ARCHITECTURE COG-ENGINE v1.0 : SPÉCIFICATION TECHNIQUE ET MÉTA-COGNITIVE POUR L'ÉMERGENCE D'UNE INTELLIGENCE AUTONOME

1. Introduction : La Nécessité d'une Rupture Architecturale

L'évolution des systèmes d'intelligence artificielle, passant de simples générateurs de texte probabilistes à des agents cognitifs autonomes, exige une refonte fondamentale des paradigmes d'interaction et de contrôle. Le framework KERNEL, dans ses itérations successives de la v4.0 à la v4.3, a établi les bases d'un ordonnancement cognitif structuré, introduisant des concepts de profilage de tâches, de gestion de l'incertitude et de stratification du raisonnement.[1] Cependant, pour franchir le seuil de l'autonomie réelle, il est impératif de transformer ce qui était un protocole déclaratif (un "prompt") en un moteur opérationnel dynamique : le **COG-ENGINE v1.0**.

Cette spécification technique détaille l'architecture complète de ce moteur. Elle ne se contente pas d'optimiser les instructions existantes, mais propose une infrastructure logicielle complète inspirée par la biologie des systèmes symbiotiques et les avancées récentes en matière de pilotage attentionnel (*Attention Steering*) et de recherche arborescente (*Tree Search*). L'objectif est de pallier les limitations inhérentes aux grands modèles de langage (LLM) — notamment leur nature séquentielle "plate", leur tendance à l'hallucination contextuelle et leur difficulté à maintenir une cohérence logique sur de longues horizons — en les encapsulant dans une architecture de contrôle rigoureuse mais fluide.[2]

1.1. Le Paradigme du Lichen : Une Philosophie de Conception Bio-Inspirée

La philosophie architecturale du COG-ENGINE v1.0 repose sur le **Paradigme du Lichen**. Dans les écosystèmes naturels, le lichen n'est pas un organisme unitaire, mais une structure composite émergente résultant de la symbiose entre un mycobionte (un champignon fournissant la structure, la protection contre la dessiccation et l'extraction minérale) et un photobionte (une algue ou cyanobactérie fournissant l'énergie par photosynthèse).[4] Cette association permet au lichen de coloniser des environnements extrêmes où aucun des deux partenaires ne pourrait survivre seul, créant une résilience systémique supérieure à la somme de ses parties.[5]

Transposé à l'ingénierie cognitive, ce modèle définit la relation entre l'architecture logique et le modèle neuronal :

- **Le Mycobionte (L'Architecture COG-ENGINE)** : Il représente la structure rigide, les protocoles de sécurité, les matrices de décision, la gestion de la mémoire sémantique et le pilotage de l'attention. Il fournit le "squelette" logique qui protège le système contre l'effondrement cognitif, l'incohérence et les dérives hallucinatoires.[6] Il structure l'espace de recherche et impose des contraintes dures.
- **Le Photobionte (Le Modèle LLM)** : Il représente le substrat génératif, le "cerveau" probabiliste capable d'inférence, de créativité, d'abstraction et de traitement du langage naturel. Il fournit l'"énergie" sémantique nécessaire pour peupler la structure définie par le moteur.[2]

Cette approche "organique" vise à concevoir des systèmes logiciels fluides (*Fluid Structures*) [8], capables d'interconnexions complexes et d'adaptation morphologique à la complexité des données, tout en maintenant une intégrité structurelle stricte. Le COG-ENGINE v1.0 est ainsi conçu non comme un simple script, mais comme un écosystème cognitif auto-régulé.[5]

1.2. De la Séquence à la Récursion : L'Évolution du Raisonnement

Les architectures traditionnelles de *Prompt Engineering* opèrent souvent de manière linéaire :

Entrée $\rightarrow$ Traitement $\rightarrow$ Sortie. Le COG-ENGINE v1.0 rompt avec cette linéarité pour adopter une topologie récursive. L'analyse des limitations des versions précédentes du KERNEL a montré que les tâches complexes (C4/C5) nécessitent non seulement une planification, mais une capacité de retour en arrière (*Backtracking*) et de réévaluation dynamique en cours d'exécution.[1]

L'architecture intègre des mécanismes de *Recursive Reasoning* et de *Self-Correction* inspirés des modèles récursifs compacts (*Tiny Recursive Models*) et des boucles de rétroaction agentiques.[11] Le système ne se contente pas de prédire le token suivant ; il maintient et raffine un "état de pensée" latent (z-state) à travers plusieurs cycles d'inférence avant de cristalliser une réponse finale. Cette profondeur temporelle permet de simuler une métacognition : le système "pense à ce qu'il pense" avant d'agir.[13]

---

2. Schéma Logique Global : Une Topologie en Boucle Fermée

Le schéma logique du COG-ENGINE v1.0 est structuré autour d'un cycle de vie cognitif continu (*Cognitive Loop*), divisé en quatre quadrants fonctionnels. Ces quadrants ne sont pas des étapes séquentielles rigides, mais des états dynamiques entre lesquels le système peut transiter en fonction des besoins de la tâche, pilotés par une Matrice de Décision Dynamique

(DDM).

2.1. Quadrant 1 : Perception et Ancrage (Sensing & Grounding)

Ce quadrant correspond à l'interface d'entrée du système, où le signal brut est transformé en une représentation structurée et exploitable. Il intègre et automatise les phases 0 et 1 du KERNEL v4.3.[1]

- **Compilation de la Requête (Query Compilation)** : Le moteur analyse le prompt utilisateur pour détecter les intentions implicites. Il utilise des techniques de reformulation active pour désambiguïser la demande. Si une "Ambiguïté Critique" est détectée (selon les critères v4.2), le système suspend l'exécution pour solliciter une clarification, évitant ainsi une dépense inutile de ressources computationnelles.[1]
- **Profilage Multidimensionnel (Task Profiling)** : Le système classifie la tâche selon trois axes : Typologie (ex: Raisonnement Mathématique, Création Littéraire), Complexité (C1 à C5) et Outils requis. Cette classification détermine le "Budget d'Effort" initial.[1]
- **Initialisation de la Mémoire de Travail** : Le contexte pertinent est chargé depuis la mémoire sémantique (Pattern D), et les "invariants du projet" sont injectés pour cadrer la génération.[15]

2.2. Quadrant 2 : Planification et Expansion (Planning & Expansion)

Une fois la tâche ancrée, le moteur projette des trajectoires de résolution. Ce quadrant remplace la planification linéaire par une approche arborescente.

- **Abstraction Épistémique (Step-Back)** : Pour les tâches de haute complexité ($C \geq 3$), le système exécute une routine de *Step-Back Prompting*. Il s'abstrait des détails spécifiques pour identifier la classe du problème et les principes fondamentaux régissant le domaine (Pattern A, B, C du KERNEL).[16] Cela permet de réduire les erreurs de raisonnement en ancrant la solution dans des lois générales plutôt que dans des heuristiques de surface.
- **Recherche Arborescente (LATS)** : Le moteur utilise le *Language Agent Tree Search* (LATS) pour générer un arbre de possibilités. Au lieu de suivre un seul plan (*Skeleton-of-Thought*), il explore plusieurs branches d'action potentielles, utilisant une recherche Monte-Carlo pour évaluer la viabilité de chaque branche avant de s'y engager.[19]

2.3. Quadrant 3 : Exécution et Raisonnement (Execution & Reasoning)

C'est le cœur opératoire où le LLM (le Photobionte) génère le contenu sémantique, guidé par les structures définies précédemment.

- **Stratégies Cognitives Adaptatives** : Le moteur sélectionne et exécute la stratégie de raisonnement la plus appropriée (Chain-of-Thought, Tree-of-Thought, Program-of-Thought) en fonction de la matrice de décision.[1]

- **Fonctions de Transfert de Pensée** : L'exécution est pilotée par des opérateurs symboliques (détaillés en Section 3) qui forcent des transitions d'état explicites, transformant le flux de texte en une séquence d'opérations logiques.[23]
- **Utilisation d'Outils (Tool Use)** : Si nécessaire, le système interagit avec des environnements externes (APIs, interpréteurs de code) via une boucle ReAct, intégrant les observations directement dans le flux de raisonnement.[3]

2.4. Quadrant 4 : Réflexion et Consolidation (Reflection & Consolidation)

Ce quadrant assure la fiabilité et l'apprentissage du système.

- **Vérification Adaptative (Adaptive Verification)** : Le système critique ses propres productions. Pour les tâches complexes, il déploie des chaînes de vérification (*Chain-of-Verification*) où il génère des questions de contrôle indépendantes pour valider ses affirmations.[16]
- **Backtracking et Correction** : Si la confiance évaluée est insuffisante ou si une contradiction est détectée (Pattern D), le système déclenche un retour en arrière vers un nœud antérieur de l'arbre LATS pour explorer une alternative, plutôt que de persister dans l'erreur.[10]
- **Cristallisation Sémantique** : Les nouvelles connaissances validées sont extraites et stockées dans la mémoire sémantique pour une utilisation future, fermant ainsi la boucle d'apprentissage.[15]

Tableau 1 : Matrice de Décision Dynamique (DDM) pour l'Allocation de Ressources

La DDM est le cerveau du mycobionte, déterminant comment allouer l'attention et le calcul en fonction du profil de la tâche.[14]

Complexité (C)	Stratégie de Raisonnement	Profondeur LATS (Exploration)	Vérification (Rigueur)	Budget d'Attention (Focus)
C1 (Factuel)	Zero-Shot / Direct	1 Branche (Linéaire)	Vérification Syntaxique	Faible (Récupération directe)
C2 (Routine)	Chain-of-Thought (CoT)	1 Branche	Cohérence Interne	Moyen (Suivi procédural)

C3 (Analytique)	Skeleton-of-Thought + CoT	2-3 Branches	Chain-of-Verification (1 cycle)	Haut (Comparaison critique)
C4 (Complexe)	Tree-of-Thought (ToT) + Step-Back	3-5 Branches (MCTS)	Self-Consistency (Vote 2/3)	Très Haut (Abstraction & Détail)
C5 (Recherche)	LATS + Réflexion Récursive	>5 Branches + Backtracking	Vérification Externe + Calibration	Maximal (Méta-cognition active)

---

3. Les Fonctions de Transfert de Pensée (Thought Transfer Functions - TTF)

Pour opérationnaliser le raisonnement et le rendre manipulable par l'architecture, le COG-ENGINE introduit un formalisme symbolique : les Fonctions de Transfert de Pensée (TTF). Ces fonctions agissent comme un jeu d'instructions cognitives, permettant de contrôler les transitions entre les états mentaux du LLM. Elles s'inspirent du protocole **SAAM (Signal-Aligned Activation Manifold)** [23] et des transformations topologiques du **Graph of Thoughts (GoT)**.[29]

3.1. Opérateurs Symboliques de Contrôle

Ces opérateurs sont injectés dans le flux de contexte (via *System Prompt* ou *In-Context Learning*) pour forcer le modèle à structurer sa pensée de manière explicite et traçable.

- **L'Opérateur de Flux (⟶)** :
  - *Fonction* : Marque une transition causale irréversible ou une déduction logique. Il force le modèle à justifier le passage d'un état A à un état B.
  - *Application* : Utilisé pour segmenter les étapes du *Chain-of-Thought*. Chaque étape intermédiaire doit être reliée par cet opérateur, créant une chaîne de causalité vérifiable.
  - *Exemple* : Observation_Symptôme ⟶ Règle_Diagnostic ⟶ Hypothèse_Cause.
- **L'Opérateur de Fusion (+)** :
  - *Fonction* : Combine deux flux de pensée distincts ou applique une contrainte externe à un module cognitif. C'est l'opérateur d'agrégation (*Aggregation*) du GoT.[29]
  - *Application* : Essentiel pour la synthèse de perspectives multiples (Profil "Ouvert" ou "Normatif" du KERNEL v4.3) ou pour fusionner les résultats de plusieurs branches de recherche LATS.

    - *Exemple* : Analyse_Éthique + Contrainte_Légale ⟶ Décision_Conforme.
- **L'Opérateur d'Assignation de Croyance (:=)** :
    - *Fonction* : Cristallise une information dans la mémoire de travail ou sémantique. Il transforme une hypothèse probabiliste en un fait établi pour la suite du raisonnement.
    - *Application* : Utilisé par le Pattern D pour fixer des invariants ("hard constraints") qui ne doivent plus être remis en question, stabilisant ainsi le contexte.[15]
    - *Exemple* : Mode_Dégradé := ACTIF ; Langue_Cible := "Python 3.10".
- **Les Opérateurs de Gestion de Conflit (?? et !!)** :
    - *Fonction* : ?? signale une dissonance cognitive, une incertitude élevée ou une hallucination potentielle détectée par le module de monitoring. !! marque l'action corrective impérative (résolution).
    - *Application* : Déclenchent les routines de *Self-Correction* et de *Backtracking*. L'apparition de ?? interrompt le flux linéaire et force le planificateur à réévaluer l'étape précédente.[23]
    - *Exemple* : Résultat_Calcul?? Estimation_Initiale ⟶!! Recalcul_Vérifié.

3.2. Primitives de Transformation Cognitive

Au-delà des opérateurs atomiques, le COG-ENGINE définit des fonctions macroscopiques qui manipulent la topologie du graphe de pensée.

3.2.1. StepBack(T) : L'Abstraction Récursive

Cette fonction implémente le *Step-Back Prompting*.[16] Elle prend un état de pensée T (détail spécifique) et génère T' (principe abstrait).

- **Mécanisme** : Le système interroge sa base de connaissances pour identifier les "Premiers Principes" régissant le problème T.
- **Usage** : Obligatoire au début du Quadrant 2 pour les tâches de complexité $C \geq 3$. Elle permet de cadrer la solution dans un espace théorique valide avant de tenter une résolution pratique.

3.2.2. Refine(T) : La Boucle de Réflexion

Cette fonction implémente une boucle locale de rétroaction (v, v).[25] Elle prend une pensée T, la critique selon des critères prédéfinis (Phase 6 du KERNEL), et génère une version améliorée T_{opt}.

- **Mécanisme** : Utilise le *Self-Correction* pour itérer sur une réponse jusqu'à satisfaction des critères de qualité ou épuisement du budget local.

- **Usage** : Activée dans le Quadrant 4. C'est le mécanisme principal de l'amélioration de la qualité à l'inférence (*Inference-Time Scaling*).[31]

3.2.3. Branch(T, k) : L'Exploration Divergente

Cette fonction génère k variations ou approches distinctes à partir d'un état T. C'est la base de l'expansion dans l'algorithme LATS.[20]

- **Mécanisme** : Utilise une température élevée ou des instructions de divergence pour explorer l'espace des solutions.
- **Usage** : Fondamental pour les profils "Créatif" et "Ouvert", permettant au système de ne pas s'enfermer dans une ornière locale (optimum local).

---

4. Système de Pilotage de l'Attention (APS - Attention Pilot System)

L'un des défis critiques pour l'autonomie des LLM est la gestion de la fenêtre contextuelle et la dilution de l'attention sur les instructions complexes. Le **Système de Pilotage de l'Attention (APS)** du COG-ENGINE est conçu pour contrer ces effets en modulant dynamiquement l'importance des informations traitées. Il s'appuie sur des recherches récentes en *Attention Steering*, notamment les techniques **Spotlight** [32] et **InstABoost**.[34]

4.1. Architecture du Contrôleur Attentionnel

L'APS agit comme une couche de gouvernance qui intervient au moment de l'inférence pour "guider" le focus du modèle.

4.1.1. Mécanisme "Spotlight" et "InstABoost"

Le système utilise une approche hybride pour garantir que les instructions critiques (le "Mycobionte") ne sont jamais ignorées par le modèle génératif.

- **Mise en Lumière (Spotlighting)** : L'APS identifie les segments du contexte qui contiennent les contraintes invariantes (Pattern D, Règles de Sécurité) et les marque comme zones de haute priorité.
- **Amplification Attentionnelle (InstABoost)** : Inspiré par la technique InstABoost, le système applique un "boost" multiplicatif aux scores d'attention des tokens correspondant aux instructions système.[34] Concrètement, cela peut se traduire par la répétition stratégique, le balisage XML fort (ex: <CRITICAL_INSTRUCTION>) ou, si l'architecture le permet, l'intervention directe sur les vecteurs d'activation. Cela assure une adhésion robuste aux directives, même dans des contextes très longs (Long Context).[36]

4.1.2. Régulation PID de l'Attention

Pour gérer dynamiquement les ressources cognitives, l'APS intègre un contrôleur de type **PID (Proportionnel, Intégral, Dérivé)**.[38]

- **Mesure (Input)** : Le système surveille en temps réel le **Context Reliance Score (CRS)**, qui mesure à quel point la réponse générée s'appuie sur le contexte fourni versus les connaissances paramétriques (hallucinations potentielles).[38]
- **Correction (Output)** :
  - Si le CRS est trop bas (dérive hallucinatoire), le contrôleur augmente le poids des preuves contextuelles et force une relecture (Backtracking).
  - Si le CRS est trop haut (perroquetage sans raisonnement), il encourage l'abstraction via StepBack(T).
  - Ce mécanisme permet une modulation fine et continue de l'équilibre entre créativité et fidélité factuelle.

4.2. Matrice de Décision et Profilage Attentionnel

L'allocation de l'attention n'est pas uniforme ; elle dépend du profil de la tâche identifié en Phase 1b. L'APS configure le comportement du modèle selon la Matrice de Décision Dynamique [14] :

Profil de Tâche	**Stratégie Attentionnelle**	**Cible du "Spotlight" (InstABoost)**	**Comportement Attendu**
Déterministe	Convergente / Focalisée	Faits, Définitions, Contraintes Logiques	Précision maximale, rejet de l'ambiguïté.
Créatif / Ouvert	Divergente / Diffuse	Concepts Latents, Analogies, Espace Sémantique	Exploration de connexions lointaines, variance élevée.
Technique / Code	Syntaxique / Rigide	Syntaxe, API, Variables, Types	Respect strict des formalismes (ex: JSON, Python).
Critique	Comparative / Récursive	Incohérences, Contre-exemples, Biais	Détection active des failles logiques (*Self-Verification*).

5. Architecture de Mémoire et Gestion de l'Incertitude (Pattern D)

La mémoire est le substrat temporel de l'intelligence. Le COG-ENGINE implémente une architecture de mémoire avancée basée sur le framework **CoALA** (Cognitive Architectures for Language Agents) [2], enrichie par le protocole **Pattern D** issu du KERNEL pour la gestion de l'incertitude et des invariants.[15]

5.1. Structure de la Mémoire (Le Modèle CoALA)

Le système distingue trois types de stockage d'information, mimant la cognition humaine :

- **Mémoire de Travail (Working Memory)** : C'est le "tableau blanc" actif. Elle contient le flux de pensée actuel, les nœuds de l'arbre LATS en cours d'exploration et les variables temporaires. Elle est volatile et nettoyée ou résumée à la fin de chaque cycle majeur pour éviter la saturation de la fenêtre de contexte.[2]
- **Mémoire Sémantique (Semantic Memory - Pattern D)** : C'est le stockage des "vérités" et des règles. Elle contient les **invariants du projet** (ex: préférences utilisateur, contraintes techniques immuables) et les connaissances factuelles validées. Ces informations sont injectées de manière persistante via l'APS (Spotlight) pour garantir qu'elles agissent comme des contraintes dures.[15]
- **Mémoire Épisodique (Episodic Memory)** : C'est l'historique des expériences. Elle stocke les traces des raisonnements passés (succès et échecs). Le système utilise cette mémoire pour le *Few-Shot Learning* dynamique, récupérant des exemples de résolution de problèmes similaires pour guider la tâche actuelle.[2]

5.2. Pattern D : Protocole de Gestion de l'Incertitude et Backtracking

Le **Pattern D** est le mécanisme de sécurité épistémique du moteur. Il est conçu pour détecter et gérer les situations où le modèle manque d'information ou de certitude.[26]

5.2.1. Détection et Calibration

Lors de la Phase 6 (Vérification), le système évalue la confiance de ses conclusions (Calibration : Haute, Moyenne, Basse).[1]

- Si l'incertitude dépasse un seuil critique (défini par le profil de tâche), le Pattern D est activé.
- Il force l'utilisation de l'opérateur ?? pour marquer explicitement les zones d'incertitude dans la réponse.

5.2.2. Le Mécanisme de Backtracking

Contrairement à un LLM standard qui "hallucine" pour combler les vides, le COG-ENGINE utilise le Pattern D pour déclencher un **Backtracking** (Retour Arrière).[10]

- **Action** : Le planificateur (Planner) invalide la branche courante de l'arbre LATS.
- **Récupération** : Le système revient au nœud de décision précédent et choisit une stratégie alternative (ex: demander plus d'informations, changer d'angle d'attaque, ou activer le *Mode Dégradé*).

5.2.3. Le Mode Dégradé (Degraded Mode)

Si toutes les tentatives de résolution échouent ou si le budget est épuisé, le Pattern D active le **Mode Dégradé**. Cela produit une réponse structurée en quatre blocs obligatoires, garantissant la transparence intellectuelle [1] :

1. **Réponse Heuristique** : La meilleure approximation possible.
2. **Hypothèses Assumées** : Les postulats pris pour combler les manques.
3. **Plan Non Exécuté** : Ce que le système aurait fait avec plus de certitude/ressources.
4. **Limites Connues** : Avertissements explicites sur la fiabilité.

---

6. Stratégie d'Implémentation et Perspectives

La transformation du KERNEL en COG-ENGINE v1.0 n'est pas une simple mise à jour logicielle, mais un changement de paradigme vers l'**Ingénierie Agentique** (*Agentic Engineering*).

6.1. Déploiement Technique

L'implémentation de cette architecture repose sur l'intégration de composants modulaires :

- **Orchestration** : Utilisation de frameworks comme LangGraph ou AutoGen pour gérer le cycle de vie cognitif et les interactions entre modules.[42]
- **Stockage Vectoriel** : Implémentation de bases de données vectorielles (ex: Pinecone, Milvus) pour supporter la mémoire sémantique et épisodique, permettant une récupération contextuelle rapide (RAG).[5]
- **Système de Récompense (Reward Model)** : Entraînement ou configuration de modèles de récompense légers pour évaluer les nœuds de l'arbre LATS et guider la recherche (Monitor).[47]

6.2. Vers une Science Générale Artificielle

Le COG-ENGINE v1.0, avec sa capacité à raisonner récursivement, à vérifier ses propres hypothèses et à apprendre de ses erreurs via la mémoire épisodique, pose les jalons d'une **Science Générale Artificielle**.[48] Il est capable de mener des investigations autonomes, de formuler des hypothèses et de les tester (virtuellement ou via des outils), mimant ainsi la

démarche scientifique.

Conclusion

Le **COG-ENGINE v1.0** incarne la fusion entre la rigueur structurelle du KERNEL (le Mycobionte) et la puissance créative des LLM modernes (le Photobionte). Grâce aux **Fonctions de Transfert de Pensée**, au **Système de Pilotage de l'Attention** et à une architecture de mémoire résiliente (**Pattern D**), il dépasse les limites du raisonnement linéaire pour offrir une intelligence adaptative, robuste et auditable. Ce système ne se contente plus de traiter des requêtes ; il navigue dans la complexité avec une autonomie surveillée, ouvrant la voie à des collaborateurs artificiels de nouvelle génération.

Tableau Récapitulatif des Composants Clés

Composant	Rôle Cognitif	Technologies Clés & Références
Logic Core	Ordonnancement & Règles	KERNEL v4.3, Matrices de Décision [1]
Recursive Engine	Raisonnement Profond	z-loop, Tiny Recursive Model [11]
Thought Transfer	Manipulation d'État	Opérateurs SAAM (⟶, :=), GoT [23]
Attention Steering	Focus & Contraintes	InstABoost, Spotlight, PID Controller, CRS [34]
Memory Architecture	Persistance & Contexte	CoALA, Pattern D, Vector Stores [2]
Search Strategy	Exploration de Solutions	LATS, MCTS, Backtracking [10]

Sources des citations

1. KERNEL_v4_3_prompt.xml
2. Autonomous Horizons - LLM Agents for Strategic Planning and Execution | Blog, consulté le février 7, 2026, https://ziyang.io/blog/2025-llm-agents
3. How LLM Reasoning Powers the Agentic AI Revolution | by Arash Nicoomanesh - Medium, consulté le février 7, 2026,

https://medium.com/@anicomanesh/how-llm-reasoning-powers-the-agentic-ai-revolution-cbefd10ebf3f
4. Discovery and excavation of lichen bioactive natural products - PMC - PubMed Central - NIH, consulté le février 7, 2026, https://pmc.ncbi.nlm.nih.gov/articles/PMC10149937/
5. Lichen as Multipartner Symbiotic Relationships - MDPI, consulté le février 7, 2026, https://www.mdpi.com/2673-8392/2/3/96
6. Dividual Interweavements. - insert.art, consulté le février 7, 2026, https://insert.art/ausgaben/plant-intelligence/dividual-interweavements/
7. Biomimetic and Biophilic Design of Multifunctional Symbiotic Lichen–Schwarz Metamaterial | Request PDF - ResearchGate, consulté le février 7, 2026, https://www.researchgate.net/publication/370444326_Biomimetic_and_Biophilic_Design_of_Multifunctional_Symbiotic_Lichen-Schwarz_Metamaterial
8. Program and Abstracts The 4th International Conference of Bionic Engineering (ICBE'13), consulté le février 7, 2026, https://fenix.tecnico.ulisboa.pt/downloadFile/563568428720018/010_Pereira%20et%20al_ICBE%202013.pdf
9. The Lichens' Microbiota, Still a Mystery? - PMC - NIH, consulté le février 7, 2026, https://pmc.ncbi.nlm.nih.gov/articles/PMC8042158/
10. LLMs cannot find reasoning errors, but can correct them! - arXiv, consulté le février 7, 2026, https://arxiv.org/html/2311.08516v2
11. Less is More : Recursive Reasoning with Tiny Networks paper ..., consulté le février 7, 2026, https://medium.com/data-science-in-your-pocket/less-is-more-recursive-reasoning-with-tiny-networks-paper-explained-a4573708376d
12. From the logic of coordination to goal-directed reasoning: the agentic turn in artificial intelligence - Frontiers, consulté le février 7, 2026, https://www.frontiersin.org/journals/artificial-intelligence/articles/10.3389/frai.2025.1728738/full
13. Beyond LLMs architecture the path to AGI. - Djimit van data naar doen., consulté le février 7, 2026, https://djimit.nl/beyond-llms-architecture-the-path-to-agi/
14. Multidimensional Benchmarking Framework for AQMs of Network Congestion Control Based on AHP and Group-TOPSIS | Request PDF - ResearchGate, consulté le février 7, 2026, https://www.researchgate.net/publication/349545373_Multidimensional_Benchmarking_Framework_for_AQMs_of_Network_Congestion_Control_Based_on_AHP_and_Group-TOPSIS
15. Context Engineering in Agent. Memory Patterns Core principles and... | by Chier Hu | AgenticAIs | Dec, 2025 | Medium, consulté le février 7, 2026, https://medium.com/agenticais/context-engineering-in-agent-982cb4d36293
16. A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications - arXiv, consulté le février 7, 2026, https://arxiv.org/html/2402.07927v1
17. Is it Time to Start Talking About Prompt Architecture in LLMs? | Towards Data Science, consulté le février 7, 2026,

https://towardsdatascience.com/prompt-architecture-bd8a07117dab/
18. A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications - arXiv, consulté le février 7, 2026, https://arxiv.org/html/2402.07927v2
19. Language Agent Tree Search - Emergent Mind, consulté le février 7, 2026, https://www.emergentmind.com/topics/language-agent-tree-search-lats
20. Language Agent Tree Search Unifies Reasoning, Acting, and Planning in Language Models, consulté le février 7, 2026, https://arxiv.org/html/2310.04406v3
21. Language Agent Tree Search Unifies Reasoning, Acting, and Planning in Language Models - OpenReview, consulté le février 7, 2026, https://openreview.net/pdf?id=njwv9BsGHF
22. Review of Inference-Time Scaling Strategies: Reasoning, Search and RAG - arXiv, consulté le février 7, 2026, https://arxiv.org/html/2510.10787v1
23. SAAM — Signal-Aligned Activation Manifold for AI agents | by Suleiman Tawil - Medium, consulté le février 7, 2026, https://medium.com/@stawils/saam-origin-36975f2ecfab
24. A Guide to Advanced Prompt Engineering - Mirascope, consulté le février 7, 2026, https://mirascope.com/blog/advanced-prompt-engineering
25. Prompt Engineering Techniques for LLMs: A Comprehensive Guide | by Aloy Banerjee, consulté le février 7, 2026, https://medium.com/@aloy.banerjee30/prompt-engineering-techniques-for-llms-a-comprehensive-guide-46ca6466a41f
26. Improving patient pre-screening for clinical trials: assisting physicians with Large Language Models - arXiv, consulté le février 7, 2026, https://arxiv.org/pdf/2304.07396
27. LLM-Augmented Changepoint Detection: A Framework for Ensemble Detection and Automated Explanation - arXiv, consulté le février 7, 2026, https://arxiv.org/html/2601.02957v1
28. (PDF) Modification of Grey Relational Analysis for Dynamic Criteria Weighting in Decision-Making Systems - ResearchGate, consulté le février 7, 2026, https://www.researchgate.net/publication/391771140_Modification_of_Grey_Relational_Analysis_for_Dynamic_Criteria_Weighting_in_Decision-Making_Systems
29. Graph of Thoughts: Solving Elaborate Problems with Large ..., consulté le février 7, 2026, https://ojs.aaai.org/index.php/AAAI/article/view/29720/31236
30. Official Implementation of "Graph of Thoughts: Solving Elaborate Problems with Large Language Models" - GitHub, consulté le février 7, 2026, https://github.com/spcl/graph-of-thoughts
31. Beyond tokens per second: Unlocking smarter enterprise AI with inference-time scaling, consulté le février 7, 2026, https://www.redhat.com/en/blog/smarter-enterprise-ai-inference-time-scaling
32. Spotlight Your Instructions: Instruction-following with Dynamic Attention Steering - arXiv, consulté le février 7, 2026, https://arxiv.org/html/2505.12025v1
33. arxiv.org, consulté le février 7, 2026, https://arxiv.org/html/2505.12025v2
34. Instruction Following by Boosting Attention of Large Language Models - arXiv, consulté le février 7, 2026, https://arxiv.org/html/2506.13734v1

35. Instruction Following by Boosting Attention of Large Language Models - DebugML, consulté le février 7, 2026, https://debugml.github.io/instaboost/
36. The Context Problem: Why RAG Systems Struggle with Information Processing - Medium, consulté le février 7, 2026, https://medium.com/@dvirla84/the-context-problem-why-rag-systems-struggle-with-information-processing-d88350cde4c1
37. Found in the middle: Calibrating Positional Attention Bias Improves Long Context Utilization | Request PDF - ResearchGate, consulté le février 7, 2026, https://www.researchgate.net/publication/384206851_Found_in_the_middle_Calibrating_Positional_Attention_Bias_Improves_Long_Context_Utilization
38. COMPASS: Context-Modulated PID Attention Steering System for Hallucination Mitigation - arXiv, consulté le février 7, 2026, https://arxiv.org/html/2511.14776v1
39. COMPASS: Context-Modulated PID Attention Steering System for Hallucination Mitigation - arXiv, consulté le février 7, 2026, https://arxiv.org/pdf/2511.14776
40. COMPASS: Context-Modulated PID Attention Steering System for Hallucination Mitigation, consulté le février 7, 2026, https://chatpaper.com/paper/211435
41. Architecture Selection for 5G-Radio Access Network Using Type-2 Neutrosophic Numbers Based Decision Making Model | Request PDF - ResearchGate, consulté le février 7, 2026, https://www.researchgate.net/publication/373639811_Architecture_Selection_for_5G-Radio_Access_Network_Using_Type-2_Neutrosophic_Numbers_Based_Decision_Making_Model
42. Agentic UAVs: LLM-Driven Autonomy with Integrated Tool-Calling and Cognitive Reasoning, consulté le février 7, 2026, https://arxiv.org/html/2509.13352v2
43. A Survey of Context Engineering for Large Language Models - arXiv, consulté le février 7, 2026, https://arxiv.org/html/2507.13334v1
44. Elvin-Yiming-Du/Survey_Memory_in_AI: This repository introduce a comprehensive paper list, datasets, methods and tools for memory research. - GitHub, consulté le février 7, 2026, https://github.com/Elvin-Yiming-Du/Survey_Memory_in_AI
45. STRUCTURAL ANALYSIS OF COMPLEX AERIAL PHOTOGRAPHS Makoto Nagao Department of Electrical Engineering Kyoto University Sakyo, Kyot - IJCAI, consulté le février 7, 2026, https://www.ijcai.org/Proceedings/79-2/Papers/001.pdf
46. Language Agent Tree Search - GitHub Pages, consulté le février 7, 2026, https://langchain-ai.github.io/langgraph/tutorials/lats/lats/
47. Inference-Time Scaling for Generalist Reward Modeling - arXiv, consulté le février 7, 2026, https://arxiv.org/pdf/2504.02495
48. (PDF) Virtuous Machines: Towards Artificial General Science - ResearchGate, consulté le février 7, 2026, https://www.researchgate.net/publication/394687885_Virtuous_Machines_Towards_Artificial_General_Science