET et OU par rapport à la forme normale conjonctive). Montrer que DNF-SAT est dans P.

**5.** Expliquer pourquoi la *«démonstration»* suivante de P $\neq$ NP est incorrecte : *« Pour vérifier qu'une instance de SAT est satisfiable, il faut tester $2^n$ affectations possibles des variables, ce qui nécessite un temps de calcul exponentiel en la taille de l'entrée donc SAT n'appartient pas à la classe P et comme SAT appartient à la classe NP, nécessairement* P $\neq$ NP*. »*

**6$^\star$.** Expliquer pourquoi l'algorithme suivant pour résoudre SAT ne démontre pas que P = NP : *« Pour vérifier qu'une instance $\varphi$ de SAT à $n$ variables est satisfiable, on construit sucessivement des ensembles de formules booléennes $\Phi_i$ en partant de $\Phi_0 = \{\varphi\}$. À chaque étape, on place dans $\Phi_{i+1}$ les deux formules $\psi_0$ et $\psi_1$ fixant la première variable pour chaque formule $\psi$ de $\Phi_i$. À la fin, il suffit de tester si $\Phi_n$ contient la formule vraie. »*

## NP-complétude (6 points)

Le jeu d'aventure *l'amulette de Yendor* a pour but d'amener un héros (représenté par le caractère `'@'`) jusqu'à la sortie (`'>'`) à travers un labyrinthe, une grille de cases, sur laquelle le héros se déplace d'une case à la fois dans l'une des quatre directions possibles (haut, bas, gauche, droite). Le labyrinthe comporte des murs infranchissables (`'#'`) et des cases de sol (`'.'`) instables sur lesquelles il peut marcher mais qui s'effondrent lorsqu'il quitte la case (elles deviennent ensuite un trou infranchissable, assimilable à un mur). Le héros peut ramasser différentes clés (`'a'`, `'b'`, ...) dans le labyrinthe qui lui permettent de franchir des portes (`'A'`, `'B'`, ...) qui lui barrent le chemin : franchir une porte nécessite d'avoir ramassé auparavant la clé correspondante. Plusieurs portes peuvent correspondre à une même clé. Un labyrinthe typique est représenté ci-dessous. Le problème de décision YENDOR consiste à décider si un labyrinthe donné en entrée possède une solution.

```
  ###
  #>#
  #.#
###A###
#.....#
#.###.#
#a# #b#
#.###.#
#..@..#
#######
```

- `'#'` : mur infranchissable
- `'.'` : sol instable
- `'>'` : sortie à atteindre
- `'@'` : le héros
- `'a'`, `'b'`, ... : clés distinctes
- `'A'`, `'B'`, ... : portes associées

**7.** Expliquer comment résoudre le puzzle ci-dessus.

**8.** Montrer que YENDOR est dans NP.

**9.** Montrer que YENDOR est NP-complet en associant un labyrinthe à toute instance de 3-SAT. On pourra s'inspirer de l'exemple ci-dessus pour trouver la réduction.