Tutorial 1 : Decision problems and Turing machines

Computational complexity theory, 5th semester.

2022

Exercice 1 — *Describe a decision problem*

Describe as a decision problem the following problems. Give the set $\mathcal{L}$ of instances and the question that should be answered.

1. (BIP ?) : Determine if a is graph bipartite.

▶ **Correction**

Version en français : Soit G un graphe non orienté, G est-il un graphe biparti ?
$\mathcal{L} = \{\text{Graphes } G \text{ non orientés }\}$, $\mathcal{L}^Y = \{\text{Graphes } G \text{ non orientés bipartis}\}$

2. (SAT ?) : Determine how to make a boolean formula true.

▶ **Correction**

Version en français : Soit une formule φ avec n variables $x_1, x_2, \ldots, x_n$ existe-t-il une affectation des variables x_i qui rende φ vrai ? (autrement dit, $\exists x_1, x_2, \ldots, x_n | \varphi(x_1, x_2, \ldots, x_n)$, ou φ est satisfiable).
$\mathcal{L} = \{$ Formule booléenne φ avec n variables $x_1, x_2, \ldots, x_n$ $\}$, $\mathcal{L}^Y = \{$ Formule booléenne φ avec n variables $x_1, x_2, \ldots, x_n$ de sorte qu'on puisse rendre la fonction φ vraie pour au moins une affectation des variables $\}$

3. (3SAT ?) : SAT restricted to conjonctive normal forms where each clause has 3 variables.

▶ **Correction**

Version en français : Soit une formule φ sous forme CNF avec 3 variables par clause, avec n variables $x_1, x_2, \ldots, x_n$, existe-t-il une affectation des variables x_i qui rende φ vrai ?
$\mathcal{L} = \{$ Formule booléenne φ sous forme CNF avec 3 variables par clause, avec n variables $x_1, x_2, \ldots, x_n$ $\}$, $\mathcal{L}^Y = \{$ Formule booléenne φ avec n variables $x_1, x_2, \ldots, x_n$ de sorte qu'on puisse rendre la fonction φ vraie pour au moins une affectation des variables $\}$

4. (ADD ?) : Add two integers.

▶ **Correction**

Version en français : Soient trois entiers x, y et z, est-ce que $x + y = z$?
$\mathcal{L} = \{x, y, z \in \mathbb{N}\}$, $\mathcal{L}^Y = \{x, y, z \in \mathbb{N} \text{ tels que } x + y = z\}$

5. (TAU ?) : Determine if a boolean formula is always true.

▶ **Correction**

Version en français : Soit une formule φ avec n variables $x_1, x_2, \ldots, x_n$ est-ce que, pour toute affectation des variables x_i, φ est vraie ? (autrement dit, $\forall x_1, x_2, \ldots, x_n | \varphi(x_1, x_2, \ldots, x_n)$ ou φ est une tautologie).
$\mathcal{L} = \{$ Formule booléenne φ avec n variables $x_1, x_2, \ldots, x_n$ $\}$, $\mathcal{L}^Y = \{$ Formule booléenne φ avec n variables $x_1, x_2, \ldots, x_n$ de sorte que φ soit toujours vraie quelle que soit l'affectation des variables $\}$

6. (3-TAU ?) : TAU restricted to disjonctive normal forms where each clause has 3 variables.

▶ **Correction**

Version en français : Soit une formule φ sous forme DNF avec 3 varialbes par clause, avec n variables $x_1, x_2, \ldots, x_n$ est-ce que, pour toute affectation des variables x_i, φ est vraie ? (autrement dit, φ est une tautologie).

$\mathcal{L} = \{$ Formule booléenne φ sous forme DNF avec 3 variables par clause, avec n variables $x_1, x_2, \ldots, x_n$ $\}$, $\mathcal{L}^Y = \{$ Formule booléenne φ avec n variables $x_1, x_2, \ldots, x_n$ de sorte que φ soit toujours vraie quelle que soit l'affectation des variables $\}$

7. (QBF ?) : Determine if a quantified boolean formula is true.

▶ **Correction**

Version en français : Soit une formule φ avec n variables $x_1, x_2, \ldots, x_n$ est-ce que $\exists x_1 \forall x_2 \exists \ldots \forall x_n | \varphi(x_1, x_2, \ldots, x_n)$?

$\mathcal{L} = \{$ Formule booléenne φ avec n variables $x_1, x_2, \ldots, x_n$ $\}$, $\mathcal{L}^Y = \{$ Formule booléenne φ avec n variables $x_1, x_2, \ldots, x_n$ de sorte que $\exists x_1 \forall x_2 \exists \ldots \forall x_n | \varphi(x_1, x_2, \ldots, x_n)$ $\}$

8. (CONNECTIVITY ?) : Connectivity of a graph.

▶ **Correction**

Version en français : Soit G un graphe non orienté, G est-il un graphe connexe ?

$\mathcal{L} = \{$Graphes G non orientés $\}$, $\mathcal{L}^Y = \{$Graphes G non orientés connexes$\}$

9. (PLANARITY ?) : Planarity of a graph.

▶ **Correction**

Version en français : Soit G un graphe non orienté, G est-il un graphe planaire ?

$\mathcal{L} = \{$Graphes G non orientés $\}$, $\mathcal{L}^Y = \{$Graphes G non orientés planaires$\}$

10. (k-COL ?) : k-colorability of a graph.

▶ **Correction**

Version en français : Soit G un graphe non orienté, G est-il un graphe k-coloriable ? Autrement dit, peut-on attribuer à chaque nœud une couleur parmi k de sortes que deux voisins n'ont pas la même couleur ?

$\mathcal{L} = \{$Graphes G non orientés $\}$, $\mathcal{L}^Y = \{$Graphes G non orientés k-coloriable $\}$

11. (CHROMA) : chromatic number of a graph.

▶ **Correction**

Version en français : Soit G un graphe non orienté et un entier k, G est-il k-coloriable ?

$\mathcal{L} = \{$Graphe G non orienté et un entier k $\}$, $\mathcal{L}^Y = \{$ Graphe G non orienté et un entier k de sorte que G est k-coloriable$\}$

Notez la différence avec le problème précédent où k n'est pas donné en entrée. Clairement on résout k-COL et CHROMA de la même manière. Mais, dans le premier cas, k est considéré comme une constante. On a le problème (3-COL) (4-COL) (27-COL). Si on a un algo en n^k il serait polynomial pour (4-COL) car on aurait alors un algo en n^4. Mais il ne serait pas polynomial pour (CHROMA) où k fait partie de l'entrée et où donc n^k serait exponentiel en k.

12. (SIZE) : number of nodes in a graph.

▶ **Correction**

Version en français : Soit G un graphe non orienté et un entier k, G est-il de taille k ?

$\mathcal{L} = \{$Graphe G non orienté et un entier k $\}$, $\mathcal{L}^Y = \{$ Graphe G non orienté et un entier k de sorte que G est de taille $k\}$

13. (INS) : Size of a maximum independant set in a graph.

▶ **Correction**

Version en français : Soit G un graphe non orienté et un entier k, G possède-t-il un stable de taille k ? (k nœuds non reliés deux à deux).
$\mathcal{L} = \{$Graphe G non orienté et un entier k $\}$, $\mathcal{L}^Y = \{$ Graphe G non orienté et un entier k de sorte que G possède un stable de taille $k\}$

14. (WINS) : weight of a maximum weighted independant set in a graph.

▶ **Correction**

Version en français : Soit G un graphe non orienté, des poids $\omega : V \to \mathbb{N}$ et un entier k, G possède-t-il un stable dont la somme des poids des nœuds dépasse k ?
$\mathcal{L} = \{$Graphe G non orienté, des poids ω sur les nœuds de G et un entier k $\}$, $\mathcal{L}^Y = \{$ Graphe G non orienté, des poids ω sur les nœuds de G et un entier k de sorte que G possède un stable de poids supérieur à k $\}$

15. (MATCHING) : Find a maximum matching in a graph.

▶ **Correction**

Version en français : Soit G un graphe non orienté et un entier k, G possède-t-il un couplage de taille supérieure à k ? (c'est à dire des arêtes sans extrémité commune)
$\mathcal{L} = \{$Graphe G non orienté et un entier k $\}$, $\mathcal{L}^Y = \{$ Graphe G non orienté et un entier k de sorte que G possède un couplage de taille supérieure à k $\}$

16. (VERTEX COVER) : Find a set of nodes of minimum size covering all the edges of a graph.

▶ **Correction**

Version en français : Soit G un graphe non orienté et un entier k, G possède-t-il un un ensemble V' d'au plus k nœuds couvrant toutes les arêtes ? (c'est à dire que chaque arête a une extrémité dans V')
$\mathcal{L} = \{$Graphe G non orienté et un entier k $\}$, $\mathcal{L}^Y = \{$ Graphe G non orienté et un entier k de sorte qu'il existe un ensemble V' de nœuds de taille inférieur à k couvrant toutes les arêtes $\}$

17. (HAM ?) : determine if a graph is hamiltonian.

▶ **Correction**

Version en français : Soit G un graphe non orienté, G est-il un graphe hamiltonien ? (possédant un cycle passant une et une fois par chaque nœud ; variante : une chaîne passant une et une fois par chaque nœud ; variante : G est orienté et un circuit/chemin passant une et une fois par chaque nœud)
$\mathcal{L} = \{$Graphes G non orientés $\}$, $\mathcal{L}^Y = \{$Graphes G non orientés hamiltoniens $\}$

18. (COHAM ?) : determine if a graph is not hamiltonian.

▶ **Correction**

Version en français : Soit G un graphe non orienté, est-ce que G n'est pas hamiltonien ?
$\mathcal{L} = \{$Graphes G non orientés $\}$, $\mathcal{L}^Y = \{$Graphes G non orientés non hamiltoniens $\}$
(Ce problème inverse les réponses de (HAM))

19. (SHP) : Find a shortest elementary path between two nodes, with non-negative weights.

▶ **Correction**

Version en français : Soit G un graphe orienté, des poids $\omega : E \to \mathbb{N}$, un nœud s et un nœud t et un entier k, G possède-t-il un chemin élémentaire (ne passant qu'une fois au plus par chaque nœud) de s à t dont la somme des poids des arcs est inférieure à k ?

20. (LOP) : Find a longest elementary path between two nodes, with non-negative weights.

▶ **Correction**

Version en français : Soit G un graphe orienté, des poids $\omega : E \to \mathbb{N}$, un nœud s et un nœud t et un entier k, G possède-t-il un chemin élémentaire (ne passant qu'une fois au plus par chaque nœud) de s à t dont la somme des poids des arcs dépasse k ?

21. (TSP) : In a complete, weighted and undirected graph, find a minimum cost cycle going through each node.

▶ **Correction**

Version en français : Soit G un graphe non orienté complet, des poids $\omega : E \to \mathbb{N}$, et un entier k, G possède-t-il un cycle hamiltonien dont la somme des poids des arêtes dépasse k ?

22. (MSPT) : Find a minimum spanning tree in a graph.

▶ **Correction**

Version en français : Soit G un graphe non orienté, des poids $\omega : E \to \mathbb{N}$ et un entier k, G possède-t-il un arbre couvrant (un sous-graphe de G sans cycle passant par chaque nœud de G) dont la somme des poids des arêtes est inférieure à k ?

23. (SET COVER) : Using sets of a subset of $\mathcal{P}(X)$, find a minimum cover of a set X.

▶ **Correction**

Version en français : Soit un ensemble X, un ensemble S de parties de X et un entier k, existe-t-il un sous-ensemble C de S couvrant X (chaque élément de X est dans au moins un ensemble de C) et de taille inférieure à k ?

24. (UST) : Find a tree spanning a subset of nodes in a graph and of minimum cost.

▶ **Correction**

Version en français : Soit G un graphe non orienté, des poids $\omega : E \to \mathbb{N}$, un ensemble X de nœuds de G et un entier k, G possède-t-il un arbre couvrant X (un sous-graphe de G sans cycle passant par chaque nœud de X et possiblement d'autres nœuds) dont la somme des poids des arêtes est inférieure à k ?

25. (DST) : Find a directed tree spanning a subset of nodes in a directed graph rooted at a specific node and of minimum cost.

▶ **Correction**

Version en français : Soit G un graphe orienté, un nœud r, des poids $\omega : E \to \mathbb{N}$, un ensemble X de nœuds de G et un entier k, G possède-t-il une arborescence enracinée en r couvrant X (un sous-graphe de G sans cycle possédant un chemin de r vers chaque nœud de X et possiblement d'autres nœuds) dont la somme des poids des arcs est inférieure à k ?

26. (GRUNDY ?) : determine if a graph has a grundy function.

▶ **Correction**

Version en français : Soit G un graphe orienté, G a-t-il une fonction de Grundy ? (c'est-à-dire que chaque nœud v a un numéro g_v de sorte que g_v est le plus petit entier positif ou nul qui n'est pas déjà attribué aux successeurs de v).

27. (SUBSET SUM ?) : Determine if a value can be obtained by summing some of the integers of a set.

▶ **Correction**

Version en français : Soit $x_1, x_2, \ldots, x_n$ et B des entiers, est-ce qu'il existe un sous-ensemble des entiers de $x_1, x_2, \ldots, x_n$ dont la somme fait B ?

28. (PARTITION ?) : Determine if a set of integers can be parted into two sets with same sum.

▶ **Correction**

Version en français : Soit $x_1, x_2, \ldots, x_n$ des entiers, est-ce qu'il existe une partition X_1, X_2 de ces entiers telle que $\sum_{x \in X_1} x = \sum_{x \in X_2} x$?

29. (KNAPSACK) : The knapsack problem.

▶ **Correction**

Version en français : Soit n objets de volume $v_1, v_2, \ldots, v_n$ et de poids $p_1, p_2, \ldots, p_n$, soit deux entiers V et P, existe-t-il un sous-ensemble $I \subset [\![1; n]\!]$ de sorte que $\sum_{i \in I} v_i \leq V$ et $\sum_{i \in I} p_i \geq P$? autrement dit un sous-ensemble des objets dont le volume ne dépasse pas V mais dont le poids dépasse P ?

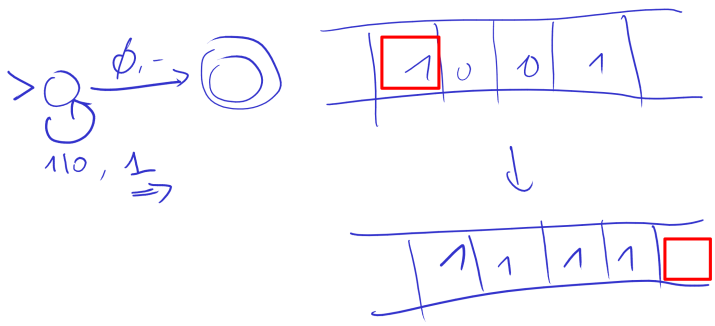
30. (LP) : Solve a linear program with real variables.
31. (ILP) : Solve a linear program with integer variables.
32. (FLOW) : Find a maximum flow in a network.
33. (MREG ?) : Determine if a Markov chain is regular.
34. (CHESS ?) : Determine if the white can force a win from a given position on a chess board.
35. (REVERSI ?) : Determine if the white can force a win from a given position on a reversi board.
36. (SUDOKU ?) : Solve a sudoku.
37. (U-SUDOKU ?) : Unicity of the solution of a sudoku.
38. (MIN SUDOKU) : Determine the minimum number of necessary clues to solve a sudoku.

Exercice 2 — *Turing machines*

For this exercise, assume that every result you must demonstrate in exercise 3 is true.

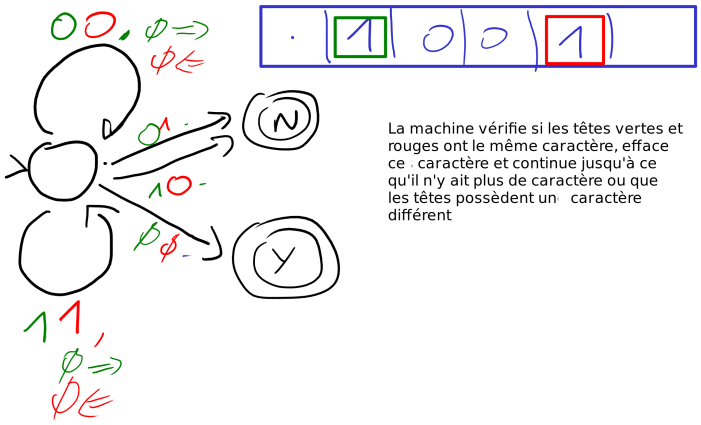
1. Describe a deterministic Turing machine that replaces the input integer by a sequence of 1.

▶ **Correction**



2. Describe a deterministic Turing machine that checks if a word is a palindrome.

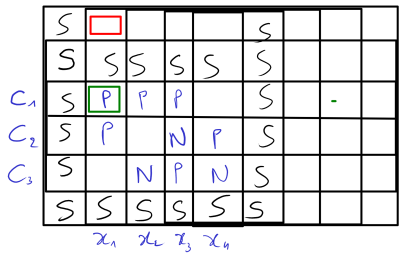
▶ **Correction**



La machine vérifie si les têtes vertes et rouges ont le même caractère, efface ce caractère et continue jusqu'à ce qu'il n'y ait plus de caractère ou que les têtes possèdent un caractère différent

3. Describe a deterministic Turing machine that solves the problem (ADD ?).
4. Describe a non-deterministic Turing machine that solves the problem (SAT ?).

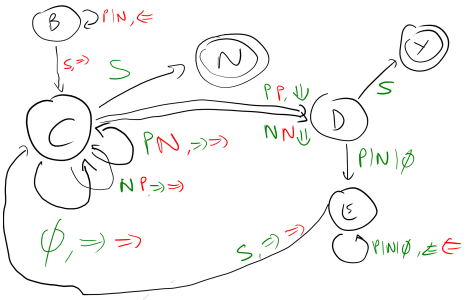
► **Correction**



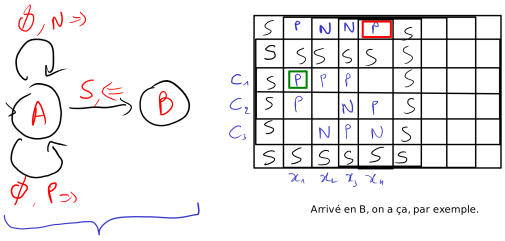
On considère ici un encodage des formules phi de la forme
phi = C1 et C2 et ... et Cm
avec Cm de la forme (a ou b ou c)
avec a, b, c de la forme x_i ; ou (non x_i)

Par exemple phi = (x1 ou x2 ou x3) et (x1 ou non x3 et x4) et (non x2 ou x3 ou non x4)

Chaque ligne correspond à une clause.
Chaque variable correspond à une colonne
(On a indiqué ici en bleu lesquels pour l'exemple)
Un P dans la ligne correspondant à C_j dans la colonne correspondant à x_i signifie que x_i apparait positivement dans C_j.
Un N signifique que (non x_i) apparait dans C_j
Un blanc signifie que ni x_i ni (non x_i) apparait dans C_j



Ici on a le vérifieur, qui vérifie que l'affectation des variables rend la formule vraie.
C parcours une clause. Si elle tombe sur une variable que le certificat a rendu VRAI et qui apparait positivement dans la clause, alors la clause est satisfaite (car c'est un OU entre des variables) et on passe à la clause suivante avec D et E.
De même si elle voit une variable que le certificat a rendu FAUX et qui apparait négativement.
D vérifie qu'on a pas atteint le bas des clauses, c'est à dire qu'on a satisfait la dernière clause.
On arrive en N si on a pas réussi à satisfaire une clause.

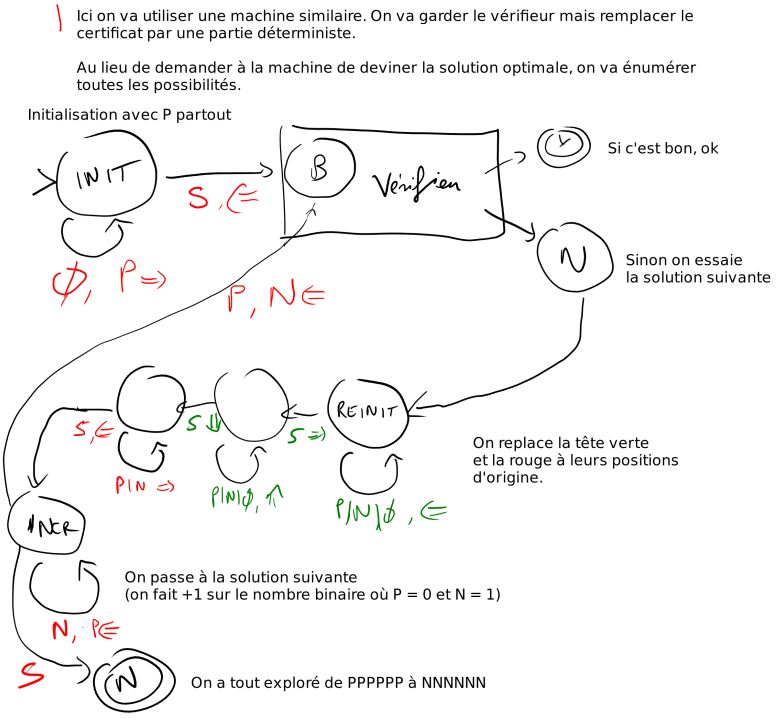


Certificat, non déterministe
Indique pour chaque variable si elle est positive ou négative

Arrivé en B, on a ça, par exemple.

5. Describe a deterministic Turing machine that solves the problem (SAT ?).

► **Correction**



6. Describe a deterministic Turing machine that solves the problem (3-COL ?).
7. Describe a non-deterministic Turing machine that solves the problem (W-INS).

Exercice 3 — *Equivalent Turing machines*

1. Let $\mathcal{M}$ be a Turing machine with an alphabet Σ, simulate $\mathcal{M}$ with a classical machine $\mathcal{M}'$.
2. Let $\mathcal{M}$ be a Turing machine, simulate $\mathcal{M}$ with a Turing machine $\mathcal{M}'$ that can only read and write 1 and 0.
3. Explain how we can encode the YES and NO answers of a decision problem by a Turing machine with no acceptation or rejection states.
4. Let $\mathcal{M}$ be a Turing machine, simulate $\mathcal{M}$ with a Turing machine $\mathcal{M}'$ that has only a half-tape (infinite on the right, finite on the left).
5. Let $\mathcal{M}$ be a Turing machine with 2 tapes, simulate $\mathcal{M}$ with a classical Turing machine $\mathcal{M}'$
6. Let $\mathcal{M}$ be a Turing machine with a 2D tape, simulate $\mathcal{M}$ with a classical Turing machine $\mathcal{M}'$.
7. Let $\mathcal{M}$ be a non-deterministic Turing machine, simulate $\mathcal{M}$ with a deterministic Turing machine $\mathcal{M}'$.