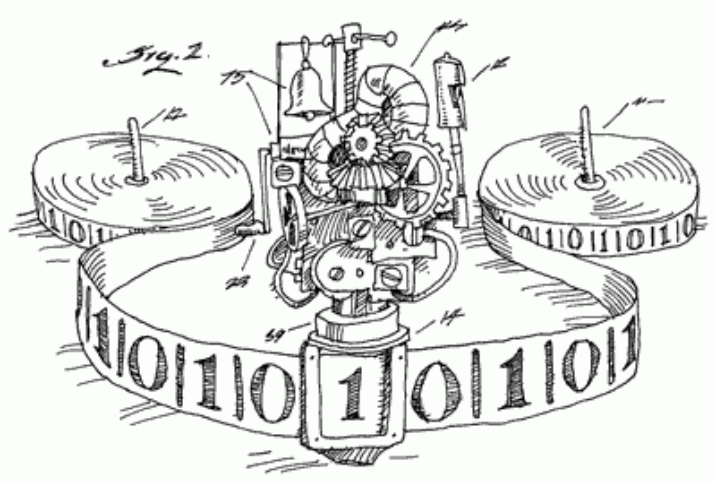
Université Cadi Ayyad
École Supérieure de Technologie - Safi
Département : Informatique
Filière : Génie Informatique

RAPPORT DE PROJET DE FIN D'ÉTUDES

Simulation d'une machine de Turing à l'aide de Python



Réalisé par :
BELCAIDA Haitam
BARKI Ayoub

Encadré par :
M. MAAROUF Hamid

Tuteur :
MLLE. ENNAIR AYA

Année Universitaire : 2022/2023

Remerciement

En préalable à ce rapport, nous souhaitons adresser tous nos remerciements aux personnes qui nous ont apporté leur aide et qui ont ainsi contribué à l'élaboration de ce rapport.

Nous tenons à remercier dans un premier temps, toute l'équipe pédagogique de l'École Supérieure de Technologie de Safi et les intervenants professionnels responsables de la formation.

Nous tenons à exprimer également notre profonde reconnaissance à notre enseignant M. **MAAROUF Hamid** qui nous a encadrés durant ce Projet de Fin d'Étude, pour l'aide et les conseils concernant les missions évoquées dans ce projet, qu'elle nous a apporté lors des différents suivis.

Nous tenons à exprimer nos sincères remerciements à notre tuteur, Mlle. **ENNAIR Aya**, pour sa précieuse contribution et son soutien tout au long de notre projet de fin d'études portant sur la création d'un simulateur de machine de Turing.

Leur expertise, leur dévouement et leur passion pour l'enseignement nous ont guidés à travers les différentes étapes de ce projet. Leur patience et leurs encouragements constants ont été précieux pour surmonter les défis techniques et conceptuels auxquels nous avons été confrontés.

Enfin, nous tenons à remercier nos proches, nos amis et tous ceux qui nous ont soutenus tout au long de ce projet. Leur encouragement constant a été une source d'inspiration pour nous.

Nous sommes conscients que ce projet n'aurait pas été possible sans l'appui indéfectible de notre encadrant et de notre tuteur, ainsi que de tous ceux qui nous ont entourés. Nous sommes profondément reconnaissants de leur confiance et de leur engagement envers notre réussite académique.

Table des matières

Table des figures

Liste des abréviations

IDE : Environnement de Développement Intégré
MT : Machine de Turing
MTD : Machine de Turing Déterministe
MTND : Machine de Turing non Déterministe
MTQ : Machines de Turing Quantiques
PFE : Projet de Fin d'Étude

Résumé

Ce rapport présente le Projet de Fin d'Études réalisé dans le cadre de la validation du Diplôme Universitaire de Technologie en Génie Informatique. L'objectif principal du projet était de concevoir et de développer un simulateur de machine de Turing en utilisant les technologies Python, CustomTkinter et GraphViz.

Le simulateur de machine de Turing est un outil fondamental en informatique théorique, permettant d'étudier les notions de calculabilité et de complexité. Le projet a exigé une analyse approfondie des concepts de la machine de Turing, ainsi que des compétences avancées en programmation et en conception logicielle.

Nous avons utilisé Python comme langage de programmation principal, en exploitant les fonctionnalités avancées de la bibliothèque CustomTkinter pour créer une interface utilisateur. De plus, GraphViz a été utilisé pour générer des visualisations graphiques des étapes de calcul de la machine de Turing.

Le rapport présente en détail les différentes phases du projet, notamment l'analyse des besoins, la conception de l'application, le développement du simulateur, les tests et les résultats obtenus. Il met également en évidence les difficultés rencontrées et les solutions mises en œuvre pour les surmonter.

Introduction Générale

Ce rapport constitue une synthèse de notre Projet de Fin d'Étude intitulé de **Simulation d'une machine de Turing à l'aide de Python**. La machine de Turing qui a joué un rôle crucial dans la contribution de la science des ordinateurs à ce qu'elle offre aujourd'hui.

En définissant un modèle abstrait de calcul, la machine de Turing a posé les bases théoriques pour l'étude des algorithmes et de la calculabilité. Elle a permis de comprendre les principes fondamentaux de la computation et a ouvert la voie à la formalisation des notions clés telles que la décidabilité et la complexité. Ces avancées ont été essentielles pour le développement ultérieur de l'informatique en tant que discipline scientifique. En influençant l'architecture des ordinateurs modernes, la machine de Turing a joué un rôle déterminant dans la conception des premiers systèmes informatiques et continue d'influencer leur évolution. Ainsi, La machine de Turing a joué un rôle essentiel dans l'édification de la science informatique de la science des ordinateurs, jetant ainsi les bases théoriques solides qui ont permis de développer le domaine tel que nous le connaissons actuellement.

En premier lieu, notre rapport se concentre sur une présentation approfondie de la machine de Turing, mettant l'accent sur son fonctionnement et ses caractéristiques essentielles. En deuxième lieu, nous examinons le lien étroit entre la science informatique et la machine de Turing, soulignant son rôle dans le développement de la théorie de la computation. Ensuite, nous nous penchons sur le fonctionnement détaillé de la machine de Turing, en analysant ses composantes et en illustrant son utilisation à travers un exemple concret. Par la suite, nous abordons la planification et la gestion du projet, mettant en évidence les différentes étapes et méthodologies mises en œuvre pour la réalisation de notre simulation de la machine de Turing. Enfin, nous présentons en détail la réalisation de notre simulateur de la machine de Turing, en mettant en évidence les choix technologiques et les résultats obtenus.

Chapitre I

La machine de Turing

1 Intoduction

La machine de Turing est un modèle abstrait de calcul qui a permis de formaliser les concepts de calculabilité et d'algorithme, et qui a influencé la conception des ordinateurs et la sécurité des communications. Ce chapitre porte sur Alan Turing, La machine de Turing et son importance dans l'histoire de l'informatique, types et variantes des machines de Turing.

2 Alan Turing : Le cerveau derrière la création de la machine de Turing

2.1 Alan Turing

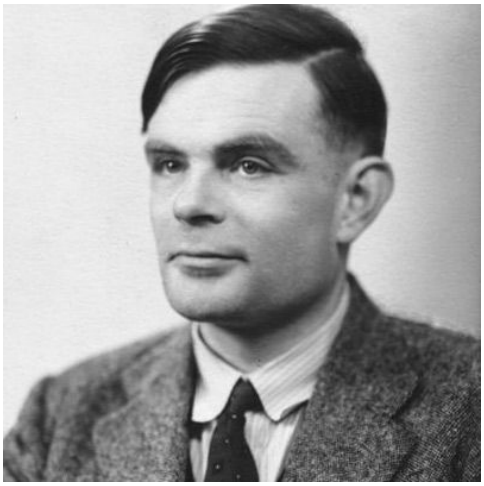


FIGURE I.1 – Alan Turing.

Alan Turing est considéré comme l'un des plus grands génies de l'informatique du XXe siècle. Il a joué un rôle essentiel dans le développement de l'ordinateur moderne, en particulier dans la cryptographie et la théorie de l'informatique. Né en 1912 en Angleterre, il était un enfant prodige et montrait une grande aptitude pour les mathématiques dès son plus jeune âge.

Turing a étudié à l'Université de Cambridge où il a obtenu un diplôme en mathématiques en 1934. En 1936, il a publié un article révolutionnaire intitulé "On Computable Numbers, with an Application to the Entscheidungsproblem". Cet article a introduit la notion de la "machine de Turing", qui est aujourd'hui considérée comme l'un des fondements de l'informatique moderne.

Pendant la Seconde Guerre mondiale, Turing a travaillé pour le gouvernement britannique en tant que cryptographe. Il a été l'un des principaux contributeurs au décryptage des codes allemands, notamment en développant la machine de Turing électromécanique, une version plus avancée de la machine de Turing.

Après la guerre, Turing a continué à travailler sur la théorie de l'informatique. En 1950, il a publié un article intitulé "Computing Machinery and Intelligence", dans lequel il a proposé le test de Turing, un test pour déterminer si une machine peut penser comme un être humain. Ce test est toujours utilisé aujourd'hui pour évaluer l'intelligence artificielle.

Malheureusement, la vie de Turing a été assombrie par des problèmes personnels. En 1952, il a été arrêté pour homosexualité, qui était alors illégale en Angleterre. Il a été condamné à la castration chimique et est décédé en 1954, apparemment d'un suicide.

Malgré sa mort tragique, Turing a laissé une marque indélébile sur le monde de l'informatique. Son travail sur la machine de Turing a jeté les bases de l'informatique moderne et a ouvert la voie à la résolution de problèmes complexes. Sa contribution à la cryptographie a aidé à mettre fin à la Seconde Guerre mondiale plus rapidement, sa contribution à la théorie de l'informatique a permis de développer l'intelligence artificielle, et son travail a inspiré des générations de chercheurs en informatique.

2.2 La machine de Turing

La machine de Turing est un modèle hypothétique d'un ordinateur conçu par Alan Turing en 1936. Elle est considérée comme l'un des fondements de l'informatique moderne, car elle a fourni un modèle théorique de l'informatique qui a permis de développer des algorithmes pour résoudre des problèmes complexes.

Le concept de la machine de Turing est simple mais puissant. Elle est basée sur une bande de papier sur laquelle des symboles peuvent être inscrits et effacés, ainsi que sur un dispositif de lecture/écriture qui peut déplacer la bande de papier d'un côté à l'autre. La bande est divisée en cases, et chaque case peut contenir un symbole. La machine de Turing peut lire un symbole de la case actuelle, écrire un nouveau symbole sur la case actuelle, et se déplacer à gauche ou à droite sur la bande.

La machine de Turing a été conçue pour exécuter des algorithmes en utilisant un ensemble d'instructions simples. Les instructions sont stockées dans une table de transition, qui indique quelle action la machine doit effectuer en fonction du symbole actuel et de l'état actuel de la machine. Par exemple, si la machine lit un symbole X dans la case actuelle et se trouve dans l'état A, la table de transition peut indiquer que la machine doit écrire un symbole Y sur la case actuelle, se déplacer d'une case à droite et passer à l'état B.

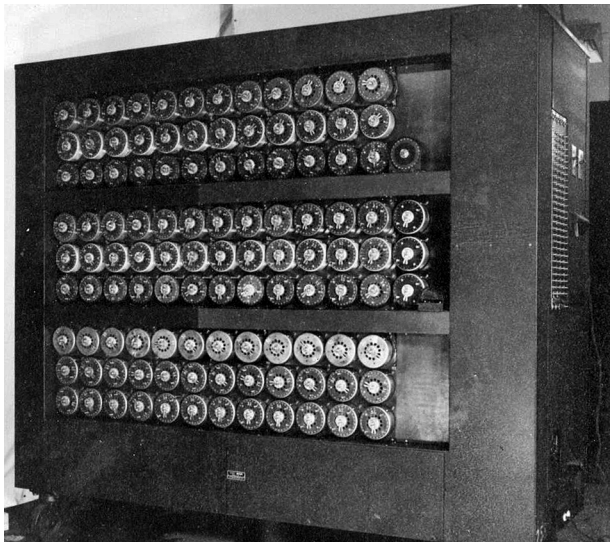


FIGURE I.2 – La machine de Turing.

La machine de Turing peut exécuter tous les algorithmes qui peuvent être exécutés par un ordinateur moderne. Elle peut effectuer des calculs mathématiques, trier des données, effectuer des opérations logiques, et bien plus encore. Elle peut également simuler d'autres machines de Turing, ce qui signifie qu'elle peut exécuter des programmes informatiques écrits dans d'autres langages de programmation.

La machine de Turing est souvent utilisée comme modèle pour décrire d'autres systèmes informatiques. Par exemple, les ordinateurs modernes sont souvent considérés comme des machines de Turing universelles, car ils peuvent exécuter n'importe quel algorithme qui peut être exécuté par une machine de Turing.

La machine de Turing a été développée à une époque où l'informatique n'existait pas encore. Elle a ouvert la voie à la résolution de problèmes complexes en fournissant un modèle théorique de l'informatique qui a permis de développer des algorithmes pour résoudre ces problèmes. Elle a également contribué à l'essor de l'intelligence artificielle en permettant de simuler des processus de pensée humaine.

3 Aperçu historique et contextuel de la proposition d'Alan Turing

Pour comprendre la naissance de la machine de Turing, il faut d'abord remonter à la logique combinatoire. Cette discipline a été introduite par George Boole au XIXe siècle pour formaliser la logique et l'algèbre. Il a introduit des concepts tels que les variables booléennes et les opérateurs logiques pour décrire la logique des circuits électriques. Les travaux de Boole ont été poursuivis par Claude Shannon dans les années 1930, qui a introduit des circuits logiques basés sur les portes logiques AND, OR et NOT. Ces circuits ont été utilisés pour la conception de machines de calcul électromécaniques.

Puis est arrivé l'automate fini, également connu sous le nom de machine à états finis. Ces machines ont été introduites dans les années 1940 pour décrire les systèmes de contrôle automatique. Un automate fini est un modèle mathématique qui consiste en un ensemble fini d'états et une transition d'état déterminée par une entrée. Les travaux de McCulloch et Pitts ont montré que les réseaux de neurones artificiels pouvaient être modélisés à l'aide d'automates finis.

Ensuite, l'automate à pile a été développé pour décrire des langages de programmation non réguliers. Un automate à pile est une extension de l'automate fini avec une pile de données, où les éléments sont insérés et retirés en haut de la pile. Cette structure de données permet de stocker des informations sur les transitions précédentes et futures de l'automate. Les automates à pile ont été introduits par John Shepherd Barron en 1960 pour décrire le fonctionnement des langages de programmation de type Algol.

C'est dans ce contexte que la proposition d'Alan Turing est arrivée. En 1936, Turing a publié un article fondateur intitulé "On Computable Numbers, with an Application to the Entscheidungsproblem" dans lequel il propose un modèle abstrait de calcul. Ce modèle, connu sous le nom de machine de Turing, est basé sur une tête de lecture-écriture qui se déplace le long d'une bande de papier infinie. La machine de Turing peut lire, écrire et effacer des symboles sur la bande, et sa tête de lecture-écriture peut se déplacer vers la gauche ou la droite. Les transitions de l'automate sont déterminées par l'état de la tête de lecture-écriture et le symbole lu.

La machine de Turing a eu un impact majeur sur la théorie du calcul et a permis de formaliser la notion de calculabilité. Il a montré que toute fonction calculable peut être calculée par une machine de Turing, et que certaines fonctions ne sont pas calculables. La machine de Turing a également permis de définir l'Hiérarchie d'automates, qui classe les automates selon leur puissance de calcul.

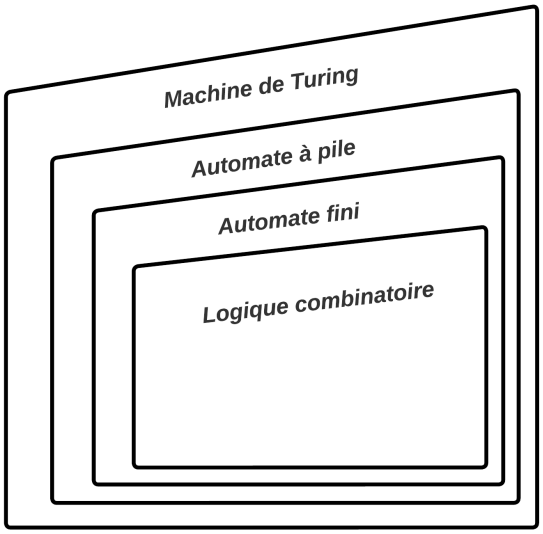


FIGURE I.3 – Hiérarchie d'automates.

En utilisant la machine de Turing, Turing a pu démontrer que certaines fonctions ma-

thématiques ne peuvent pas être calculées. Il a utilisé cet outil pour résoudre le problème de la décision, également appelé le problème de l'arrêt, qui consiste à déterminer si une machine de Turing s'arrêtera sur une entrée donnée. En utilisant une preuve par l'absurde, Turing a démontré que le problème de la décision est indécidable, c'est-à-dire qu'il n'existe pas d'algorithme qui puisse résoudre ce problème pour tous les cas.

Les travaux de Turing ont également eu un impact sur la cryptographie, qui est l'étude des techniques de sécurisation des communications. Pendant la Seconde Guerre mondiale, Turing a travaillé pour le gouvernement britannique pour casser les codes secrets utilisés par les Allemands. Il a utilisé une machine de Turing pour casser le code Enigma, qui était utilisé pour chiffrer les communications militaires allemandes. Les travaux de Turing ont eu un impact important sur la victoire des Alliés pendant la guerre.

4 Importance de la machine de Turing dans l'histoire de l'informatique

La machine de Turing est un élément clé de l'histoire de l'informatique, qui a permis de formaliser les concepts de calculabilité et d'algorithme, et qui a influencé la conception et l'implémentation des ordinateurs modernes. En effet, avant l'arrivée de la machine de Turing, la notion de calculabilité n'était pas formalisée, et il n'y avait pas de définition précise de ce qu'était un algorithme. La machine de Turing a permis de formaliser ces concepts et de donner une définition précise de ce qu'est un algorithme.

Cette machine n'est pas simplement un modèle abstrait de calcul, mais elle peut également être implémentée dans un ordinateur physique. Les ordinateurs modernes fonctionnent sur le principe de la machine de Turing, avec des processeurs qui exécutent des instructions stockées dans la mémoire. Elle est donc considérée comme un modèle théorique de l'informatique, mais elle a également des applications pratiques dans la conception et l'implémentation des ordinateurs.

Les travaux de Turing ont également montré que certaines fonctions mathématiques ne peuvent pas être calculées, ce qui a des implications importantes pour la théorie de l'informatique. En effet, la machine de Turing a permis de développer la théorie du calcul et de l'Hiérarchie d'automates, qui permet de classer les problèmes de calcul selon leur complexité.

La machine de Turing est un modèle abstrait de calcul qui a permis de formaliser les concepts de calculabilité et d'algorithme, et qui a influencé la conception et l'implémentation des ordinateurs modernes. Elle a également permis de développer la théorie du calcul et de l'Hiérarchie d'automates, qui a des implications importantes pour la théorie de l'informatique. Enfin, les travaux de Turing sur la machine de Turing ont eu des implications importantes pour la cryptographie et la sécurité des communications, ce qui montre que la machine de Turing n'est pas simplement un modèle abstrait, mais qu'elle a également des applications pratiques dans le monde réel.

5 Types des machines de Turing

Les types de machines de Turing sont une classification basée sur les capacités de la machine à traiter des informations. En général, il y a quatre types de machines de Turing

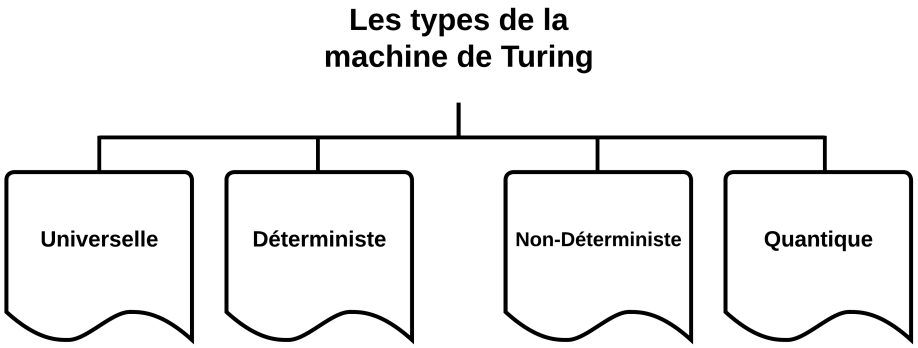


FIGURE I.4 – Les types de la machine de Turing.

5.1 Machine de Turing universelle

Une machine de Turing universelle est un type spécifique de machine de Turing qui peut simuler n'importe quelle autre machine de Turing. Elle a été conçue par Alan Turing en 1937, peu après l'invention de la machine de Turing standard. La machine de Turing universelle utilise un programme (ou "table de transition") spécifique pour simuler l'exécution d'une autre machine de Turing.

En d'autres termes, une machine de Turing universelle est capable d'exécuter n'importe quel algorithme qui peut être exécuté sur une machine de Turing. Elle peut être considérée comme une sorte de "machine virtuelle" pour les machines de Turing. Les machines de Turing universelles ont une grande importance en théorie de la calculabilité, car elles montrent que toutes les machines de Turing sont équivalentes en termes de puissance de calcul, et que tout algorithme qui peut être exécuté sur une machine de Turing peut être exécuté sur une machine de Turing universelle.

5.2 Machine de Turing déterministe

Une Machine de Turing Déterministe est une machine de Turing dont le comportement est entièrement déterminé par la règle de transition qu'elle suit. Cette règle de transition définit la prochaine action à effectuer pour la machine en fonction de son état courant et du symbole lu sur la bande. En d'autres termes, à chaque étape, la machine de Turing déterministe suit une et une seule transition possible en fonction de l'état et du symbole courant. Cela rend le comportement de la machine de Turing déterministe facilement prévisible.

Les machines de Turing déterministes sont souvent utilisées pour modéliser des algorithmes simples et pour effectuer des calculs sur des données structurées. Par exemple,

elles peuvent être utilisées pour trier des données, rechercher des éléments dans une liste ou exécuter des calculs mathématiques simples.

Les machines de Turing déterministes ont l'avantage d'être faciles à comprendre et à analyser, car leur comportement est déterminé de manière prévisible. Cependant, elles ont des limitations en termes de puissance de calcul, car elles ne peuvent pas simuler tous les types de problèmes de manière efficace. Pour des problèmes plus complexes, d'autre type des machines de Turing peuvent être nécessaires.

5.3 Machine de Turing non déterministe

Une Machine de Turing non Déterministe est une variante de la machine de Turing qui, contrairement à la machine de Turing déterministe, peut avoir plusieurs transitions possibles à partir d'un même état et d'un même symbole lu sur la bande. Au lieu de suivre une seule transition déterministe, la MTND peut se diviser en plusieurs chemins de calcul possibles, chacun étant déterminé par une probabilité ou une heuristique.

Autrement dit, une MTND peut explorer plusieurs options de calcul simultanément, ce qui lui permet de résoudre certains problèmes de manière plus efficace qu'une machine de Turing déterministe. Cela en fait un outil puissant pour modéliser des algorithmes probabilistes et pour résoudre des problèmes de décision complexes tels que la recherche de solutions optimales à un problème.

Cependant, la complexité de la MTND est également plus élevée, car elle nécessite de vérifier tous les chemins de calcul possibles pour déterminer la solution optimale. De plus, la MTND n'est qu'une abstraction théorique et n'a pas encore été implémentée dans la pratique. Elle est principalement utilisée pour définir des classes de complexité pour les problèmes de calcul.

5.4 Machine de Turing quantique

Les Machines de Turing Quantiques sont des machines hypothétiques qui utilisent des principes quantiques pour effectuer des calculs. Elles sont basées sur des qubits, qui sont des unités de stockage de l'information quantique. Contrairement aux bits classiques, les qubits peuvent exister dans des états de superposition, ce qui signifie qu'ils peuvent représenter plusieurs états simultanément.

L'utilisation de qubits permet aux MTQ d'explorer plusieurs chemins de calcul simultanément, ce qui peut accélérer certaines opérations. Par exemple, pour factoriser un entier très grand en utilisant une machine de Turing classique, il faut essayer toutes les combinaisons possibles de nombres premiers jusqu'à trouver la bonne solution. Cela peut prendre beaucoup de temps, même avec des ordinateurs très rapides. En revanche, une MTQ pourrait explorer toutes les combinaisons possibles simultanément, ce qui réduirait considérablement le temps nécessaire pour trouver la solution.

Cependant, les MTQ sont soumises à des contraintes strictes, telles que la décohérence,

qui est la perte d'information quantique due à l'interaction avec l'environnement, et l'effet tunnel, qui peut causer des erreurs de calcul. Ces contraintes limitent la taille et la durée des opérations quantiques pouvant être effectuées sur une MTQ.

En dépit de ces défis, les MTQ sont étudiées et développées activement par de nombreuses entreprises et institutions de recherche dans le monde entier, dans l'espoir de réaliser un jour des opérations quantiques plus avancées et résoudre certains problèmes qui sont considérés comme insolubles par les machines de Turing classiques.

6 Variantes des machines de Turing

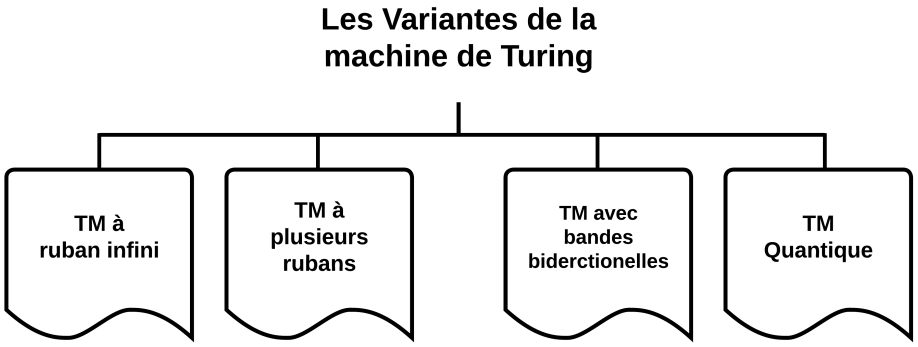


FIGURE I.5 – Les variantes de la machine de Turing.

Une variante de la machine de Turing est une version modifiée de la machine de Turing standard qui peut avoir des caractéristiques spécifiques ajoutées, supprimées ou modifiées. Les variantes de la machine de Turing sont souvent créées pour étudier des problèmes de calcul spécifiques ou pour permettre à la machine de résoudre des problèmes qui ne sont pas résolus par la machine de Turing standard.

6.1 Machine de Turing à ruban infini

La machine de Turing à ruban infini est une variante de la machine de Turing standard qui a été introduite pour surmonter les limitations de la machine de Turing standard. Dans la machine de Turing standard, le ruban est de taille finie et ne peut contenir qu'un nombre limité de symboles. En conséquence, la machine de Turing standard ne peut traiter que des entrées de taille finie et est incapable de résoudre certains problèmes complexes.

En revanche, la machine de Turing à ruban infini possède un ruban de longueur infinie dans les deux directions, ce qui lui permet de traiter des entrées de longueur arbitraire. Cela signifie que la machine de Turing à ruban infini est capable de résoudre des problèmes beaucoup plus complexes que la machine de Turing standard.

La machine de Turing à ruban infini a été utilisée pour démontrer que certains problèmes de décision sont indécidables, c'est-à-dire qu'il n'existe pas d'algorithme qui puisse résoudre ces problèmes pour toutes les entrées. Cela a conduit à des avancées importantes en informatique théorique, en particulier dans la théorie de la complexité computationnelle.

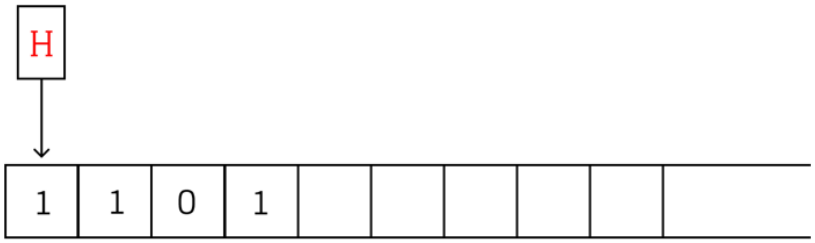


FIGURE I.6 – Représentation d'une machine de Turing a ruban infini.

6.2 Machine de Turing à plusieurs rubans

Une machine de Turing à plusieurs rubans est une extension de la machine de Turing standard qui dispose de plusieurs rubans plutôt que d'un seul. Cette extension permet de stocker plusieurs bandes de données en même temps, ce qui peut permettre un traitement plus rapide et plus efficace de l'information.

Chaque ruban peut stocker une partie de l'entrée ou du calcul en cours, et la machine peut déplacer les têtes de lecture/écriture sur les différents rubans simultanément. Cela permet à la machine de traiter plusieurs parties de l'entrée en parallèle, réduisant ainsi le temps de calcul.

Les machines de Turing à plusieurs rubans ont été introduites par Marvin Minsky en 1961 dans son livre "Computation : Finite and Infinite Machines". Cette extension de la machine de Turing standard a été utilisée pour démontrer des propriétés fondamentales de la théorie de la complexité algorithmique.

Les machines de Turing à plusieurs rubans sont souvent utilisées dans la conception de langages de programmation et dans la théorie de la complexité algorithmique pour prouver l'équivalence entre différents modèles de calcul et pour explorer les limites de la computation.

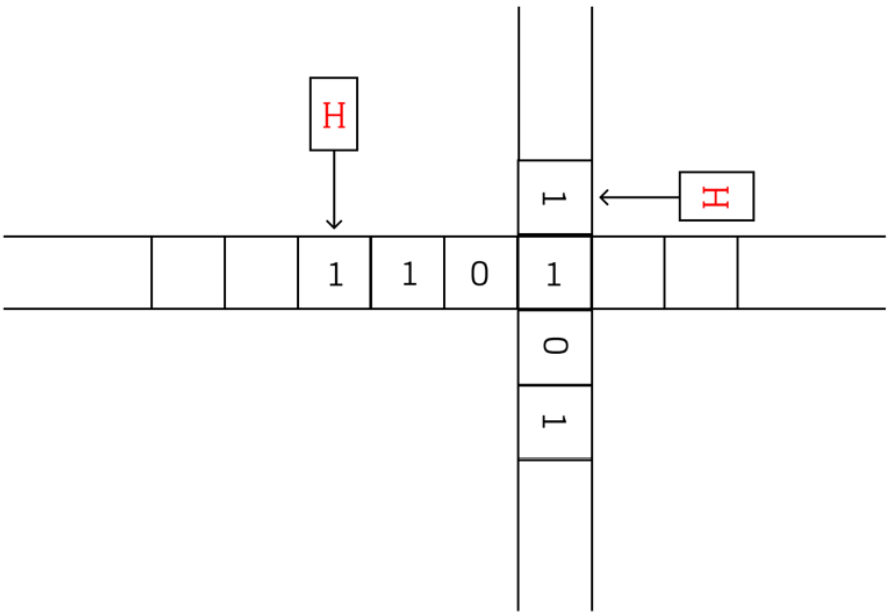


FIGURE I.7 – Représentation d'une machine de Turing a plusieurs rubans.

6.3 Machine de Turing avec bandes bidirectionnelles

Une machine de Turing à bandes bidirectionnelles est une variante de la machine de Turing dans laquelle les bandes ont des cellules qui peuvent être lues et écrites dans les deux sens, c'est-à-dire à la fois de gauche à droite et de droite à gauche. Contrairement à la machine de Turing à bande simple, qui ne permet de déplacer la tête de lecture/écriture que dans une seule direction, la machine de Turing à bandes bidirectionnelles permet de se déplacer dans les deux directions.

Cette fonctionnalité supplémentaire permet une plus grande flexibilité dans la conception et la mise en œuvre d'algorithmes pour résoudre certains problèmes. Par exemple, certains algorithmes de traitement du langage naturel nécessitent de parcourir les chaînes de caractères dans les deux sens, ce qui peut être plus facilement accompli avec une machine de Turing à bandes bidirectionnelles. De même, certains problèmes d'optimisation peuvent bénéficier de la capacité de se déplacer dans les deux directions pour chercher des solutions dans différentes directions.

Cependant, l'utilisation de plusieurs directions peut rendre la conception de l'algorithme plus complexe et augmenter le nombre d'états nécessaires pour représenter le comportement de la machine de Turing. En général, l'utilisation d'une bande bidirectionnelle peut améliorer l'efficacité et la vitesse de certains algorithmes, mais cela peut également augmenter la complexité de leur mise en œuvre.

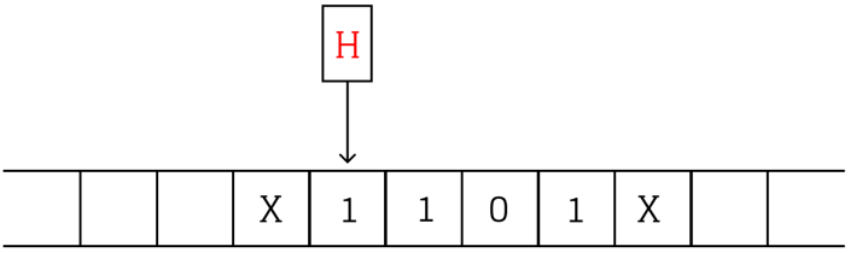


FIGURE I.8 – Représentation d'une machine de Turing a bande bidirectionnel.

6.4 Machine de Turing à plusieurs têtes

Une machine de Turing à plusieurs têtes est une variante de la machine de Turing standard qui dispose de plusieurs têtes de lecture/écriture sur la bande de la machine. Chaque tête peut se déplacer indépendamment des autres têtes et peut effectuer des opérations de lecture, d'écriture ou de déplacement sur la bande.

La présence de plusieurs têtes sur la machine de Turing permet d'effectuer des opérations plus complexes et plus rapides que sur une machine de Turing standard. Par exemple, une tête peut se déplacer rapidement vers une partie de la bande où une autre tête a déjà écrit des données et ainsi lire ces données sans avoir besoin de se déplacer sur toute la longueur de la bande.

Les machines de Turing à plusieurs têtes sont utilisées dans la théorie de la complexité pour démontrer que certains problèmes peuvent être résolus en temps polynomial, c'est-à-dire en un temps raisonnable, contrairement à d'autres problèmes qui nécessitent un temps exponentiel. Ces machines sont également utilisées en informatique théorique pour l'analyse de la complexité des algorithmes et pour la conception de programmes efficaces.

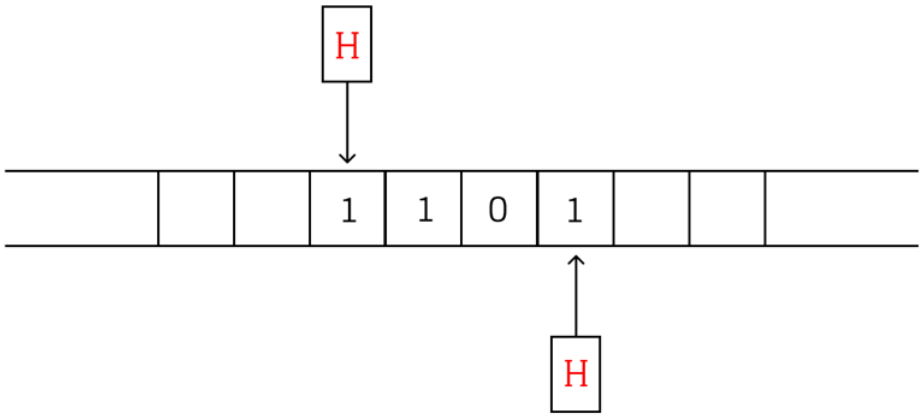


FIGURE I.9 – Représentation d'une machine de Turing a plusieurs têtes.

7 Conclusion

En conclusion, Alan Turing, le mathématicien visionnaire, a développé la machine de Turing, un modèle abstrait de calcul qui a joué un rôle fondamental dans l'histoire de l'informatique. Son travail a permis de formaliser les concepts de calculabilité et d'algorithme, et a influencé la conception des ordinateurs modernes. L'héritage d'Alan Turing et de sa machine de Turing se manifeste encore aujourd'hui dans de nombreux aspects de la technologie et de la sécurité informatique.

Chapitre II

Science informatique et la machine de Turing

1 Intoduction

Ce chapitre explore l'importance de la machine de Turing dans la science informatique. Nous examinerons son utilisation pour la résolution de problèmes informatiques et la détermination de la calculabilité. Nous aborderons également les limitations de la machine de Turing et ses contributions à la compréhension des algorithmes informatique et à la théorie de la complexité.

2 Utilisation de la machine de Turing

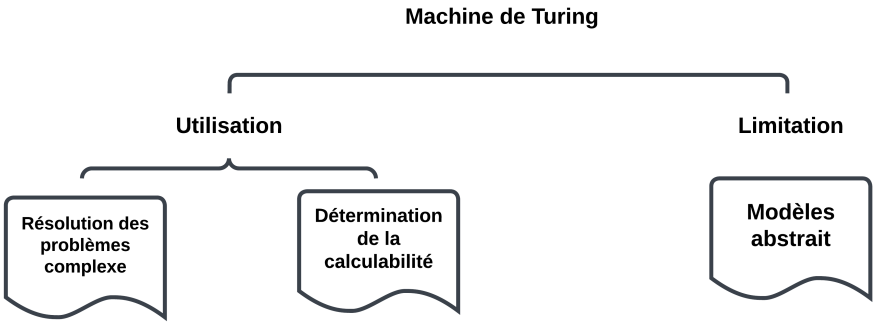


FIGURE II.1 – Utilisation et limitation de la machine de Turing.

La machine de Turing est un outil théorique important dans le domaine de la science informatique. Elle peut être utilisée pour résoudre des problèmes algorithmiques complexes en simulant le comportement d'un ordinateur programmable. En effet, la machine de Turing peut être programmée pour effectuer des opérations mathématiques telles que la multiplication, l'addition et la soustraction, ainsi que pour résoudre des problèmes de logique et de décision. Sa capacité à simuler le comportement de n'importe quel ordinateur programmable en fait un outil très utile pour résoudre des problèmes informatiques complexes.

En plus de résoudre des problèmes informatiques, la machine peut également être utilisée pour déterminer la calculabilité des problèmes. Un problème est considéré comme calculable s'il peut être résolu par la même machine. Si un problème ne peut pas être résolu, il est considéré comme non calculable. Cette distinction entre les problèmes calculables et non calculables est importante dans la science informatique, car elle aide à déterminer les limites de la calculabilité.

La machine de Turing présente également des limitations. Elle est un modèle abstrait qui ne peut pas être construit physiquement en raison de sa bande de lecture-écriture infinie. En outre, cette machine ne peut pas résoudre tous les problèmes, car certains problèmes sont trop complexes pour être résolus par des algorithmes. Ces limitations soulignent l'importance de la recherche continue dans le domaine de la science informatique pour trouver de nouveaux outils et algorithmes pour résoudre des problèmes encore plus complexes.

2.1 Résolution de problèmes informatiques

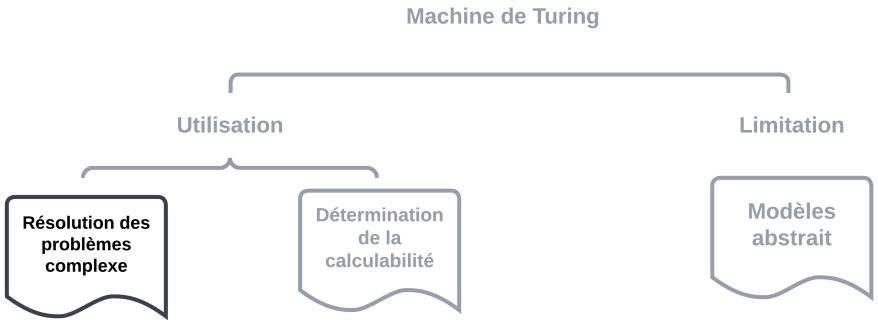


FIGURE II.2 – Résolution des problèmes informatiques.

La machine de Turing joue un rôle essentiel dans la résolution de nombreux problèmes mathématiques et informatiques. Elle offre une perspective algorithmique pour résoudre des défis tels que la décision du problème de l'arrêt, la reconnaissance des langages formels, la construction de compilateurs, l'optimisation des programmes, la cryptographie, la vérification de programmes et la conception de protocoles de communication. En effet, la machine de Turing est capable de traiter ces problèmes en suivant un algorithme bien défini qui guide ses étapes de calcul. Elle peut manipuler des symboles et effectuer des opérations sur des bandes, tout en fournissant une représentation abstraite des calculs.

Néanmoins, il existe une classe de problèmes qui échappent à la résolution de manière algorithmique. Ces problèmes sont qualifiés d'intrinsèquement non calculables, car il n'existe aucun algorithme qui puisse les résoudre en un nombre fini d'étapes. L'exemple classique est le problème de l'arrêt, où il est impossible de déterminer, pour tout programme et toute donnée d'entrée, si le programme s'arrêtera ou continuera indéfiniment. Peu importe l'algorithme ou la machine de calcul utilisée, il n'existe pas de solution générale à ce problème. Cela met en évidence les limites fondamentales de la résolution

algorithmique.

La machine de Turing offre un modèle puissant pour résoudre un large éventail de problèmes mathématiques et informatiques en utilisant des algorithmes. Cependant, elle rencontre ses limites lorsqu'il s'agit de problèmes intrinsèquement non calculables, tels que le problème de l'arrêt. Ces problèmes posent des défis qui vont au-delà des capacités de tout modèle de calcul, soulignant ainsi la complexité et les frontières de la résolution algorithmique.

2.2 Détermination de la calculabilité

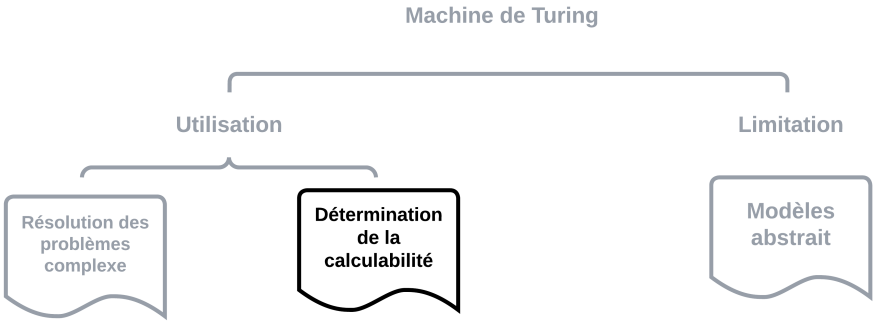


FIGURE II.3 – Détermination de calculabilité.

L'évaluation de la calculabilité d'un problème constitue une préoccupation essentielle au sein du domaine de l'informatique théorique. Cette évaluation repose sur la détermination de la possibilité de résoudre un problème de manière algorithmique, c'est-à-dire d'établir l'existence d'un algorithme permettant de le résoudre en un nombre fini d'étapes. Afin d'étudier cette calculabilité, différents modèles de calcul sont utilisés, tels que la machine de Turing, le lambda-calcul et les fonctions récursives.

Pour évaluer la calculabilité d'un problème en utilisant la machine de Turing, il est nécessaire de décrire de manière précise le problème à résoudre et de construire une machine de Turing qui puisse le résoudre. Cette machine doit être formellement décrite à l'aide d'une liste de règles décrivant comment elle manipule les symboles sur sa bande. Une fois que la machine est construite, il convient de déterminer si elle résout le problème en un nombre fini d'étapes pour toutes les données d'entrée. Si tel est le cas, alors le problème est considéré comme calculable.

L'impossibilité de construire une machine de Turing résolvant le problème ne signifie pas nécessairement que le problème est non calculable. Il est tout à fait envisageable qu'une autre machine de Turing puisse résoudre le problème. En revanche, si l'on parvient à prouver l'inexistence d'un algorithme permettant de résoudre le problème en un nombre fini d'étapes, alors le problème est intrinsèquement non calculable.

Il est important de souligner que la calculabilité d'un problème ne garantit pas forcément sa facilité de résolution pratique. En effet, le temps d'exécution de l'algorithme peut

être très long pour certaines instances du problème, rendant ainsi sa résolution difficile en pratique. Ainsi, il existe des problèmes calculables considérés comme étant difficiles, voire impossibles à résoudre en pratique.

En conclusion, l'évaluation de la calculabilité d'un problème à l'aide de la machine de Turing consiste à construire une machine de Turing qui résout le problème et à vérifier si celle-ci le fait en un nombre fini d'étapes pour toutes les données d'entrée. Si tel est le cas, le problème est considéré comme calculable. Dans le cas contraire, il peut être nécessaire d'explorer d'autres machines de Turing ou de démontrer l'existence d'une limitation intrinsèque à la calculabilité du problème.

2.3 Limitations de la machine de Turing

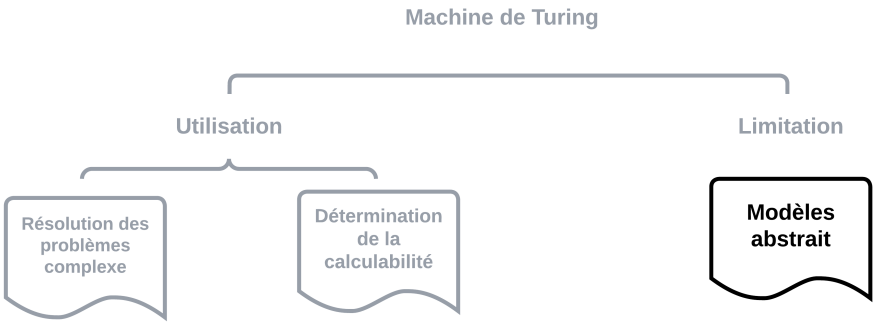


FIGURE II.4 – Limitations de la machine de Turing.

La machine de Turing est un modèle fondamental de l'informatique théorique, qui permet de formaliser de manière précise les étapes nécessaires à la résolution d'un problème. Cependant, malgré son utilité, la machine de Turing présente certaines limitations qui la rendent inadaptée pour modéliser certains types de calculs.

L'une des principales limitations de la machine de Turing est qu'elle est un modèle purement séquentiel. Cela signifie que la machine ne peut traiter qu'une seule entrée à la fois et ne peut pas effectuer de calculs parallèles. Cette limitation est importante car de nombreux problèmes dans le monde réel nécessitent des calculs parallèles pour être résolus de manière efficace. Par exemple, la simulation de systèmes physiques complexes ou le traitement d'images et de vidéos peuvent bénéficier de calculs parallèles pour accélérer les temps de calcul.

Une autre limitation de la machine de Turing est qu'elle est un modèle déterministe. Cela signifie que la machine ne peut effectuer qu'une seule action pour chaque état de la bande, en fonction de l'état actuel de la machine et du symbole présent sur la bande. Cette limitation peut rendre la résolution de certains problèmes plus difficile, car il est souvent plus facile de résoudre un problème en utilisant des stratégies non déterministes ou probabilistes.

Enfin, la machine de Turing ne peut pas résoudre certains problèmes qui nécessitent une mémoire infinie. En effet, la machine de Turing dispose d'une bande de stockage finie, ce qui signifie qu'elle ne peut pas traiter des données infinies ou des problèmes qui nécessitent un nombre infini d'étapes de calcul. Cette limitation est importante car certains problèmes mathématiques, tels que la recherche de solutions pour des équations différentielles, nécessitent un traitement de données infinies pour être résolus.

En conclusion, la machine de Turing est un modèle important de l'informatique théorique, mais elle présente des limitations importantes qui la rendent inadaptée pour modéliser certains types de calculs. Les limites de la machine de Turing ont conduit au développement de modèles de calcul alternatifs, tels que les machines de Turing non déterministes ou les machines quantiques, qui permettent de modéliser des types de calculs différents de ceux modélisables par la machine de Turing classique.

3 Contributions de la machine de Turing à la science informatique

3.1 Impact sur la compréhension des algorithmes informatiques

L'impact de la machine de Turing sur la compréhension des algorithmes informatiques est considérable. En effet, la machine de Turing fournit un modèle théorique qui permet de formaliser de manière précise les étapes nécessaires à la résolution d'un problème, en décomposant le problème en une série d'étapes simples et en les reliant les unes aux autres de manière logique et séquentielle. Cette formalisation permet de comprendre la complexité des algorithmes et d'analyser leur efficacité, ce qui est crucial dans le développement de logiciels et d'applications informatiques.

La machine de Turing a également contribué à la théorie de la calculabilité, qui est un domaine clé de l'informatique théorique. La théorie de la calculabilité étudie les limites de ce qui peut être calculé par des algorithmes, en déterminant quels problèmes sont résolubles et quels problèmes ne le sont pas. Cette théorie a conduit à la découverte de nombreux problèmes algorithmiques qui sont intrinsèquement difficiles à résoudre, tels que le problème de la décision, le problème de la coloration de graphes ou le problème du voyageur de commerce. La théorie de la calculabilité a également permis de déterminer les limites de la complexité des algorithmes, en déterminant que certains problèmes nécessitent un temps de calcul exponentiel pour être résolus, tandis que d'autres peuvent être résolus en temps polynomial.

Enfin, la machine de Turing a également inspiré le développement de modèles de calcul alternatifs, tels que les machines de Turing non déterministes ou les machines quantiques. Ces modèles de calcul permettent de modéliser des types de calculs différents de ceux modélisables par la machine de Turing classique, et ont conduit à des avancées importantes dans des domaines tels que la cryptographie, la recherche d'informations ou la modélisation de systèmes physiques complexes.

En conclusion, la machine de Turing a eu un impact considérable sur la compréhension

des algorithmes informatiques, en fournissant un modèle théorique précis qui permet de formaliser les étapes nécessaires à la résolution d'un problème. La machine de Turing a également contribué à la théorie de la calculabilité, en déterminant les limites de ce qui peut être calculé par des algorithmes, et a inspiré le développement de modèles de calcul alternatifs qui ont conduit à des avancées importantes dans de nombreux domaines de l'informatique.

3.2 Contribution à la théorie de la complexité

La théorie de la complexité étudie la quantité de ressources (temps, espace, etc.) nécessaires pour résoudre des problèmes de calcul. La machine de Turing a permis de formaliser et de définir rigoureusement ce qu'est un problème de calcul et comment il peut être résolu.

Plus précisément, la machine de Turing a introduit le concept de complexité de temps et de complexité d'espace. La complexité de temps mesure le nombre d'étapes de calcul nécessaires pour résoudre un problème, tandis que la complexité d'espace mesure la quantité de mémoire nécessaire. Ces concepts sont fondamentaux pour évaluer l'efficacité des algorithmes et pour classer les problèmes en fonction de leur difficulté intrinsèque.

La machine de Turing a également permis de formaliser la notion de problème indécidable, c'est-à-dire un problème pour lequel il n'existe pas d'algorithme qui puisse toujours donner une réponse correcte. L'exemple le plus célèbre est le problème de l'arrêt, qui consiste à déterminer, étant donné un programme et une entrée, si ce programme s'arrêtera ou non sur cette entrée. Alan Turing a démontré que le problème de l'arrêt est indécidable en utilisant une machine de Turing.

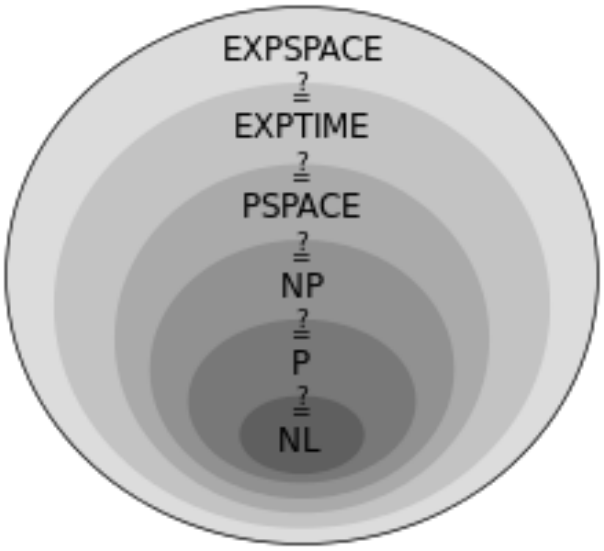


FIGURE II.5 – Les classes de complexité.

4 Conclusion

En conclusion, la machine de Turing a joué un rôle fondamental dans la science informatique en permettant la résolution de problèmes, la détermination de la calculabilité et en contribuant à la théorie de la complexité. Malgré ses limitations, cet outil abstrait a jeté les bases de l'informatique moderne et continue d'influencer de nombreux domaines de recherche et d'application.

Chapitre III

Fonctionnement de la machine de Turing

1 Introduction

Dans ce chapitre, nous étudierons le fonctionnement de la machine de Turing. Nous examinerons sa description, en mettant l'accent sur le ruban infini et les symboles. Nous aborderons également les notions d'états et de transitions. Enfin, nous illustrerons le fonctionnement de la machine de Turing à travers un exemple concret de son utilisation.

2 Description du modèle de la machine de Turing

Une machine de Turing est un objet abstrait composé de quatre éléments :

- Le ruban infini est un support de stockage pour les données, et chaque case du ruban peut contenir un symbole issu d'un alphabet fini.
- Une tête de lecture/écriture pouvant lire et écrire sur le ruban, et pouvant se déplacer vers la gauche ou la droite du ruban.
- Un registre d'état qui retient l'état actuel de la machine. Pour un être humain faisant du calcul mental, on pourrait comparer cela au fait de retenir l'état d'avancement du calcul (par exemple les retenues).
- Une table d'actions qui lie (en fonction de l'état courant) un symbole lu à une action à effectuer par la tête de lecture/écriture (déplacement et/ou écriture).

Ces quatre éléments combinés permettent à la machine de Turing de résoudre des problèmes algorithmiques de manière abstraite, posant ainsi les fondements de l'informatique moderne.

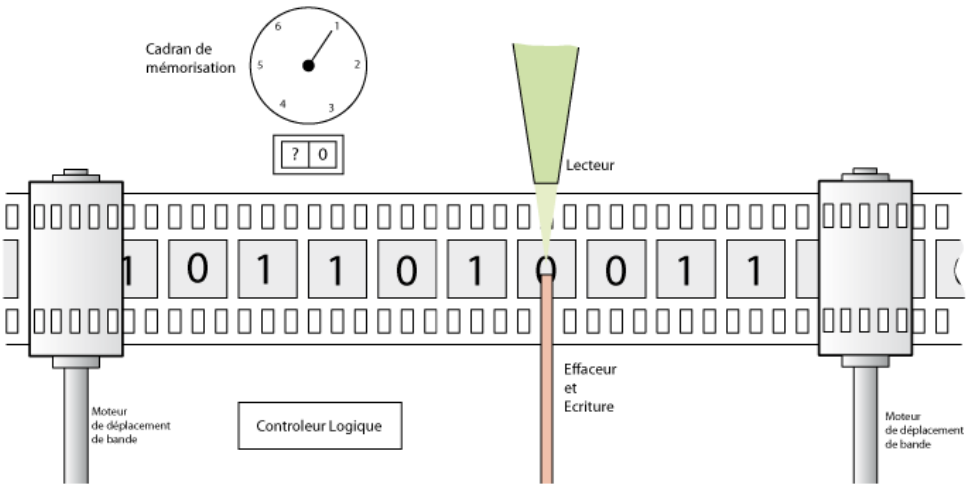


FIGURE III.1 – Schéma illustratif de fonctionnement d'une machine de Turing.

3 Ruban infini et symboles

Le ruban infini est l'un des éléments les plus importants de la machine de Turing, car il permet de stocker les données nécessaires pour effectuer des calculs. Le ruban est infini dans la mesure où il n'a pas de limite physique et peut contenir une quantité illimitée de données. Chaque case du ruban peut contenir un symbole issu d'une liste finie appelée alphabet. Les symboles sont lus et écrits par la tête de lecture/écriture de la machine.

4 Tête de lecture et déplacement

La tête de lecture/écriture est l'élément de la machine de Turing qui lit et écrit les symboles sur le ruban. La tête est attachée à un bras mobile qui peut se déplacer vers la gauche ou la droite le long du ruban. Lorsque la tête se déplace, elle lit le symbole présent sur la case et effectue une action en fonction de la table d'actions. Cette table d'actions est basée sur l'état actuel de la machine et le symbole présent sur la case. Elle spécifie l'action à effectuer (par exemple, déplacer la tête vers la gauche ou écrire un nouveau symbole sur la case) et l'état suivant de la machine.

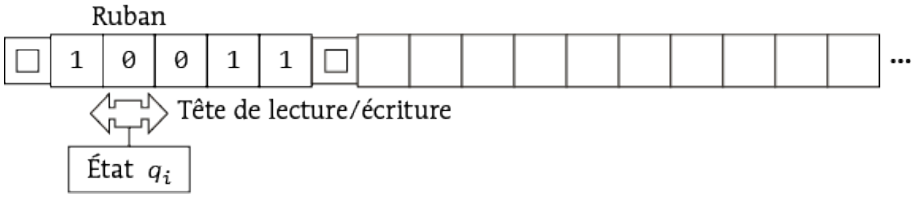


FIGURE III.2 – Schéma illustratif d'une tête de lecture et un ruban.

5 États et transitions

Les états constituent une composante essentielle de la machine de Turing. Le registre d'état permet de stocker l'état actuel de la machine. Il y a trois types d'états dans une machine de Turing : l'état initial, les états intermédiaires et les états acceptants.

L'état initial est l'état dans lequel se trouve la machine de Turing lorsqu'elle démarre. Cet état est défini par l'algorithme à exécuter. Les états intermédiaires sont des états transitoires qui permettent de faire avancer le calcul. La machine de Turing passe par plusieurs états intermédiaires pendant son fonctionnement, en fonction de la lecture des symboles sur le ruban et des actions à effectuer.

Enfin, les états acceptants sont des états correspondant à une fin possible de l'algorithme. Si la machine de Turing atteint un état acceptant, cela signifie que le calcul a été effectué avec succès. Les états acceptants sont définis par l'algorithme à exécuter, et il peut y en avoir plusieurs, en fonction du problème à résoudre.

Les transitions entre les états sont également importantes dans une machine de Turing. Elles sont déterminées par la table d'actions, qui spécifie l'action à effectuer en fonction de l'état actuel de la machine et du symbole présent sur la case lue par la tête de lecture/écriture. Les transitions permettent à la machine de Turing de se déplacer sur le ruban, de lire et d'écrire des symboles, et de changer d'état en fonction du problème à résoudre. En résumé, les états et les transitions sont des éléments clés de la machine de Turing, qui lui permettent d'effectuer des calculs complexes sur le ruban infini.

6 Exemple de fonctionnement de la machine de Turing

6.1 Représentation mathématique

La machine de Turing peut être représentée mathématiquement par un ensemble de règles et de transitions. Ces règles spécifient comment la machine réagit à un symbole donné lu sur le ruban, en fonction de son état actuel.

Pour illustrer cela, prenons l'exemple d'une machine de Turing simple qui effectue la multiplication de deux nombres binaires. Supposons que nous ayons deux nombres binaires A et B sur le ruban, et que nous voulions calculer leur produit. La machine de Turing peut être configurée avec différents états et transitions pour réaliser cette opération.

Lorsque la machine commence, elle se trouve dans un état initial et sa tête de lecture est positionnée sur le premier symbole du nombre A. En fonction de ce symbole et de son état actuel, la machine effectue une transition spécifique. Elle peut déplacer sa tête de lecture vers la droite ou vers la gauche, écrire un nouveau symbole sur le ruban et passer à un nouvel état.

Ces transitions sont définies dans un tableau ou une fonction de transition, qui peut être représentée de manière mathématique.

Pour illustrer les expressions mathématiques du fonctionnement d'une machine de Turing dans l'exemple de la multiplication binaire, nous pouvons utiliser des notations symboliques pour représenter les états, les symboles et les transitions.

Supposons que nous ayons deux nombres binaires A = 101 et B = 11, et nous voulons calculer leur produit. Nous utilisons une machine de Turing pour effectuer cette opération.

Supposons également que l'instruction donnée soit

$$(p, 1, 0, R, q) \tag{III.1}$$

Ce qui signifie que la machine est à l'état p et que sa tête de lecture pointe sur le caractère 1. L'instruction spécifie que nous devons effacer le caractère 1 et le remplacer par 0, puis déplacer la tête de lecture vers la droite et enfin passer à l'état q.

Voici le déroulement de l'exécution de cette instruction :

1. Au début, nous avons le nombre A = 101 et le nombre B = 11 sur le ruban.
2. La machine de Turing est à l'état p et la tête de lecture pointe sur le premier 1 de A.
3. Selon l'instruction *(p, 1, 0, R, q)*, la machine efface le 1 et le remplace par 0.
4. Le ruban est maintenant mis à jour, A devient 100 et B reste inchangé.
5. La tête de lecture se déplace vers la droite, passant au prochain symbole sur le ruban.
6. La machine est maintenant à l'état q et la tête de lecture pointe sur le symbole suivant dans A.

Ce processus se répète pour chaque instruction donnée et pour chaque symbole du ruban, jusqu'à ce que la machine atteigne un état final et que le résultat de la multiplication soit obtenu.

Il est important de noter que cet exemple ne couvre qu'une seule instruction et une étape du processus de multiplication. Pour réaliser la multiplication complète de deux nombres binaires, plusieurs instructions et transitions seraient nécessaires pour gérer les différentes étapes du calcul.

6.2 Représentation à l'aide d'un graphe

Une autre façon de représenter le fonctionnement d'une machine de Turing est d'utiliser un graphe. Dans cette représentation, les états de la machine sont représentés par des nœuds du graphe, et les transitions entre les états sont représentées par des arêtes.

Prenons l'exemple de la multiplication binaire avec une machine de Turing. Nous pouvons créer un graphe où chaque nœud représente un état de la machine, et les arêtes représentent les transitions entre les états.

Chaque arête est étiquetée avec les informations pertinentes, telles que le symbole lu, le symbole écrit, le déplacement de la tête de lecture et le nouvel état.

En reliant les différents nœuds de l'état initial à l'état final à l'aide des transitions appropriées, nous construisons un graphe qui décrit le flux d'exécution de la machine de Turing pour réaliser la multiplication binaire.

Cette représentation graphique offre une visualisation claire du fonctionnement de la machine, en montrant les transitions entre les états et les actions à effectuer à chaque étape du processus de calcul. Elle facilite également la compréhension et l'analyse du comportement de la machine.

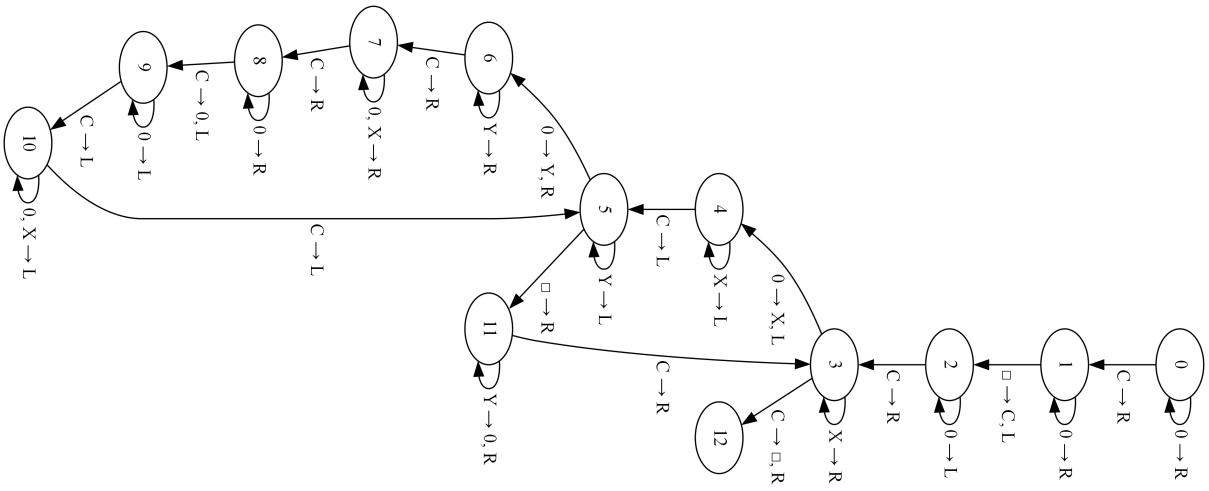


FIGURE III.3 – Exemple de représentation graphique.

7 Conclusion

En conclusion, ce chapitre nous a permis de plonger dans le fonctionnement essentiel de la machine de Turing. Nous avons exploré les éléments clés de son modèle, tels que le ruban infini, les symboles, les états et les transitions. À travers un exemple concret, nous avons également examiné sa représentation mathématique et sa visualisation graphique.

Chapitre IV

Plannification et gestion de projet

1 Introduction

La réussite d'un projet dépend largement d'une planification et d'une gestion efficaces. Dans ce chapitre, nous explorerons les différentes facettes de la planification et de la gestion de projet, en mettant l'accent sur une approche méthodologique en cascade. De plus, nous étudierons les outils et les technologies qui soutiennent cette méthodologie, afin de garantir une exécution fluide et efficace du projet.

2 Méthodologie en cascade

La méthode de gestion de projet en cascade, également connue sous le nom de modèle en cascade, est un cadre de gestion de projet linéaire et séquentiel. Dans cette approche, chaque phase du projet est réalisée de manière séquentielle, avec une transition claire entre les étapes. Le processus commence par une phase de planification approfondie, suivie de l'exécution, puis du contrôle et de la validation des résultats obtenus. Une fois qu'une phase est terminée, elle ne peut généralement pas être revisitée. Chaque étape du projet doit être achevée avant de passer à la suivante, ce qui rend la méthode de gestion en cascade moins flexible que d'autres approches. Cependant, elle offre une structure claire et prévisible, ce qui facilite la gestion des projets complexes. Cette méthode est souvent utilisée lorsque les exigences du projet sont stables et bien définies dès le départ, et lorsque les changements ultérieurs sont peu probables ou coûteux.

Voici quelques principes clés de la méthodologie en cascade :

- Séquentialité : Les phases du projet sont exécutées de manière séquentielle, une après l'autre, sans chevauchement significatif. Chaque phase doit être terminée avant de passer à la suivante.
- Planification approfondie : Une planification détaillée est effectuée avant le début du projet. Les tâches, les ressources nécessaires, les délais et les objectifs sont clairement définis dans un plan global.
- Validation à la fin des phases : Chaque phase du projet est validée avant de passer à la suivante. Cela garantit que les résultats obtenus sont conformes aux attentes et aux spécifications définies lors de la planification.

- Documentation détaillée : La méthodologie en cascade encourage la création de documentation détaillée à chaque étape du projet. Cela facilite la compréhension et la traçabilité des décisions prises et des résultats obtenus.
- Gestion du risque préliminaire : La gestion des risques est souvent effectuée en amont du projet, avec une identification et une évaluation préliminaires des risques potentiels. Les stratégies d'atténuation des risques sont généralement intégrées dans le plan global du projet.

La méthodologie en cascade convient mieux aux projets où les exigences sont stables et bien définies dès le départ.

2.1 Diagramme de Gantt

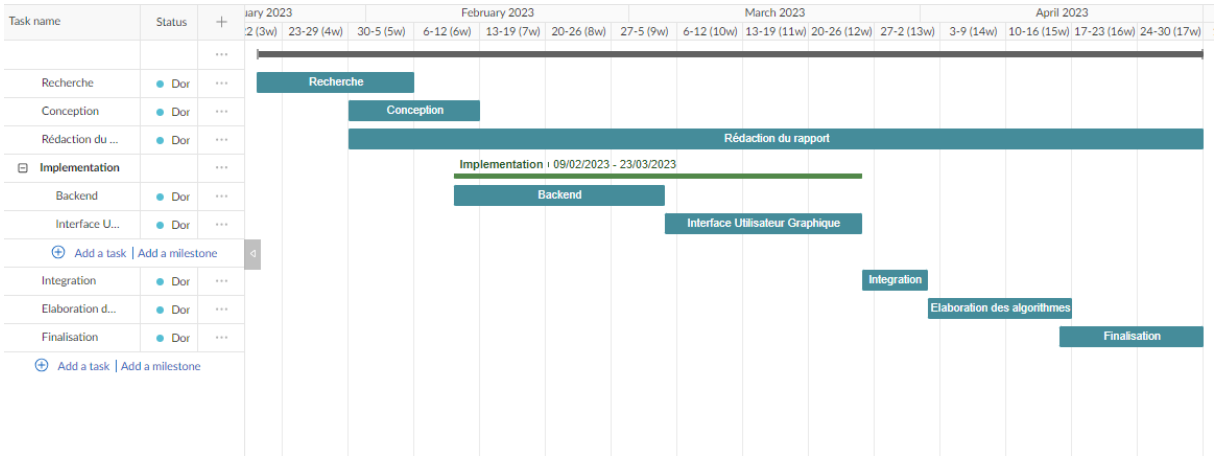


FIGURE IV.1 – Diagramme de Gantt.

3 Outils et technologies

3.1 Langage de programmation

FIGURE IV.2 – Python.

Python est un langage de programmation interprété, orienté objet et de haut niveau avec une sémantique dynamique. Ses structures de données intégrées de haut niveau, combinées à un typage dynamique et à une liaison dynamique, le rendent très attrayant pour le développement rapide d'applications, ainsi que pour une utilisation en tant que langage

de script ou de collage pour connecter des composants existants entre eux. La syntaxe simple et facile à apprendre de Python met l'accent sur la lisibilité et réduit donc le coût de la maintenance du programme. Python prend en charge les modules et les packages, ce qui encourage la modularité des programmes et la réutilisation du code. L'interpréteur Python et la vaste bibliothèque standard sont disponibles gratuitement sous forme de source ou binaire pour toutes les principales plates-formes et peuvent être distribués gratuitement.

Souvent, le choix de Python revient à la productivité accrue qu'il offre. Comme il n'y a pas d'étape de compilation, le cycle édition-test-débogage est incroyablement rapide. Le débogage des programmes Python est simple : une bogue ou une mauvaise entrée ne provoquera jamais d'erreur de segmentation. Au lieu de cela, lorsque l'interpréteur découvre une erreur, il déclenche une exception. Lorsque le programme n'attrape pas l'exception, l'interpréteur imprime une trace de pile. Un débogueur au niveau de la source permet d'inspecter les variables locales et globales, d'évaluer des expressions arbitraires, de définir des points d'arrêt, de parcourir le code ligne par ligne, etc. Le débogueur est écrit en Python lui-même, témoignant du pouvoir introspective de Python. D'un autre côté, le moyen le plus rapide de déboguer un programme est souvent d'ajouter quelques instructions d'impression à la source : le cycle rapide édition-test-débogage rend cette approche simple très efficace.

3.2 Bibliothèque

3.2.1 CustomTkinter

FIGURE IV.3 – CustomTkinter.

CustomTkinter est une bibliothèque d'interface utilisateur python basée sur Tkinter, qui fournit de nouveaux widgets modernes et entièrement personnalisables. Ils sont créés et utilisés comme des widgets Tkinter normaux et peuvent également être utilisés en combinaison avec des éléments Tkinter normaux. Les widgets et les couleurs des fenêtres s'adaptent à l'apparence du système ou au mode défini manuellement ("clair", "sombre"), et tous les widgets et fenêtres CustomTkinter prennent en charge la mise à l'échelle HighDPI (Windows, macOS). Avec CustomTkinter, on obtiendra un look cohérent et moderne sur toutes les plateformes de bureau (Windows, macOS, Linux).

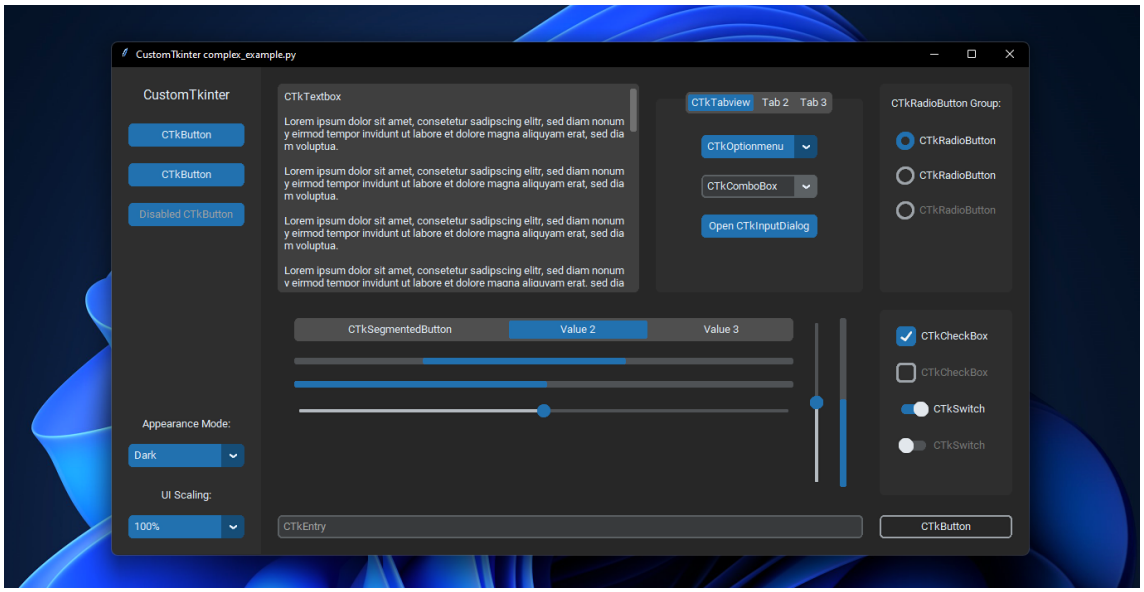


FIGURE IV.4 – Exemple d'interface grahique sur Windows 11 avec le mode sombre et le thème "bleu".

3.2.2 GraphViz - Graph Visualization

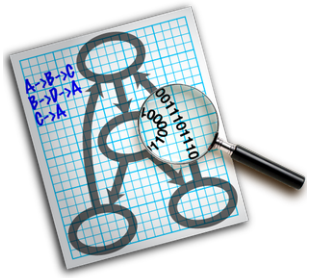


FIGURE IV.5 – GraphViz.

GraphViz, c'est un acronyme pour Graph Visualization (La visualisation de graphes en français) est un moyen open source pour représenter des informations structurelles sous forme de diagrammes de graphes abstraits et de réseaux. Le dessin automatique de graphiques a de nombreuses applications importantes en génie logiciel, en conception de bases de données et de sites Web, en mise en réseau et en interfaces visuelles pour de nombreux autres domaines.

Les programmes de mise en page prennent des descriptions de graphiques dans un langage textuel simple et créent des diagrammes dans plusieurs formats utiles tels que des images et SVG pour les pages Web, Postscript pour inclusion dans PDF ou d'autres documents ; ou afficher dans un navigateur de graphiques interactif.

Il possède aussi de nombreuses fonctionnalités utiles pour les diagrammes concrets, telles que des options pour les couleurs, les polices, les dispositions des nœuds tabulaires, les styles de ligne, les hyperliens et les formes personnalisées.

En pratique, les graphiques sont généralement générés à partir de sources de données externes, mais ils peuvent également être créés et édités manuellement, soit sous forme de fichiers texte bruts, soit dans un éditeur graphique.

3.3 Environnement de développement

FIGURE IV.6 – PyCharm.

PyCharm est un Environnement de Développement Intégré (IDE) très utilisé pour la programmation en Python. Il offre aux développeurs une gamme complète d'outils et de fonctionnalités qui améliorent leur productivité lorsqu'ils travaillent sur des projets Python.

L'IDE PyCharm propose des fonctionnalités spécifiquement adaptées au développement Python, telles que l'analyse de code, l'autocomplétion intelligente, la navigation dans le code, les capacités de débogage et l'intégration de contrôle de version. Il prend également en charge les frameworks web comme Django et Flask, facilitant ainsi le développement d'applications web.

L'interface de PyCharm permet aux développeurs de personnaliser la disposition des fenêtres, des éditeurs et des panneaux selon leurs préférences. L'éditeur de code puissant offre une coloration syntaxique, un formatage du code et des options de refactorisation pour maintenir un code propre et cohérent.

Le logiciel prend en charge l'exécution et le débogage du code Python, aussi bien en local qu'à distance. Il facilite la gestion des environnements virtuels et l'exécution de tests unitaires. De plus, PyCharm s'intègre aux systèmes de contrôle de version populaires tels que Git, Mercurial et Subversion, ce qui facilite la collaboration et la gestion du contrôle de version.

3.4 Git et Github

3.4.1 Git

FIGURE IV.7 – Git.

Git est un logiciel de gestion de versions décentralisé. C'est un logiciel libre et gratuit, créé en 2005 par Linus Torvalds, auteur du noyau Linux, et distribué selon les termes de la licence publique générale GNU version 2. Le principal contributeur actuel de Git, et ce depuis plus de 16 ans, est Junio C Hamano.

Depuis les années 2010, il s'agit du logiciel de gestion de versions le plus populaire dans le développement logiciel et web, qui est utilisé par des dizaines de millions de personnes, sur tous les environnements (Windows, Mac, Linux)3. Git est aussi le système à la base du célèbre site web GitHub, le plus important hébergeur de code informatique.

3.4.2 Github

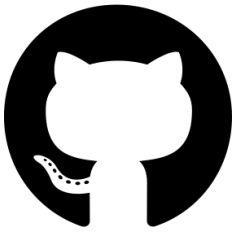


FIGURE IV.8 – Github.

GitHub est une plateforme de développement collaboratif et de gestion de code source basée sur Git. Elle permet aux développeurs de travailler ensemble sur des projets logiciels, de partager leur code, de suivre les modifications, de collaborer et de gérer les versions de leur code.

Voici quelques éléments clés de GitHub :

- Contrôle de version : GitHub utilise Git, un système de contrôle de version distribué, pour permettre aux développeurs de suivre les modifications apportées à leur code. Git enregistre l'historique complet des modifications, ce qui facilite le suivi des modifications, le retour en arrière et la collaboration entre plusieurs personnes.
- Hébergement de code : GitHub héberge les dépôts Git, ce qui signifie que les développeurs peuvent stocker leurs projets et leur code source sur la plateforme. Cela offre un emplacement centralisé pour le code, ce qui permet aux équipes de travailler ensemble plus facilement.
- Collaboration : GitHub facilite la collaboration entre développeurs grâce à des fonctionnalités telles que les demandes de fusion (pull requests), les problèmes (issues)

et les commentaires. Les développeurs peuvent suggérer des modifications de code, examiner les modifications apportées par d'autres membres de l'équipe, signaler des problèmes, discuter des changements et travailler ensemble pour améliorer le projet.

4 Conclusion

Ce chapitre souligne l'importance de la planification et de la gestion de projet dans le développement logiciel. Nous avons examiné la méthodologie en cascade pour les projets avec des exigences stables. Nous avons également exploré des outils tels que Python, CustomTkinter, GraphViz, PyCharm, Git et GitHub pour améliorer la gestion de projet. Une approche méthodique et l'utilisation d'outils adéquats permettent une gestion efficace des projets de développement logiciel.

Chapitre V

Réalisation d'une simulation de la machine de Turing

1 Introduction

Ce chapitre porte sur l'élaboration des différentes algorithme executable par la machine de Turing qui réalise des tâches spécifique. Nous expliqueront la logique du conception d'une algorithme ainsi que la représentaion mathématique des instructions qui constitue une algorithme

2 Algorithmes appliqués

2.1 Algorithme pour additionner deux nombres

Un nombre est représenté en format binaire dans différents automates finis. Par exemple, 5 est représenté par 101. Cependant, dans le cas d'une addition utilisant une machine de Turing, le format unaire est suivi. Dans le format unaire, un nombre est représenté soit par des uns, soit par des zéros. Par exemple, 5 sera représenté par une séquence de cinq zéros ou de cinq uns : 5 = 1 1 1 1 1 ou 5 = 0 0 0 0 0. Utilisons les zéros pour la représentation.

Une machine de Turing peut être conçue pour effectuer une addition en utilisant sa bande pour représenter les nombres à ajouter et ses états pour contrôler le processus d'addition. Voici une description du processus :

- Les deux nombres à additionner sont placés sur la bande, séparés par un symbole blanc.
- La machine de Turing commence au chiffre le plus à gauche du premier nombre et se déplace vers la droite, un chiffre à la fois, en vérifiant la valeur de chaque chiffre.
- Si un chiffre est 0, la machine écrit un 0 à la position correspondante de la somme. Si c'est 1, il écrit un 1.
- Si les deux chiffres sont 1, la machine écrit un 0 et ajoute un report de 1 à la position suivante.
- Lorsque la machine de Turing atteint la fin de l'entrée, elle ajoute tout report restant et écrit le résultat sur la bande.

Pour additionner 2 nombres à l'aide d'une machine de Turing, ces deux nombres sont donnés en entrée à la machine de Turing séparés par un "c".

Exemples - (2 + 3) seront donnés sous la forme 0 0 c 0 0 0 :

```
Entrée : 0 0 c 0 0 0 // 2 + 3
Sortie : 0 0 0 0 0 // 5

Entrée : 0 0 0 0 c 0 0 0 // 4 + 3
Sortie : 0 0 0 0 0 0 0 // 7
```

FIGURE V.1 – Illustration du concept d'additionner de deux nombres.

Approche utilisée

Convertissez un 0 dans le premier nombre en X, puis traversez toute l'entrée et convertissez le premier blanc rencontré en 0. Ensuite, déplacez-vous vers la gauche en ignorant tous les 0 et "c". Venez la position juste à côté de X, et répétez la même procédure jusqu'au moment où nous obtenons un "c" au lieu de X au retour. Convertissez le c en blanc et l'addition est terminée.

Les étapes de déroulement de l'algorithme

Étape 1 : Convertissez 0 en X et passez à l'étape 2. Si le symbole est "c", convertissez-le en blanc (B), déplacez-vous vers la droite et passez à l'étape 6.

Étape 2 : continuez à ignorer les 0 et déplacez-vous vers la droite. Ignorez "c", déplacez-vous vers la droite et passez à l'étape 3.

Étape 3 : continuez à ignorer les 0 et déplacez-vous vers la droite. Convertissez un blanc (B) en 0, déplacez-vous vers la gauche et passez à l'étape 4.

Étape 4 : continuez à ignorer les 0 et déplacez-vous vers la gauche. Ignorez "c", déplacez-vous vers la gauche et passez à l'étape 3.

Étape 5 : continuez à ignorer les 0 et déplacez-vous vers la gauche. Ignorez un X, déplacez-vous à droite et passez à l'étape 1.

Étape-6 : Fin.

Représentation graphique

Le schéma suivant représente les étapes de déroulement de l'algorithme d'une manière graphique, où un cercle représente l'état actuelle, la fléshe représente l'état suivante qui sera atteint si la condition est vérifier

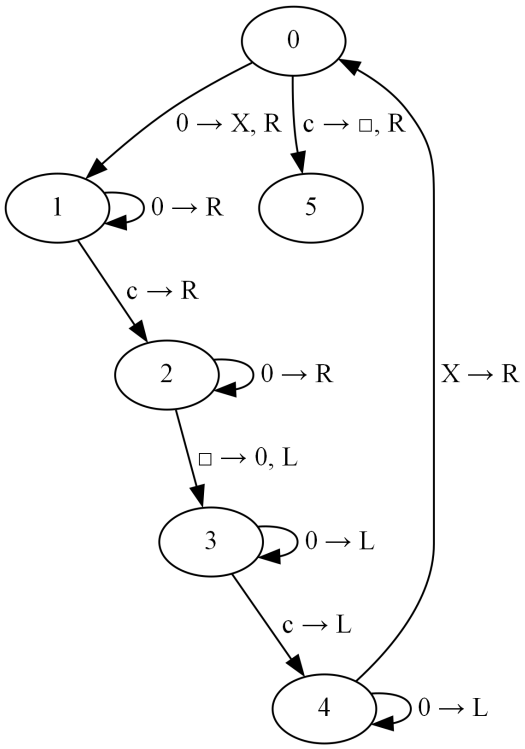


FIGURE V.2 – Représentation graphique de l'algorithme d'addition de deux nombre.

2.2 Algorithme pour inverser une chaîne de caractères

Approche utilisé

Marquer la première entrée en remplaçant le symbole a par 0 et le symbole b par 1. Si un symbole blanc (B) est rencontré, passer à l'état de fin (-3), ensuite se déplacer vers la droite jusqu'à la première case vide (B) et écrire le symbole (). Ensuite, se déplacer vers la gauche. Puis se déplacer vers la gauche jusqu'à rencontrer un symbole 0, 1 ou un symbole blanc (B). Si un symbole X est rencontré, continuer à se déplacer vers la gauche. Si le premier caractère marqué (a ou b) est trouvé, le remplacer par un symbole blanc (B). Si un symbole 0 est trouvé, se déplacer vers la droite jusqu'à la prochaine case vide (B), puis se déplacer vers la gauche jusqu'à trouver le symbole (). Enfin, se déplacer vers la gauche pour revenir à la phase 2. Si un symbole 1 est trouvé, suivre un processus similaire à la phase 3a, mais avec des actions légèrement différentes. Et finalement en passe vers l'étape de nettoyage où on remplacent tous les symboles X par des symboles blancs (B) et continuer à se déplacer vers la droite jusqu'à atteindre le symbole (). Enfin, se déplacer vers la droite pour atteindre l'état de fin (-3), indiquant ainsi la fin de l'algorithme.

Les étapes de déroulement de l'algorithme

- **Étape 1 :** Marquer la première entrée en remplaçant le symbole a par 0 et le symbole b par 1. Si un symbole blanc (B) est rencontré, passer à l'état de fin (-3).
- Étape 2 : Se déplacer vers la droite jusqu'à la première case vide (B) et écrire le symbole . Ensuite, se déplacer vers la gauche.
- **Étape 3 :** Se déplacer vers la gauche jusqu'à rencontrer un symbole 0, 1 ou un symbole blanc (B). Si un symbole X est rencontré, continuer à se déplacer vers la

gauche. Si le premier caractère marqué (a ou b) est trouvé, le remplacer par un symbole blanc (B).

- **Étape 4a :** Si un symbole 0 est trouvé, se déplacer vers la droite jusqu'à la prochaine case vide (B), puis se déplacer vers la gauche jusqu'à trouver le symbole . Enfin, se déplacer vers la gauche pour revenir à la phase 2.
- **Étape 4b :** Si un symbole 1 est trouvé, suivre un processus similaire à la phase 3a, mais avec des actions légèrement différentes.
- **Étape de nettoyage :** Remplacer tous les symboles X par des symboles blancs (B) et continuer à se déplacer vers la droite jusqu'à atteindre le symbole . Enfin, se déplacer vers la droite pour atteindre l'état de fin (-3), indiquant ainsi la fin de l'algorithme.

Représenation graphique

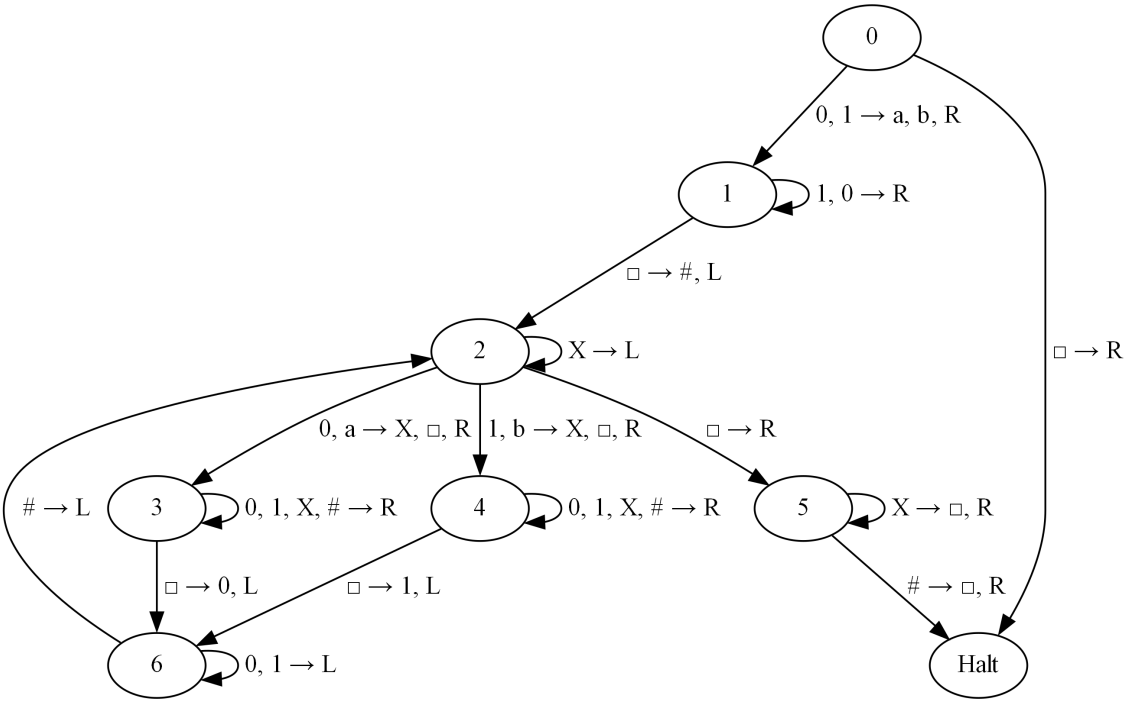


FIGURE V.3 – Représentation graphique de l'algorithme d'inverser une liste d'entrée.

2.3 Algrithme pour le langage L = { $a^n b^m a^{n+m} \mid n, m \geq 1$}

$$L = \{a^n b^m a^{n+m} \mid n, m \geq 1\}$$

représente un type de langage où nous n'utilisons que 2 caractères, c'est-à-dire a et b. La première partie du langage peut être n'importe quel nombre de "a" (au moins 1). La deuxième partie soit un nombre quelconque de "b" (au moins 1). La troisième partie du langage est un nombre de "a" dont le nombre est la somme du nombre de a dans la

première partie de la chaîne et du nombre de b dans la deuxième partie de la chaîne. Toute chaîne de ce type entrant dans cette catégorie sera acceptée par ce langage. Le début et la fin de la chaîne sont marqués par le signe $.

Exemples :

```
Entrée : aabbbaaaaa // n=2, m=3
Sortie : Acceptée

Entrée : aabaaaa // n=2, m=1
Sortie : Non acceptée
```

FIGURE V.4 – Exemple d'un résultat de simulation.

Approche utilisée

Convertissez « a » dans la première partie en « X », puis déplacez-vous vers la droite en ignorant tous les symboles intermédiaires. Lorsque "a" est rencontré juste après "b", convertissez-le en "Z" et déplacez-vous vers la gauche et arrêtez-vous à la position juste à côté de "X". Répétez la procédure ci-dessus.

Lorsque tous les a de la première partie ont été convertis, appliquez le même processus sur la deuxième partie. Convertissez « b » en « Y » et « a » en « Z » dans la troisième partie. Lorsque l'intégralité de la première et de la deuxième partie a été convertie et si la troisième partie est également convertie, la chaîne sera acceptée, sinon non.

Les étapes de déroulement de l'algorithme

- **Étape 0 :** Convertissez « a » en « X », déplacez-vous vers la droite et passez à l'état 1. Si le symbole est « b », ignorez-le, déplacez-vous vers la droite et passez à l'état-4.
- **Étape 1 :** Si le symbole est "a", ignorez-le et déplacez-vous vers la droite, restez dans le même état. Si le symbole est "b", ignorez-le, déplacez-vous vers la droite et passez à l'état-2.
- **Étape 2 :** Si le symbole est "Z", ignorez-le et déplacez-vous vers la droite, restez dans le même état. Si le symbole est "b", ignorez-le et déplacez-vous vers la droite, restez sur le même état et si le symbole est "a", convertissez-le en "Z", déplacez-vous vers la gauche et passez à l'état-3.
- **Étape 3 :** Si le symbole est "Z", ignorez-le et déplacez-vous vers la gauche, restez dans le même état. Si le symbole est "b", ignorez-le et déplacez-vous vers la gauche, restez dans le même état. Si le symbole est "a", ignorez-le et déplacez-vous vers la

gauche, restez dans le même état, et si le symbole est "X", ignorez-le et déplacez-vous vers la droite, passez à l'état-0.

- **Étape 4 :** Si le symbole est "b", ignorez-le et déplacez-vous vers la gauche, passez à l'état 5, et si le symbole est "Z", ignorez-le et déplacez-vous vers la gauche, passez à l'état-5.
- **Étape 5 :** Convertissez « b » en « Y », déplacez-vous vers la droite et passez à l'état 6, et si le symbole est « Z », ignorez-le et déplacez-vous vers la droite, passez à l'état-8.
- **Étape 6 :** Si le symbole est "Z", ignorez-le et déplacez-vous vers la droite, restez dans le même état. Si le symbole est "b", ignorez-le et déplacez-vous vers la droite, restez sur le même état, et si le symbole est "a", convertissez-le en "Z", déplacez-vous vers la gauche et passez à l'état-7.
- **Étape 7 :** Si le symbole est "Z", ignorez-le et déplacez-vous vers la gauche, restez dans le même état. Si le symbole est "b", ignorez-le et déplacez-vous vers la gauche, restez dans le même état, et si le symbole est "Y", ignorez-le et déplacez-vous vers la droite, passez à l'état-5.
- **Étape 8 :** Si le symbole est "Z", ignorez-le et déplacez-vous vers la droite, restez dans le même état, et si le symbole est "$", ignorez-le et déplacez-vous vers la gauche, passez à l'état-9.
- **Étape 9 :** Chaîne *ACCEPTÉE*.

Représentation graphique

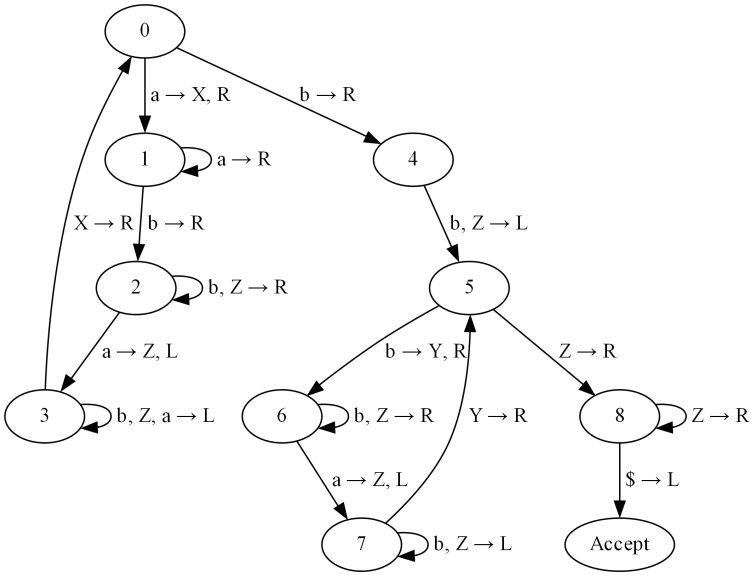


FIGURE V.5 – Représentation graphique de l'algorithme pour un langage de la concaténation équilibrée.

2.4 Algorithme pour le language L = { $0^n1^n2^n \mid n \geq 1$}

$$L = \{0^n1^n2^n \mid n \geq 1\}$$

représente un type de langage où nous n'utilisons que 3 caractères, c'est-à-dire 0, 1 et 2. Au début, le langage a un certain nombre de 0 suivis d'un nombre égal de 1, puis suivis d'un nombre égal de 2. Toute chaîne de ce type entrant dans cette catégorie sera acceptée par ce langage. Le début et la fin de la chaîne sont marqués par le signe $.

Approche utilisée

Remplacez d'abord un 0 de l'avant par X, puis continuez à vous déplacer vers la droite jusqu'à ce que vous trouviez un 1 et remplacez ce 1 par Y. Encore une fois, continuez à vous déplacer vers la droite jusqu'à ce que vous trouviez un 2, remplacez-le par Z et déplacez-vous vers la gauche. Maintenant, continuez à vous déplacer vers la gauche jusqu'à ce que vous trouviez un X. Lorsque vous l'avez trouvé, déplacez-vous vers la droite, puis suivez la même procédure que ci-dessus.

Une condition survient lorsque vous trouvez un X immédiatement suivi d'un Y. À ce stade, nous continuons à nous déplacer vers la droite et continuons à vérifier que tous les 1 et 2 ont été convertis en Y et Z. Sinon, la chaîne n'est pas acceptée. Si nous atteignons $, la chaîne est acceptée.

Les étapes de déroulement de l'algorithme

- **Étape-1 :**

  Remplacez 0 par X et déplacez-vous vers la droite, passez à l'état Q1.

- **Étape 2 :**

  Remplacez 0 par 0 et déplacez-vous vers la droite, Restez dans le même état.

  Remplacez Y par Y et déplacez-vous vers la droite, Restez sur le même état.

  Remplacez 1 par Y et déplacez-vous vers la droite, passez à l'état Q2.

- **Étape 3 :**

  Remplacez 1 par 1 et déplacez-vous vers la droite, restez dans le même état.

  Remplacez Z par Z et déplacez-vous vers la droite, restez sur le même état.

  Remplacez 2 par Z et déplacez-vous vers la droite, passez à l'état Q3.

- **Étape 4 :**

  Remplacer 1 par 1 et déplacer vers la gauche, Rester sur le même état.

  Remplacer 0 par 0 et déplacer vers la gauche, Rester sur le même état.

  Remplacer Z par Z et déplacer vers la gauche, Rester sur le même état.

  Remplacer Y par Y et déplacer vers la gauche, Rester sur même état

  Remplacez X par X et déplacez-vous vers la droite, passez à l'état Q0.

- **Étape 5 :**

  Si le symbole est Y, remplacez-le par Y et déplacez-vous vers la droite et passez à l'état Q4

  Sinon, passez à l'étape 1

- **Étape 6 :**

  Remplacez Z par Z et déplacez-vous vers la droite, restez dans le même état Remplacez Y par Y et déplacez-vous vers la droite, restez sur le même état

Représentation graphique

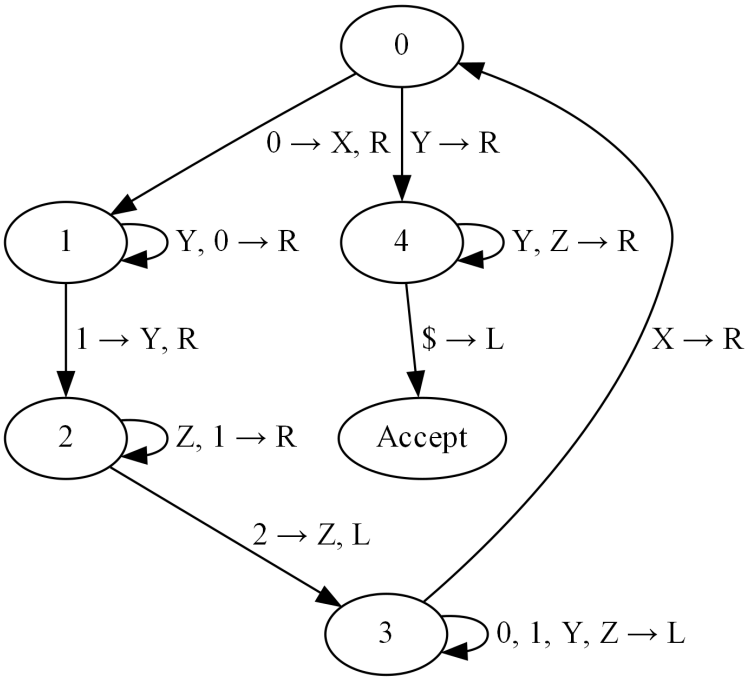


FIGURE V.6 – Représentation graphique de l'algorithme pour le langage

3 Elaboration de l'interface graphique et fonctionnement de l'application

3.1 Aperçu général

Ce simulateur de machine de Turing offre toutes les fonctionnalités d'une machine de Turing bidirectionnelle à bande infinie, à ceci près que la machine simulée ne dispose pas d'une bande infiniment longue (bien qu'elle soit suffisamment longue pour tous les usages pratiques). Cela est dû aux limites fondamentales de la mémoire des ordinateurs. Outre la simulation d'une machine de Turing, ce programme est également capable de simuler une machine de Turing unidirectionnelle à bande infinie. Cette machines peut être visualisées à la fois comme des bandes simulées et comme des textes fournissant des informations pour chaque étape. Dans le programme lui-même, l'utilisateur peut écrire les différents états et changements d'état que la machine doit suivre. Ces textes peuvent ensuite être

sauvegardés sous forme de fichiers .tm et chargés dans le simulateur pour être utilisés ultérieurement.

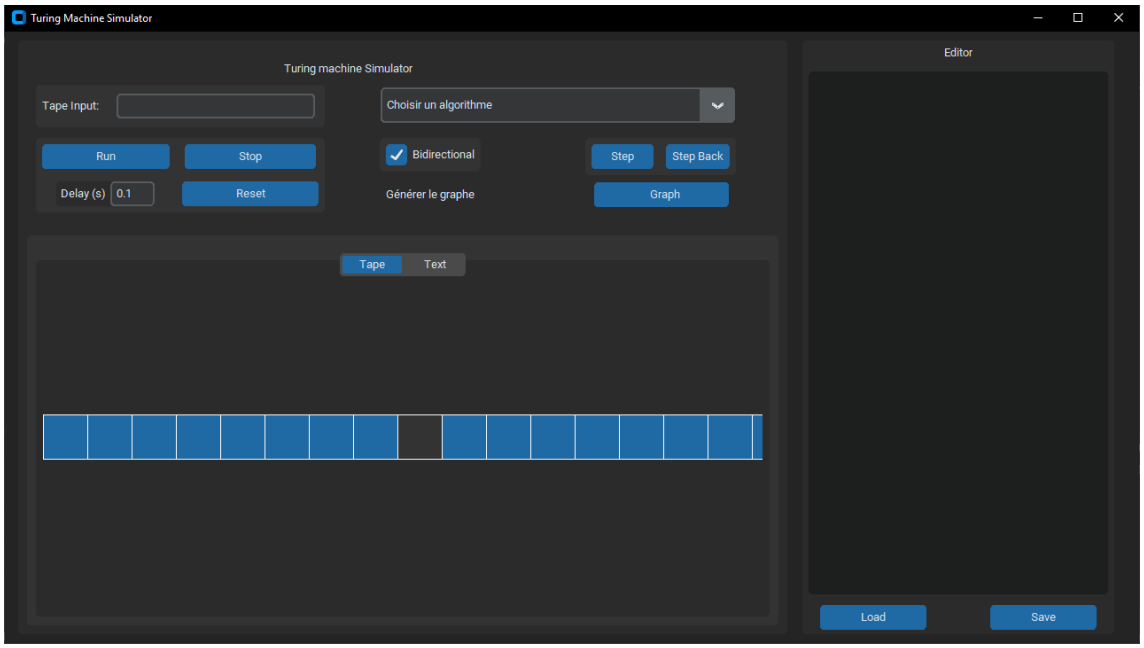


FIGURE V.7 – Simulateur au démarrage.

3.2 L'éditeur de text

La fenêtre de l'éditeur est située à droite de la fenêtre du programme. C'est ici que nous pouvons charger, écrire et enregistrer des fichiers .tm qui défieront les fonctionnalités de notre machine de Turing.

3.2.1 Le fichier .tm

Le fichier avec l'extension ".tm" représente généralement un fichier de description de machine de Turing. Il est utilisé pour définir le comportement et la structure d'une machine de Turing. Il contient des informations sur la machine de Turing, telles que l'ensemble des états, l'alphabet de la bande, les règles de transition, l'état initial et les états finaux. Il sert de représentation textuelle de la configuration de la machine de Turing.

3.2.2 Création d'un fichier .tm

Les instructions pour la machine de Turing simulée sont écrites sous forme de fichier .tm. Ceux-ci peuvent être créés soit en utilisant l'éditeur de ce programme, soit en créant fichier texte ordinaire et en l'enregistrant avec l'extension .tm. Les instructions sont écrites au format suivant :

- Les états sont toujours des entiers avec 0 étant l'état initial.
- -1 désigne un état d'arrêt et d'acceptation, -2 est un état d'arrêt et de rejet et -3 est un état d'arrêt général. Tous les autres numéros d'État doivent être non négatifs.

- L dit à la machine de se déplacer à gauche et R lui dit de se déplacer à droite. De plus, si vous écrivez une machine de Turing à deux bandes, S indique à la machine de rester là où elle se trouve. S ne peut être utilisé que pour deux machines de turing à bande.
- Un " B " majuscule représente le symbole vide.
- Pour une machine de Turing à bande unique, chaque ligne du fichier .tm prend la forme :

  ÉTAT SYMBOLE NOUVEAU-ÉTAT NOUVEAU-SYMBOLE DIRECTION

  Donc l'expression

  2 b 5 c L

  indique à la machine que si elle est dans l'état 2 et lit le symbole b, remplacez le b par un c, déplacez-vous vers la gauche et passez à l'état 5.
- Pour inclure des commentaires, on précède chaque ligne de commentaire par un .

3.2.3 Sauvegarde d'un fichier .tm

Après qu'on a déterminé les instructions que notre machine va suivre, on peut maintenant les sauvegarder sur notre ordinateur pour les utiliser à un autre moment, pour se faire :

- Nous devons nous assurer que les instructions suivent exactement le format décrit ci-dessus. Bien que tout ce que nous écrivons dans la fenêtre de l'éditeur puisse être enregistré en tant que fichier. tm, s'il ne suit pas le format approprié, le simulateur ne pourra pas s'exécuter.
- On clique sur le bouton Enregistrer situé en bas à droite de la section éditeur
- Après, on choisie le répertoire dans lequel nous souhaitons enregistrer le fichier, indiquez un nom de fichier, puis cliquer sur Enregistrer.

Cela créera un fichier. tm dans le répertoire donné qui pourra être utilisé ultérieurement.

3.2.4 Chargement d'un fichier .tm

Pour charger un fichier .tm créé précédemment :

- On clique sur le bouton chargé situé en bas à gauche de la section éditeur.
- Accédez à notre fichier, sélectionnez-le et l'ouvrir.

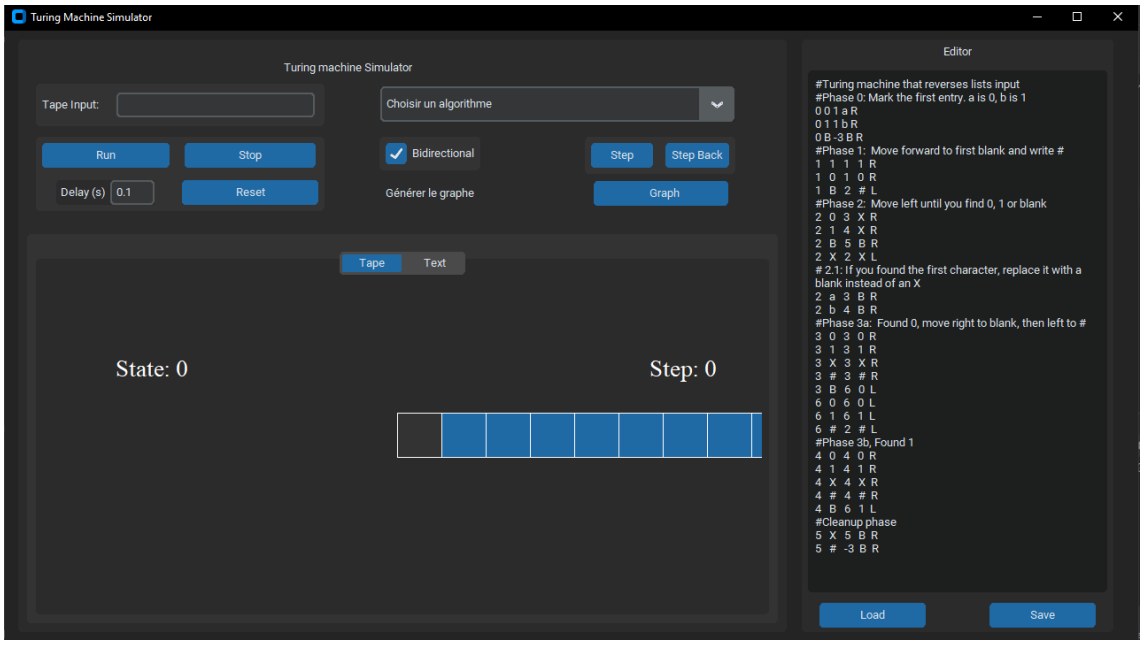


FIGURE V.8 – Simulateur avec fichier .tm chargé.

Nous devrions maintenant voir la liste des instructions pour notre machine de Turing dans la fenêtre de l'éditeur. Nous pouvons ensuite les modifier comme nous le voyons adéquat et les enregistrer comme tout texte qui est écrit dans la fenêtre de l'éditeur.

3.3 Simulateur

La fenêtre du simulateur est située sur le côté gauche de la fenêtre du programme. C'est ici que nous pouvons constater une visualisation de notre machine de Turing fonctionnant à la fois graphiquement et sous forme de texte.

3.3.1 Lecture du simulateur

Dans ce programme, nous pouvons simuler la machine de Turing de deux manières. Il s'agit soit d'une représentation graphique du ruban, soit d'un texte. À tout temps d'exécution des machines, vous pouvez passer d'une simulation de la machine à l'autre comme une vraie cassette et la voir comme du texte. Cela se fait en basculant entre les onglets "tape" et "texte" situés en haut au centre de la fenêtre du simulateur.

Lors de la simulation de la machine en tant que bande :

- L'état actuel et l'étape sont situés au-dessus de la bande elle-même.
- La cellule en cours de lecture par la machine est la cellule grise

Lors de la simulation de la machine sous forme de texte :

- Chaque étape est écrite sous la forme d'un morceau de texte avec l'étape en cours, l'état actuel et ce qui est actuellement écrit sur la bande.
- La lettre avec un signe d'insertion en dessous est celle lue par la machine

Bien que l'onglet "bande" soit utile pour obtenir une compréhension visuelle du fonctionnement de notre machine de Turing, nous recommandons de suivre toute analyse réelle dans l'onglet "texte" car nous somme limité à ne voir qu'une section de la bande entière dans l'onglet "bande" alors que nous pouvons facilement faire défiler toutes les étapes dans l'onglet "texte".

3.3.2 Exécution d'une machine de Turing

Afin de faire fonctionner une machine de Turing, nous procédons comme suit :

1. Chargez un fichier. tm dans l'éditeur ou enregistrez ce que vous avez écrit sur votre machine. Chaque fois qu'une machine est chargée dans le simulateur, vous verrez alors une étiquette "state" et "step" apparaître dans la fenêtre tape and text.

2. Tapez dans la zone de texte intitulé "Tape Input" située en haut au centre de la fenêtre du simulateur ce que nous voulons que l'entrée de bande initiale soit.

3. Entrez le délai souhaité entre chaque étape dans la zone de texte intitulée "Delay" située en bas à gauche de la fenêtre du simulateur. Le délai par défaut est de 0,1 s

4. Cliquez sur le bouton Exécuter situé directement au-dessus de la zone de texte "Delay".

5. Si vous souhaitez arrêter la course à tout moment, appuyez sur le bouton "Stop" situé à droite du bouton "Run". Si vous souhaitez réinitialiser la machine, cliquez sur le bouton "Réinitialiser" situé directement sous le bouton "Stop".

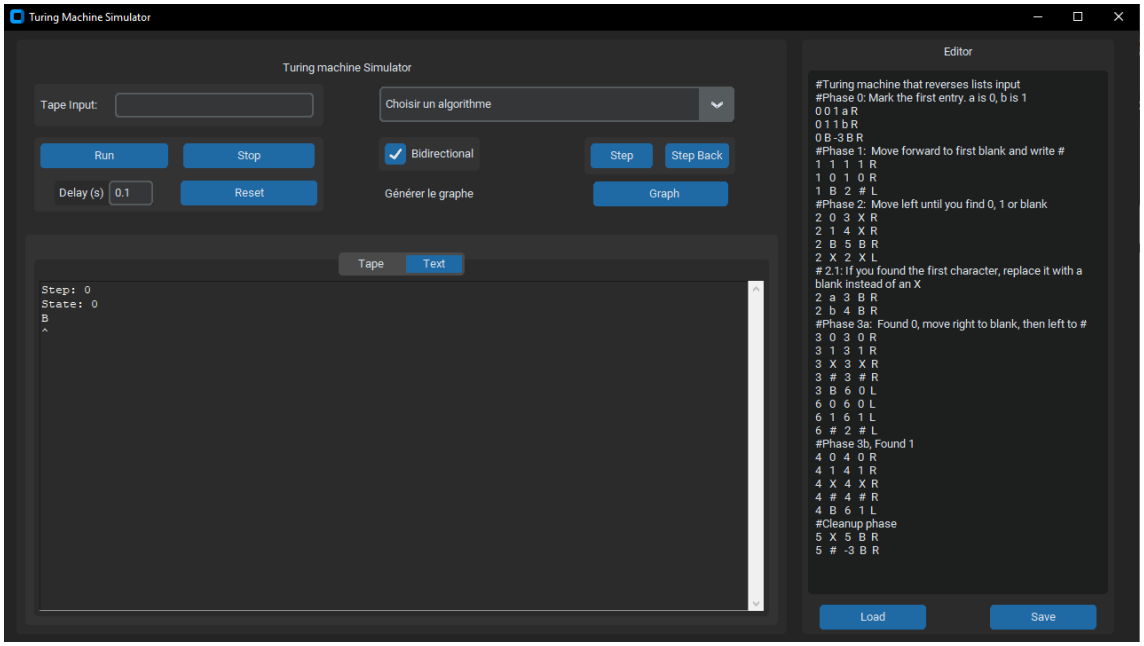


FIGURE V.9 – Simulateur dans l'onglet texte.

3.3.3 Passer étape par étape à travers l'éxecution

Si nous souhaitons avoir plus de contrôle sur le fonctionnement de notre machine de Turing, nous pouvons passer étape par étape en utilisant les boutons "Step" et "Step Back" situés en haut à droite de la fenêtre du simulateur. Notez que si nous appuyons sur le bouton" Reculer " à l'étape 0, rien ne se produira. Si nous appuyons sur le bouton" Step " après l'arrêt de la machine, nous ajouterons des étapes supplémentaires, mais la machine restera à l'arrêt. Ces fonctionnalités peuvent être utilisées conjointement avec les fonctionnalités "Run" et "Stop", comme prévu.

3.3.4 Problème de l'arrêt

En raison des limitations fondamentales des machines de Turing et du calcul, il est impossible de savoir si notre machine de Turing s'arrêtera un jour. C'est le problème de l'arrêt et est discuté dans le chapitre ci-dessus. Si nous exécutons une machine de Turing qui n'arrête jamais le programme ne répondra probablement plus. À ce stade, nous vous recommandons de fermer et de redémarrer le programme. Alternativement, pour aider à résoudre ce problème, nous avons défini le nombre maximum d'étapes autorisé pour une exécution donnée à 200 000 étapes. Une fois que le programme a déterminé toutes ces étapes, il devrait répondre à nouveau et afficher l'exécution de 200 000 étapes. Cela peut prendre un certain temps, donc redémarrer le programme est probablement notre meilleure option.

3.4 Génération du graphe

En plus de simuler une machine de Turing, nous avons inclus un programme qui convertit automatiquement un chier. tm en un graphe orienté facile à suivre. Ceci est utile si l'utilisateur souhaite avoir une représentation visuelle de la machine de Turing sans avoir à tester différentes chaînes d'entrée.

Cette fonctionnalité a été développé dans le seul but de représenter graphiquement les fichiers .tm associés aux machines de Turing à bande unique.

4 Aperçu technique

Le code du simulateur est divisé en deux classes. La première de ces classes est *turing machines.py*. Cette classe sert de backend pour le simulateur. C'est ici que le code réel pour simuler une machine de Turing est contenu. L'autre classe est *TMGUI.py*. Cette classe est une interface graphique construite à l'aide de la bibliothèque Python *CustomTkinter* et permet à l'utilisateur d'interagir réellement avec le programme.

Le code graphique utilise *GraphViz*, un programme de génération de graphes orientés, afin de convertir un fichier .tm en un graphe orienté des transitions d'état. Ce code a été initialement développé afin de fournir une belle animation pour les machines de Turing à inclure dans le simulateur lui-même.

Problèmes détectés

Le seul problème connu à l'heure actuelle est que le programme commence à prendre du retard si une entrée de bande trop importante est entrée, ou si le programme est exécuté pendant une longue période sur de nombreuses machines ou sur de nombreuses entrées différentes. On ne sait pas si cela est dû à une fuite de mémoire dans notre programme ou à une limitation de la part de Python, bien que des efforts considérables aient été déployés pour atténuer ce problème. Le redémarrage du programme résout ce décalage.

Conclusion

Ce projet a permis de mettre en lumière l'importance de la machine de Turing dans le domaine de la science informatique. En définissant un modèle abstrait de calcul, cette machine a jeté les bases théoriques pour l'étude des algorithmes, de la calculabilité et de la complexité. Son influence sur l'architecture des ordinateurs modernes et son rôle déterminant dans la conception des premiers systèmes informatiques sont indéniables.

La réalisation de notre simulateur de la machine de Turing à l'aide de Python a permis de mieux comprendre son fonctionnement et ses composantes essentielles. À travers un exemple concret, nous avons pu illustrer son utilisation et son application pratique. La planification et la gestion du projet ont été abordées, mettant en évidence les différentes étapes et méthodologies mises en œuvre pour mener à bien cette simulation.

Grâce à ce projet, nous avons pu approfondir nos connaissances en informatique théorique et en programmation, tout en acquérant des compétences pratiques dans le développement d'un simulateur. Les choix technologiques effectués ont permis d'obtenir des résultats satisfaisants, et nous avons pu constater l'efficacité de ce modèle abstrait dans la résolution de problèmes.

Ce Projet de Fin d'Étude a été une expérience enrichissante qui a renforcé notre compréhension de la machine de Turing et de son impact sur le domaine de la science informatique. Nous espérons que ce rapport servira de ressource utile pour ceux qui souhaitent approfondir leurs connaissances sur ce sujet fondamental.

Webographie

- **https ://plato.stanford.edu/entries/turing-machine/** (consulté le 23/01/2023)
- **https ://www.geeksforgeeks.org/turing-machine-in-toc/** (consulté le 28/01/2023)
- **https ://courses.engr.illinois.edu/cs374/sp2021/scribbles/B-2021-02-23.pdf** (consulté le 28/01/2023)
- **https ://erik-engheim.medium.com/turing-machines-for-dummies-81e8e25471b2** (consulté le 29/01/2023)
- **https ://youtu.be/PvLaPKPzq2I** (consulté le 30/01/2023)
- **https ://turingmachinesimulator.com/** (consulté le 02/02/2023)
- **https ://docs.python.org/3/** (consulté le 05/02/2023)
- **https ://customtkinter.tomschimansky.com/documentation/** (consulté le 06/03/2023)
- **https ://graphviz.org/** (consulté le 10/04/2023)
- **https ://stackoverflow.com/** (consulté le 15/05/2023)