UNIVERSITÉ DU QUÉBEC À MONTRÉAL

# LES GRANDS MODÈLES DE LANGAGE PEUVENT-ILS COMPRENDRE LES AUTOMATES ?

MÉMOIRE

PRÉSENTÉ

COMME EXIGENCE PARTIELLE

DE LA MAÎTRISE EN INFORMATIQUE

PAR

AHMED AZIZ ATTIA

AOÛT 2025

## REMERCIEMENTS

Ce mémoire marque l'aboutissement d'un chapitre fondamental de mon parcours académique et personnel. Il n'aurait jamais pu voir le jour sans le soutien précieux et constant de nombreuses personnes, que je souhaite remercier du fond du cœur.

Tout d'abord, je tiens à exprimer ma profonde gratitude à mes parents, restés en Tunisie, dont l'amour inconditionnel, les sacrifices silencieux et les encouragements constants ont toujours été ma plus grande source de force.

Je remercie également ma famille ici au Canada, en particulier mon oncle et sa famille, pour leur accueil bienveillant, leur hospitalité et leur présence rassurante au quotidien. Leur soutien affectif a rendu cette aventure loin de chez moi beaucoup plus douce.

Je tiens à adresser ma reconnaissance la plus sincère à mon directeur de recherche, pour son encadrement rigoureux, ses conseils éclairés et sa confiance indéfectible. Sa rigueur scientifique, sa disponibilité et son sens du détail ont profondément enrichi ma réflexion et guidé ce travail vers la maturité.

Un merci sincère à mes amis, qui m'ont accompagné avec patience, humour et solidarité à travers les hauts et les bas de cette période de recherche. Leurs conseils, leurs encouragements et leur simple présence ont souvent suffi à redonner de l'énergie lorsque les moments de doute surgissaient.

Je remercie aussi chaleureusement l'Université du Québec à Montréal (UQAM) pour m'avoir offert un environnement d'apprentissage stimulant, riche et accueillant. Le dynamisme de ses équipes, la diversité de sa communauté et l'accessibilité de ses ressources ont contribué à faire de mon séjour académique une expérience humaine et intellectuelle marquante.

Enfin, à toutes celles et ceux qui, de près ou de loin, ont contribué à la réalisation de ce mémoire enseignants, collègues, personnel administratif ou soutiens de l'ombre, je vous adresse mes plus sincères remerciements. Ce mémoire vous appartient un peu à toutes et à tous.

Ahmed Aziz Attia

**TABLE DES MATIÈRES**

**TABLE DES FIGURES**

**LISTE DES TABLEAUX**

## RÉSUMÉ

L'émergence récente des modèles de langage de très grande taille (LLM) tels que GPT-4, DeepSeek, Claude 3, Gemini ou LLaMA 3 bouleverse l'automatisation des activités de production logicielle, depuis la rédaction de spécifications jusqu'aux étapes de vérification et de test. Dans ce contexte, les *automates finis* demeurent un formalisme central : ils décrivent les comportements séquentiels de nombreux composants logiciels, et ramènent les tests fonctionnels à la simple vérification de l'acceptation d'une chaîne d'entrée.

Cette étude interroge les capacités de ces modèles à *raisonner symboliquement* sur des automates finis. Deux problèmes fondamentaux sont évalués : (i) la **vérification d'acceptation**, consistant à déterminer si un automate accepte une chaîne donnée ; (ii) la **génération d'une séquence acceptante minimale**, qui requiert la production d'une chaîne de longueur minimale menant vers un état d'acceptation. Nous combinons pour cela cinq formats de représentation (Liste de transitions, Liste d'adjacence, DOT, Langage naturel contrôlé, et un format hybride) avec quatre stratégies de *prompting* (Question fermée, Approche normale, Approche assistée et Approche générique), explorant l'effet de ces choix sur les performances des LLM.

Trois axes complémentaires structurent notre analyse expérimentale : la complexité structurelle des automates (Nombre d'états), la richesse de leur alphabet, et la longueur des chaînes d'entrée à traiter. Cette combinaison permet d'évaluer la robustesse des modèles face à des variations de structure, de contenu symbolique et de profondeur séquentielle, dans des conditions proches des applications réelles.

Les résultats font apparaître des tendances robustes : les formats textuels, en particulier ceux encadrés par une approche assistée, obtiennent la meilleure précision pour la vérification d'acceptation. En revanche, la génération de la chaîne minimale reste extrêmement dépendante de la stratégie de prompt adopté. On observe que certains modèles, bien qu'ayant correctement identifié les transitions intermédiaires, échouent à synthétiser une réponse minimale, révélant une difficulté persistante dans le suivi des dépendances longues et la planification séquentielle.

En termes de temps de calcul, la vérification d'acceptation nécessite entre 9 et 10 heures selon le nombre d'états, et entre 15 et 16 heures pour les alphabets élargis. La génération de séquences minimales dure de 12 à 13 heures sur les automates de tailles croissantes, et jusqu'à 20 heures avec de grands alphabets. Le benchmark sur la longueur des chaînes présente un coût supplémentaire, entre 11 et 13 heures, la complexité augmentant proportionnellement avec la longueur des séquences à traiter.

En synthèse, nos observations offrent une cartographie fine des capacités et limites actuelles des LLMs dans des tâches formelles impliquant des automates finis, et fournissent des repères méthodologiques pour sélectionner un format de représentation et une stratégie de *prompt engineering* adaptés aux exigences de la vérification logicielle.



**Mots-clés :**

Grands modèles de langage (LLMs), Théorie des automates, Vérification formelle, Génération de séquences, Ingénierie des prompts, Raisonnement symbolique, Représentations structurées.

## INTRODUCTION

La montée en puissance des grands modèles de langage (Large Language Models, LLMs), tels que GPT-4 (OpenAI, 2023), Gemini (DeepMind, 2023), GitHub Copilot (GitHub, 2023), LLaMA (AI, 2023b), et d'autres, a révolutionné le domaine de l'intelligence artificielle. Entraînés sur des quantités massives de données textuelles, ces modèles ont montré des capacités remarquables en compréhension du langage naturel, en génération et en interprétation textuelle.

Les LLMs se distinguent aussi par leur polyvalence, capables de traiter des tâches variées comme la traduction, le résumé, le raisonnement logique ou la programmation assistée (Brown *et al.*, 2020a). Fondés sur des architectures de type transformeur exploitant l'attention pour modéliser des dépendances complexes à longue portée (Vaswani *et al.*, 2017), ils se révèlent très puissants, mais rencontrent encore des défis majeurs (Chen *et al.*, 2021) dans la manipulation de structures mathématiques formelles et l'exécution de raisonnements symboliques rigoureux (Marcus, 2020).

Les structures formelles comme les automates finis déterministes (en anglais Deterministic Finite Automata ou DFA) (Hopcroft *et al.*, 2001), qui reposent sur des définitions mathématiques précises et des propriétés algébriques bien définies, constituent donc un excellent terrain d'évaluation pour ces capacités (Weiss *et al.*, 2021). Les DFA offrent un cadre rigoureux pour modéliser les comportements séquentiels (Kuhlmann *et al.*, 2011). Grâce à leur formalisme clair et leurs propriétés algébriques robustes, les automates constituent un excellent banc d'essai pour évaluer si les LLMs peuvent internaliser, manipuler et raisonner sur des structures formelles. Il convient de noter que, bien que divers types d'automates existent tels que les automates non-déterministes (NFA), les automates à pile (pour les langages context-free) et même les machines de Turing notre étude se concentre sur les automates finis, et plus particulièrement sur les automates finis déterministes (DFA). Plusieurs raisons motivent ce choix :

1. **Simplicité et clarté mathématique :** Les DFA possèdent une définition formelle très concise, ce qui permet d'isoler précisément l'aspect du raisonnement symbolique sans la complexité supplémentaire induite par la non-déterminisme ou les mécanismes de mémoire utilisés dans les automates à pile et les machines de Turing.
2. **Équivalence théorique avec les NFA :** Bien que les NFA soient théoriquement équivalents aux DFA en termes de puissance d'expression, le comportement déterministe des DFA facilite l'analyse et la

comparaison des performances des LLMs sur des tâches de reconnaissance et de génération, réduisant ainsi l'ambiguïté dans l'extraction des transitions.

3. **Pertinence pratique :** Les automates finis sont largement utilisés dans la conception de compilateurs, l'analyse syntaxique et la vérification formelle. Leur utilisation dans des applications concrètes permet de mieux transposer les résultats expérimentaux de notre étude aux scénarios réels.
4. **Limitations en matière de raisonnement symbolique :** En se concentrant sur les DFA, nous pouvons examiner de manière rigoureuse et ciblée comment les LLMs traitent un cadre formel rigoureux, ce qui serait davantage embrouillé dans des modèles plus puissants (par exemple, les automates à pile ou les machines de Turing) où des mécanismes de mémoire additionnels interviennent.

Pour un approfondissement théorique, le Chapitre 1 présente l'évolution historique, les concepts fondamentaux et les applications pratiques de la théorie des automates.

Les DFAs constituent la pierre angulaire de multiples systèmes critiques (Clarke *et al.*, 2018), (Angluin, 1987) :

**Vérification formelle :**

- Model checking temporel (LTL/CTL) via la construction d'automates de Büchi (Tripakis, )
- Vérification d'invariants par analyse d'accessibilité (algorithme de Tarjan (Tarjan, 1972))
- Synthèse de contrôleurs réactifs (théorie des jeux à états finis (Pnueli et Rosner, 1989))

**Ingénierie des langages :**

- Analyse lexicale par DFAs déterministes optimisés (algorithme de Hopcroft (Hopcroft *et al.*, 2001))
- Implémentation efficace d'expressions régulières (compilation vers DFA via NFA (Thompson, 1968))
- Validation syntaxique de documents XML/JSON par automates à piles (Alur et Madhusudan, 2004)

**Analyse de données :**

- Reconnaissance de motifs dans les séries temporelles (algorithme de Viterbi (Lou, 1995))
- Filtrage de flux réseau avec automates temporisés (modèles de Alur-Dill (Alur et Dill, 1994))
- Apprentissage automatique de modèles d'états finis (méthode d'Angluin (Angluin, 1987))

**Les défis contemporains incluent :**

- L'extension aux systèmes hybrides (combinaison états discrets/continus (Henzinger, 1996))
- La gestion de l'aléatoire (automates probabilistes de Rabin (Rabin, 1963))
- L'optimisation énergétique des implémentations matérielles (DFAs asynchrones (Diekert et Muscholl, 2011))

Cette théorie demeure active, comme en témoignent les travaux récents sur les automates quantiques (Deutsch *et al.*, 2021) ainsi que les applications en biologie computationnelle, avec des modèles récents pour l'analyse de la migration collective des cellules (Vayadande *et al.*, 2022).

Cette sélection est particulièrement pertinente car ces structures constituent un pont entre la théorie des langages formels et de nombreuses applications pratiques, tout en présentant différents niveaux de complexité analytique selon leur classe (déterministes, non-déterministes, avec epsilon-transitions, etc.) (Hopcroft *et al.*, 2001). Les automates finis sont des modèles fondamentaux en informatique théorique. Utilisés dans des domaines variés comme la vérification de logiciels, la reconnaissance de motifs, la linguistique computationnelle, l'analyse lexicale des compilateurs, la conception de circuits électroniques, les systèmes embarqués, la bioinformatique, et la modélisation de protocoles réseau, ils jouent un rôle clé dans la modélisation de systèmes complexes. Néanmoins, leur nature formelle et structurée représente un défi pour les LLMs, qui sont principalement conçus pour manipuler des textes en langage naturel. Cette recherche vise donc à explorer si les LLMs peuvent comprendre et manipuler ces structures de manière efficace, notamment dans des contextes où la précision formelle et la rigueur mathématique sont essentielles. Les automates finis, par leur nature à la fois simple (représentation graphique compacte) et riche (propriétés algébriques non-triviales), offrent un cadre d'analyse privilégié pour évaluer la capacité des LLMs à internaliser des structures discrètes régies par des règles formelles (Bubeck *et al.*, 2023). De plus, les automates finis, avec leurs états discrets et transitions bien définies, constituent un cadre idéal pour tester les limites du raisonnement symbolique des modèles de langage actuels. Ces limites sont particulièrement importantes à explorer à l'heure où les LLMs sont de plus en plus utilisés comme assistants dans l'enseignement des sciences informatiques et comme outils d'aide à la programmation dans des environnements industriels où la fiabilité et la correction formelle sont critiques (Bengio, 2024).

## 0.1 Problématique et Questions de Recherche

L'objectif principal de ce mémoire est d'analyser la capacité des LLMs à interpréter des automates finis à partir de différentes représentations (textuelles et formelles) et à résoudre des problèmes fondamentaux liés à ces structures. Plus précisément, nous posons les questions suivantes :

- Dans quelle mesure un LLM peut-il identifier les états et les transitions d'un automate à partir d'une description textuelle ?
- Est ce que les LLMs peuvent comprendre le probléme d'acceptance de séquence et la génération

d'une séquence acceptante dans le contexte des automates ?

— Peut-il déterminer si un mot donné est accepté par un automate en se basant uniquement sur sa représentation textuelle ?
— Comment la performance varie-t-elle en fonction du format de représentation de l'automate (tableaux, graphes, descriptions formelles) ?
— Existe-t-il des stratégies de requêtage (*prompt engineering*) permettant d'améliorer la précision des réponses des LLMs ?

## 0.2 Objectifs et Contributions

Cette étude se propose d'apporter plusieurs contributions significatives :

1. Développement d'un protocole expérimental rigoureux permettant d'évaluer la compréhension des automates par les LLMs.
2. Comparaison de plusieurs formats de représentation des automates et de leur impact sur la performance des modèles.
3. Proposition de stratégies de requêtage optimisées pour améliorer l'interprétation des structures formelles par les LLMs.
4. Analyse des limites des LLMs et des perspectives d'amélioration pour leur application dans des tâches de raisonnement formel.

Les contributions de cette étude apportent une perspective nouvelle en isolant les aspects de compréhension formelle des automates et en quantifiant l'influence des représentations sur les performances des LLMs.

## 0.3 Méthodologie

Notre méthodologie s'articule autour de plusieurs axes expérimentaux :

1. **Génération d'automates finis** : Nous concevons des automates aléatoires de complexité variable, que nous traduisons ensuite en plusieurs formats représentatifs (texte libre, structures tabulaires ou graphiques).
2. **Génération de cas de test** : Pour chaque automate, nous générons des chaînes d'entrée et des séquences candidates, que nous validons par simulation afin de disposer de résultats de référence pour l'évaluation.

3. **Stratégies de requêtage** : Nous explorons plusieurs styles de prompt engineering, allant de requêtes simples à des approches guidées ou hybrides, pour analyser l'effet du cadrage sur la performance des LLMs.
4. **Évaluation et analyse** : Les réponses des LLMs sont comparées aux résultats attendus, et des métriques telles que le taux de réussite et la marge d'erreur sont calculées. L'ensemble des expériences est répété sur de multiples itérations pour garantir la robustesse des résultats. Afin de confirmer la significativité des écarts entre formats, nous appliquons également des **analyses statistiques** : une *ANOVA à un facteur* pour détecter les différences globales, suivie d'un *test post-hoc de Tukey* pour identifier les comparaisons significatives.

Ce protocole expérimental nous permet de quantifier précisément l'impact de la représentation et du style de requêtage sur la capacité des LLMs à simuler des transitions d'états dans les automates finis.

## 0.4 Organisation du Document

Le présent document est structuré de la manière suivante :

— **Chapitre 1 : État de l'Art** – Revue des travaux antérieurs portant sur les LLMs, le prompt engineering et la théorie des automates, avec une attention particulière aux limites identifiées dans la compréhension des structures formelles.

— **Chapitre 2 : Méthodologie** – Description détaillée de notre protocole expérimental, incluant la génération d'automates, les différents formats de représentation, les stratégies de requêtage, et les cas de test utilisés.

— **Chapitre 3 : Résultats Expérimentaux** – Présentation et analyse des résultats obtenus, avec des comparaisons entre les différentes approches et formats de représentation.

— **Chapitre 4 : Discussion et Perspectives** – Discussion des implications de nos résultats, des limites rencontrées et des pistes de recherche futures.

Cette introduction a ainsi présenté le contexte, la motivation et la méthodologie de notre étude, tout en positionnant notre travail comme une contribution novatrice à l'intersection entre les LLMs et la théorie des automates. Elle sert de fondement pour l'exploration détaillée des capacités déductives des LLMs face à des tâches formelles, et souligne l'importance d'un prompt engineering efficace pour surmonter les défis posés par les représentations structurées.

# CHAPITRE 1
# ÉTAT DE L'ART

## 1.1 Les Fondements de la Théorie des Automates

### 1.1.1 Historique et Concepts Clés

La théorie des automates constitue l'un des piliers de l'informatique théorique. Ses origines remontent aux premières tentatives de formaliser le concept de calcul. Dès les travaux pionniers d'Alan Turing (Turing *et al.*, 1936), qui introduit en 1936 le concept de *machine de Turing* pour définir la notion d'algorithme et les limites du calcul, jusqu'aux études de Stephen Kleene (Kleene, 1951) sur les expressions régulières et la classification des langages formels, la théorie des automates a profondément influencé la compréhension et la modélisation des systèmes de calcul.

Les automates ont par la suite été utilisés pour formaliser la notion de calcul et de traitement de l'information. Claude Shannon, notamment, a contribué à la théorie de la communication en introduisant des concepts qui trouvent des échos dans la modélisation des systèmes automatiques (Shannon, 1948). Dans ce contexte, les **automates finis** se sont imposés comme des outils indispensables non seulement pour la modélisation des langages formels, mais aussi pour la conception de compilateurs, l'analyse syntaxique et la vérification de systèmes logiciels.

Après avoir présenté leur rôle historique et leur importance pratique, il est nécessaire de préciser ce que l'on entend formellement par *automate fini*.

Un automate fini est défini comme un quintuplé $(Q, \Sigma, \delta, q_0, F)$, où :

- $Q$ est un ensemble fini d'états,
- $\Sigma$ est un alphabet de symboles d'entrée,
- $\delta$ est une fonction de transition associant à un état et un symbole d'entrée un état suivant (ou un ensemble d'états dans le cas non déterministe),
- $q_0$ est l'état initial, et
- $F$ est l'ensemble des états acceptants (finaux).

Un automate est *complet* si la fonction $\delta$ est totale. En théorie, il peut exister des automates non complets (certaines transitions ne sont pas définies). Toutefois, dans notre protocole expérimental, nous imposons la

complétude par construction, afin de garantir que chaque paire (état, symbole) possède une transition bien définie. Cela évite des comportements indéterminés et simplifie la simulation algorithmique, en assurant que l'exécution de toute chaîne est toujours définie.

Une *exécution* (ou chemin) dans un automate correspond à une séquence de transitions. Pour une chaîne d'entrée $w = a_1 a_2 \dots a_n$ de longueur $n$, une exécution peut être représentée par une suite d'états $q_0, q_1, \dots, q_m$ avec $0 \leq m \leq n$, où $q_0$ est l'état initial et, lorsque la transition $\delta(q_i, a_{i+1})$ est définie, $q_{i+1} = \delta(q_i, a_{i+1})$. Si la suite peut être prolongée jusqu'à $q_n$ et que $q_n \in F$, alors la chaîne $w$ est dite acceptée ; sinon, elle est rejetée. La vérification de l'acceptation constitue une étape fondamentale dans l'analyse des automates, en particulier lorsque le modèle présente une grande complexité (de nombreux états, transitions et alphabets étendus).

### 1.1.2 Exemple d'Automate

Figure 1.1 présente un exemple simple d'automate déterministe (DFA) avec deux états et des transitions définies pour les symboles de $\Sigma = \{a, b, c\}$. L'état initial est indiqué par la flèche pointant vers l'état de départ, et l'état final sont représentés par un double cercle. Cet automate accepte par exemple la chaîne `bbbacbba` (les chaînes se terminant par $a$ ou $c$ sont acceptées) et rejette la chaîne `bbbacbbb`.

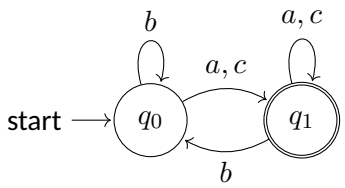


Figure 1.1 Exemple d'automate déterministe complet

Les concepts fondamentaux incluent :

- **Les états et transitions** : Un automate est défini par un ensemble fini d'états et un ensemble de transitions. Chaque transition, déclenchée par un symbole d'entrée, modélise le passage d'un état à un autre. Ce formalisme permet de capturer des comportements séquentiels et conditionnels de manière précise.
- **Les conditions d'acceptation** : La définition d'états spécifiques, appelés états finaux ou d'acceptation, est cruciale pour déterminer si une chaîne de caractères appartient au langage modélisé par l'automate.

— **Les représentations formelles** : Divers formats de représentation existent pour décrire les automates. Parmi eux, on trouve les tables de transitions, les listes d'adjacence et les graphes structurés, tels que ceux générés par l'outil DOT de Graphviz, qui facilitent la visualisation et l'analyse des modèles.

### 1.1.3 Exemple d'automate appliqué à une énigme classique

Considérez l'énigme suivante : `Un homme (M) doit traverser une rivière de l'Est à l'Ouest avec un loup (W), une chèvre (G) et un chou (C). Il dispose d'un petit bateau qui peut transporter uniquement lui-même et un seul de ses compagnons à la fois. Le loup mangera la chèvre si laissés seuls ensemble, et la chèvre mangera le chou dans les mêmes conditions. Comment peut-il traverser la rivière en toute sécurité ?`

Un DFA peut modéliser les différents états correspondant à la répartition des compagnons sur les deux rives et décrire les transitions (déplacements) autorisés, permettant de définir une séquence d'actions qui assure une traversée sécurisée (Sipser, 1996). La Figure 1.2 illustre un tel DFA où chaque état représente une configuration différente du déplacement du groupe.

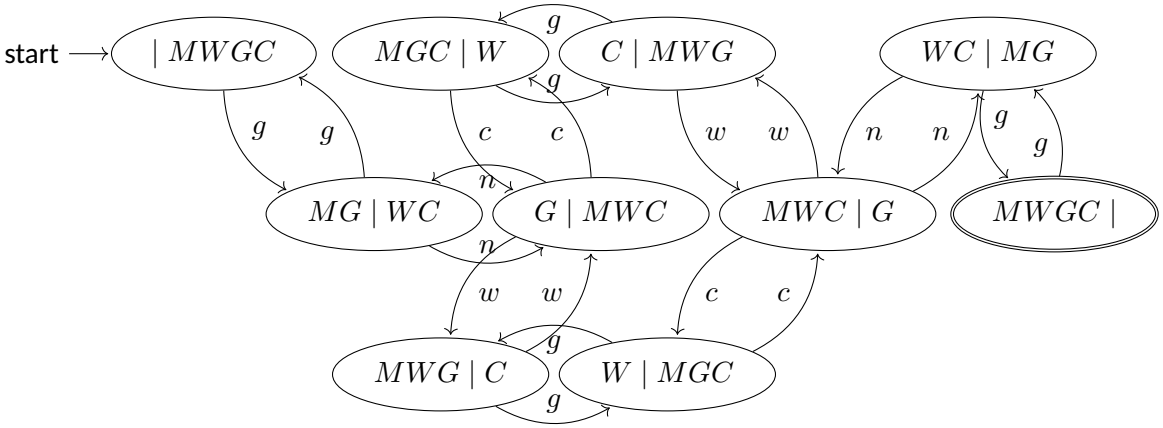


Figure 1.2 DFA modélisant l'énigme du traversier

En utilisant l'automate présenté en Figure 1.2, nous avons pu identifier deux séquences d'actions courtes et valides dans $\Sigma = \{g, c, w, n\}$ qui permettent de transporter en toute sécurité tous les compagnons de l'autre côté de la rivière : $gnwgcng$ et $gncgwng$. ChatGPT-4.0 a réussi à fournir l'une de ces séquences lorsqu'on lui a soumis uniquement la description textuelle du problème, mais il n'a pas réussi à déduire la seconde séquence lorsque on lui demande de générer une seconde solution. Cela suggère que les réponses

de ChatGPT-4.0 semblent reposer davantage sur des éléments de connaissances mémorisées que sur un véritable raisonnement analytique appliqué à la description du problème, limitant ainsi sa capacité à explorer toutes les solutions possibles dans des tâches complexes de génération séquentielle.

### 1.1.4 Applications et Enjeux Pratiques

Les automates finis trouvent des applications dans de nombreux domaines, attestant de leur transversalité et de leur pertinence :

- **Vérification de logiciels** : Les automates sont utilisés pour modéliser les comportements attendus des systèmes logiciels et vérifier leur conformité aux spécifications (Clarke *et al.*, 2018). Par exemple, l'analyse de protocoles de communication et la vérification formelle de systèmes critiques reposent sur des modèles d'automates pour détecter des erreurs potentielles.
- **Conception de langages de programmation** : La syntaxe et la sémantique des langages de programmation sont souvent définies à l'aide d'automates déterministes ou non-déterministes. Ces modèles facilitent l'analyse lexicale et la construction des parsers (Aho *et al.*, 2006) garantissant une interprétation correcte du code source.
- **Traitement des signaux et reconnaissance de motifs** : Dans le domaine du traitement de données, les automates sont exploités pour détecter des séquences ou motifs spécifiques au sein de grands volumes d'informations, comme dans l'analyse de séquences biologiques (Durbin *et al.*, 1998) ou la reconnaissance de formes dans des signaux numériques.

Malgré leur efficacité, leur déploiement pratique se heurte à des défis, comme la complexité accrue pour les systèmes à grande échelle ou les environnements dynamiques. Ces enjeux ont motivé des travaux en optimisation algorithmique et en mise à jour automatisée de modèles (Angluin, 1987), assurant la pertinence des automates dans l'informatique moderne.

En résumé, si la première partie de l'introduction expose la motivation de l'utilisation des DFAs comme outil d'évaluation du raisonnement formel des LLMs, ce sous-chapitre détaille l'évolution historique, les principes clés et les applications pratiques qui sous-tendent notre étude.

## 1.2 Les Modèles de Langage de Grande Taille (LLMs)

### 1.2.1 Évolution et Architectures Modernes

Les grands modèles de langages (LLMs, pour Large Language Models) ont connu une croissance exponentielle au cours de la dernière décennie, révolutionnant la manière dont les machines interagissent avec le langage humain. Cette révolution a été catalysée par l'introduction de l'architecture Transformer par Vaswani et al. en 2017 (Vaswani *et al.*, 2017), qui a remplacé les approches séquentielles classiques comme les réseaux récurrents (RNN, LSTM, GRU) (Gao et Glowacka, 2016).

Contrairement aux modèles précédents qui traitaient les séquences mot à mot, les Transformers permettent le traitement parallèle de séquences entières, avec un mécanisme central d'attention (notamment l'attention multi-tête) permettant au modèle de pondérer l'importance relative de chaque mot dans un contexte donné (Devlin *et al.*, 2019). Ce mécanisme est à la base des performances impressionnantes des LLMs, car il permet de capturer des dépendances longues et complexes dans le texte.

L'évolution des LLMs peut être résumée en plusieurs étapes clés :

- **L'ère des modèles autoregressifs** : GPT (Generative Pre-trained Transformer) de OpenAI (Brown *et al.*, 2020b), entraîné en deux phases (pré-entraînement non supervisé, puis fine-tuning supervisé), a marqué une rupture avec les approches antérieures. GPT-3, avec 175 milliards de paramètres, a démontré pour la première fois une compétence quasi-généralisée dans de nombreuses tâches NLP sans entraînement spécifique.
- **L'avènement de l'entraînement instructif** : Les modèles comme InstructGPT (Ouyang *et al.*, 2022) et ChatGPT ont été affinés pour suivre des instructions humaines via des techniques de *Reinforcement Learning from Human Feedback* (RLHF), ce qui a permis d'améliorer drastiquement la pertinence des réponses dans des contextes conversationnels.
- **Les modèles multi-tâches et multilingues** : T5 (Raffel *et al.*, 2020) (Text-To-Text Transfer Transformer) et mT5 ont redéfini l'idée du pretraining, en formulant toutes les tâches NLP comme des tâches de génération de texte. De leur côté, les modèles comme BLOOM (Le Scao *et al.*, 2023) ont été conçus pour fonctionner dans plusieurs langues, ouvrant la voie à une inclusion linguistique plus large.
- **La croissance exponentielle des paramètres** : Le passage de GPT-2 (1.5B) à GPT-4 (estimé à plusieurs centaines de milliards de paramètres (OpenAI, 2023)) illustre l'intuition que "plus grand = meilleur"

en matière de performances. Cette croissance s'est également accompagnée d'une augmentation des besoins en données, en puissance de calcul, et en techniques d'optimisation comme le Mixture of Experts (MoE) (Shazeer et Stern, 2018).

— **L'intégration multimodale** : Des modèles comme Flamingo (Alayrac *et al.*, 2022), Gemini (DeepMind, 2023) ou GPT-4V (Achiam *et al.*, 2023) ont ajouté la capacité de traiter à la fois du texte, des images et, dans certains cas, de l'audio ou de la vidéo, créant ainsi des systèmes multimodaux capables de raisonner sur des données hétérogènes.

### 1.2.2 Capacités de Raisonnement et Limites Intrinsèques

Les LLMs modernes ne se contentent plus de prédire des mots : ils démontrent des capacités de raisonnement de plus en plus complexes, parfois comparables à celles attendues d'un être humain. Une de leurs forces les plus fascinantes est leur aptitude à généraliser à partir de descriptions ou d'instructions abstraites, en appliquant ces connaissances à des cas concrets, même lorsqu'ils n'ont jamais vu exactement la même situation durant leur entraînement.

Cette capacité à inférer des règles générales à partir d'exemples particuliers, à raisonner à partir d'un prompt flou, voire à simuler un comportement algorithmique, révèle une forme d'abstraction cognitive qui interpelle les spécialistes du raisonnement automatique et de la modélisation symbolique

Parmi les capacités cognitives simulées par les LLMs :

— **Le raisonnement logique et arithmétique** : Des prompts structurés permettent à des modèles comme GPT-4 ou Claude 3 d'effectuer des raisonnements mathématiques ou des preuves simples, notamment via le *chain-of-thought prompting* (Wei *et al.*, 2022).

— **La compréhension pragmatique et contextuelle** : Les LLMs peuvent adapter leur discours à des contextes spécifiques, répondre de manière cohérente à des dialogues complexes et simuler divers styles de langage.

— **L'exécution implicite de tâches spécialisées** : Sans entraînement supplémentaire, des LLMs peuvent rédiger du code, analyser des bases de données, générer des articles scientifiques, ou même jouer à des jeux de stratégie.

Cependant, ces capacités s'accompagnent de limitations importantes :

— **Hallucinations** : Les modèles peuvent produire des réponses fausses mais formulées de manière

convaincante (Ji *et al.*, 2023), particulièrement dans les domaines spécialisés (droit, médecine, etc.).

— **Biais et éthique** : Étant entraînés sur des données issues d'internet, les LLMs peuvent reproduire ou amplifier des biais sociaux, culturels, ou politiques (Bender *et al.*, 2021).

— **Manque de traçabilité** : Le processus de raisonnement interne d'un LLM n'est pas explicite. Il n'existe pas de "logique formelle" identifiable entre l'entrée et la sortie, ce qui complique les processus d'explication et de vérification.

— **Performances inégales selon le domaine** : Si les LLMs sont efficaces dans des tâches générales, ils sont parfois incompétents dans des raisonnements symboliques (Wei *et al.*, 2022) complexes, comme en logique formelle ou en mathématiques avancées, sauf avec des prompts extrêmement bien calibrés ou des stratégies spécialisées (Mondorf et Plank, 2024).

#### 1.2.2.1 Comparaison entre modèles existants

Cette section présente une comparaison des modèles de LLMs actuels, en mettant en évidence leurs tailles, organisations et spécificités principales. Le tableau suivant synthétise ces informations pour faciliter la lecture et la compréhension des différences entre les modèles.

| **Modèle** | **Taille (params)** | **Organisation** | **Spécificités** |
|---|---|---|---|
| GPT-3.5 | 175B | OpenAI | Haute performance, réponses parfois factuellement incorrectes. |
| GPT-4 | 500B | OpenAI | Accès via ChatGPT Plus/API, raisonnement amélioré. |
| Claude 3 | Inconnu | Anthropic | Excellente gestion des dialogues complexes. |
| Gemini 2.0 Flash | 20B-30B | Google DeepMind | Multimodal, avec capacités de raisonnement avancées. |
| Mistral 7B | 7B | Mistral AI | Open-source, optimisé pour les déploiements locaux. |
| DeepSeek-V3-0324 | 42B | DeepAI Labs | Recherche sémantique précise dans des contextes complexes. |
| LLaMA 2 | 70B | Meta | Open-source, licence restrictive, multilingue. |
| LLaMA 3 | 405B | Meta | Pré-entraîné sur 15T tokens, multimodal. |
| GitHub Copilot | 12B-175B | GitHub (Microsoft) | Assistant de codage intégrant des modèles d'OpenAI, Anthropic et Google. |
| Perplexity AI | 1B-405B | Perplexity | Système de recherche conversationnelle et synthèse d'informations. |
| Qwen | 1,8B-72B | Alibaba | Modèle open-source multilingue, surpassant DeepSeek-V3 sur plusieurs benchmarks. |
| Doubao | Inconnu | ByteDance | LLM économique, optimisé pour les usages à faible coût. |

Tableau 1.1 Comparaison simplifiée des modèles de LLMs et variantes associées

*(B) Nombre de paramètres en milliards estimé selon plusieurs analyses techniques indépendantes.*

Explications détaillées :

GPT-3.5 : Ce modèle autoregressif possède 175 milliards de paramètres (Brown *et al.*, 2020b). Il a été évalué sur des benchmarks standard en NLP, montrant de bonnes performances pour la génération de texte et la complétion de phrases. Des limitations sont signalées, notamment la génération occasionnelle d'informations incorrectes ou incohérentes.

GPT-4 : Évalué autour de 500 milliards de paramètres (valeur non officiellement confirmée), GPT-4 (OpenAI, 2023) est également un modèle autoregressif, conçu pour la génération de texte et la résolution de tâches complexes. Il a été testé sur des benchmarks incluant raisonnement logique et compréhension contextuelle.

Claude 3 : La taille exacte de ce modèle n'est pas publiée. Claude 3 (Anthropic, 2024) est conçu pour l'interaction conversationnelle et la génération de texte. Il a été évalué sur des tâches de dialogue ouvertes et des tests de cohérence contextuelle.

Gemini 2.0 Flash (Google DeepMind) : Ce modèle multimodal (DeepMind, 2024) peut traiter simultanément texte, images et autres types de données. Ses performances ont été étudiées sur des tâches de raisonnement et de compréhension multimodale.

Mistral 7B : Ce modèle open-source dispose de 7 milliards de paramètres (AI, 2023a). Il est conçu pour des déploiements locaux et a été évalué sur des tâches de génération de texte standard, offrant des performances comparables à d'autres modèles de taille similaire.

DeepSeek-V3-0324 : Avec 42 milliards de paramètres (DeepSeek, 2025), ce modèle est spécialisé dans la recherche sémantique et la récupération d'informations. Il a été testé sur des ensembles de données complexes pour l'extraction précise de passages pertinents.

LLaMa 2 : Disponible en tailles 7B, 13B et 70B (AI, 2023c), ce modèle open-source a été pré-entraîné sur de larges corpus multilingues. Il a été évalué sur des tâches NLP standard et certaines tâches multilingues.

LLaMa 3 : Ce modèle est proposé en versions 8B, 70B et 405B (AI, 2024), pré-entraîné sur environ 15T tokens. Il a été évalué sur des tâches multilingues et multimodales, montrant des capacités générales de génération et compréhension de texte.

GitHub Copilot : Lancé en 2021 (GitHub, 2023), Copilot s'intègre aux environnements de développement et

fournit des suggestions de code en temps réel. Il repose initialement sur Codex (GPT-3) et utilise désormais des modèles plus récents comme GPT-4 pour la génération de code et la complétion de fonctions.

Perplexity AI : Ce système (AI, 2022) est orienté vers la recherche conversationnelle et la synthèse d'informations. Il a été évalué sur des tâches de récupération d'information et de génération de réponses à partir de contextes complexes.

Qwen : Développé par Alibaba Cloud (Group, 2025), Qwen 2.5-Max est disponible en versions 1,8B à 72B. Il a été évalué sur des benchmarks de compréhension du langage, résolution de problèmes et génération de code.

Doubao : Ce modèle (ByteDance, 2025), développé par ByteDance, a été testé sur des tâches de génération de texte et de compréhension de langage naturel. Il utilise une architecture de type « mixture of experts » pour optimiser l'utilisation des ressources et la performance sur des tâches standard.

### 1.2.3 Adoption Accélérée des LLMs depuis 2023

Depuis l'intégration de ChatGPT dans les environnements Microsoft (Edge, Office 365, Copilot), l'adoption des modèles de langage de grande taille (LLMs) s'est rapidement généralisée dans les milieux industriels et professionnels. D'après le rapport *State of AI 2024* (Benaich et Capital, 2024), près de 73 % des grandes entreprises technologiques et 52 % des entreprises du Fortune 500 ont amorcé une intégration active d'outils fondés sur des LLMs dans leurs chaînes de production, notamment dans les domaines du support client, de la génération de code, de l'analyse de texte, ou de la rédaction de rapports techniques.

Du côté académique, l'intérêt pour les LLMs s'est traduit par une croissance exponentielle de la littérature scientifique : le nombre de publications indexées par Semantic Scholar contenant les mots-clés *large language model*, *prompt engineering* ou *transformer architecture* a plus que triplé entre 2022 et 2024. Cette explosion est soutenue par la multiplication des ressources pédagogiques en ligne, telles que les spécialisations en IA générative proposées par DeepLearning.AI, Stanford ou Hugging Face, qui figurent parmi les cours les plus suivis sur Coursera et edX en 2024.

## 1.3 Le Prompt Engineering : Approches et Défis

### 1.3.1 Définition et Méthodologies

Le prompt engineering désigne l'art de concevoir des requêtes structurées et optimisées pour orienter efficacement le comportement des modèles de langage de grande taille (LLMs). Cette discipline, en constante évolution, s'appuie sur plusieurs principes fondamentaux :

- **La clarté de la requête** : Une formulation précise permet de guider le modèle vers une interprétation correcte du problème.
- **La décomposition en sous-tâches** : La division d'un problème complexe en étapes successives (chain-of-thought) facilite la compréhension et améliore la qualité des réponses.
- **L'adaptation contextuelle** : Les prompts doivent être ajustés en fonction du domaine d'application : informatique théorique, écriture académique, médecine, art, etc. (Giray, 2023; Meskó, 2023; Oppenlaender *et al.*, 2024).

Plusieurs études ont montré que la qualité du prompt est un facteur déterminant pour obtenir des résultats pertinents, particulièrement dans des tâches de raisonnement formel comme la vérification de l'acceptation de mots par un automate (Wei *et al.*, 2022; Shum *et al.*, 2023).

### 1.3.2 Revue des Techniques Appliquées

La littérature récente a introduit diverses techniques de prompt engineering, chacune adaptée à des contextes spécifiques :

1. **Prompt Standard (Vanilla Prompting)** : Il s'agit d'une requête directe sans contexte ni exemple supplémentaire. Bien que simple, cette approche peut être insuffisante pour des tâches complexes nécessitant un raisonnement approfondi (Wei *et al.*, 2022).
2. **Prompt en Chaîne de Pensée (Chain-of-Thought)** : Cette méthode incite le modèle à détailler son processus de décision avant de fournir la réponse finale. Elle a montré des améliorations significatives dans des tâches de raisonnement mathématique et de compréhension contextuelle (Wei *et al.*, 2022).
3. **Auto-CoT (Automatic Chain-of-Thought)** : Cette approche automatise la génération de chaînes de pensée en regroupant les requêtes similaires et en générant des raisonnements pour des requêtes représentatives. Elle permet d'améliorer les performances sans nécessiter de données d'entraînement manuelles (Zhang *et al.*, 2022).

4. **Self-Consistency Prompting** : Cette technique génère plusieurs chaînes de raisonnement pour une même requête et sélectionne la réponse la plus cohérente parmi elles. Elle améliore la précision en exploitant la diversité des chemins de raisonnement possibles (Wang *et al.*, 2022).
5. **Prompt Décomposé (Decomposed Prompting)** : Cette méthode modulaire divise une tâche complexe en sous-tâches gérées par des fonctions spécialisées, orchestrées par un contrôleur central. Elle est particulièrement efficace pour les problèmes nécessitant une planification ou une exécution séquentielle (Khot *et al.*, 2022).
6. **Prompt de Rôle (Role Prompting)** : En assignant un rôle spécifique au modèle (par exemple, "Vous êtes un expert en automates finis"), cette technique guide le style et le contenu de la réponse, améliorant la pertinence dans des contextes spécialisés (Kong *et al.*, 2023).
7. **Prompt Multi-Tâche (Multi-Task Prompting)** : Cette approche combine plusieurs tâches dans une seule requête, permettant au modèle de gérer des actions interconnectées simultanément. Elle est utile pour des scénarios complexes nécessitant une compréhension contextuelle approfondie (Gozzi et Di Maio, 2024).
8. **Prompt Graphique (Graph Prompting)** : En intégrant des graphes de connaissances, cette technique améliore la capacité des modèles à gérer des informations complexes et interconnectées, en facilitant le raisonnement structuré (Tian *et al.*, 2024).
9. **Prompt Socratique (Socratic Questioning)** : Cette méthode utilise une série de questions exploratoires pour approfondir la compréhension d'un sujet, favorisant l'analyse critique et la réflexion approfondie (Qi *et al.*, 2023).

Ces techniques ont été expérimentées dans divers domaines, allant de la rédaction académique à l'analyse de données médicales, et démontrent chacune des avantages et des limites spécifiques en fonction du contexte d'utilisation.

| **Technique** | **Avantages** | **Inconvénients** |
|---|---|---|
| Prompt Standard | Simplicité, rapidité | Moins efficace pour des tâches complexes |
| Chaîne de Pensée (CoT) | Améliore le raisonnement, explicite les étapes | Peut être verbeux, nécessite une formulation soignée |
| Auto-CoT | Automatisation, réduction du besoin de données manuelles | Peut générer des raisonnements moins pertinents |
| Self-Consistency | Améliore la précision en agrégeant plusieurs raisonnements | Coûteux en ressources computationnelles |
| Prompt Décomposé | Modularité, efficacité pour les tâches complexes | Complexité de mise en œuvre |
| Prompt de Rôle | Pertinence contextuelle, personnalisation des réponses | Risque de stéréotypes, dépend de la qualité des données d'entraînement |
| Prompt Multi-Tâche | Efficacité, gestion de tâches interconnectées | Risque de surcharge cognitive pour le modèle |
| Prompt Graphique | Raisonnement structuré, gestion de données complexes | Nécessite des graphes de connaissances bien structurés |
| Prompt Socratique | Favorise l'analyse critique, approfondissement des sujets | Peut être moins direct, nécessite une formulation précise |
| Prompt de Vérification/Édition | Améliore la précision, alignement sur des faits vérifiables | Processus en plusieurs étapes, nécessite des ressources supplémentaires |

Tableau 1.2 Comparaison des techniques de prompt engineering

## 1.4 Comparaison des Approches et Perspectives d'Amélioration

### 1.4.1 Forces des Approches Existantes

Les techniques actuelles de prompt engineering présentent plusieurs atouts significatifs, consolidés par des recherches récentes :

- Les prompts peuvent être adaptés à une variété de tâches, allant de la génération de texte à la résolution de problèmes complexes, sans nécessiter de modifications des paramètres internes des modèles.
- L'utilisation de techniques telles que la chaîne de pensée (*chain-of-thought*) permet aux modèles de détailler leur processus de décision, améliorant ainsi la qualité des réponses pour des tâches nécessitant un raisonnement logique
- Le prompt engineering a démontré son efficacité dans divers domaines, notamment l'éducation, la santé, le droit et la création artistique, illustrant sa polyvalence

### 1.4.2 Limites et Défis à Relever

Malgré ces avancées, plusieurs défis demeurent :

- Les modèles de langage sont particulièrement sensibles aux variations, même minimes, dans la formulation des prompts (Polo *et al.*, 2024). Des études ont montré que des modifications subtiles, telles que la ponctuation ou l'ordre des mots, peuvent entraîner des variations de performance significatives, allant jusqu'à 76 points de précision dans certains cas (Akmal, nd). Cette sensibilité complique la reproductibilité des résultats et souligne la nécessité de concevoir des prompts rigoureux et méthodiques (Chatterjee *et al.*, 2024).
- De légères variations dans la formulation des prompts peuvent entraîner des différences significatives dans les réponses des modèles, rendant la reproductibilité des résultats difficile (Errica *et al.*, 2024).
- Le manque de transparence dans le processus de raisonnement des LLMs, compliquant l'identification des sources d'erreur. Les modèles de langage sont souvent perçus comme des "boîtes noires", ce qui complique la compréhension de leur processus de raisonnement/décision et l'identification des sources d'erreur
- Des attaques telles que l'injection de prompt peuvent manipuler les modèles pour produire des sorties non désirées ou malveillantes, posant des risques en matière de sécurité et d'éthique (Zhuo

*et al.*, 2023).

— Les modèles peuvent reproduire ou amplifier des biais présents dans leurs données d'entraînement, même avec des prompts soigneusement conçus, ce qui soulève des préoccupations éthiques.

## 1.5 Conclusion de l'État de l'Art

L'examen de la littérature révèle que, plus qu'une simple convergence, c'est une interaction riche et prometteuse qui se dessine entre la théorie des automates et les capacités émergentes des LLMs via le prompt engineering. Les automates finis offrent un cadre formel rigoureux pour tester les aptitudes des modèles à manipuler des structures symboliques, tandis que les LLMs, grâce à leur pouvoir d'abstraction et de raisonnement pas-à-pas, proposent de nouveaux moyens d'automatiser la vérification et l'optimisation des systèmes logiciels. Toutefois, cette synergie soulève des défis cruciaux : robustesse des prompts, cohérence sur de longues séquences, fiabilité des résultats et traçabilité des raisonnements qui doivent être surmontés pour garantir la sécurité et la reproductibilité des approches. La suite de cette recherche s'attachera à développer des méthodes hybrides combinant la puissance des LLMs et la rigueur formelle des automates, tout en adressant ces enjeux fondamentaux.

# CHAPITRE 2
# MÉTHODOLOGIE

Dans cette section, nous présentons en détail notre méthodologie, illustrée schématiquement à la Figure 2.1. Notre approche s'articule autour de quatre axes expérimentaux majeurs :

1. **Génération d'automates finis** : Création aléatoire de DFAs (automates finis déterministes) en contrôlant le nombre d'états, le ratio d'états acceptants et la taille de l'alphabet.
2. **Représentation textuelle** : Conversion de chaque automate en divers formats de représentation, afin d'examiner l'impact de ces formats sur la compréhension par les modèles de langage.
3. **Formulation de la requête (prompting)** : Construction de prompts à partir des représentations textuelles pour interroger le LLM sur deux problématiques centrales (vérification de l'acceptation d'une chaîne et génération d'une des chaînes les plus courtes acceptées).
4. **Vérification des réponses** : Comparaison systématique des réponses fournies par le LLM aux résultats attendus, par le biais de simulations d'exécutions et d'algorithmes d'optimisation.

Dans cette étude, nous avons choisi d'utiliser le modèle ChatGPT de la plateforme OpenAI, spécifiquement la variante `gpt-4o-mini-2024-07-18`, afin de satisfaire plusieurs critères méthodologiques essentiels.

Premièrement, ce choix s'inscrit dans une logique de reproductibilité et de comparabilité avec les travaux antérieurs de l'écosystème OpenAI, où des modèles tels que GPT-3.5 et GPT-4 ont été largement utilisés. Cela permet de situer nos résultats dans un cadre expérimental validé par la communauté scientifique.

Deuxièmement, l'API de ChatGPT est stable, bien documentée, et facilement intégrable dans des scripts automatisés. Elle permet de gérer efficacement un grand nombre de requêtes sans surcharge technique ni coûts imprévisibles, contrairement à certains modèles open-source souvent mieux notés en benchmarks, mais dont l'intégration API reste limitée ou peu standardisée.

Enfin, des contraintes budgétaires propres à un projet de maîtrise ont fortement influencé notre choix. Des modèles haut de gamme comme Claude 3 ou Gemini 2.0 Flash, bien que performants, imposent des frais d'accès élevés. De leur côté, des alternatives open-source comme Mistral 7B, LLaMA 2/3, ou DeepSeek-V3 nécessitent un déploiement local, souvent basé sur GPU, ainsi qu'un encadrement technique spécialisé difficile à mobiliser dans notre environnement académique.

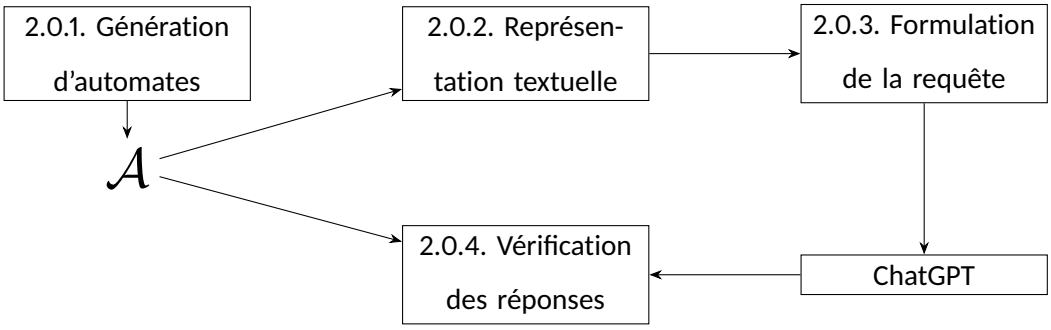


Figure 2.1 Représentation schématique de notre méthodologie.

### 2.0.1 Génération d'automates

La génération aléatoire d'un automate fini déterministe (DFA) suit trois étapes inspirées des principes de :

1. **Initialisation :** Trois paramètres fondamentaux sont définis pour chaque expérimentation : le nombre d'états de l'automate, la taille de son alphabet, et la longueur des chaînes d'entrée. Afin d'isoler l'influence de chaque facteur, nous avons adopté une stratégie de variation contrôlée : pour évaluer l'impact du nombre d'états, l'alphabet est fixé à trois lettres et les chaînes à 20 caractères; pour l'étude de la taille de l'alphabet , nous fixons à 10 le nombre d'états; et pour analyser l'effet de la longueur des chaînes, nous gardons constants les états et l'alphabet. Cette approche permet une évaluation fine et ciblée de la robustesse des LLM selon la complexité structurelle et symbolique des automates testés.
2. **Assignation des transitions :** Pour chaque paire (état, symbole) de l'alphabet, une transition est assignée aléatoirement vers l'un des états, garantissant ainsi que la fonction de transition $\delta$ soit définie pour chaque état et chaque symbole. Chaque automate généré est donc *complet*, la complétude étant assurée par construction.
3. **Définition de l'état initial et des états acceptants :** Un état de départ est choisi aléatoirement dans l'ensemble des états, et les états acceptants sont sélectionnés selon un ratio défini.

Pour illustrer ce mécanisme de génération, la figure 2.2 donne un exemple d'automate fini déterministe (DFA) aléatoirement généré :

La figure 2.2 présente un automate fini déterministe (DFA) à deux états et six transitions. L'alphabet est $\Sigma = \{a, b, c\}$, l'ensemble des états est $Q = \{q_0, q_1\}$, la fonction de transition est définie par

$$\delta(q_0, a) = q_1, \quad \delta(q_0, b) = q_0, \quad \delta(q_0, c) = q_1,$$

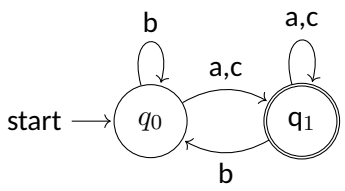


Figure 2.2 Exemple d'automate fini déterministe généré aléatoirement.

$$\delta(q_1, a) = q_1, \quad \delta(q_1, b) = q_0, \quad \delta(q_1, c) = q_1,$$

Les transitions sont définies de sorte que toute lecture d'un «b» en $q_0$ reste en $q_0$, tandis que «a» ou «c» font passer à $q_1$, etc. l'état initial est $q_0$ et l'ensemble des états acceptants est $F = \{q_1\}$.
Cet automate accepte la chaîne `bbbacbba` et rejette la chaîne `bbbacbbb`. Elle accepte précisément les mots se terminant par «a» ou «c».

Le pseudo-code ci-dessous illustre ce processus :

```
function generate_automaton(num_states, num_accepting, alphabet_size):
    states = [0, 1, ..., num_states-1]
    alphabet = [symbol_1, symbol_2, ..., symbol_alphabet_size]
    accepting_states = random_sample(states, num_accepting)
    transitions = {}
    for state in states:
        for symbol in alphabet:
            transitions[state][symbol] = random_choice(states)
    start_state = choose_start_state(states)
    return Automaton(states, alphabet, transitions, start_state, accepting_states)
```

### 2.0.2 Représentation textuelle

Une fois l'automate généré, nous le convertissons en divers formats de représentation afin d'évaluer comment la forme de présentation influence la compréhension par les LLMs. Nous utilisons les formats suivants :

- **Langage Naturel Contrôlé (Controlled Natural Language - CNL)**
- **Liste d'Adjacence (Adjacency List)**
- **Tableau de Transitions (Transition Table)**

— **Format DOT**

— **Format de représentation Combiné (Combined Representation)**

Le choix de ces formats est motivé par l'objectif d'explorer l'impact des différentes représentations sur la compréhension des modèles dans les tâches d'interprétation d'automates. Nous décrivons ci-dessous chaque format, ses forces, ses faiblesses et les raisons de son inclusion dans l'étude. Pour chacune de ces représentations, nous fournissons sa grammaire BNF et illustrons la conversion à l'aide de l'exemple de la figure 2.2.

#### 2.0.2.1 Langage Naturel Contrôlé :

Ce format utilise une description narrative en anglais contrôlé pour représenter l'automate, en s'appuyant sur la Figure 2.2. Il explicite de manière lisible le nombre d'états, l'état initial, les états acceptants et chaque transition, ce qui tire parti des compétences des LLMs sur le langage naturel. Toutefois, sa verbosité et son absence de structure rigide peuvent rendre l'analyse plus laborieuse pour un modèle.

**Grammaire BNF :**

$$
\begin{aligned}
\langle \text{CNL\_description} \rangle &::= \langle \text{number\_states} \rangle \\
&\quad \langle \text{initial\_state} \rangle \\
&\quad \langle \text{accepting\_states} \rangle \\
&\quad \langle \text{transitions\_header} \rangle \\
&\quad \langle \text{list\_transitions} \rangle, \\
\langle \text{number\_states} \rangle &::= \text{The automaton has } \langle \text{INT} \rangle \text{ states.} \\
\langle \text{initial\_state} \rangle &::= \text{The start state is } \langle \text{INT} \rangle. \\
\langle \text{accepting\_states} \rangle &::= \text{The accepting states are : } \langle \text{list\_int} \rangle. \\
\langle \text{transitions\_header} \rangle &::= \text{Transitions are as follows :} \\
\langle \text{list\_transitions} \rangle &::= \langle \text{transition} \rangle \langle \text{list\_transitions} \rangle \\
&\quad | \langle \text{transition} \rangle, \\
\langle \text{transition} \rangle &::= \text{From state } \langle \text{INT} \rangle, \text{ on input } \langle \text{STRING} \rangle, \text{ move to state } \langle \text{INT} \rangle. \\
\langle \text{list\_int} \rangle &::= \langle \text{INT} \rangle, \langle \text{list\_int} \rangle \\
&\quad | \langle \text{INT} \rangle.
\end{aligned}
$$

**Exemple :**

```
The automaton has 2 states. The start state is 1. The accepting states are: 1.
Transitions are as follows:
From state 0, on input 'a', move to state 0.
From state 0, on input 'b', move to state 1.
From state 0, on input 'c', move to state 0.
From state 1, on input 'a', move to state 0.
From state 1, on input 'b', move to state 0.
From state 1, on input 'c', move to state 1.
```

Ce format, bien qu'expressif et proche du langage humain, présente un compromis entre lisibilité et rigueur structurelle. Les LLMs tirent profit de la syntaxe narrative pour interpréter les transitions, en particulier lorsqu'un raisonnement pas-à-pas est attendu. Toutefois, l'absence de hiérarchie formelle ou de structure tabulaire peut générer des confusions dans le suivi des états, notamment pour les longues chaînes de transitions ou dans les cas où plusieurs formats sont comparés. Nos expérimentations montrent que ce format, combiné à des prompts explicites et à une approche assistée, constitue l'un des meilleurs compromis entre accessibilité humaine et efficacité machine.

#### 2.0.2.2 Format de Liste d'Adjacence :

Ce format exploite une structure hiérarchique dans laquelle chaque état est utilisé comme une clé, et la valeur associée correspond à un dictionnaire décrivant les transitions depuis cet état. L'état initial ainsi que les états acceptants sont explicitement indiqués. Bien que ce format soit très structuré et facilement interprétable par des systèmes informatiques grâce à sa nature explicite, sa structure imbriquée peut poser des défis aux LLMs, qui peuvent trouver moins intuitif de décoder une représentation en dictionnaires imbriqués.

**Grammaire BNF :**

$$
\begin{aligned}
\langle \text{ADJ\_description} \rangle &::= \{\text{"start\_state"} : \langle \text{INT} \rangle,\ \text{"accepting\_states"} : [\langle \text{list\_int} \rangle], \langle \text{list\_transition} \rangle\} \\
\langle \text{list\_int} \rangle &::= \langle \text{INT} \rangle, \langle \text{list\_int} \rangle \quad | \quad \langle \text{INT} \rangle \\
\langle \text{list\_transition} \rangle &::= \langle \text{transition} \rangle, \langle \text{list\_transition} \rangle \quad | \quad \langle \text{transition} \rangle \\
\langle \text{transition} \rangle &::= \text{"}\langle \text{INT} \rangle\text{"}\ \{\langle \text{list\_output} \rangle\} \\
\langle \text{list\_output} \rangle &::= \langle \text{output} \rangle, \langle \text{list\_output} \rangle \quad | \quad \langle \text{output} \rangle \\
\langle \text{output} \rangle &::= \text{"}\langle \text{STRING} \rangle\text{"} : \langle \text{INT} \rangle
\end{aligned}
$$

**Exemple :**

```
{ "start_state": 0, "accepting_states": [1],
  "0": {"a": 1, "b": 0, "c": 1},
  "1": {"a": 0, "b": 1, "c": 1} }
```

Ce format permet de représenter de façon explicite chaque état et ses sorties, facilitant ainsi l'analyse algorithmique. Il est particulièrement utile pour illustrer la structure de l'automate et pour permettre aux LLMs d'identifier clairement les relations entre les états et les symboles d'entrée, même si la nature hiérarchique imbriquée peut parfois compliquer la tâche pour des modèles de langage qui sont optimisés pour le texte en langage naturel. Des travaux comme ceux de (Zhang *et al.*, 2025) ont montré que ce type de représentation structurelle peut améliorer la robustesse de l'analyse algorithmique lorsqu'il est utilisé conjointement avec des approches hybrides combinant plusieurs formats.

#### 2.0.2.3 Liste de Tableau de Transitions :

Ce format fournit une représentation détaillée et granulaire des transitions de l'automate, chaque état étant exprimé comme une liste de paires associant un symbole à l'indice de l'état suivant. Il inclut explicitement l'état initial pour chaque transition, offrant une description exhaustive de la configuration et des déplacements au sein de l'automate. Bien que ce format soit très précis et explicite, sa dépendance aux indices d'états peut en limiter la lisibilité et rendre son interprétation moins intuitive pour les humains et potentiellement pour les LLMs.

**Grammaire BNF :**

$$
\begin{aligned}
\langle \text{Table\_List\_description} \rangle &::= [\langle \text{list\_states} \rangle] \\
\langle \text{list\_states} \rangle &::= \langle \text{state\_transitions} \rangle \ \langle \text{list\_states} \rangle \\
&\quad \mid \langle \text{state\_transitions} \rangle \\
\langle \text{state\_transitions} \rangle &::= [\langle \text{list\_pairs} \rangle] \\
\langle \text{list\_pairs} \rangle &::= \langle \text{pair} \rangle \ \langle \text{list\_pairs} \rangle \\
&\quad \mid \langle \text{pair} \rangle \\
\langle \text{pair} \rangle &::= \{\text{"symbol"} : \langle \text{STRING} \rangle, \\
&\qquad \text{"next\_state"} : \langle \text{INT} \rangle, \\
&\qquad \text{"accepting\_state"} : \langle \text{BOOL} \rangle, \\
&\qquad \text{"is\_start\_state"} : \langle \text{BOOL} \rangle\}
\end{aligned}
$$

**Exemple :**

```
[[{"symbol": "a", "next_state": 1, "accepting_state": False,
    "is_start_state": True},
    {"symbol": "b", "next_state": 0,
      "accepting_state": False, "is_start_state": True},
    {"symbol": "c", "next_state": 1,
      "accepting_state": False, "is_start_state": True} ] ,
[{"symbol": "a", "next_state": 0, "accepting_state": True,
    "is_start_state": False},
    {"symbol": "b", "next_state": 1,
      "accepting_state": True, "is_start_state": False},
    {"symbol": "c", "next_state": 1,
      "accepting_state": True, "is_start_state": False} ] ]
```

Ce format, en offrant une granularité fine dans la description de chaque transition, permet une analyse minutieuse de la structure de l'automate. Bien que sa syntaxe soit parfois moins lisible pour un observateur humain en raison de l'utilisation d'indices numériques, elle est particulièrement utile pour des applications nécessitant une interprétation algorithmique précise ou une mise en œuvre dans des systèmes informa-

tiques.

#### 2.0.2.4 Format DOT

Ce format offre une représentation visuelle de l'automate en utilisant la syntaxe DOT de Graphviz. Il permet d'obtenir un graphe orienté intuitif, idéal pour le raisonnement basé sur des structures de type graphe. Cependant, cette approche nécessite que les modèles de langage soient capables de traiter des données structurées sous forme visuelle ou graphique.

**Grammaire BNF :**

$$
\begin{aligned}
\langle \text{DOT\_description} \rangle &::= \text{digraph } \{ \langle \text{graph\_settings} \rangle \langle \text{list\_nodes} \rangle \langle \text{list\_edges} \rangle \} \\
\langle \text{graph\_settings} \rangle &::= \text{rankdir = LR}; \\
&\quad\ \text{start} \rightarrow \langle \text{INT} \rangle; \\
\langle \text{list\_nodes} \rangle &::= \langle \text{node} \rangle \langle \text{list\_nodes} \rangle \mid \langle \text{node} \rangle \\
\langle \text{node} \rangle &::= \langle \text{INT} \rangle \ [\text{shape=doublecircle}]; \\
\langle \text{list\_edges} \rangle &::= \langle \text{edge} \rangle \langle \text{list\_edges} \rangle \mid \langle \text{edge} \rangle \\
\langle \text{edge} \rangle &::= \langle \text{INT} \rangle \rightarrow \langle \text{INT} \rangle \ [\text{label="} \langle \text{STRING} \rangle \text{"}]; \\
\langle \text{INT} \rangle &::= \text{integer value} \\
\langle \text{STRING} \rangle &::= \text{single character enclosed in double quotes}
\end{aligned}
$$

**Exemple :**

```
digraph {
    rankdir = LR; start[style=invis];
    start -> 0;
    0 -> 1 [label="a"];
    0 -> 0 [label="b"];
    1 -> 0 [label="a"];
    1 -> 1 [label="b"];
    1 [shape=doublecircle];
}
```

Ce format DOT permet ainsi de générer des visualisations claires et structurées, facilitant la compréhension globale de la dynamique des transitions de l'automate. Il est particulièrement avantageux pour des analyses qui nécessitent une interprétation graphique de la structure de l'automate, bien qu'il suppose que l'utilisateur ou le modèle puisse traiter des représentations visuelles ou des données en graphes.

#### 2.0.2.5 Format de représentation combiné

Ce format hybride combine les avantages du langage naturel contrôlé et de la liste d'adjacence afin de maximiser la clarté et la structure, ce qui permet aux LLMs de mieux interpréter la configuration de l'automate.

**Grammaire BNF :**

$$\langle \text{Combined\_description} \rangle ::= \text{Here is the automaton described in two formats :}$$
$$- \langle \text{CNL\_description} \rangle$$
$$- \langle \text{ADJ\_description} \rangle$$

**Exemple :**

```
Here is the automaton described in two formats:
- The automaton has 2 states. The start state...
- { "start_state": 0, "accepting_states":...
```

Ce double encodage exploite à la fois l'intuitivité du langage naturel et la rigueur des structures imbriquées. Nos résultats expérimentaux suggèrent qu'il favorise une meilleure compréhension par les LLMs, en réduisant les ambiguïtés d'interprétation observées avec les formats isolés.

### 2.0.3 Formulation de la requête (Prompting)

La formulation de la requête consiste à transformer la représentation textuelle de l'automate en un prompt destiné à interroger le LLM, visant à mettre en évidence deux problématiques essentielles :

1. **Le problème d'acceptation :** Vérifier si une chaîne $w$ donnée est acceptée par l'automate en simulant une exécution des transitions.

2. **La génération de la plus courte chaîne acceptée :** Déterminer la plus courte séquence de symboles de $\Sigma$ qui conduit l'automate à un état acceptant.

Cette approche combinée permet d'évaluer la capacité du modèle à vérifier, par simulation via la fonction `process_string` (décrite à la Section 2.0.4), qu'une chaîne conduisant à un état acceptant est générée, ainsi qu'à explorer l'espace des solutions en produisant la chaîne minimale via une recherche en largeur à l'aide de la fonction `find_minimal_accepting_string` (voir Section 2.0.4). Ainsi, nous assurons que la solution retournée est à la fois correcte et optimisée.

Par ailleurs, nous appliquons différentes stratégies de prompt engineering :

— **Approche normale** : simple question naïve posée sans contexte détaillé.

— **Approche question-fermée (closed question)** : ressemble à l'approache normale mais impose au LLM de répondre uniquement par oui (Yes) ou non (No)

— **Approche assistée** : Demande au LLM de réafficher, à chaque étape (lecture de la lettre en cours, simulation de la transition correspondantes), les lettres non encore traitées dans le probléme de l'acceptance d'une séquence ou de lui guider étape par étape (Identifications de l'état intial de l'automate, des états acceptants, etc ...) dans le probléme de la génération d'un mot acceptant

— **Approche générique** : division du problème en étapes intermédiaires afin de stimuler au maximum le processus de raisonnement du LLM.

— **Approche hybride** : combinaison de la représentation du format language naturel controlé et la liste d'adjacence pour stimuler au maximum la compréhension et le raisonnement du LLM.

Pour mieux situer nos approches dans l'écosystème des techniques de prompt engineering, nous établissons ici une correspondance entre nos méthodes expérimentales et certaines stratégies reconnues dans la littérature.

| Notre approche | Technique de la littérature | Justification de la correspondance |
|---|---|---|
| Approche normale | Prompt standard | Les deux consistent en une requête directe, sans contexte ni raisonnement intermédiaire. |
| Approche question fermée | Prompt standard (variation fermée) | Version contraignante de l'approche normale, imposant une réponse binaire (oui/-non), utile pour forcer la clarté. |
| Approche assistée | Prompt socratique | Utilise un questionnement étape par étape, similaire au questionnement socratique, pour guider la réflexion du LLM à travers les étapes du raisonnement. |
| Approche générique | Chain-of-Thought (CoT) prompting | Encourage la décomposition logique du problème en sous-étapes, comme le CoT qui explicite le raisonnement du modèle. |
| Approche hybride | Prompt décomposé (Decomposed Prompting) | Combine plusieurs représentations pour diviser la tâche et stimuler un raisonnement structuré et modulaire, à l'image du prompting décomposé. |

Tableau 2.1 Correspondance entre nos approches de prompt engineering et les techniques de la littérature

Cette comparaison montre que bien que nos approches soient adaptées à notre contexte expérimental spécifique (reconnaissance ou génération avec des automates finis), elles s'inscrivent dans la continuité des stratégies de prompt engineering explorées dans des domaines variés. Cela renforce la validité de notre méthodologie, tout en mettant en valeur notre effort d'adaptation contextualisée aux contraintes posées par les tâches de traitement d'automates.

### 2.0.4 Vérification des réponses

La phase de vérification est cruciale pour assurer la robustesse de notre méthode. Elle se décompose en deux volets :

— **Vérification de l'acceptation :** Afin de valider que la chaîne $w$ est acceptée par l'automate, nous

procédons à une simulation complète du parcours de $w$ à travers les états définis par la fonction de transition $\delta$. Concrètement, nous débutons à partir de l'état initial $q_0$ et, pour chaque symbole de la chaîne, nous appliquons successivement $\delta$ afin de déterminer l'état suivant. À l'issue du traitement de la chaîne, si l'état atteint fait partie de l'ensemble $F$ des états acceptants, la chaîne est considérée comme valide. Cette méthode assure une vérification déterministe et robuste de l'exécution de la chaîne dans l'automate. En simulant minutieusement chaque transition, nous garantissons que la chaîne respecte l'ensemble des contraintes imposées par la structure formelle de l'automate.

— **Vérification de la génération optimale :** Nous vérifions que la chaîne générée par le LLM respecte l'instruction de produire la chaîne acceptée la plus courte. Pour ce faire, nous procédons en deux étapes complémentaires. D'une part, nous simulons l'exécution de l'automate sur la chaîne proposée à l'aide de la méthode `process_string` pour confirmer qu'elle est bien acceptée. D'autre part, nous utilisons une recherche en largeur (BFS) implémentée dans la fonction `find_minimal_accepting_string` afin d'obtenir la solution minimale au sens de la longueur, et nous comparons cette solution avec celle générée par le LLM. Cette démarche combinée assure que la chaîne proposée est à la fois correcte et optimale.

Ce processus de vérification est répété sur de multiples itérations afin de réduire l'incertitude statistique et d'assurer la validité des résultats expérimentaux.

Ainsi, notre méthodologie, qui combine la génération aléatoire d'automates, leur conversion en multiples représentations textuelles, la formulation ciblée de requêtes et un processus rigoureux de vérification, permet d'évaluer en profondeur la performance des LLMs dans des tâches de compréhension et de génération basées sur des structures formelles.

# CHAPITRE 3
# EXPÉRIMENTATION

Cette étude propose une évaluation systématique des capacités des modèles de langage de grande taille (LLMs) à raisonner sur des automates finis, à travers une série d'expériences soigneusement paramétrées. Nous combinons quatre stratégies de *prompt engineering* avec cinq formats de représentation, dans le but de dégager les conditions les plus favorables à une interprétation correcte par les modèles.

Le protocole expérimental repose sur les éléments suivants :

- **Environnement d'exécution :** Toutes les étapes de génération aléatoire des automates, de simulation et de vérification, ainsi que les calculs statistiques, ont été réalisées en local sur mon ordinateur de bureau au laboratoire transdisciplinaire de recherche sur les écosystèmes informatiques (LATECE) de l'UQAM. La machine utilisée était équipée d'un processeur Intel Core i7-12700 (2.10 GHz), 32 Go de mémoire vive, système d'exploitation Windows 11 Pro 64, et Python 3.12.3. Les scripts ont été développés et exécutés avec Visual Studio Code (version 1.102.1).
- **Échantillonnage :** Pour chaque valeur de la variable indépendante (nombre d'états, taille de l'alphabet ou longueur des séquences), nous générons **200 automates aléatoires** afin d'assurer la robustesse statistique des résultats. Cette taille d'échantillon a été choisie après observation empirique : en dessous de ce seuil, les résultats présentaient une variabilité excessive (bruit), tandis qu'à partir de 200 exemples, les performances du modèle se stabilisent, rendant les courbes plus fiables.
- **Paramètres de référence :**
  - Lors de l'analyse de l'influence du **nombre d'états** (de 2 à 20), l'alphabet est fixé à $\Sigma = \{a, b, c\}$ et la longueur des mots est fixée à 20. Cette configuration permet d'évaluer l'impact pur de la complexité structurelle.
  - Pour les benchmarks sur la **taille de l'alphabet** et la **longueur des chaînes**, le nombre d'états est fixé à 10. Ce choix s'appuie sur des résultats préliminaires indiquant que les LLMs tendent à adopter une dynamique stable au-delà de 8 à 10 états, rendant ce point un excellent compromis pour isoler l'effet des autres paramètres.
  - La **longueur des chaînes** testées varie de 10 à 20 caractères. Cette plage permet de couvrir à la fois des scénarios simples et des cas plus complexes, tout en garantissant que les chaînes

génèrent des comportements suffisamment longs pour mettre à l'épreuve les capacités de suivi des transitions par les modèles.

  - La proportion d'états acceptants est fixée à **50%**, répartie aléatoirement. Cette parité assure une distribution équilibrée, condition nécessaire à une mesure équitable de la précision.

- **Métriques d'évaluation :** La performance des LLMs est mesurée par la **précision moyenne** (proportion de réponses correctes) sur les 200 cas testés. Une **marge d'erreur à 95%** est calculée à l'aide de la formule suivante :

$$\text{marge} = 1{,}96 \times \sqrt{\frac{p(1-p)}{n}}$$

  où $p$ est la précision observée et $n = 200$.

- **Analyses statistiques complémentaires :** Afin d'évaluer la significativité des écarts entre les formats de représentation, une **analyse de variance (ANOVA à un facteur)** est appliquée à chaque benchmark. Elle permet de tester si les différences de performance sont dues au hasard ou à des variations systématiques induites par le format. En cas de résultat significatif ($p < 0{,}05$), un **test post-hoc de Tukey HSD** est réalisé pour identifier les paires de formats présentant une différence statistiquement significative. Cette approche renforce la robustesse des conclusions tirées à partir des moyennes observées.

Par ailleurs, deux axes d'analyse complémentaires structurent notre exploration :

- **Complexité structurelle :** En faisant croître linéairement le nombre d'états, le nombre de transitions ($|\delta| = |Q| \times |\Sigma|$) augmente proportionnellement, simulant la montée en complexité des systèmes à modéliser. Cette configuration permet de tester la capacité des LLMs à conserver un raisonnement cohérent malgré l'accroissement de la structure à manipuler.
- **Expressivité des représentations :** Nous analysons l'influence du format d'entrée du plus formel (DOT) au plus sémantique (langage naturel contrôlé), ainsi qu'un format hybride sur la performance des modèles. Cette diversité de représentations permet d'observer l'effet combiné de la structure, de la lisibilité humaine et de la compatibilité avec les architectures de type Transformer.

## 3.1 Vérification de l'acceptation d'une séquence

Cette section présente les résultats détaillés des expérimentations menées pour évaluer la capacité du modèle à vérifier si une chaîne donnée est acceptée par un automate fini. L'étude s'articule en trois volets :

— **Section 3.1.1** : Impact du nombre d'états.
— **Section 3.1.2** : Impact de la taille de l'alphabet.
— **Section 3.1.3** : Impact de la longueur des chaînes d'entrée.

Chaque sous-section explore l'évolution des performances du modèle pour une approche de questionnement donnée (assistée, générique, normale ou question fermée), en maintenant les autres paramètres constants pour isoler l'effet étudié. **Temps d'exécution :** chaque série de 200 tests a nécessité un temps d'exécution important :

— Benchmark sur le **nombre d'états** : environ **9 à 10 heures** pour couvrir les automates de 2 à 20 états.
— Benchmark sur la **taille de l'alphabet** : **15 à 16 heures**, les alphabets plus riches augmentant la complexité symbolique.
— Benchmark sur la **longueur de chaîne** : temps estimé entre **11 et 13 heures**, la complexité de traitement augmentant proportionnellement avec la longueur des séquences à analyser.

Ces durées reflètent non seulement le volume de requêtes effectuées mais également le coût de traitement cognitif imposé au modèle par les transitions longues et les entrées plus variées.

### 3.1.1 Effet du nombre d'états sur la vérification d'acceptation

Les figures 3.1 à 3.4 présentent les courbes de précision obtenues lors de la tâche de vérification d'acceptation, selon le nombre d'états de l'automate. Chaque graphique illustre l'évolution de la précision en fonction de la complexité croissante, pour une stratégie de questionnement donnée (approche assistée, générique, normale ou fermée).

Le tableau 3.6 présente une synthèse des résultats agrégés pour l'ensemble des paramètres, avec les précisions moyennes et les marges d'erreur à 95%. Cette synthèse permet d'identifier rapidement les combinaisons format/approche les plus efficaces, ainsi que les zones critiques où les modèles peinent à raisonner correctement.

#### 3.1.1.0.1 Prompt utilisé – Approche assistée

```
Here is the automaton in {format_type} format: {automaton_description}
Does the automaton accept the string '{mot}'?
```

Decompose the string processing step by step:

- At each step, indicate the current state and the remaining input.
- Apply the transition corresponding to the next symbol.
- Finally, answer only "yes" or "no".

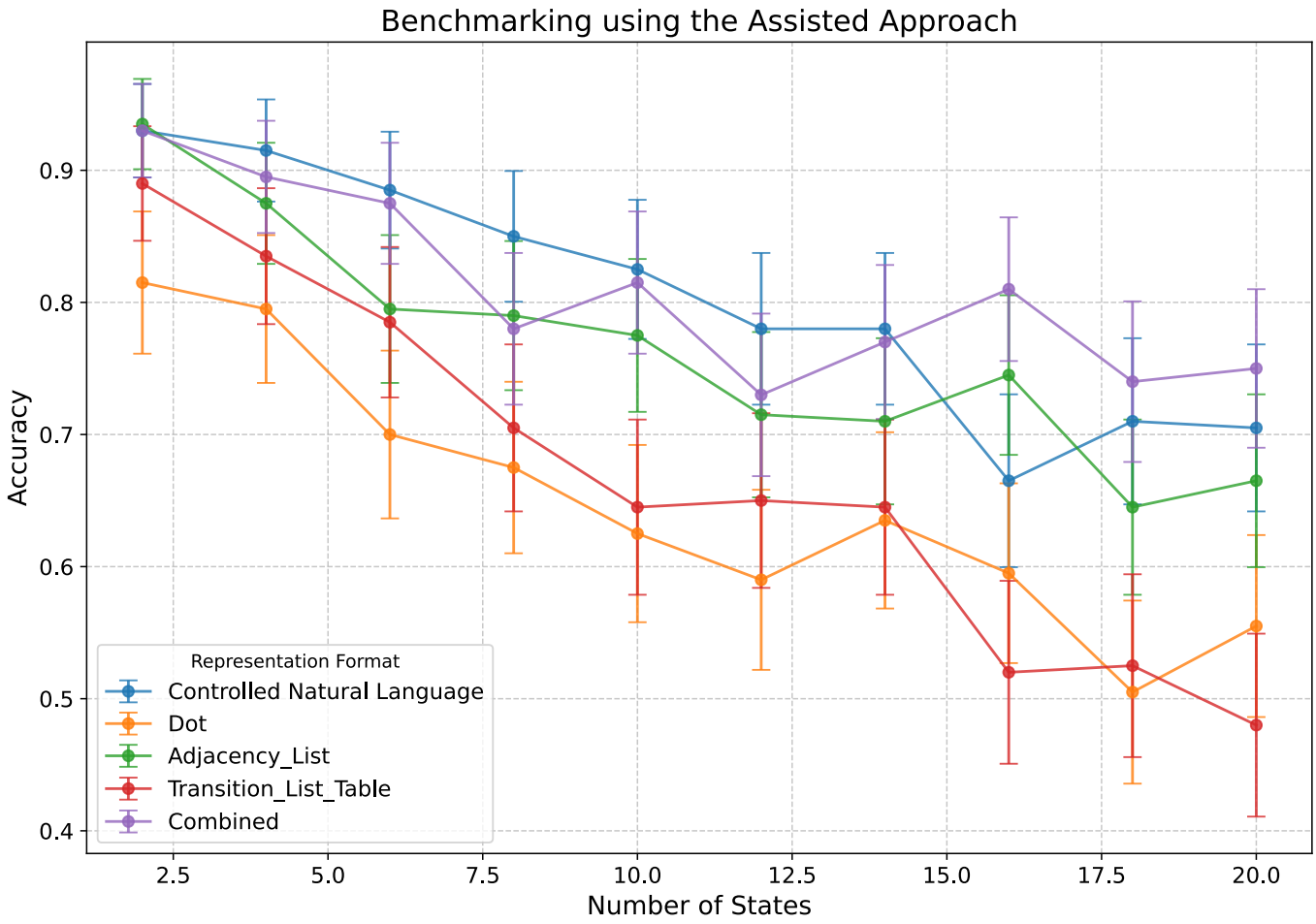


Figure 3.1 Performance de l'approche assistée.

Les résultats (figure 3.1) obtenus avec l'approche Assistée montrent des performances globalement élevées et une variabilité en fonction du format de représentation. Nous observons les tendances suivantes :

- **Liste d'adjacence (Adjacency_List) :** La précision atteint un maximum de 0.935 pour un automate de 2 états, et descend jusqu'à 0.645 pour 20 états. Ce format offre donc d'excellents résultats pour les automates de petite taille, mais la variabilité augmente avec la complexité, entraînant une précision minimale relativement basse.
- **Format combiné (Combined) :** Les précisions varient de 0.930 (maximum) à 0.730 (minimum). Le format combiné semble présenter une meilleure stabilité que la liste d'adjacence seule, avec des valeurs minimales plus élevées pour les automates de grande taille.

— **Langage naturel contrôlé (Controlled Natural Language) :** Ce format atteint également un maximum de 0.930 et présente un minimum de 0.665. Les résultats indiquent une bonne robustesse du format langage naturel contrôlé, favorisant l'interprétation du modèle grâce à sa dimension sémantique.

— **DOT :** La précision observée avec le format DOT est légèrement inférieure, avec un maximum de 0.815 et un minimum de 0.505. Cette performance réduite peut s'expliquer par la complexité inhérente aux représentations graphiques, qui semblent moins adaptées aux capacités d'interprétation des LLMs dans le cadre de cette approche.

— **Tableau de transitions (Transition_List_Table) :** Les résultats oscillent entre un maximum de 0.890 et un minimum de 0.480. Bien que les performances maximales soient bonnes, la variabilité élevée et la précision minimale laissent penser que ce format peut être moins adapté pour des automates complexes.

**Analyse statistique**

Afin de confirmer la significativité des différences observées entre les formats de représentation, une analyse de variance (ANOVA à un facteur) a été menée.

| Source | SS | DF | MS | F | p-valeur | $\eta^2$ (np2) |
|---|---|---|---|---|---|---|
| Format | 0.231 | 4 | 0.0577 | 5.73 | <0.001 | 0.34 |
| Résidu (erreur) | 0.453 | 45 | 0.0101 | – | – | – |

Tableau 3.1 ANOVA – Effet du format dans l'approche Assistée

Les résultats de l'ANOVA (tableau 3.1) indiquent une différence significative entre les formats ($F = 5{,}73, p < 0{,}001$), ce qui suggère que le format de représentation influence effectivement les performances du modèle. La ligne du *résidu* (erreur intra-groupe) représente la variabilité non expliquée par le facteur « Format » ; elle ne possède pas de valeur de $F$ ou de $p$ car ces mesures ne sont calculées que pour les effets testés.

Un test post-hoc de Tukey (tableau 3.2) a ensuite été réalisé afin d'identifier les paires de formats pour lesquelles la différence est statistiquement significative. Les résultats ci-dessous présentent uniquement les comparaisons ayant une $p$-valeur inférieure à 0,05 :

Chaque ligne du test de Tukey présente la différence moyenne observée entre les deux formats comparés, la

| Format A | Format B | Différence | p-valeur | Statistique $T$ | Hedges' $g$ |
|---|---|---|---|---|---|
| Langage naturel contrôlé | DOT | +0.156 | 0.0098 | 3.47 | 1.55 |
| Langage naturel contrôlé | Tableau de transitions | +0.137 | 0.0304 | 3.04 | 1.11 |
| Format combiné | DOT | +0.161 | 0.0072 | 3.58 | 1.79 |
| Format combiné | Tableau de transitions | +0.142 | 0.0228 | 3.15 | 1.24 |

Tableau 3.2 Test post-hoc de Tukey

valeur $p$, la statistique $T$ (analogue au score $t$ dans un test de Student) ainsi que la taille de l'effet (*Hedges' $g$*), permettant d'apprécier l'ampleur des écarts mesurés. À titre d'exemple, des valeurs de $g > 1$ sont généralement considérées comme de très grands effets.

Ces résultats confirment quantitativement l'intuition selon laquelle les formats offrant à la fois une structure explicite et une richesse sémantique (langage naturel contrôlé, combiné) sont mieux compris par les LLMs dans l'approche Assistée. À l'inverse, les formats plus visuels ou hiérarchisés, tels que DOT ou les tableaux de transitions, affichent des performances moindres, probablement dues à leur éloignement du paradigme textuel.

En synthèse, l'approche **assistée** affiche les meilleures performances avec les formats **langage naturel contrôlé** et **combiné**, et des résultats plus stables face à l'augmentation du nombre d'états. En revanche, les formats **DOT** et **tableau de transitions** présentent une baisse plus marquée de la précision pour des automates de grande taille, soulignant l'importance du choix du format dans le succès du *prompt engineering*.

#### 3.1.1.0.2 Prompt utilisé – Approche générique

```
Here is the automaton in {format_type} format :{automaton_description}
Does the automaton accept the string '{mot}'?
Break down the reasoning into intermediate steps.
```

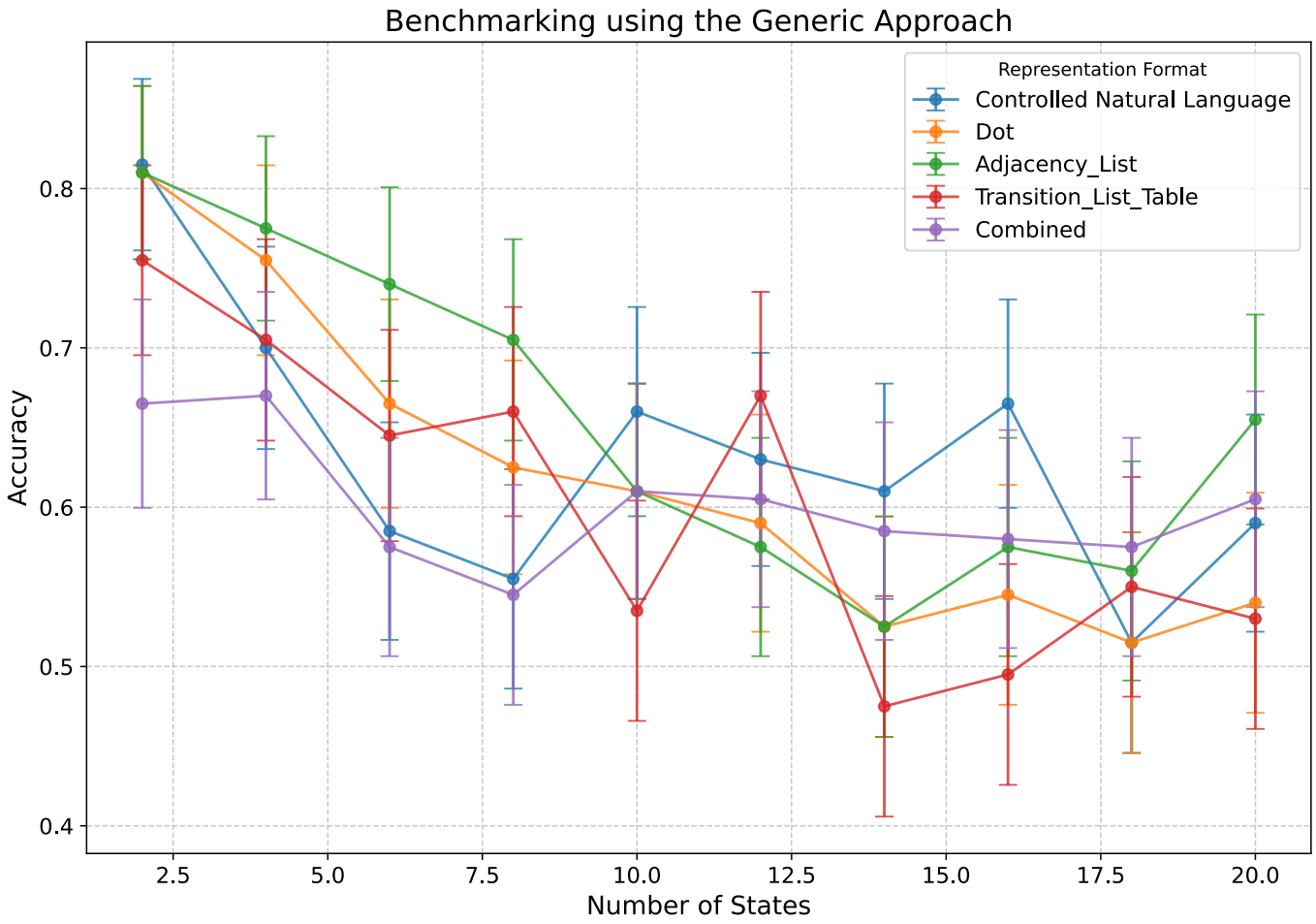


Figure 3.2 Performance de l'approche générique

Les résultats (figure 3.2) obtenus avec l'approche Generic, qui sollicite le modèle pour décomposer le problème en étapes intermédiaires sans fournir de guidance explicite, montrent des performances globalement modérées et relativement proches d'un format à l'autre :

- **Format Liste d'adjacence :** La précision atteint un maximum de 0.810 pour 2 états et descend à 0.525 pour 20 états. Cette tendance suggère que, bien que le modèle parvienne à traiter efficacement des automates simples sous ce format, la complexité accrue des automates de grande taille entraîne une dégradation notable de la performance.
- **Format Combiné :** Les précisions varient de 0.670 (pour 4 états) à 0.545 (pour 8 états), avec une stabilisation autour de 0.605 pour 20 états. L'intégration de plusieurs représentations semble offrir

une performance modérée, mais ne parvient pas à égaler les résultats obtenus avec le format Liste d'adjacence pour les automates les plus simples.

— **Format Langage naturel contrôlé :** Ce format affiche une précision maximale de 0.815 et une précision minimale de 0.515, indiquant que la richesse contextuelle apportée par ce format est bénéfique pour des automates de petite taille, mais que son efficacité diminue avec la complexité.

— **Format DOT :** La représentation graphique (DOT) fournit des performances comparables à celles de la Liste d'adjacence, avec des précisions variant entre 0.810 et 0.515. Toutefois, l'interprétation des données visuelles peut présenter des défis supplémentaires pour le modèle.

— **Format Tableau de transitions :** Ce format, plus structuré, présente la plus grande variabilité avec une précision maximale de 0.755 et une précision minimale de 0.475. Ces résultats suggèrent que l'approche Generic a plus de difficultés à traiter un format très numérisé et hiérarchisé.

**Analyse statistique**

Afin de vérifier la significativité des écarts de performance entre les différents formats dans l'approche *Generic*, une analyse de variance (ANOVA à un facteur) a été réalisée.

| Source | SS | DF | MS | F | p-valeur | $\eta^2$ (np2) |
|---|---|---|---|---|---|---|
| Format | 0.0191 | 4 | 0.0048 | 0.63 | 0.644 | 0.05 |
| Résidu (erreur) | 0.3403 | 45 | 0.0076 | – | – | – |

Tableau 3.3 ANOVA – Effet du format dans l'approche générique

Les résultats de l'ANOVA révèlent qu'il n'y a **pas de différence significative** entre les performances moyennes obtenues selon le format utilisé ($p = 0.644$). Le faible effet mesuré ($\eta^2 = 0.05$) indique une variance très modeste expliquée par le format. Aucun test post-hoc n'a révélé de contraste significatif entre les formats. Les différences observées relèvent donc d'une variabilité intrinsèque au comportement du modèle.

L'approche *générique*, qui repose sur une décomposition implicite du raisonnement sans guidance explicite, conduit à des performances modérées et relativement uniformes entre les formats. Contrairement à l'approche *Assisted*, ici aucune structure n'émerge clairement comme étant statistiquement supérieure. Cela suggère que, sans cadrage fort, les LLMs peinent à tirer parti des caractéristiques spécifiques des formats textuels, graphiques ou structurés.

#### 3.1.1.0.3 Prompt utilisé – Approche normale

```
Here is the automaton in {format_type} format: {automaton_description}
Does the automaton accept the string '{mot}'?
```

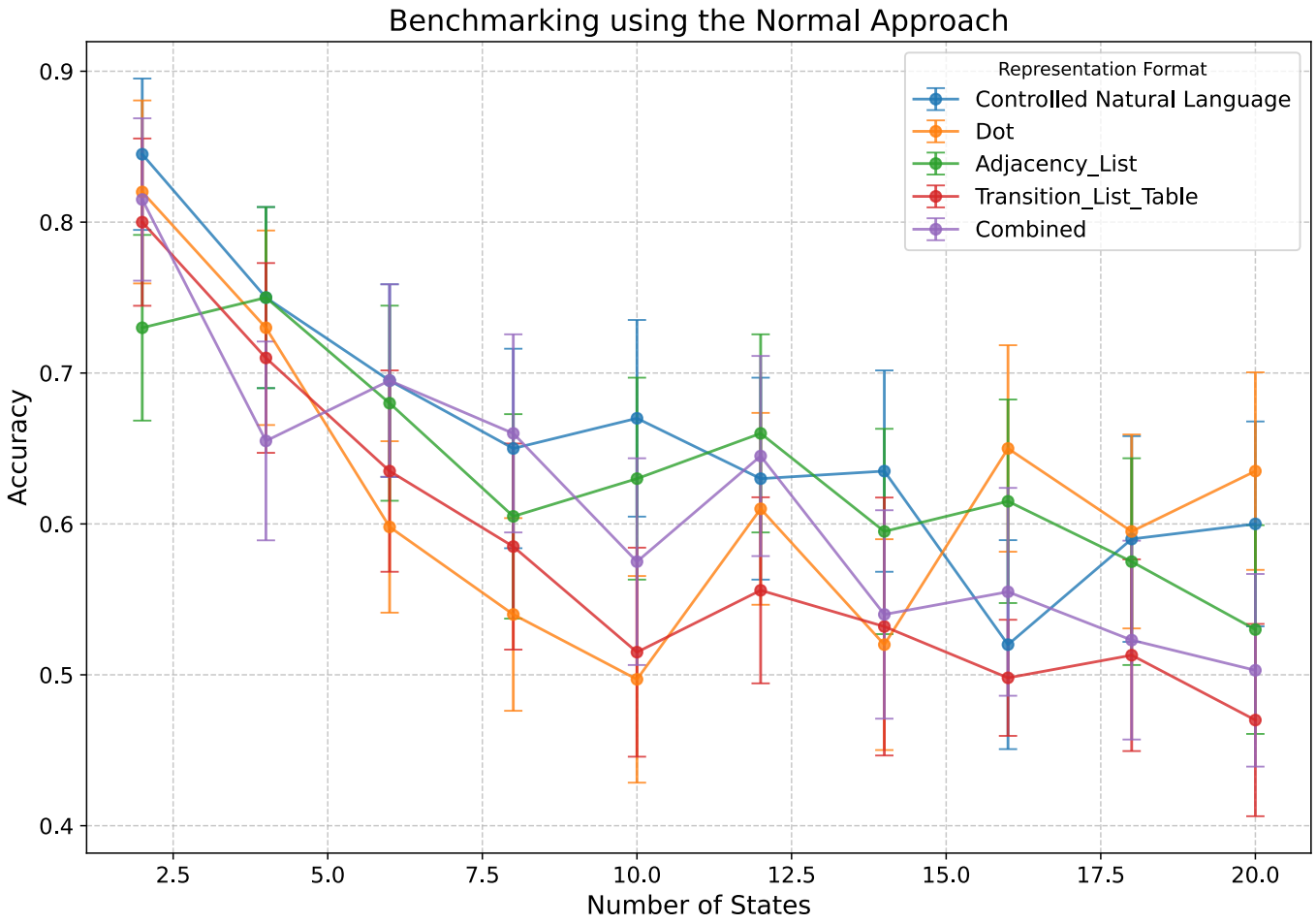


Figure 3.3 Performance de l'approche normale.

Les résultats (figure 3.3) obtenus avec l'approche normale, qui consiste à poser la question directement sans guider le modèle, montrent des performances modérées, avec une variabilité modérée selon le format de représentation utilisé. Plus précisément :

— **Liste d'adjacence :** La précision varie entre 0.730 et 0.750, avec un maximum à 0.750 et un mini-

mum à 0.530 pour un automate de 20 états. Cela indique que, bien que ce format permette une bonne représentation de la structure de l'automate, la performance décroît pour des automates plus complexes.

— **Format combiné :** La précision atteint un maximum de 0.815 pour de petits automates, mais chute jusqu'à 0.503 pour des automates plus grands, suggérant que l'intégration de plusieurs formats peut améliorer la compréhension pour de petits jeux de données, alors qu'elle devient moins efficace lorsque la complexité augmente.

— **Langage Naturel Contrôlé :** Ce format affiche des résultats relativement stables, avec une précision maximale de 0.845 et une valeur minimale de 0.520. La richesse contextuelle du format langage naturel contrôlé semble en effet favoriser une meilleure interprétation par le modèle.

— **Format DOT :** La représentation graphique via DOT produit des précisions variant entre 0.820 et 0.497, ce qui indique que, malgré son intuitivité visuelle, la complexité du graphe peut rendre l'interprétation moins fiable pour des automates de grande taille.

— **Tableau de transitions :** Le format le plus structuré se traduit par des performances plus faibles, avec des précisions allant de 0.800 à 0.470, ce qui suggère que cette approche, bien que rigoureuse, est moins adaptée à une interprétation par le LLM dans le cadre de questions directes.

**Analyse statistique**

Pour évaluer si les différences observées entre les formats sont significatives dans l'approche *Normale*, une ANOVA à un facteur a été réalisée.

| Source | SS | DF | MS | F | p-valeur | $\eta^2$ (np2) |
|---|---|---|---|---|---|---|
| Format | 0.0324 | 4 | 0.0081 | 0.95 | 0.44 | 0.08 |
| Résidu (erreur) | 0.3832 | 45 | 0.0085 | – | – | – |

Tableau 3.4 ANOVA – Effet du format dans l'approche normale

Les résultats indiquent qu'aucune différence significative n'est détectée entre les formats ($p = 0.44$). Le faible effet mesuré ($\eta^2 = 0.08$) suggère que le type de représentation a un impact limité sur la performance du modèle dans ce cadre.

Le test post-hoc de Tukey n'a révélé aucune paire de formats avec une différence significative ($p > 0.34$

pour toutes les comparaisons), confirmant que la variabilité observée est probablement attribuable au bruit expérimental ou à des fluctuations naturelles dans les réponses du modèle.

En résumé, l'approche *normale* montre que la performance du LLM dépend modérément du format de représentation. Aucune configuration ne se démarque de manière significative. Les formats offrant un peu plus de contexte sémantique peuvent faciliter l'interprétation pour des automates simples, mais cet avantage disparaît pour des structures plus complexes. À l'inverse, les formats très structurés ou graphiques affichent des performances plus hétérogènes sans être significativement inférieurs.

#### 3.1.1.0.4 Prompt utilisé – Approche question fermée

```
Here is the automaton in {format_type} format : {automaton_description}
Does the automaton accept the string '{mot}'? Please answer only "yes" or "no".
```

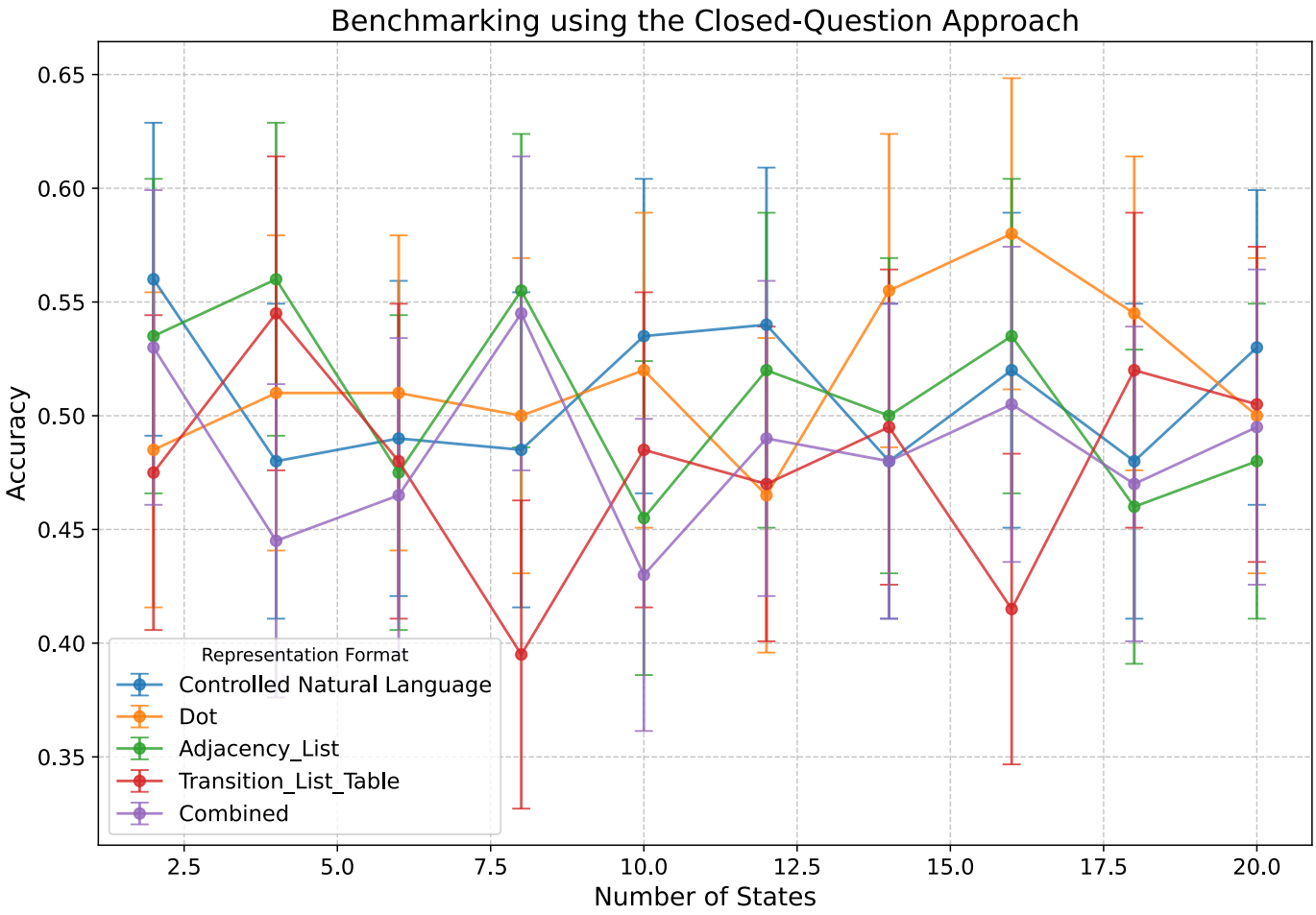


Figure 3.4 Performance de l'approche question fermée

Nous observons (figure 3.4) que l'approche question fermée affiche des performances relativement faibles et peu discriminantes dans la vérification d'une séquence, quelle que soit la représentation utilisée. En effet, les précisions mesurées restent globalement faibles, indiquant une capacité limitée du modèle à répondre efficacement par une réponse binaire dans ce contexte.

— **Liste d'adjacence :** La précision varie de 0.455 à 0.560, avec un maximum de 0.560 et un minimum de 0.455.

— **Format combiné :** Les résultats s'étendent de 0.430 à 0.545, le score le plus élevé étant 0.545 et le plus faible 0.430.

— **Langage Naturel Contrôlé :** La précision oscille entre 0.480 et 0.560, avec des valeurs maximales et

minimales respectivement de 0.560 et 0.480.

— **Format DOT :** La plage de précision est de 0.465 à 0.580, le meilleur résultat atteignant 0.580 et le plus bas 0.465.

— **Tableau de transitions :** Les précisions varient de 0.395 à 0.545, avec une précision maximale de 0.545 et une minimale de 0.395.

**Analyse statistique.** Une analyse de variance (ANOVA) a été réalisée pour évaluer si les différences observées entre les formats sont statistiquement significatives (tableau 3.5).

| Source | SS | DF | MS | F | p-valeur | $\eta^2$ (np2) |
|---|---|---|---|---|---|---|
| Format | 0.0112 | 4 | 0.0028 | 2.02 | 0.11 | 0.15 |
| Résidu (erreur) | 0.0621 | 45 | 0.0014 | – | – | – |

Tableau 3.5 ANOVA – Effet du format dans l'approche Closed Question

Les résultats de l'ANOVA indiquent qu'aucune différence significative n'est détectée entre les formats ($p = 0.11$). Bien que la taille de l'effet soit modérée ($\eta^2 = 0.15$), elle n'est pas suffisante pour conclure à un impact clair du format sur les performances.

Le test post-hoc de Tukey confirme ce résultat : toutes les comparaisons entre paires de formats présentent des $p$-valeurs supérieures à 0.15, ce qui invalide l'hypothèse de différences significatives. Cela renforce l'idée que la nature binaire de la tâche, combinée à l'absence d'explication demandée au modèle, réduit son efficacité indépendamment du format utilisé.

Ces résultats confirment que l'approche question fermée, bien qu'intuitive et simple à implémenter, n'exploite pas le plein potentiel de raisonnement des LLMs. L'absence de différences significatives entre les formats suggère que, sans contextualisation ou raisonnement explicite, le format de l'automate n'apporte que peu d'avantages dans ce type d'interaction.

Tableau 3.6 synthétise la précision moyenne obtenue, sur l'ensemble des tailles d'automates (de 2 à 20 états), pour chacune des approches de requêtage et pour chaque format de représentation, indépendamment de la significativité statistique intra-approche.

| Approche | Liste d'adjacence | Tableau de transitions | DOT | Langage naturel contrôlé | Combiné |
|---|---|---|---|---|---|
| Question fermée | 0.48 ± 0.069 | 0.51 ± 0.067 | 0.49 ± 0.065 | 0.47 ± 0.069 | 0.48 ± 0.068 |
| Normale | 0.59 ± 0.068 | 0.60 ± 0.067 | 0.61 ± 0.064 | 0.63 ± 0.065 | 0.61 ± 0.065 |
| Assistée | **0.77 ± 0.058** | **0.67 ± 0.063** | **0.65 ± 0.065** | **0.80 ± 0.059** | **0.81 ± 0.059** |
| Générique | 0.65 ± 0.066 | 0.60 ± 0.066 | 0.62 ± 0.067 | 0.63 ± 0.065 | 0.60 ± 0.067 |

Tableau 3.6 Précision moyenne par format de représentation selon le nombre d'états

Nous constatons que :

- L'approche **question fermée** affiche des performances médiocres, avec des précisions se situant autour de 0.48 à 0.51, ce qui est proche d'une réponse aléatoire. Cela suggère que limiter le modèle à une réponse binaire ne permet pas d'extraire suffisamment d'informations détaillées sur le fonctionnement de l'automate.
- L'approche **normale** améliore la performance avec des précisions comprises entre 0.59 et 0.63, indiquant que la question directe laisse plus de latitude au modèle pour exploiter ses capacités de raisonnement, même si elle n'atteint pas les meilleurs résultats.
- L'approche **assistée** atteint les meilleurs scores, avec des précisions maximales allant jusqu'à 0.81, particulièrement pour les formats en Langage Naturel Contrôlé et Combiné. Cette méthode, qui guide le LLM étape par étape dans le processus de reconnaissance, permet d'obtenir une performance nettement supérieure, quelle que soit la représentation utilisée.
- Enfin, l'approche **générique** se situe en position intermédiaire, avec des scores de 0.60 à 0.65, ce qui suggère que la décomposition du problème en étapes intermédiaires, sans guidance explicite, améliore la performance par rapport à la méthode normale, mais reste inférieure à l'approche assistée.

Ces résultats démontrent clairement que la stratégie de prompt adoptée influence fortement la précision du LLM dans l'analyse des automates.

Les sections suivantes (Sections 3.1.2 et 3.1.3) prolongent cette analyse en étudiant l'impact de deux autres facteurs : la taille de l'alphabet (Section 3.1.2) et la longueur de la chaîne d'entrée (Section 3.1.3). Les mêmes approches de sollicitation (assistée, générique, normale et question fermée) sont systématiquement réutilisées pour garantir la comparabilité des résultats.

### 3.1.2 Effet de la taille de l'alphabet

Cette sous-section explore l'effet de la taille de l'alphabet sur la capacité des modèles à raisonner correctement sur l'acceptation d'une séquence. Pour cela, le nombre d'états est fixé à 10 et la longueur des chaînes d'entrée est maintenue à 20 symboles. Seule la taille de l'alphabet varie, entre 3 et 26 lettres.

Les Figures 3.5 à 3.8 présentent les courbes de précision obtenues selon les quatre stratégies de questionnement. Chaque figure contient cinq courbes, correspondant aux cinq formats de représentation étudiés.

Nous observons que les performances générales diminuent progressivement avec l'augmentation de la taille de l'alphabet, en raison de la complexité symbolique croissante. Néanmoins, cette dégradation varie fortement selon la stratégie de prompt et le format de représentation utilisé.

#### 3.1.2.0.1 Performances globales par approche :

**Approche assistée** : Les résultats (figure 3.5) demeurent robustes même pour les alphabets étendus. Le format combiné atteint la précision moyenne la plus élevée (0.899), suivi de près par le langage naturel controlé (0.882) et la Liste d'adjacence (0.839). Le format DOT (0.789) reste modérément performant, tandis que la Table de transitions, plus rigide, obtient un score moyen de 0.733.

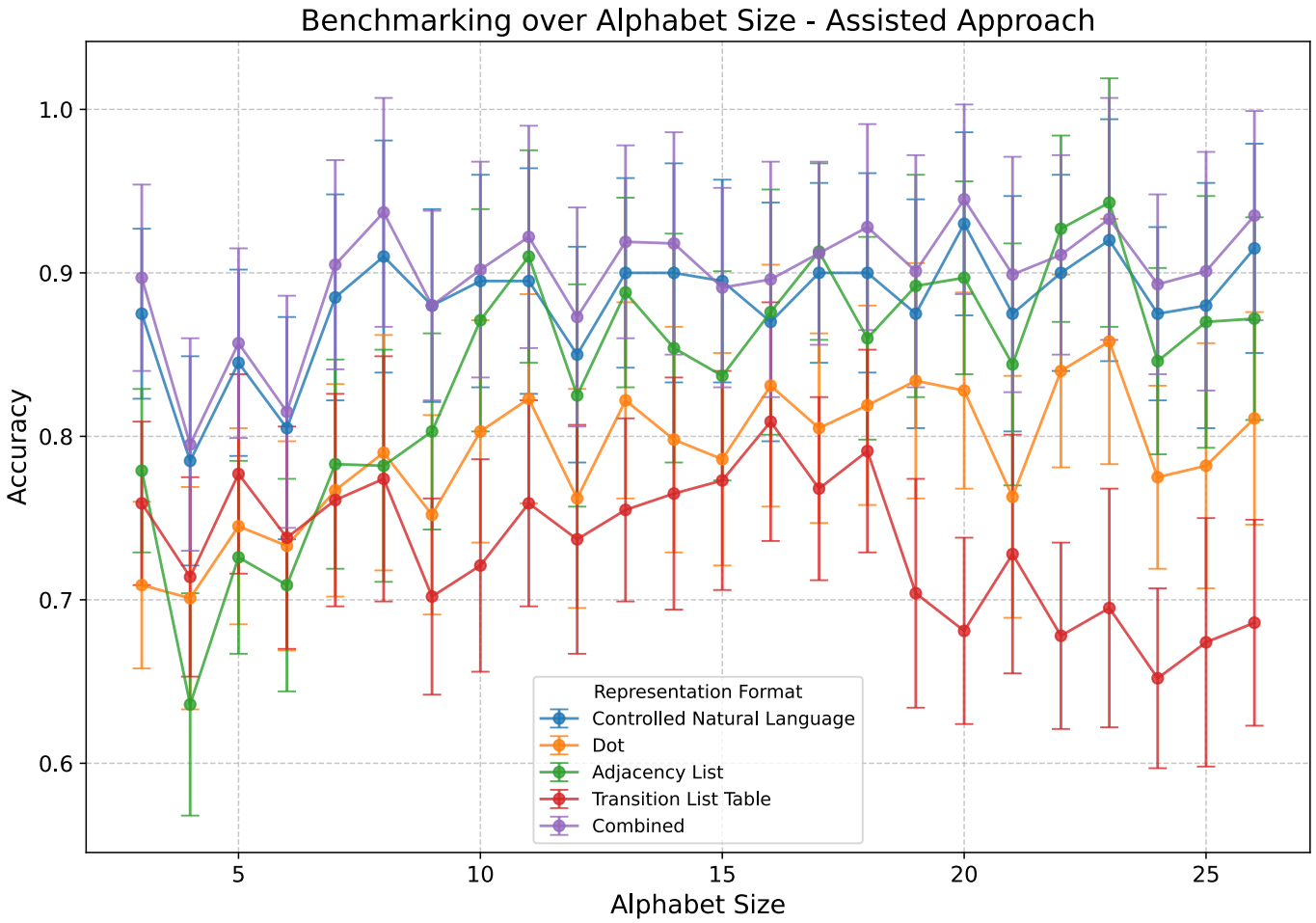


Figure 3.5 Performance de l'approche assistée

Afin de vérifier si les différences de précision entre les formats sont statistiquement significatives pour l'approche assistée, une analyse de variance (ANOVA à un facteur) a été réalisée (tableau 3.7)

| Source | SS | DF | MS | F | p-valeur | $\eta^2$ (np2) |
|---|---|---|---|---|---|---|
| Format | 0.443 | 4 | 0.1107 | 48.81 | <0.001 | 0.63 |
| Résidu | 0.261 | 115 | 0.0023 | – | – | – |

Tableau 3.7 ANOVA pour l'effet du format (Approche assistée – alphabet variable)

Les résultats indiquent une différence hautement significative entre les formats ($F = 48.81$, $p < 0.001$), avec un effet de taille important ($\eta^2 = 0.63$).

Un test post-hoc de Tukey révèle que :

- Le **format combiné** surpasse tous les autres de manière significative ($p < 0.001$), avec des écarts moyens allant jusqu'à +0.165 (vs. Table de transitions).
- Le **format langage naturel contrôlé** est lui aussi nettement meilleur que DOT et la Table de transitions (avec $p < 0.01$).
- Le format Liste d'adjacence obtient des scores intermédiaires : il est significativement meilleur que DOT ($p = 0.0035$) et la tableau de transitions ($p < 10^{-10}$), mais moins performant que langage naturel contrôlé et Combiné.

Ces résultats confirment que les représentations riches en structure (Combiné) ou en langage naturel contrôlé facilitent l'analyse par les LLMs, même lorsque la taille de l'alphabet augmente. À l'inverse, les formats purement structurels comme les tableaux de transitions voient leur précision chuter de manière significative dans un espace symbolique plus large.

**Approche générique** : Les précisions (figure 3.6) chutent d'environ 10 à 12% par rapport à l'approche assistée, avec des moyennes globales situées entre 0.715 (Combiné) et 0.615 (Table de transitions).

Une ANOVA a été menée afin d'évaluer l'influence du format de représentation sur les performances de l'approche générique. (tableau 3.8)

| Source | SS | DF | MS | F | p-valeur | $\eta^2$ (np2) |
|---|---|---|---|---|---|---|
| Format | 0.481 | 4 | 0.1203 | 89.30 | <0.001 | 0.76 |
| Résidu | 0.155 | 115 | 0.0013 | – | – | – |

Tableau 3.8 ANOVA pour l'effet du format (Approche générique – alphabet variable)

Les résultats confirment une différence hautement significative entre les formats ($F = 89.30$, $p < 0.001$), avec un très fort effet de taille ($\eta^2 = 0.76$), suggérant que le format joue un rôle central dans les performances du modèle.

Le test post-hoc de Tukey révèle que :

- Le **format combiné** et le **langage naturel controlé** sont significativement meilleurs que tous les autres formats ($p < 10^{-8}$), sauf entre eux ($p = 0.99$).

— Le format **DOT** est significativement inférieur à tous les formats narratifs ou hybrides ($p < 0.001$), et reste cependant meilleur que le format **tableau de transitions**.

— Le **tableau de transitions** est le format le moins performant, avec un écart moyen de –0.17 par rapport au langage naturel controlé.

Ces résultats confirment que, même sans guidance explicite, les LLMs parviennent mieux à traiter des formats riches et sémantiques (combiné, langage naturel controlé), tandis que les représentations tabulaires imposent des contraintes qui freinent leur capacité de généralisation.

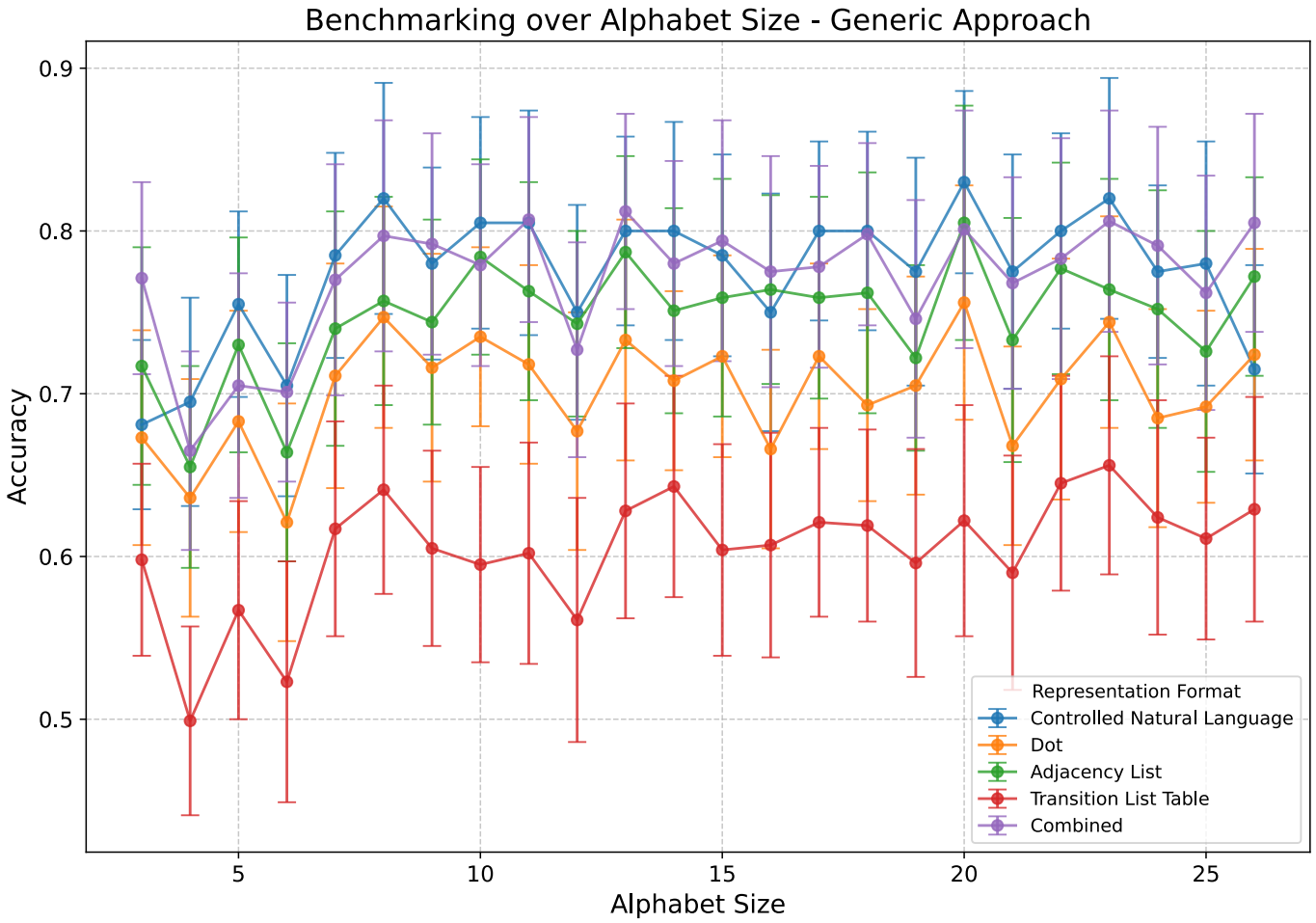


Figure 3.6 Performance de l'approche générique

**Approche normale** : La tendance décroissante est plus marquée (figure 3.7), les performances tombant jusqu'à 0.521 pour le langage naturel contrôlé, et autour de 0.55 à 0.57 pour les autres formats.

Les résultats (tableau 3.9) montrent un effet significatif du format de représentation sur la précision ($F =$

| Source | SS | DF | MS | F | p-valeur | $\eta^2$ (np2) |
|---|---|---|---|---|---|---|
| Format | 0.266 | 4 | 0.0665 | 14.93 | <0.001 | 0.34 |
| Résidu | 0.512 | 115 | 0.0045 | – | – | – |

Tableau 3.9 ANOVA pour l'effet du format (Approche normale – alphabet variable)

$14.93$, $p < 0.001$), avec un effet de taille modéré ($\eta^2 = 0.34$).

Le test post-hoc de Tukey indique que :

- La **liste d'adjacence** surpasse significativement les formats **DOT** ($p < 10^{-6}$) et **Tableau de transitions** ($p < 10^{-7}$), avec des écarts de précision moyens de plus de 0.12.
- Les formats **combiné** et **langage naturel contrôlé** occupent une position intermédiaire, ne différant pas significativement entre eux ($p = 0.33$), ni avec la liste d'adjacence.
- Les performances des formats **DOT** et **tableau de transitions** restent inférieures, avec une différence non significative entre eux ($p = 0.99$), suggérant une difficulté partagée à traiter des formats structurés sans guidage explicite.

Ces résultats corroborent les observations empiriques selon lesquelles les formats narratifs ou semi-structurés permettent une meilleure exploitation des capacités de raisonnement implicite du modèle, tandis que les formats strictement tabulaires ou graphiques limitent cette faculté.

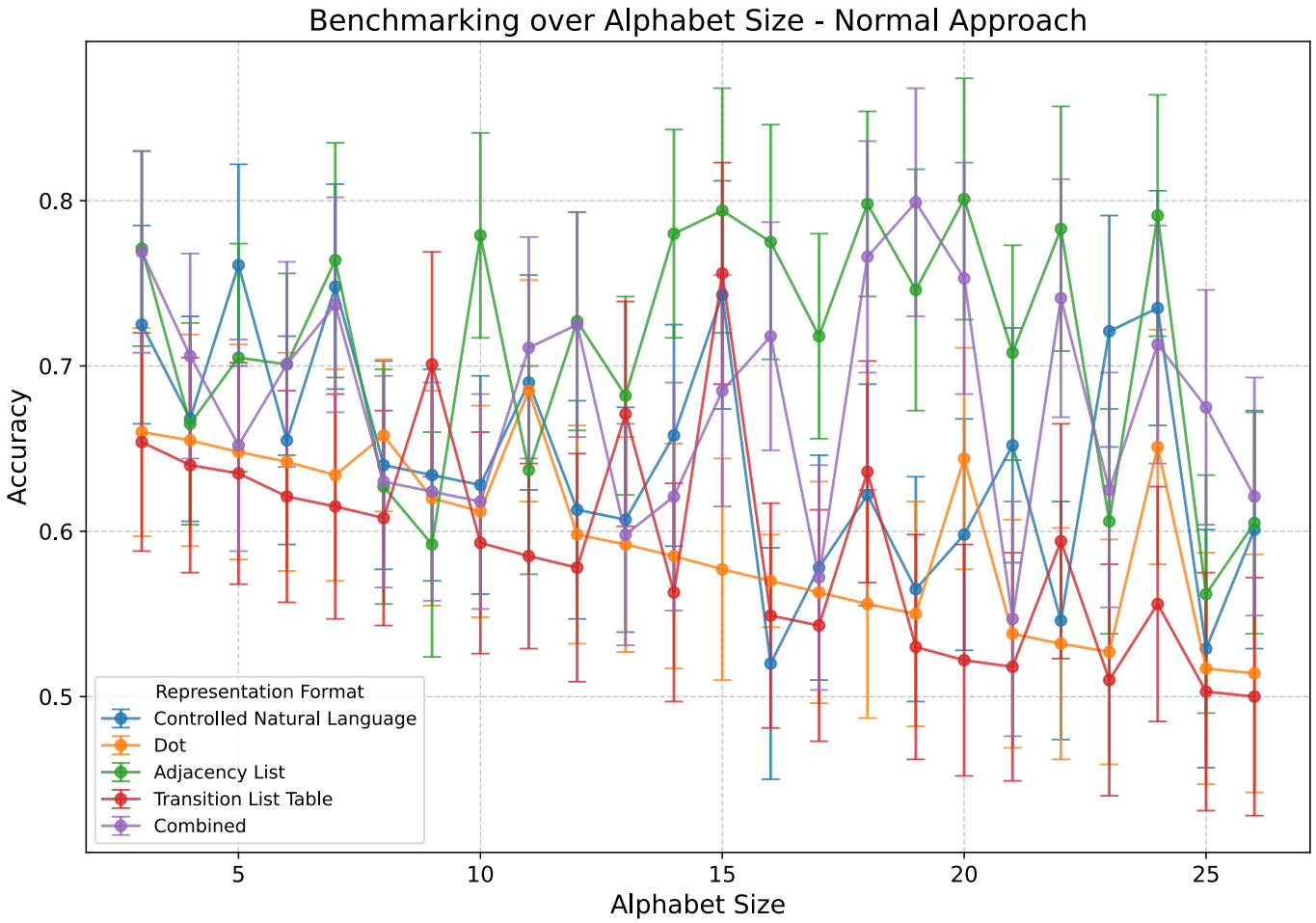


Figure 3.7 Performance de l'approche normale

**Approche question fermée** : Sans surprise, cette approche affiche les résultats (figure 3.8) les plus faibles, frôlant parfois les performances aléatoires, en particulier avec les formats structurés comme le tableau de transitions (0.484 de moyenne).

| Source | SS | DF | MS | F | p-valeur | $\eta^2$ (np2) |
|---|---|---|---|---|---|---|
| Format | 0.029 | 4 | 0.0073 | 3.09 | 0.0185 | 0.10 |
| Résidu | 0.272 | 115 | 0.0024 | – | – | – |

Tableau 3.10 ANOVA pour l'effet du format Approche Closed Question

Les résultats de l'ANOVA (tableau 3.10) révèlent une différence significative entre les formats ($F = 3.09$,

$p = 0.0185$), bien que l'effet soit modéré ($\eta^2 = 0.10$).

L'analyse post-hoc de Tukey indique une seule différence significative :

— Le format **tableau de transitions** est significativement moins performant que la **liste d'adjacence** ($p = 0.035$), avec une différence moyenne de précision de 0.041.

Aucune autre comparaison n'a révélé de différences statistiquement significatives ($p > 0.05$), ce qui suggère que les performances restent relativement homogènes entre les formats dans cette approche. Cela corrobore l'idée que le modèle, contraint à répondre par *« yes »* ou *« no »*, n'exploite pas efficacement les différences structurelles des représentations, limitant ainsi l'effet du format sur la performance.

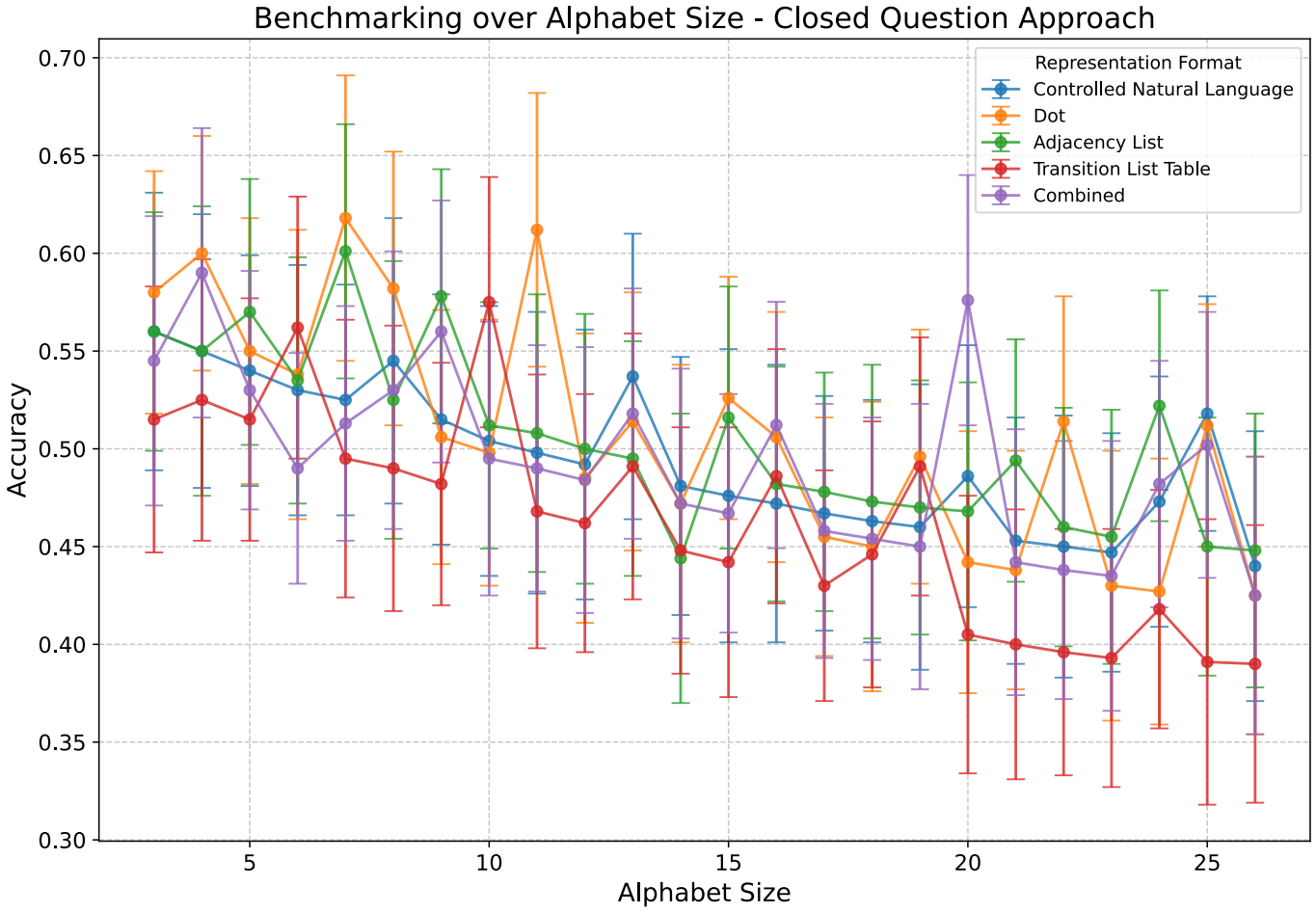


Figure 3.8 Performance de l'approche question fermée

#### 3.1.2.0.2 Résumé des résultats

| Approche | Liste d'adjacence | Tableau de transitions | DOT | Langage naturel contrôlé | Combiné |
|---|---|---|---|---|---|
| Question fermée | 0.495 ± 0.067 | 0.484 ± 0.066 | 0.490 ± 0.065 | 0.502 ± 0.064 | 0.492 ± 0.066 |
| Normale | 0.629 ± 0.060 | 0.576 ± 0.063 | 0.568 ± 0.062 | 0.587 ± 0.062 | 0.579 ± 0.061 |
| Assistée | **0.839 ± 0.056** | **0.733 ± 0.058** | **0.789 ± 0.057** | **0.882 ± 0.055** | **0.899 ± 0.055** |
| Générique | 0.701 ± 0.060 | 0.638 ± 0.061 | 0.675 ± 0.062 | 0.728 ± 0.059 | 0.715 ± 0.059 |

Tableau 3.11 Précision moyenne par format de représentation selon la taille de l'alphabet.

Nous constatons que (tableau 3.11), comme pour l'analyse fondée sur le nombre d'états, l'approche assistée permet de préserver une bonne stabilité des performances, même dans un espace symbolique plus vaste. L'impact du format de représentation reste déterminant : les formats narratifs et hybrides facilitent l'interprétation par le modèle, tandis que les formats structurels rigides demeurent plus sensibles à l'augmentation de la complexité.

### 3.1.3 Effet de la longueur des chaînes d'entrée

Cette sous-section évalue l'impact de la longueur des chaînes d'entrée sur la capacité du modèle à vérifier correctement l'acceptation par un automate. Pour isoler cet effet, nous avons fixé le nombre d'états à 10 et la taille de l'alphabet à 3 lettres. Seule la longueur de la chaîne d'entrée varie, entre 10 et 20 symboles.

Les figures 3.9 à 3.12 présentent les courbes de précision obtenues selon les quatre stratégies de questionnement. Chaque figure regroupe cinq courbes correspondant aux cinq formats de représentation.

De manière générale, l'augmentation de la longueur des chaînes d'entrée tend à stabiliser les performances. Contrairement aux autres benchmarks, cette variable introduit moins de bruit dans les résultats et semble favoriser une progression lente. Cette stabilité relative s'explique par le volume accru de contexte présent dans les séquences longues, qui peut renforcer la compréhension des dynamiques de transition à condition que le format et la stratégie de requêtage orientent efficacement le raisonnement du modèle.

#### 3.1.3.0.1 Performances globales par approche :

**Approche assistée** : C'est à nouveau l'approche la plus performante (figure 3.9). Les formats combiné (0.926) et langage naturel contrôlé (0.925) obtiennent les meilleures moyennes, suivis de très près par la liste d'adjacence (0.911). Le format DOT reste solide (0.880), tandis que la table de transitions reste en retrait (0.784).

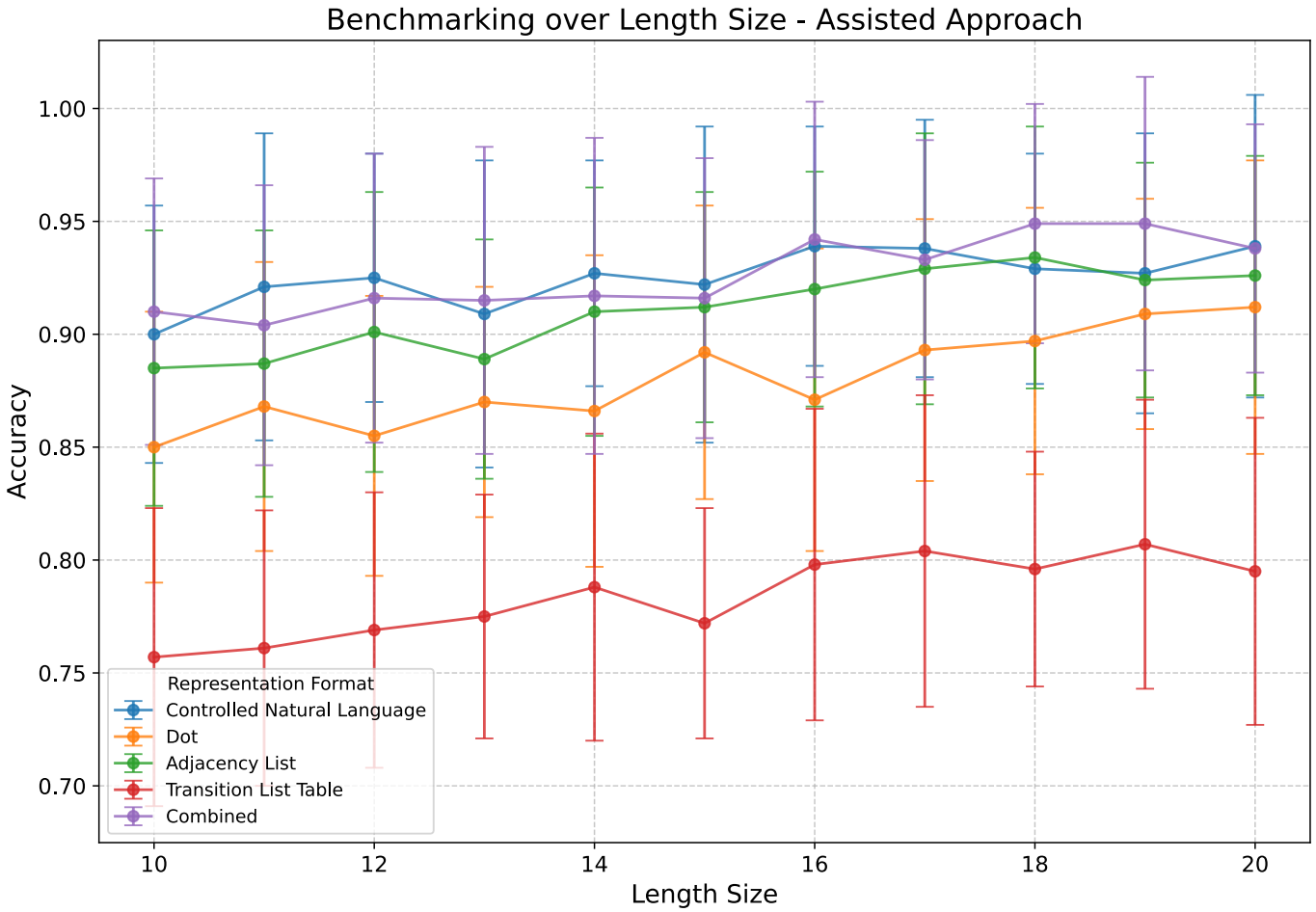


Figure 3.9 Performance de l'approche assistée

| Source | SS | DF | MS | F | p-valeur | $\eta^2$ (np2) |
|---|---|---|---|---|---|---|
| Format | 0.157 | 4 | 0.0391 | 131.04 | $<$ 0.0001 | 0.91 |
| Résidu | 0.015 | 50 | 0.0003 | – | – | – |

Tableau 3.12 ANOVA pour l'effet du format (Approche assistée – longueur des chaînes)

Les résultats de l'ANOVA (tableau 3.12) révèlent un effet fortement significatif du format sur les performances ($F = 131.04$, $p < 10^{-25}$), avec une taille d'effet très élevée ($\eta^2 = 0.91$), indiquant que le choix du format explique l'essentiel de la variance observée.

| Format A | Format B | Différence | p-valeur | Stat. $T$ | Hedges' $g$ |
|---|---|---|---|---|---|
| Liste d'adjacence | DOT | +0.030 | 0.0013 | 4.12 | 1.49 |
| Liste d'adjacence | Tableau de transitions | +0.127 | $<$0.0001 | 17.21 | 6.89 |
| Langage naturel contrôlé | DOT | +0.045 | $<$0.0001 | 6.08 | 2.49 |
| Langage naturel contrôlé | Tableau de transitions | +0.141 | $<$0.0001 | 19.17 | 8.96 |
| Combiné | DOT | +0.046 | $<$0.0001 | 6.24 | 2.34 |
| Combiné | Tableau de transitions | +0.142 | $<$0.0001 | 19.33 | 8.07 |
| DOT | Tableau de transitions | +0.096 | $<$0.0001 | 13.09 | 4.75 |

Tableau 3.13 Test post-hoc de Tukey (Approche assistée – longueur des chaînes)

Les comparaisons post-hoc (tableau 3.13) révèlent que les formats narratifs (langage naturel contrôlé, combiné) et textuels (liste d'adjacence) sont tous significativement plus performants que les formats visuels (DOT) et structurés (tableau de transitions), avec des tailles d'effet élevées (Hedges' $g$ entre 1.5 et 8). Ces résultats renforcent l'hypothèse selon laquelle les représentations explicites et contextuelles facilitent la compréhension des transitions par les LLMs, en particulier lorsqu'ils sont guidés de manière pas à pas.

**Approche générique** : Les résultats (figure 3.10) sont également élevés, avec une moyenne de 0.858 pour le format Combiné et 0.855 pour le langage naturel contrôlé. Même les formats les plus faibles (DOT et tableau

de transitions) dépassent les 0.71, ce qui confirme que la longueur des chaînes bénéficie particulièrement à cette stratégie.

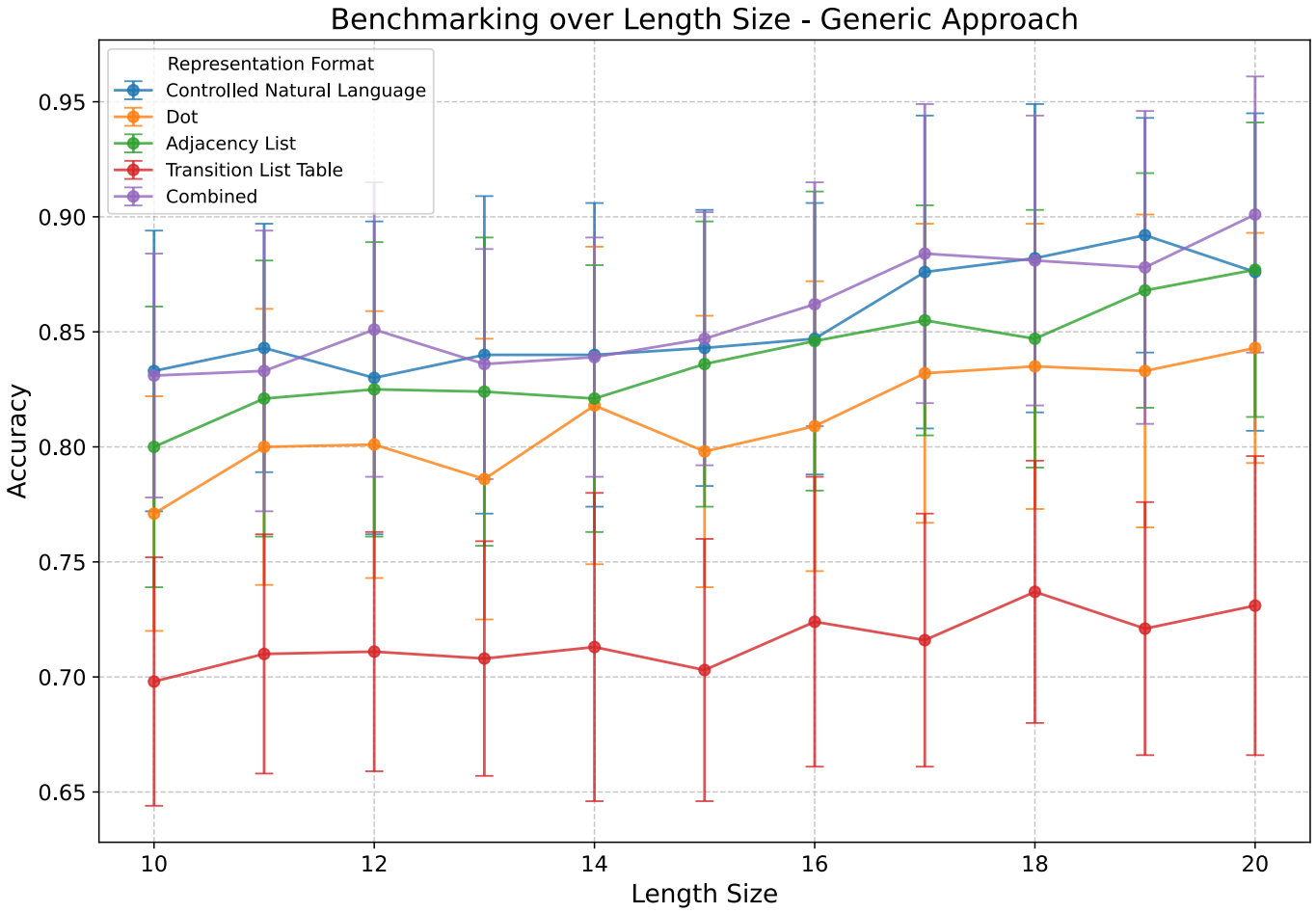


Figure 3.10 Performance de l'approche générique

Pour approfondir l'interprétation des différences entre formats dans l'approche Générique, une comparaison statistique (tableau 3.14) rigoureuse a été réalisée.

| **Source** | **SS** | **DF** | **MS** | **F** | **p-valeur** | $\eta^2$ **(np2)** |
|---|---|---|---|---|---|---|
| Format | 0.153 | 4 | 0.0382 | 84.70 | $< 0.0001$ | 0.87 |
| Résidu | 0.023 | 50 | 0.0005 | – | – | – |

Tableau 3.14 ANOVA sur les formats (Approche générique – longueur des chaînes)

L'effet du format s'avère statistiquement très significatif ($p < 10^{-20}$), avec une taille d'effet considérable ($\eta^2 = 0.87$), indiquant que la variation des performances dépend majoritairement du format utilisé.

| **Format A** | **Format B** | **Différence** | **p-valeur** | **Stat.** $T$ | **Hedges'** $g$ |
|---|---|---|---|---|---|
| Liste d'adjacence | DOT | +0.027 | 0.037 | 2.95 | 1.13 |
| Liste d'adjacence | Tableau de transitions | +0.123 | $<$0.0001 | 13.53 | 6.48 |
| Langage naturel contrôlé | DOT | +0.043 | $<$0.001 | 4.78 | 1.85 |
| langage naturel contrôlé | Tableau transitions | +0.139 | $<$0.0001 | 15.36 | 7.56 |
| Combiné | DOT | +0.047 | $<$0.0001 | 5.19 | 1.93 |
| Combiné | Tableau de transitions | +0.143 | $<$0.0001 | 15.77 | 7.23 |
| DOT | Tableau de transitions | +0.096 | $<$0.0001 | 10.58 | 5.09 |

Tableau 3.15 Test de Tukey HSD (Approche générique – longueur des chaînes)

Ces résultats (tableau 3.15) confirment que les performances sont plus faibles avec les formats **DOT** et **tableau de transitions**, en comparaison aux formats **combiné**, **langage naturel contrôlé** et **liste d'adjacence**. Les tailles d'effet (Hedges' $g$) dépassent souvent 5, signalant des différences très marquées. Ainsi, même sans guidance explicite, les LLMs bénéficient des formats riches ou textuels lorsqu'ils doivent traiter des chaînes longues.

**Approche normale** : Cette approche montre des progrès clairs à mesure que la séquence s'allonge (figure 3.11). Le format langage naturel contrôlé atteint une moyenne de 0.788, suivi du combiné (0.784) et de la liste d'adjacence (0.773). Même les formats structurels affichent une progression (DOT : 0.736, Tableau : 0.631).

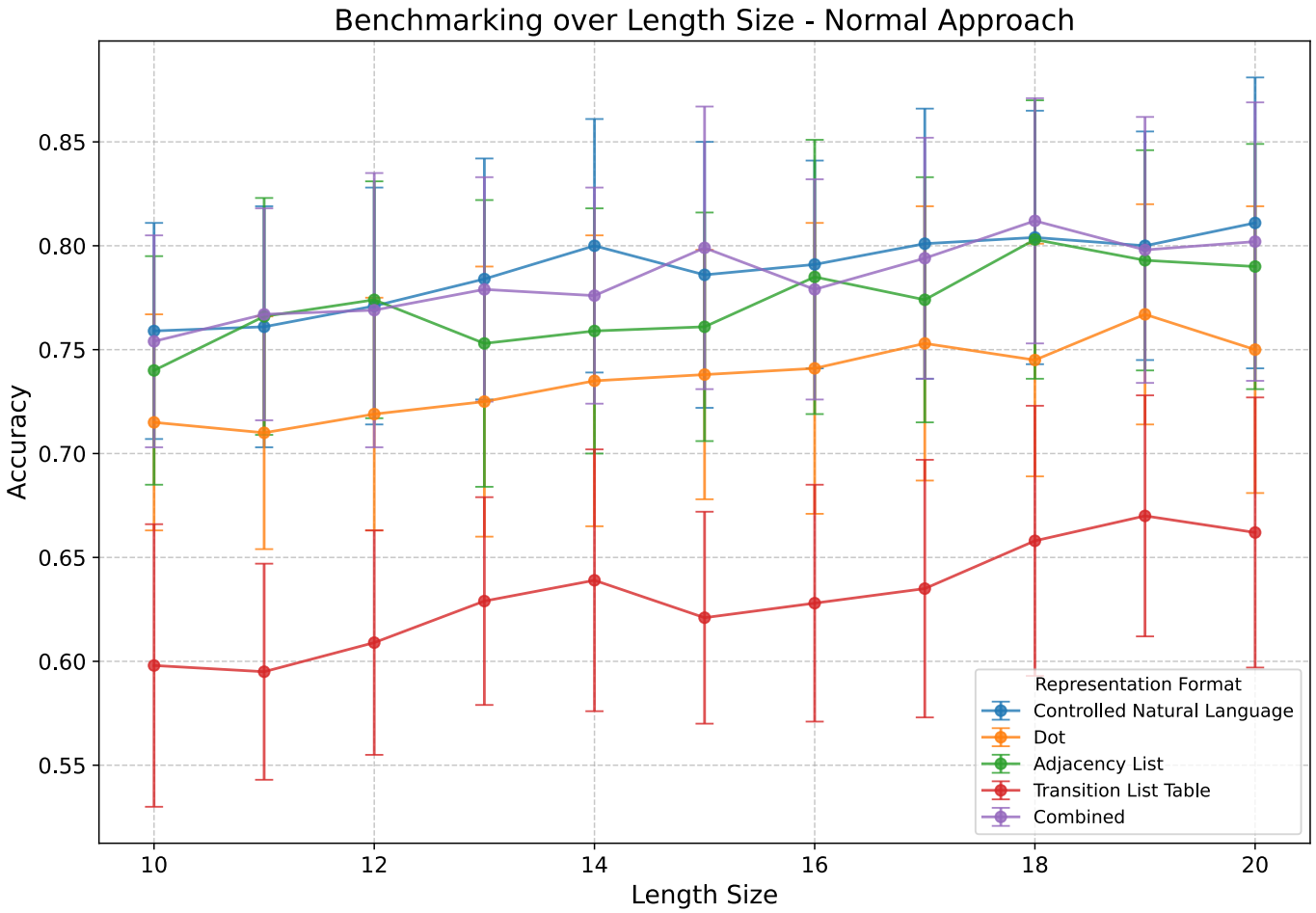


Figure 3.11 Performance de l'approche normale

Une étude statistique détaillée (tableau 3.16) a été réalisée afin d'évaluer si le format de représentation a un effet significatif sur les performances observées dans l'approche Normale.

| **Source** | **SS** | **DF** | **MS** | **F** | **p-valeur** | $\eta^2$ **(np2)** |
|---|---|---|---|---|---|---|
| Format | 0.153 | 4 | 0.0382 | 84.70 | $< 0.0001$ | 0.87 |
| Résidu | 0.023 | 50 | 0.0005 | – | – | – |

Tableau 3.16 ANOVA sur les formats (Approche normale – longueur des chaînes)

Les résultats révèlent un effet fortement significatif du format sur la performance ($p < 10^{-20}$), avec une taille d'effet très élevée ($\eta^2 = 0.87$), soulignant que le choix du format est déterminant dans cette configu-

ration.

| **Format A** | **Format B** | **Différence** | **p-valeur** | **Stat.** $T$ | **Hedges'** $g$ |
|---|---|---|---|---|---|
| Liste d'adjacence | DOT | +0.027 | 0.037 | 2.95 | 1.13 |
| Liste d'adjacence | Tableau de transitions | +0.123 | $<$0.0001 | 13.53 | 6.48 |
| Langage naturel contrôlé | DOT | +0.043 | $<$0.001 | 4.78 | 1.85 |
| Langage naturel contrôlé | Tableau de transitions | +0.139 | $<$0.0001 | 15.36 | 7.56 |
| Combiné | DOT | +0.047 | $<$0.0001 | 5.19 | 1.93 |
| Combiné | Tableau de transitions | +0.143 | $<$0.0001 | 15.77 | 7.23 |
| DOT | Tableau de transitions | +0.096 | $<$0.0001 | 10.58 | 5.09 |

Tableau 3.17 Test de Tukey HSD (Approche normale – longueur des chaînes)

Les comparaisons post-hoc (tableau 3.17) confirment que les formats **langage naturel contrôlé** et **combiné** offrent des performances significativement supérieures aux formats **DOT** et **tableau de transitions**, avec des tailles d'effet très marquées ($g > 5$ pour certains contrastes). Le format **liste d'adjacence** se situe dans une zone intermédiaire, performant contre les formats structurés mais non significativement différent des formats textuels.

**Approche question fermée** : C'est la seule stratégie pour laquelle l'augmentation de la longueur n'entraîne pas de gain substantiel (figure 3.12). Les scores restent modérés : autour de 0.619 pour le langage naturel contrôlé, 0.596 pour la liste d'adjacence, et 0.574 pour DOT. Le tableau de transitions reste, comme toujours, en dernière position (0.541).

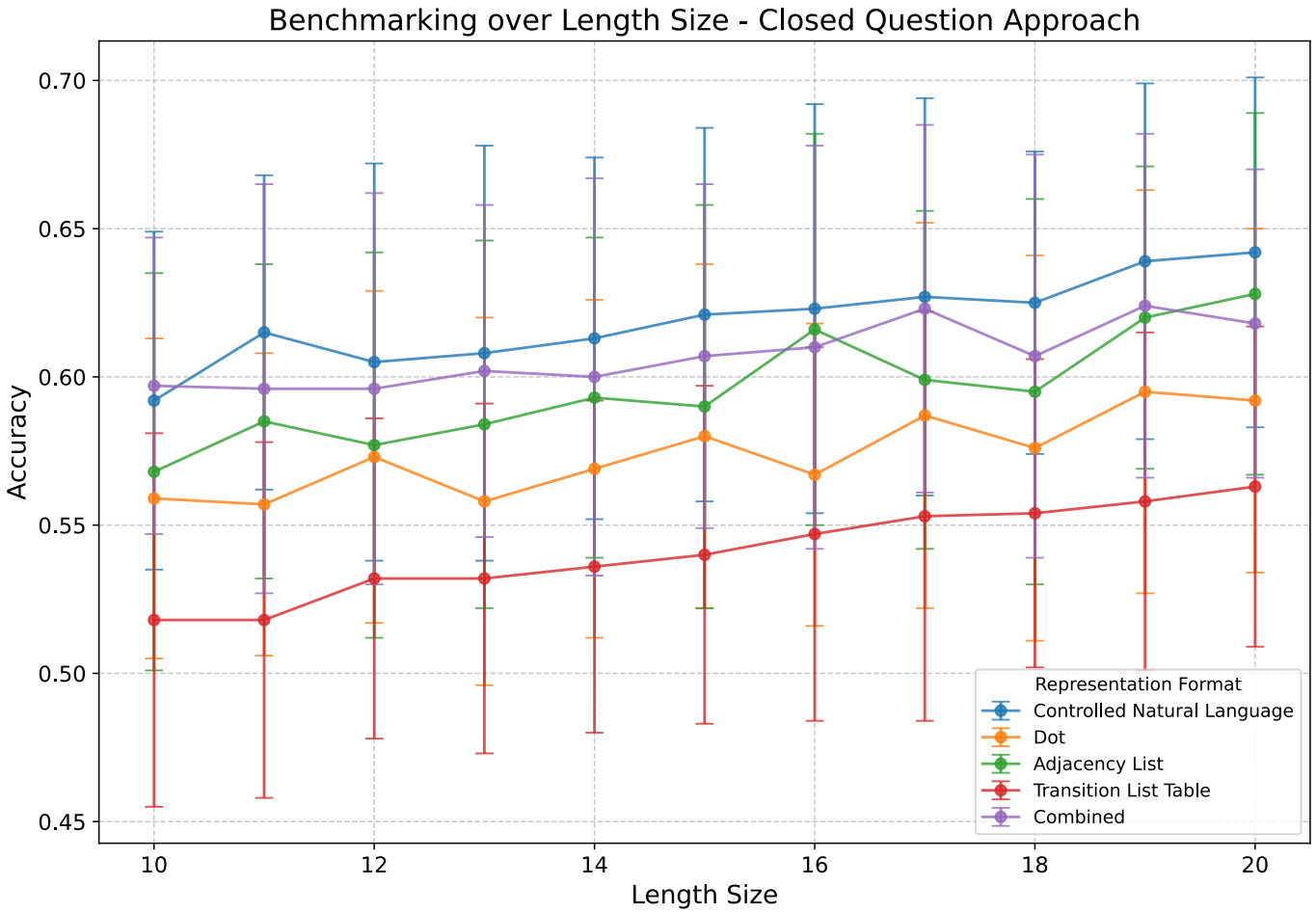


Figure 3.12 Performance de l'approche Closed Question

Pour déterminer si le format de représentation influence les performances dans cette stratégie, une analyse statistique a été conduite.

| Source | SS | DF | MS | F | p-valeur | $\eta^2$ (np2) |
|---|---|---|---|---|---|---|
| Format | 0.0419 | 4 | 0.0105 | 48.07 | $< 0.0001$ | 0.79 |
| Résidu | 0.0109 | 50 | 0.0002 | – | – | – |

Tableau 3.18 ANOVA sur les formats (Approche question fermée – longueur des chaînes)

Les résultats de l'ANOVA (tableau 3.18) indiquent un effet significatif du format ($p < 10^{-15}$), avec une taille d'effet considérable ($\eta^2 = 0.79$), confirmant que même dans un cadre de réponse fermée, la représentation

textuelle impacte la performance.

#### 3.1.3.0.2 Résumé des résultats

| Approche | Liste d'adjacence | Tableau de transitions | DOT | Langage naturel contrôlé | Combiné |
|---|---|---|---|---|---|
| Question fermée | 0,596 ± 0,062 | 0,541 ± 0,062 | 0,574 ± 0,059 | 0,619 ± 0,061 | 0,607 ± 0,061 |
| Normale | 0,773 ± 0,064 | 0,631 ± 0,061 | 0,736 ± 0,059 | 0,788 ± 0,062 | 0,784 ± 0,063 |
| Assistée | **0,911 ± 0,057** | **0,784 ± 0,060** | **0,880 ± 0,061** | **0,925 ± 0,056** | **0,926 ± 0,059** |
| Générique | 0,838 ± 0,062 | 0,716 ± 0,064 | 0,811 ± 0,061 | 0,855 ± 0,065 | 0,858 ± 0,065 |

Tableau 3.19 Précision moyenne par format de représentation selon la longueur des chaînes d'entrée.

Ces résultats (tableau 3.19) suggèrent que l'allongement des chaînes contribue à une meilleure stabilité des performances, en particulier pour les approches assistée et générique. Le surplus de contexte disponible semble permettre au modèle de maintenir une trajectoire de raisonnement plus fluide et moins sujette à des erreurs de transition. Toutefois, cette stabilité dépend fortement de la structuration explicite apportée par le prompt. Les stratégies non-guidées ou fermées, n'intégrant pas ce cadre de raisonnement, ne parviennent pas à exploiter pleinement cette richesse contextuelle, ce qui limite leur potentiel d'amélioration.

**Résultats pour l'énigme**

Afin d'illustrer concrètement l'efficacité de nos méthodes, nous avons appliqué nos approches de prompt engineering à l'exemple de l'énigme classique du traversier (cf. figure 1.2). Cette étude de cas a été intégrée dans le protocole expérimental à travers le benchmark sur le nombre d'états, puisque la séquence d'entrée ainsi que l'alphabet y sont déjà fixés par la nature même de l'énigme, rendant les autres axes de variation inapplicables. Nous avons testé quatre séquences acceptantes (par exemple, `gnwgcng`, `gncgwng`, `gnwgcwgcwgcng` et `gncgwcgwcgwng`), ainsi que quatre séquences non acceptantes obtenues en dupliquant une lettre dans chaque séquence acceptante.

Même si cet exemple repose sur un échantillon de petite taille, il reflète les mêmes tendances observées avec les automates générés aléatoirement. Plus précisément :

— L'approche **question fermée** se révèle inefficace, avec une précision très faible pour les chaînes ac-

| Format | Question fermée | Normale | Assistée | Générique |
|---|---|---|---|---|
| **Langage naturel contrôlé** | 0% : 100% | 25% : 50% | 100% : 100% | 75% : 75% |
| **Liste d'adjacence** | 25% : 75% | 75% : 25% | 100% : 100% | 75% : 75% |
| **DOT** | 50% : 50% | 75% : 75% | 50% : 100% | 75% : 75% |
| **Tableau de transitions** | 0% : 100% | 0% : 100% | 0% : 75% | 0% : 100% |
| **Combiné** | 0% : 100% | 50% : 75% | 100% : 100% | 75% : 75% |

Tableau 3.20 Taux de vrais positifs et négatifs par format et approche dans l'énigme

ceptées (proche de 0%), malgré une identification correcte des chaînes rejetées.

- L'approche **normale** améliore légèrement la capacité du modèle à détecter les chaînes acceptées, sans atteindre un niveau optimal.
- L'approche **assistée** se démarque en obtenant une performance parfaite (100% pour les chaînes acceptées et rejetées), démontrant ainsi l'efficacité du guidage étape par étape.
- L'approche **générique** présente des performances intermédiaires, avec environ 75% de précision pour les deux catégories.

Ces résultats, synthétisés dans le tableau 3.20, mettent en lumière l'importance du format de représentation dans l'interaction avec le LLM. En particulier, le **langage naturel contrôlé** et le format **combiné** se révèlent les plus efficaces pour résoudre l'énigme, tandis que les formats plus structurés (comme le **tableau de transitions**) affichent des performances moindres. Cette analyse confirme que le choix de la stratégie de prompt et du format de représentation est déterminant pour optimiser la précision et la robustesse des réponses des LLMs dans les tâches de vérification d'automates.

**Discussion**

Les résultats expérimentaux mettent en évidence plusieurs limitations et biais empirique inhérents aux modèles de langage actuels, en particulier lorsqu'ils sont sollicités pour résoudre des tâches de vérification d'automates finis. Plusieurs points méritent d'être discutés en détail :

- **Biais vers le langage naturel :** Les formats narratifs (**langage naturel contrôlé**, **combiné**) surpassent les formats structurés (**liste d'adjacence**, **DOT**, **tableau de transitions**), car les LLMs sont entraînés sur du texte naturel. Leur capacité à interpréter des relations sémantiques favorise les structures

idiomatiques. À l'inverse, des formats trop formels induisent une interprétation hors contexte et des erreurs implicites.

— **Catastrophic forgetting :** Sur de longues chaînes (ex. `abcaaa...`), le modèle oublie des lettres, faussant la simulation. Ce phénomène, lié aux limites de mémoire des transformers, est aggravé par des répétitions et entraîne des omissions critiques. Des rappels explicites améliorent partiellement la précision.

— **Erreurs de transition :** ChatGPT confond parfois les transitions, en attribuant un symbole ou un état à une mauvaise règle. Cela survient surtout avec des transitions ambiguës, provoquant des substitutions plausibles mais incorrectes. L'absence de vérification étape par étape aggrave ces erreurs.

— **Incohérences d'acceptation :** Le modèle peut indiquer que la chaîne est acceptée même si l'état final n'est pas acceptant, ou inversement. Ces contradictions traduisent un décalage entre le raisonnement séquentiel et le verdict final.

— **Impact de la longueur et de la complexité des automates :** La performance du modèle tend à se dégrader à mesure que le nombre d'états dans l'automate augmente. La complexité accrue, notamment en termes de dépendances à long terme et de transitions interconnectées, expose davantage les limites des LLMs en matière de gestion de séquences longues, exacerbant ainsi les problèmes de "forgetting" et d'erreurs de transition. L'augmentation du nombre d'états multiplie les chemins possibles, rendant la tâche plus sujette aux omissions et confusions. Nous observons une décroissance quasi linéaire de la précision à partir de 8 états, puis une chute plus rapide au-delà de 12 états. Cette dégradation met en évidence les limites du `max_tokens` et la saturation de l'attention sur de longs prompts. Des stratégies de découpage en sous-automates ou un raisonnement hiérarchique pourraient permettre de pallier cette perte de performance.

— **Impact de la taille de l'alphabet et de la longueur des chaînes :** La performance des modèles est sensible à la complexité symbolique (taille de l'alphabet) et à la profondeur de la séquence (longueur de chaîne). Lorsque l'alphabet s'élargit, on observe une dégradation progressive des performances, particulièrement marquée pour les approches moins guidées ou basées sur des formats rigides, due à la dispersion attentionnelle sur un espace symbolique plus vaste. Contrairement à ce que l'on pourrait anticiper, l'augmentation de la longueur des chaînes d'entrée n'entraîne pas de dégradation notable. Au contraire, les résultats montrent une plus grande stabilité et parfois une légère amélioration pour les approches guidées, telles que l'assistée ou la générique, car les longues séquences fournissent un cadre contextuel dense, consolidant le raisonnement du modèle. Toutefois, pour les approches plus limitées (comme les questions fermées), ni la richesse symbolique ni la longueur ne suffisent à

compenser l'absence de raisonnement intermédiaire explicite. Cela démontre que le facteur déterminant reste la stratégie de guidage, plus que la seule complexité en entrée.

En conclusion, ces différents biais et limitations montrent que, malgré leur puissance linguistique, les LLMs restent fragiles face aux exigences de cohérence et de rigueur des tâches formelles.

## 3.2 Génération optimale de séquences acceptées

Cette section évalue la capacité des modèles de langage à générer des séquences non seulement acceptées, mais également optimales c'est-à-dire de longueur minimale. Pour ce faire, nous analysons deux indicateurs complémentaires : la *précision* (*accuracy*) des chaînes générées (c'est-à-dire leur validité) et le *taux de minimalité* (*minimal rate*), mesurant la proportion de chaînes effectivement optimales. Ces mesures sont analysées en fonction du nombre d'états de l'automate, pour chaque stratégie de requêtage et format de représentation.

### 3.2.1 Approche assistée

Le prompt utilisé dans cette approche guidait le modèle à travers un raisonnement pas-à-pas explicite :

```
Here is a finite automaton in {format_type} format: {automaton_description}
Generate the SHORTEST POSSIBLE string that this automaton accepts.
Think step by step and identify an accepted sequence by:
1. Identifying the initial state.
2. Listing possible transitions leading to an accepting state.
3. Generating a single string that follows those transitions.
```

La figure 3.13 présente les résultats obtenus. Les formats langage naturel contrôlé, Liste d'adjacence et Combined maintiennent de bonnes performances jusqu'à 10 états. Le *MinimalRate* suit une courbe parallèle à la précision, avec des valeurs particulièrement élevées en taille réduite (> 0.4 pour les formats langage naturel contrôlé et Transition List Table à 2 états).

### 3.2.2 Approche générique

L'approche générique ne propose qu'un guidage léger du raisonnement, avec un prompt plus ouvert :

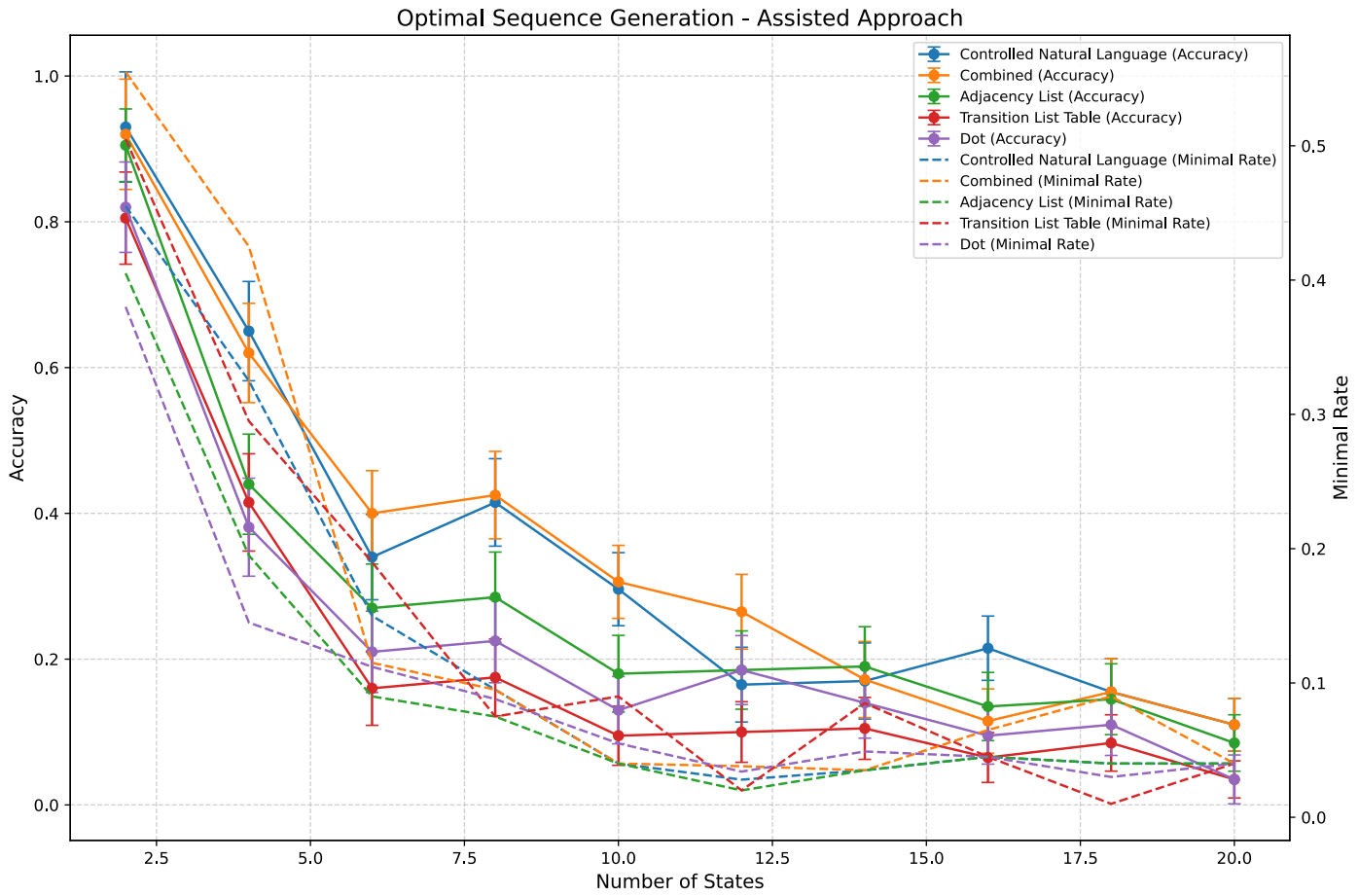


Figure 3.13 Performance de l'approche assistée (précision et taux de minimalité).

```
Here is a finite automaton in {format_type} format: {automaton_description}
Generate the SHORTEST POSSIBLE string that this automaton accepts. Break the resolution
into intermediate steps.
```

Comme l'illustre la figure 3.14, la précision décroît plus rapidement que dans l'approche assistée, notamment au-delà de 10 états. Le *MinimalRate*, bien que plus bas, reste corrélé à la précision globale. Les formats Adjacency List et Transition List Table semblent offrir la meilleure robustesse, malgré un fléchissement dès 8 états.

### 3.2.3 Approche normale

Le prompt utilisé dans cette approche est minimaliste et direct :

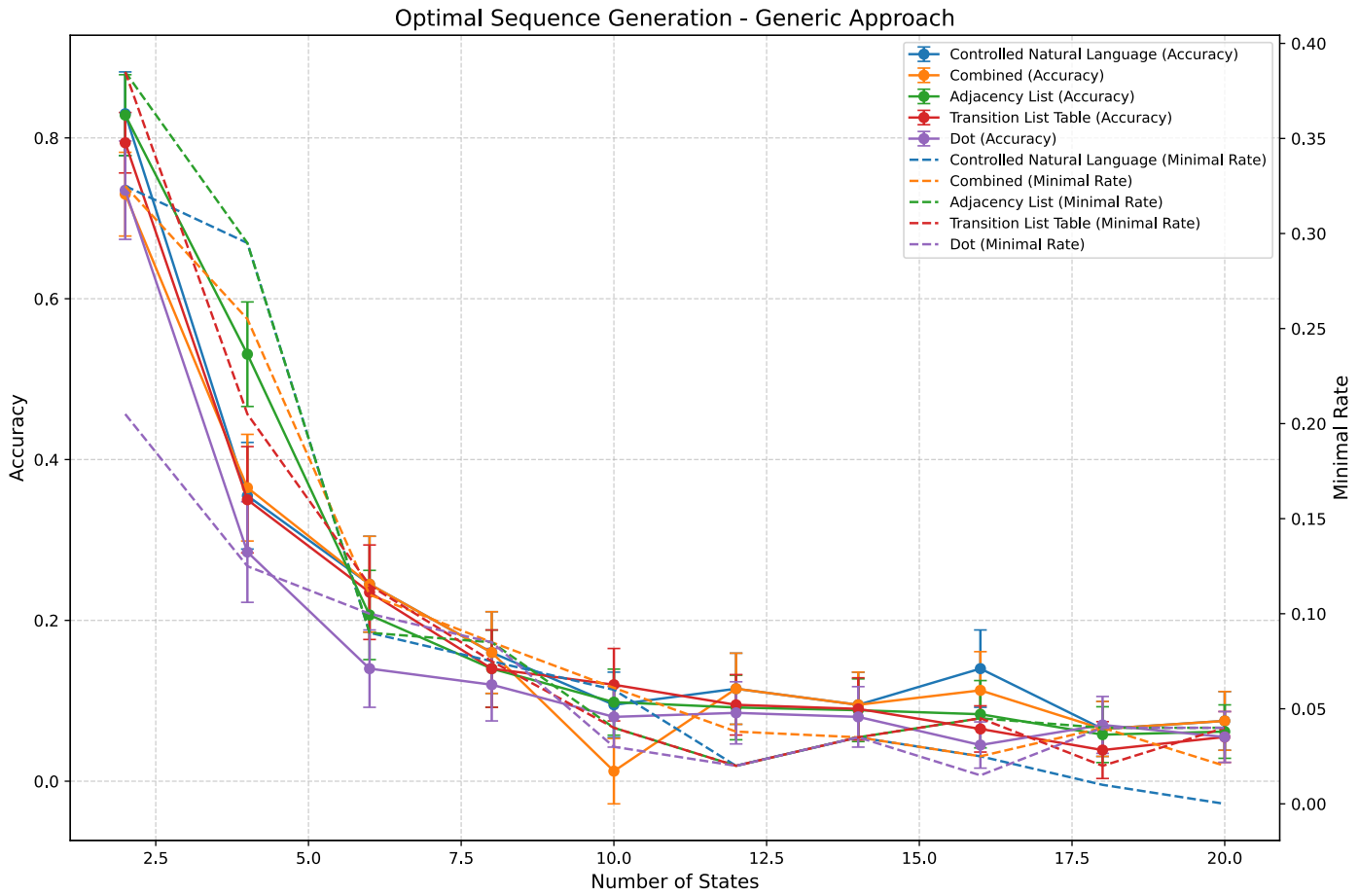


Figure 3.14 Performance de l'approche générique (précision et taux de minimalité).

```
Here is a finite automaton in {format_type} format: {automaton_description}
Generate the SHORTEST POSSIBLE string that this automaton accepts.
```

Comme montré dans la figure 3.15, cette approche atteint de bons scores en précision pour les petits automates, mais chute rapidement après 6 états. Le taux de minimalité reste faible, ne dépassant que rarement les 0.1 au-delà de 10 états, même pour les formats langage naturel contrôlé et Combined.

### 3.2.4 Approche question fermée

Enfin, l'approche question fermée impose une contrainte stricte à la sortie attendue :

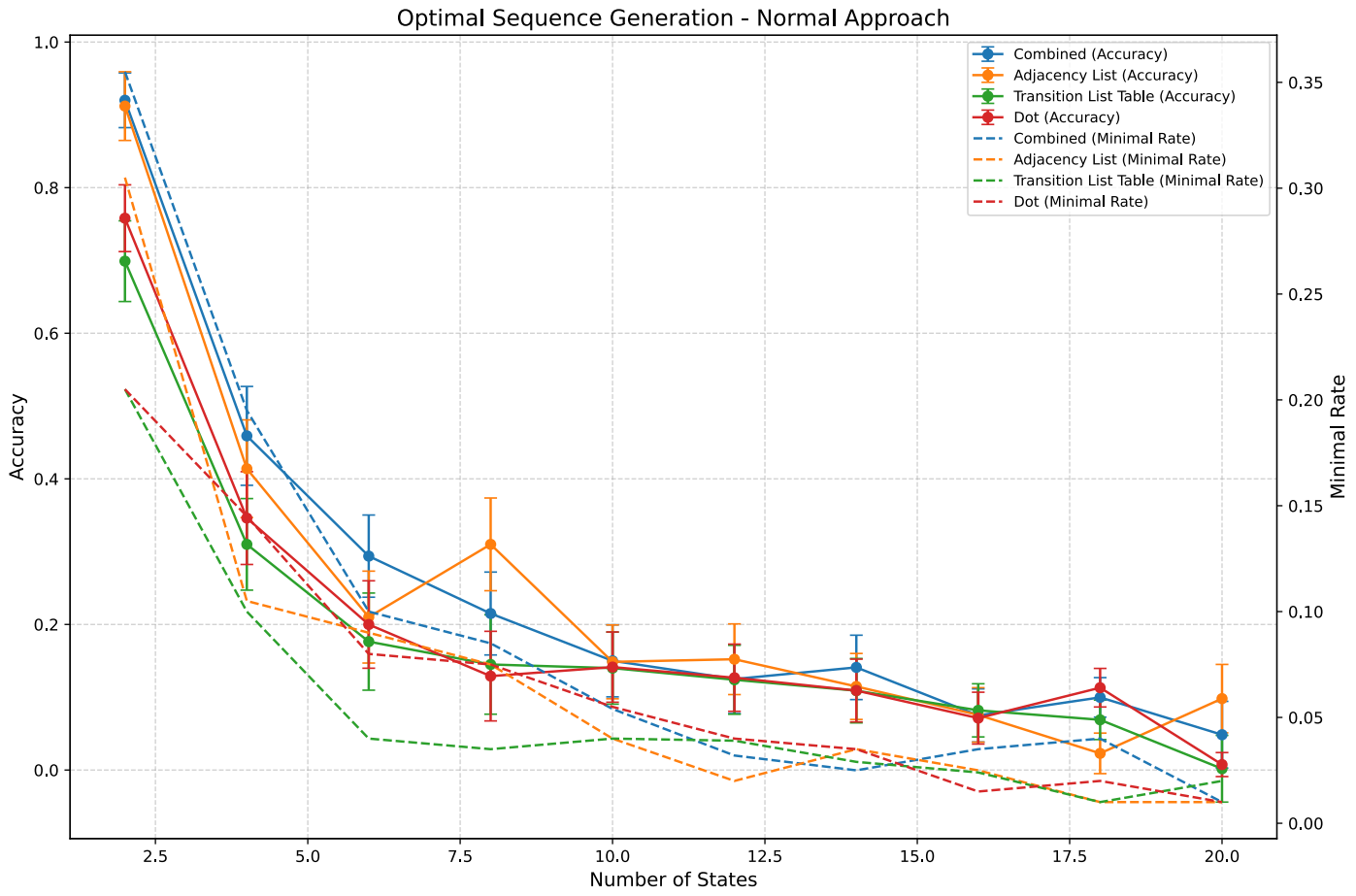


Figure 3.15 Performance de l'approche normale (précision et taux de minimalité).

```
Here is a finite automaton in {format_type} format: {automaton_description}
Generate the shortest possible string that this automaton accepts. Respond only with
the string.
```

Les résultats de la figure 3.16 montrent une nette baisse des deux indicateurs au-delà de 6 états. Malgré cela, certains formats comme liste d'adjacence montrent une résilience relative dans les tailles moyennes, tandis que langage naturel contrôlé chute brutalement après 12 états.

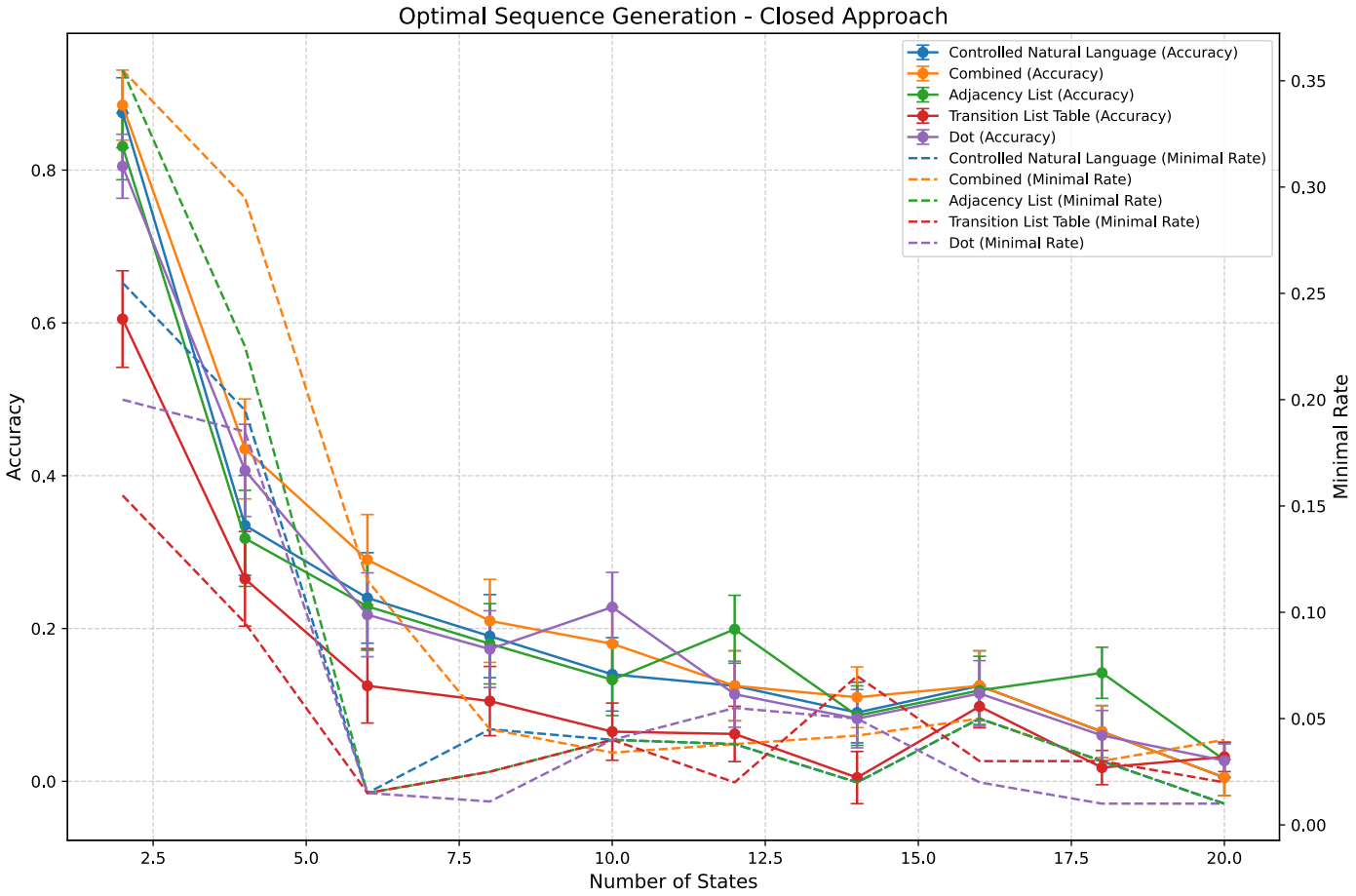


Figure 3.16 Performance de l'approche fermée (précision et taux de minimalité).

### 3.2.5 Résumé des résultats

| Approche | | 2 états | | 20 états | |
|---|---|---|---|---|---|
| **Type** | **Format** | **Précision** | **Minimalité** | **Précision** | **Minimalité** |
| Assistée | Langage naturel contrôlé | 0.93 | 0.455 | 0.11 | 0.04 |
| | Combiné | 0.92 | 0.555 | 0.11 | 0.04 |
| | Liste d'adjacence | 0.905 | 0.405 | 0.085 | 0.04 |
| | Tableau de transitions | 0.805 | 0.505 | 0.035 | 0.04 |
| | DOT | 0.82 | 0.38 | 0.035 | 0.04 |
| Générique | Langage naturel contrôlé | 0.83 | 0.325 | 0.075 | 0 |
| | Combiné | 0.73 | 0.325 | 0.075 | 0.02 |
| | Liste d'adjacence | 0.8283 | 0.385 | 0.0617 | 0.04 |
| | Tableau de transitions | 0.794 | 0.385 | 0.055 | 0.04 |
| | DOT | 0.735 | 0.205 | 0.055 | 0.04 |
| Normale | Langage naturel contrôlé | 0.94 | 0.365 | 0.0485 | 0.01 |
| | Combiné | 0.92 | 0.355 | 0.0485 | 0.01 |
| | Liste d'adjacence | 0.912 | 0.305 | 0.0981 | 0.01 |
| | Tableau de transitions | 0.699 | 0.205 | 0.0018 | 0.02 |
| | DOT | 0.758 | 0.205 | 0.00751 | 0.01 |
| Question fermée | Langage naturel contrôlé | 0.875 | 0.255 | 0.005 | 0.01 |
| | Combiné | 0.885 | 0.355 | 0.005 | 0.04 |
| | Liste d'adjacence | 0.831 | 0.355 | 0.028 | 0.01 |
| | Tableau de transitions | 0.605 | 0.155 | 0.032 | 0.02 |
| | DOT | 0.805 | 0.2 | 0.027 | 0.01 |

Tableau 3.21 Précision et minimalité par approche et format pour différentes tailles d'automates.

L'analyse finale, synthétisés dans le tableau 3.21, révèle une chute systématique des performances à mesure que la complexité des automates augmente particulièrement marquée dans les approches normale et fermée. Les résultats montrent que, malgré des scores de précision acceptables pour des automates simples (2 états), les modèles échouent massivement à maintenir des séquences minimales pour des automates de 20 états, avec des taux de minimalité proches de zéro.

Les approches assistée et générique démontrent une meilleure robustesse, atteignant des taux de minimalité initiaux élevés (jusqu'à 0.555 pour le format combiné), mais elles finissent par s'effondrer pour les automates complexes, s'alignant souvent sur les performances médiocres des approches directes.

Le *MinimalRate* s'impose ainsi comme un indicateur critique, révélant des performances bien plus catastrophiques que ne le laissent supposer les taux d'acceptation seuls.

Nous avons également effectué des tests statistiques (ANOVA, Tukey) sur les mesures de précision et de minimalité. Toutefois, étant donné la faiblesse générale des performances notamment en minimalité et la clarté des écarts observés, ces analyses n'ont pas été incluses, car elles n'apporteraient aucune information complémentaire significative.

### 3.2.6 Analyse complémentaire : taille de l'alphabet et longueur des séquences

Nous avons également réalisé des benchmarks en faisant varier deux autres facteurs : la taille de l'alphabet et la longueur des séquences. Les résultats obtenus, bien que non présentés ici sous forme de figures afin d'éviter d'alourdir le document, suivent des tendances très similaires à celles observées lors de l'augmentation du nombre d'états. En particulier, les performances des modèles en termes de précision d'acceptation et de taux de minimalité décroissent rapidement dès que la complexité structurelle augmente qu'il s'agisse d'un alphabet plus large ou de chaînes plus longues à générer.

### 3.2.7 Discussion

Les résultats expérimentaux liés à la génération de la plus courte chaîne acceptée révèlent de profondes limitations des modèles de langage de type LLMs, comme ChatGPT, surtout en comparaison avec la simple tâche de validation d'une séquence. Plusieurs constats critiques émergent de cette évaluation :

— **Effondrement précoce des performances :** Alors que les performances en validation restaient acceptables jusqu'à 10 états, la génération de séquences minimales montre une chute drastique dès les automates à 4 états. Le MinimalRate tombe souvent en dessous de 0.05 au-delà de 10 états, indépendamment du format ou de l'approche.

— **Taux de minimalité très faible :** Même lorsque le modèle parvient à générer une séquence acceptée, il échoue majoritairement à produire une chaîne de longueur minimale. Par exemple, dans les approches normale et fermée, MinimalRate chute à 0.01 ou même 0.00 dès 16 à 20 états. Seules les approches guidées (assistée, générique) atteignent temporairement des taux autour de 0.3 à 0.5, mais ces performances ne se maintiennent pas à mesure que la taille augmente.

— **Incohérence du raisonnement global :** Les modèles échouent souvent à exploiter leurs propres raisonnements intermédiaires. Ils identifient parfois correctement plusieurs séquences candidates (`ab`, `ca`, `bc`), mais concluent par une chaîne incorrecte, non reliée logiquement à leur processus de réflexion. Cela révèle une incapacité à agréger localement l'information pour former une décision cohérente au niveau global.

— **Effet accentué de *catastrophic forgetting* :** Lorsqu'une chaîne minimale implique des répétitions (comme `abba`, `aabbaacc`), les modèles ont tendance à omettre ou permuter certains symboles. Le résultat est souvent une chaîne incomplète ou non acceptée. Cet effet s'amplifie avec la taille de l'automate, ce qui nuit autant à la précision qu'au taux de minimalité.

— **Erreurs dans l'interprétation des transitions :** Des transitions simples comme « état 2, sur `a`, aller à l'état 5 » sont fréquemment mal interprétées. Le modèle génère des chemins impossibles ou ignore des états acceptants. Ces erreurs structurelles contribuent directement à l'échec de génération de chaînes valides et minimales.

— **Absence de stratégie algorithmique :** Générer une chaîne minimale dans un automate revient souvent à une recherche de chemin optimal (type BFS). Aucun comportement de ce type n'est détectable dans les réponses des modèles. Ceux-ci explorent des chemins au hasard ou selon des heuristiques implicites, inadaptées à la nature combinatoire du problème.

— **Impact exponentiel du nombre d'états :** L'effet de la taille est plus que linéaire. Au-delà de 6, les performances s'effondrent rapidement. Le MinimalRate chute parfois à 0 pour certains formats. Cela indique que le modèle est incapable de gérer la croissance combinatoire du graphe des transitions.

En résumé, la génération de chaînes minimales révèle crûment les limites actuelles des LLMs face aux problèmes nécessitant un raisonnement symbolique et algorithmique.

## PERSPECTIVES

Ce mémoire s'inscrit dans le cadre d'un projet de maîtrise en recherche, avec des contraintes inhérentes : des ressources techniques limitées, un budget restreint et une durée d'exécution bornée. Dans ce contexte, notre objectif a été de concevoir une expérimentation cohérente, reproductible et scientifiquement pertinente sur la capacité des modèles de langage à raisonner sur des automates finis.

Le choix du modèle `gpt-4o-mini-2024-07-18` repose sur un équilibre entre coût, accessibilité et stabilité. Il permet de lancer des milliers de requêtes via une API bien documentée, tout en s'inscrivant dans un cadre reproductible utilisé par d'autres travaux. Certes, l'étude aurait pu être enrichie par l'utilisation de modèles récents comme Claude 3, Gemini 2.0 Flash, ou encore des alternatives open-source telles que Mistral 7B, LLaMA 3, ou DeepSeek-V3. Toutefois, leur coût d'accès élevé ou leurs exigences en infrastructure matérielle dépassent largement les moyens disponibles dans un cadre de recherche de maîtrise. Le modèle choisi représente ainsi un compromis pertinent, à la fois économiquement viable et scientifiquement solide.

Les deux tâches étudiées vérification d'acceptation et génération de chaînes minimales sont centrales dans les applications des automates. Elles ont été évaluées à travers des formats de représentation variés et des stratégies de prompting graduées. Chaque expérience s'est concentrée sur un facteur à la fois (nombre d'états, taille de l'alphabet, longueur des chaînes), avec les autres paramètres fixés pour garantir des comparaisons rigoureuses. Par exemple, la fixation à 10 états dans certains cas découle d'observations montrant que les performances se stabilisent au-delà de ce seuil. Le choix d'un échantillon de 200 automates par configuration s'appuie également sur des tests initiaux indiquant que cette taille assure une précision statistique suffisante, évitant les résultats bruités.

Ce travail constitue une première base solide, adaptable à des modèles plus puissants, des tâches plus complexes ou des architectures hybrides. Il ouvre des perspectives vers des approches neuro-symboliques et des outils formels intégrés, capables de dépasser les limites actuelles des LLMs dans la manipulation de structures computationnelles rigoureuses.

# CONCLUSION

Dans ce mémoire, nous avons évalué la capacité des grands modèles de langage en particulier gpt-4o-mini-2024-07-18 à comprendre et à raisonner sur des automates finis, un pilier fondamental de l'informatique théorique. L'étude s'est concentrée sur deux tâches centrales : la vérification de l'acceptation d'une chaîne et la génération de séquences minimales acceptées, en mobilisant diverses représentations d'automates et stratégies de sollicitation (prompting).

Cinq formats ont été testés : liste d'adjacence, table de transitions, format DOT, langage naturel contrôlé (CNL), et une représentation hybride. Quatre stratégies de prompting : Question fermée, Normale, Assistée, et Générique, ont permis de comparer différents niveaux de guidage. L'approche assistée, combinée à des formats narratifs, a montré les meilleures performances pour la tâche de vérification, soulignant l'affinité des LLMs avec des formulations linguistiques explicites.

En revanche, la génération de chaînes d'acceptation minimales reste une tâche difficile. Les résultats se sont dégradés avec la complexité croissante des automates, même pour de petites structures. Les erreurs fréquentes incluent des transitions incohérentes, des omissions de symboles, et des divergences entre les raisonnements intermédiaires et la réponse finale symptômes d'une limitation dans la mémoire symbolique et la stabilité logique des modèles.

Ces résultats révèlent un contraste important : les LLMs peuvent exceller dans la compréhension de descriptions narratives mais peinent face à des manipulations symboliques rigoureuses. Cela appelle à l'exploration d'approches hybrides mêlant raisonnement formel, mémoire persistante, et ingénierie avancée de prompts, pour mieux aligner les capacités des LLMs avec les exigences de la vérification formelle et des langages calculables.

**BIBLIOGRAPHIE**


Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S. *et al.* (2023). GPT-4 Technical Report. *arXiv preprint arXiv :2303.08774*.

Aho, A. V., Lam, M. S., Sethi, R. et Ullman, J. D. (2006). *Compilers : Principles, Techniques, and Tools* (2nd éd.). Boston, MA, USA : Addison-Wesley. Known as the "Dragon Book", the definitive reference for lexical analysis (finite automata) and parsing (pushdown automata). Récupéré de https://dl.acm.org/doi/10.5555/1177220

AI, M. (2023a). Announcing Mistral 7B. Accessed : 2025-05-05. Récupéré de https://mistral.ai/news/announcing-mistral-7b

AI, M. (2023b). Llama : Open Foundation Language Models. https://ai.meta.com/llama. Accessed : 2025-05-03.

AI, M. (2023c). Meta and Microsoft Introduce the Next Generation of Llama. Accessed : 2025-05-05. Récupéré de https://about.fb.com/news/2023/07/llama-2/

AI, M. (2024). Introducing Meta Llama 3 : The most capable openly available LLM. Accessed : 2025-05-05. Récupéré de https://ai.meta.com/blog/meta-llama-3/

AI, P. (2022). Perplexity AI. Accessed : 2025-05-05. Récupéré de https://www.perplexity.ai/

Akmal, M. A. B. B. (n.d.). The ultimate guide to prompt engineering – techniques, examples, and success stories. https://www.xgrid.co/resources/guide-to-prompt-engineering/. Accessed : October 15, 2023.

Alayrac, J.-B., Donahue, J., Luc, P., Miech, A., Barr, I., Hasson, Y., Lenc, K., Mensch, A., Millican, K., Reynolds, M. *et al.* (2022). Flamingo : A visual language model for few-shot learning. *Advances in neural information processing systems*, *35*, 23716–23736.

Alur, R. et Dill, D. L. (1994). A theory of timed automata. *Theoretical computer science*, *126*(2), 183–235.

Alur, R. et Madhusudan, P. (2004). Visibly pushdown languages. Dans *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, 202–211.

Angluin, D. (1987). Learning regular sets from queries and counterexamples. *Information and computation*, *75*(2), 87–106.

Anthropic (2024). Introducing the Claude 3 Model Family. Accessed : 2025-05-05. Récupéré de https://www.anthropic.com/news/claude-3-family

Benaich, N. et Capital, A. S. (2024). State of AI Report 2024. https://www.stateof.ai/. Accessed May 3, 2025.

Bender, E. M., Gebru, T., McMillan-Major, A. et Shmitchell, S. (2021). On the dangers of stochastic parrots : Can language models be too big ?. Dans *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, 610–623.

Bengio, Y. (2024). Paris AI Safety Breakfast #3 : Yoshua Bengio. Online event organized by Future of Life Institute. Discussed risks of LLMs in code generation, emphasizing their tendency to produce subtle errors/vulnerabilities despite coherent outputs. Highlighted the need for formal verification frameworks (like automata) to validate AI outputs in critical environments. Event date : October 16, 2024. Récupéré de https://futureoflife.org/ai-policy/breakfast-3-yoshua-bengio/

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A. *et al.* (2020a). Language models are few-shot learners advances in neural information processing systems 33.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I. et Amodei, D. (2020b). Language models are few-shot learners. Dans H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, et H. Lin (dir.). *Advances in Neural Information Processing Systems*, volume 33, 1877–1901. Curran Associates, Inc. Récupéré de https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf

Bubeck, S., Chadrasekaran, V., Eldan, R., Gehrke, J., Horvitz, E., Kamar, E., Lee, P., Lee, Y. T., Li, Y., Lundberg, S. *et al.* (2023). Sparks of Artificial General Intelligence : Early experiments with GPT-4.

ByteDance (2025). Bytedance launches AI model Doubao 1.5 Pro. Accessed : 2025-05-05. Récupéré de https://www.techinasia.com/news/bytedance-launches-ai-model-doubao-15-pro

Chatterjee, A., Renduchintala, H. K., Bhatia, S. et Chakraborty, T. (2024). POSIX : A Prompt Sensitivity Index For Large Language Models. *arXiv preprint arXiv :2410.02185*.

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. D. O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G. *et al.* (2021). Evaluating large language models trained on code. *arXiv preprint arXiv :2107.03374*.

Clarke, E. M., Grumberg, O. et Peled, D. A. (2018). *Model Checking* (2nd éd.). Cambridge, MA, USA : MIT Press.

DeepMind, G. (2023). Gemini : Google's multimodal AI models. https://deepmind.google/technologies/gemini/. Accessed : 2025-05-03.

DeepMind, G. (2024). Gemini 2.0 flash. Accessed : 2025-05-05. Récupéré de https://deepmind.google/technologies/gemini/flash/

DeepSeek (2025). DeepSeek-V3-0324 release. Accessed : 2025-05-05. Récupéré de https://api-docs.deepseek.com/news/news250325

Deutsch, A., Nava-Sedeño, J. M., Syga, S. et Hatzikirou, H. (2021). BIO-LGCA : A cellular automaton modelling class for analysing collective cell migration. *PLoS computational biology*, *17*(6), e1009066.

Devlin, J., Chang, M.-W., Lee, K. et Toutanova, K. (2019). Bert : Pre-training of deep bidirectional

transformers for language understanding. Dans *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics : human language technologies, volume 1 (long and short papers)*, 4171–4186.

Diekert, V. et Muscholl, A. (2011). Trace theory.

Durbin, R., Eddy, S. R., Krogh, A. et Mitchison, G. (1998). *Biological sequence analysis : probabilistic models of proteins and nucleic acids*. Cambridge university press.

Errica, F., Siracusano, G., Sanvito, D. et Bifulco, R. (2024). What did i do wrong ? quantifying LLMs' sensitivity and consistency to prompt engineering. *arXiv preprint arXiv :2406.12334*.

Gao, Y. et Glowacka, D. (2016). Deep gate recurrent neural network. Dans *Asian conference on machine learning*, 350–365. PMLR.

Giray, L. (2023). Prompt engineering with ChatGPT : A guide for academic writers. *Annals of Biomedical Engineering*, *51*, 3. `http://dx.doi.org/10.1007/s10439-023-03272-4`

GitHub (2023). Github copilot : Your AI pair programmer. `https://github.com/features/copilot`. Accessed : 2025-05-03.

Gozzi, M. et Di Maio, F. (2024). Comparative analysis of prompt strategies for LLMs : Single-task vs. multitasking prompts.

Group, A. (2025). Alibaba unveils advanced Qwen 3 AI as chinese tech rivalry intensifies. Accessed : 2025-05-05. Récupéré de `https://www.reuters.com/business/media-telecom/alibaba-unveils-advanced-qwen-3-ai-chinese-tech-rivalry-intensifies-2025-04-29/`

Henzinger, T. A. (1996). The theory of hybrid automata. Dans *Proceedings 11th Annual IEEE Symposium on Logic in Computer Science*, 278–292. IEEE.

Hopcroft, J. E., Motwani, R. et Ullman, J. D. (2001). Introduction to automata theory, languages, and computation. *Acm Sigact News*, *32*(1), 60–65.

Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y. J., Madotto, A. et Fung, P. (2023). Survey of hallucination in natural language generation. *ACM computing surveys*, *55*(12), 1–38.

Khot, T., Trivedi, H., Finlayson, M., Fu, Y., Richardson, K., Clark, P. et Sabharwal, A. (2022). Decomposed prompting : A modular approach for solving complex tasks. *arXiv preprint arXiv :2210.02406*.

Kleene, S. C. (1951). Representationof events in nerve nets and finite automata. *CE Shannon and J. McCarthy*.

Kong, A., Zhao, S., Chen, H., Li, Q., Qin, Y., Sun, R., Zhou, X., Wang, E. et Dong, X. (2023). Better zero-shot reasoning with role-play prompting. *arXiv preprint arXiv :2308.07702*.

Kuhlmann, M., Gómez-Rodríguez, C. et Satta, G. (2011). Dynamic programming algorithms for transition-based dependency parsers. Dans *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics : Human Language Technologies*, 673–682.

Le Scao, T., Fan, A., Akiki, C., Pavlick, E., Ilić, S., Hesslow, D., Castagné, R., Luccioni, A. S., Yvon, F., Gallé, M. *et al.* (2023). Bloom : A 176b-parameter open-access multilingual language model.

Lou, H.-L. (1995). Implementing the Viterbi algorithm. *IEEE Signal processing magazine*, *12*(5), 42–52.

Marcus, G. (2020). The next decade in AI : four steps towards robust artificial intelligence. *arXiv preprint arXiv :2002.06177*.

Meskó, B. (2023). Prompt engineering as an important emerging skill for medical professionals : tutorial. *Journal of medical Internet research*, *25*, e50638.

Mondorf, P. et Plank, B. (2024). Beyond Accuracy : Evaluating the Reasoning Behavior of Large Language Models – A Survey. Récupéré de `https://arxiv.org/abs/2404.01869`

OpenAI (2023). GPT-4 Technical Report. `https://openai.com/research/gpt-4`. Accessed : 2025-05-03.

Oppenlaender, J., Linder, R. et Silvennoinen, J. (2024). Prompting AI art : An investigation into the creative skill of prompt engineering. Récupéré de `https://arxiv.org/abs/2303.13534`

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A. *et al.* (2022). Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, *35*, 27730–27744.

Pnueli, A. et Rosner, R. (1989). On the synthesis of a reactive module. Dans *Proceedings of the 16th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, 179–190.

Polo, F. M., Xu, R., Weber, L., Silva, M., Bhardwaj, O., Choshen, L., de Oliveira, A. F. M., Sun, Y. et Yurochkin, M. (2024). Efficient multi-prompt evaluation of llms. *arXiv preprint arXiv :2405.17202*.

Qi, J., Xu, Z., Shen, Y., Liu, M., Jin, D., Wang, Q. et Huang, L. (2023). The art of SOCRATIC QUESTIONING : Recursive thinking with large language models. *arXiv preprint arXiv :2305.14999*.

Rabin, M. O. (1963). Probabilistic automata. *Information and control*, *6*(3), 230–245.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W. et Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, *21*(140), 1–67.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell system technical journal*, *27*(3), 379–423.

Shazeer, N. et Stern, M. (2018). Adafactor : Adaptive learning rates with sublinear memory cost. Dans *International Conference on Machine Learning*, 4596–4604. PMLR.

Shum, K., Diao, S. et Zhang, T. (2023). Automatic prompt augmentation and selection with chain-of-thought from labeled data. *arXiv preprint arXiv :2302.12822*.

Sipser, M. (1996). Introduction to the theory of computation. *ACM Sigact News*, *27*(1), 27–29.

Tarjan, R. (1972). Depth-first search and linear graph algorithms. *SIAM journal on computing*, *1*(2), 146–160.

Thompson, K. (1968). Programming techniques : Regular expression search algorithm. *Communications of the ACM*, *11*(6), 419–422.

Tian, Y., Song, H., Wang, Z., Wang, H., Hu, Z., Wang, F., Chawla, N. V. et Xu, P. (2024). Graph neural prompting with large language models. Dans *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 19080–19088.

Tripakis, S.

Turing, A. M. *et al.* (1936). On computable numbers, with an application to the entscheidungsproblem. *J. of Math*, *58*(345-363), 5.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. et Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

Vayadande, K., Pokarne, R., Phaldesai, M., Bhuruk, T., Patil, T. et Kumar, P. (2022). Simulation of conway's game of life using cellular automata. *SIMULATION*, *9*(01), 1–12.

Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E. H., Narang, S., Chowdhery, A. et Zhou, D. (2022). Self-Consistency Improves Chain of Thought Reasoning in Language Models. *arXiv preprint arXiv :2203.11171*.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D. *et al.* (2022). Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, *35*, 24824–24837.

Weiss, G., Goldberg, Y. et Yahav, E. (2021). Thinking like transformers. Dans *International Conference on Machine Learning*, 11080–11090. PMLR.

Zhang, J., Wang, T., Wu, H., Huang, Z., Wu, Y., Chen, D., Song, L., Zhang, Y., Rao, G. et Yu, K. (2025). SR-LLM : Rethinking the structured representation in large language model. *arXiv preprint arXiv :2502.14352*.

Zhang, Z., Zhang, A., Li, M. et Smola, A. (2022). Automatic Chain of Thought Prompting in Large Language Models. *arXiv preprint arXiv :2210.03493*.

Zhuo, T. Y., Huang, Y., Chen, C. et Xing, Z. (2023). Red teaming ChatGPT via Jailbreaking : Bias, Robustness, Reliability and Toxicity. Récupéré de `https://arxiv.org/abs/2301.12867`