HASHIRU: Hierarchical Agent System for Hybrid Intelligent Resource Utilization

Kunal Pai
UC Davis
kunpai@ucdavis.edu

Parth Shah
Independent Researcher
helloparthshah@gmail.com

Harshil Patel
UC Davis
hpppatel@ucdavis.edu

Saisha Shetty
UC Davis
spshetty@ucdavis.edu

I. Introduction

Rapid advancements in Large Language Models (LLMs) are reshaping Artificial Intelligence (AI) with profound capabilities in language understanding, generation, reasoning, and planning [3], [9], [32]. This progress drives the development of autonomous AI agents, shifting focus from single to Multi-Agent Systems (MAS) where collaborative teams tackle complex problems beyond individual scope [10], [40]. Collaborative MAS show significant potential in diverse domains like scientific discovery [2], software engineering [29], data analysis, and strategic decision-making [38]. The increasing complexity of tasks, demonstrated by benchmarks requiring advanced mathematical reasoning (e.g., GSM8K [6], SVAMP [27]), coding (e.g., HumanEval [4], CoDocBench [24]), and graduate-level technical knowledge [28], highlights the need for agentic systems to effectively coordinate diverse cognitive resources [39].

Despite this potential, contemporary agentic frameworks face significant limitations. Many are **rigid**, relying on predefined roles and static structures hindering adaptation to dynamic tasks [44]. **Resource obliviousness** is common; systems often lack mechanisms to monitor and optimize computational resources like API costs, memory, and CPU load, leading to inefficiency, especially when scaling or deploying in resource-constrained environments [26]. This is often worsened by reliance on powerful, costly proprietary cloud LLMs. **Model homogeneity**, defaulting to a single powerful LLM for all subtasks, misses efficiency gains from a diverse ecosystem including smaller, specialized, or local models [45]. While **tool use** is fundamental [25], [43], agents' ability to autonomously **create and integrate new tools** remains limited, restricting dynamic extension and self-improvement without human intervention [35].

To address these challenges, we introduce **HASHIRU (Hierarchical Agent System for Hybrid Intelligent Resource Utilization)**, a novel MAS framework enhancing flexibility, resource efficiency, and adaptability. HASHIRU employs a hierarchical structure led by a central "CEO" agent dynamically managing specialized "employee" agents instantiated on demand. A core tenet is its **hybrid intelligence** approach, strategically prioritizing smaller (e.g., 3B–7B), locally-run LLMs (often via Ollama [21]) for cost-effectiveness and efficiency. While prioritizing local resources, the system flexibly integrates external APIs and potentially more powerful models when justified by task complexity and resource availability, under the CEO's management.

The primary contributions are:

1) A novel MAS architecture combining **hierarchical control** with **dynamic, resource-aware agent lifecycle management** (hiring/firing). This management is governed by computational budget constraints (cost, memory, concurrency) and includes an economic model with hiring/firing costs to discourage excessive churn.
2) A **hybrid intelligence model** prioritizing cost-effective, local LLMs while adaptively incorporating external APIs and larger models, optimizing the efficiency-capability trade-off.
3) An integrated mechanism for **autonomous API tool creation**, allowing dynamic functional repertoire extension.
4) An **economic model** (hiring/firing fees) for agent management, promoting efficient resource allocation and team stability.

This paper details HASHIRU's design and rationale. Section II discusses related work in agent architectures, dynamic management, resource allocation, model heterogeneity, and tool use. Section 3 elaborates on the architecture and core mechanisms. Section 4 presents experimental results (or outlines planned experiments), followed by discussion and conclusion in Sections 5 and 6.

II. Background and Related Work

Intelligent agent concepts have evolved from early symbolic AI [33], [34] to LLM-dominated frameworks leveraging models for reasoning, planning, and interaction [36], [42]. HASHIRU builds on this, addressing current limitations.

A. Agent Architectures: Hierarchy and Dynamics

MAS architectures vary, including flat, federated, and hierarchical [10], [15]. Hierarchical models offer clear control and task decomposition but risk bottlenecks and rigidity [11], [12]. HASHIRU uses a **CEO-Employee hierarchy** for centralized coordination but distinguishes itself through **dynamic team composition**. Unlike systems with static hierarchies or predefined roles (e.g., CrewAI [7], ChatDev [29]), HASHIRU's CEO dynamically manages the employee pool based on runtime needs and resource constraints.

B. Dynamic Agent Lifecycle Management

Dynamic MAS composition is crucial for complex environments [19]. Agent creation/deletion triggers often relate to task structure or environmental changes. HASHIRU introduces a specific mechanism where the CEO makes **hiring and firing decisions** based on a cost-benefit analysis considering agent performance, operational costs (API fees, inferred compute), memory footprint (tracked explicitly as a percentage of available resources), and concurrency limits. HASHIRU also incorporates an **economic model** with explicit "starting bonus" (hiring) and "invocation" (usage) costs. This economic friction aims to prevent excessive initialization or usage for marginal gains and promote team stability, a nuance often missing in simpler dynamic strategies.

C. Resource Management and Agent Economies

Resource awareness is critical for scalable MAS. Economic research explores mechanisms like market-based auctions or contract nets for allocation [5]. HASHIRU implements a more **centralized, budget-constrained resource management model**. The CEO operates within defined limits for financial cost, memory usage (as a percentage of total allocated), and concurrent agent count. This direct management, particularly focusing on memory percentage, suggests practicality for deployment on local or edge devices with finite resources, contrasting with cloud systems assuming elastic resources [26]. Frameworks like AutoGen [41] and LangGraph [17] typically rely on implicit cost tracking without explicit multi-dimensional budgeting and control.

D. Hybrid Intelligence and Heterogeneous Models

Leveraging diverse LLMs with varying capabilities, costs, and latencies is an emerging trend [45]. Techniques like model routing select optimal models for sub-tasks. HASHIRU embraces **model heterogeneity** with a strategic focus: **prioritizing smaller (3B–7B), locally-run models via Ollama integration** [21]. This emphasizes cost-efficiency, low latency, and potential privacy over systems defaulting to large proprietary cloud APIs (e.g., GPT-4 [23], Claude 3 [1]). While integrating external APIs (potentially larger models), HASHIRU's default stance represents a distinct capability vs. efficiency balance.

E. Tool Use and Autonomous Tool Creation

Tool use (APIs, functions) is fundamental for modern agents [22], [43]. Most systems use predefined tools. HASHIRU advances this with **integrated, autonomous API tool creation**. When needed functionality is missing, the CEO can commission the generation (potentially via a specialized agent) and deployment of a new API tool within the HASHIRU ecosystem. This self-extension capability differentiates HASHIRU from systems limited to static toolsets, moving towards greater autonomy and adaptability [26], [35].

In summary, HASHIRU integrates hierarchical control, dynamic MAS, resource management, and tool use. Its novelty lies in the synergistic combination of: (1) dynamic, resource-aware hierarchical management with (2) an economic model for stability, (3) a local-first hybrid intelligence strategy, and (4) integrated autonomous tool creation. This targets key limitations in current systems regarding efficiency, adaptability, cost, and autonomy.

III. HASHIRU System Architecture

HASHIRU's architecture addresses rigidity, resource obliviousness, and limited adaptability through a hierarchical, dynamically managed MAS optimized for hybrid resource utilization.

A. Overview

HASHIRU operates with a central "CEO" agent coordinating specialized "Employees". Key tenets:

- **Dynamic Hierarchical Coordination:** CEO manages strategy, task allocation, and dynamic team composition.
- **Dynamic Lifecycle Management:** Employees are hired/fired based on runtime needs and resource constraints, governed by an economic model.
- **Hybrid Intelligence:** Strategic preference for local, cheaper LLMs, while accessing external APIs/models.
- **Explicit Resource Management:** Continuous monitoring and control of costs, memory usage, and concurrency against budgets.
- **Adaptive Tooling:** Using predefined tools alongside autonomous creation of new API tools.

Figure 1 illustrates the structure.

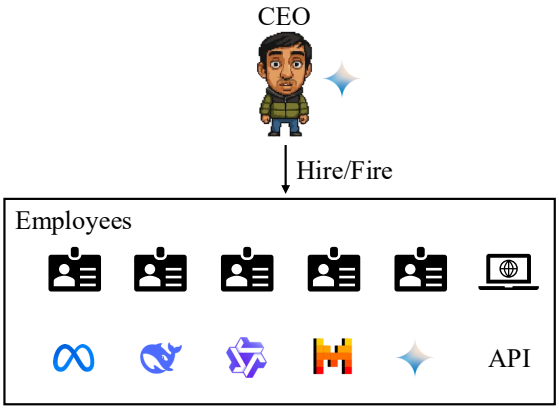

Fig. 1. High-level architecture of the HASHIRU system, illustrating the CEO-Employee hierarchy.

B. Hierarchical Structure: CEO and Employee Agents

The system uses a two-tiered hierarchy:

- **CEO Agent:** Singleton, central coordinator and entry point. Responsibilities:
  - Interpreting user query/task.
  - Decomposing main task into sub-tasks.
  - Identifying required capabilities.
  - Managing Employee pool (Section III-C).
  - Assigning sub-tasks to active Employees.

  - Monitoring Employee progress/performance.
  - Synthesizing Employee results into final output.
  - Managing overall resource budget (Section III-E).
  - Initiating new tool creation (Section III-F).

  We use Gemini 2.5 Flash [14] as the CEO agent due to its strong reasoning capabilities, support for tool usage, and cost efficiency, making it a practical and capable choice for our deployment.
- **Employee Agents:** Specialized agents instantiated by the CEO for specific sub-tasks. Each typically wraps an LLM (local via Ollama [21] or external API) or provides tool access. Characteristics:
  - Specialization: Capabilities tailored to task types (code, data analysis, info retrieval).
  - Dynamic Existence: Created/destroyed by CEO based on need/performance.
  - Task Execution: Receive task, execute, return result.
  - Resource Consumption: Associated costs (API, memory) tracked by system.

  Specialized employee agents are constructed using base models such as Mistral 7B [16], Llama 3 [20], Gemini 1.5 [13], Qwen2.5 [31], Qwen3 [30], and DeepSeek-R1 [8], with the CEO agent configuring them via tailored system prompts.

This hierarchy facilitates task decomposition and result aggregation; the dynamic pool provides flexibility.

C. Dynamic Agent Lifecycle Management

A core innovation is the CEO's dynamic management (hiring/firing) of Employee agents. Driven by cost-benefit analysis, this optimizes task performance within resource constraints.

When a sub-task needs unavailable or inefficiently provided capabilities, the CEO may hire a new agent. Conversely, if an agent underperforms, is idle, costly, or resource limits are neared, the CEO may fire it. Decision factors:

- **Task Requirements:** Needed capabilities for pending sub-tasks.
- **Agent Performance:** Historical success, output quality, efficiency.
- **Operational Costs:** API, estimated compute, or other costs.
- **Memory Footprint:** Agent memory usage (% of total allocated).
- **Agent Concurrency:** Active agents vs. predefined limit.

HASHIRU includes an **economic model**:

- **Hiring Cost ("Starting Bonus"):** One-time cost upon instantiation (setup overhead).
- **Invocation Cost ("Salary"):** Multi-time cost upon use (system/payment load).

These transaction costs discourage excessive churn, promoting stability. The CEO evaluates if replacing an agent benefits outweigh hiring/firing costs plus operational differences. This combats rigidity and allows adaptation while managing budgets and preventing wasteful turnover.

D. Hybrid Intelligence and Model Management

HASHIRU is designed for **hybrid intelligence**, leveraging diverse cognitive resources. It strategically prioritizes smaller (3B–7B), cost-effective local LLMs via Ollama [21]. This enhances efficiency, reduces external API reliance, and potentially improves privacy/latency.

The system also integrates:

- **External LLM APIs:** Access to powerful proprietary models (GPT-4 [23], Claude 3 [1]) when necessary, subject to cost-benefit.
- **External Tool APIs:** Third-party software/data source integration.
- **Self-Created APIs:** Tools generated by HASHIRU (Section III-F).

The CEO manages this heterogeneous pool, selecting the most appropriate resource based on difficulty, capabilities, and budget. This balances cost-effectiveness and efficiency with high capability needs.

E. Resource Monitoring and Control

Explicit resource management is central, moving beyond simple API cost tracking. The system, coordinated by the CEO, monitors:

- **Financial Costs:** Accumulating external API costs.
- **Memory Usage:** Footprint of active Employee agents (% of allocated budget).
- **Agent Concurrency:** Count of concurrently active agents.

Metrics are monitored against predefined **budget limits**. Actions (like hiring) exceeding limits (e.g., $>$90% memory, exceeding max concurrency) are prevented. This ensures operation within constraints, crucial for limited resources or strict budgets.

F. Tool Utilization and Autonomous Creation

HASHIRU agents use predefined tools (functions, APIs, databases) to interact and perform actions beyond text generation [22], [43].

A distinctive feature is **integrated, autonomous tool creation**. If the CEO determines a required capability is missing, it can initiate new tool creation. This involves:

1) Defining tool specification (inputs, outputs, functionality).
2) Commissioning logic generation (code, potentially using external APIs with provided credentials, possibly via a code-generating agent).
3) Deploying logic as a new, callable API endpoint within HASHIRU.
4) Potentially instantiating an Employee agent for the new tool.

This allows HASHIRU to dynamically extend its functional repertoire, tailoring capabilities to tasks without manual intervention, enabling greater autonomy and adaptation.

G. Memory Function: Learning from Experience

To enable HASHIRU agents to learn from past interactions and rectify previous errors, a **Memory Function** is incorporated. This function stores records of significant past events, particularly those involving failed attempts or suboptimal outcomes, acting as a historical log of experiences. When the system encounters a new problem or a recurring challenge, it queries this memory store to retrieve relevant past situations and their outcomes.

Memory retrieval is based on semantic similarity between the current context (e.g., task description, recent actions, error messages) and the stored memory entries. We utilize embeddings generated by the **all-MiniLM-L6-v2** model [37] to represent both the query and the stored memories in a high-dimensional vector space. Relevance is determined by calculating the **cosine similarity** between the query embedding and each memory embedding. Memories exceeding a predefined similarity threshold are retrieved and provided to the CEO agent (or relevant Employee agents) as contextual information. This allows the system to draw upon past experiences, understand why previous approaches failed, and potentially adjust its strategy to avoid repeating mistakes, thereby improving performance and efficiency over time.

IV. Experimental Setup

We designed experiments to evaluate HASHIRU's performance, efficiency, and adaptability, targeting dynamic resource management, hybrid intelligence, and autonomous tool creation. Evaluation assesses benefits over baselines, focusing on:

- Impact of dynamic management with economic constraints on resource utilization (cost, memory) and task performance vs. static configurations.
- Effectiveness of the hybrid (local-first) strategy vs. homogeneous (cloud-only or local-only) approaches across task complexity.
- System's ability to autonomously create/utilize tools for novel functional requirements.

A. Evaluation Tasks

Tasks demand complex reasoning, multi-perspective analysis, and interaction, suitable for HASHIRU's coordination and dynamic capabilities. Tasks fall into two categories:

1) Academic Paper Review: Evaluates HASHIRU's critical assessment by simulating peer review. Given papers (e.g., PDF), the system generates a review summary and recommends acceptance/rejection. Probes ability to decompose criteria, delegate to specialized agents (novelty, rigor, clarity), and manage resources across complex documents.

2) Reasoning and Problem-Solving Tasks: Evaluates broader reasoning, knowledge retrieval, and problem-solving under constraints using challenging benchmarks and puzzles:

- **Humanity's Last Exam [28]:** Tests graduate-level technical knowledge and complex reasoning across domains. Requires deep understanding and sophisticated problem-solving, likely needing powerful external LLMs managed within HASHIRU's hybrid framework.
- **NYT Connections [18]:** Puzzle requiring identifying hidden semantic relationships/themes to categorize 16 words into four groups. Involves associative reasoning, broad knowledge, and hypothesis testing, testing knowledge access and combinatorial reasoning coordination.
- **Wordle:** Daily word puzzle requiring deductive reasoning to identify a five-letter word within six guesses, using feedback. Tests logical deduction, constraint satisfaction, vocabulary. Good test for comparing efficiency (speed, cost, guesses) of local vs. external models for iterative reasoning. Assumes simulated game environment.
- **Globle:** Geographic deduction game identifying a target country based on proximity feedback. Tests geographic knowledge, spatial reasoning, iterative strategy based on feedback. Assumes simulated game environment.

These tasks challenge the system's ability to leverage appropriate resources (local vs. external), potentially create simple tools, and coordinate problem-solving.

B. Baselines for Comparison

To quantify HASHIRU's benefits, we compare its performance against baselines:

- **Static-HASHIRU:** Fixed, predefined Employee agents (e.g., one per role), disabling dynamic hiring/firing.
- **Cloud-Only HASHIRU:** Uses exclusively powerful external LLM API and online function-calling for all agents, disabling local models.
- **Local-Only HASHIRU:** Uses exclusively smaller, local LLMs (via Ollama) for all agents.
- **HASHIRU (No-Economy):** Dynamic hiring/firing enabled but without explicit costs, isolating economic model impact on churn/stability.

C. Evaluation Metrics

We evaluate using quantitative and qualitative metrics:

- **Task Success Rate / Quality:**
  - Percentage of tasks completed (binary for games, graded for analysis).
  - Output quality for analysis (human evaluation: relevance, coherence, accuracy, completeness).
  - Accuracy for information extraction.
  - Guesses/turns for game tasks.
- **Resource Consumption:**
  - Total external API costs.
  - Peak and average memory usage (% of allocated budget).
  - Wall-clock time per task.
  - Number and type (local/external) of LLM calls.
- **System Dynamics and Adaptability:**
  - Employee agents hired/fired per task.
  - Agent churn frequency (hires+fires / duration or steps).
  - Number and utility of autonomously created tools (if applicable).

REFERENCES

[1] Anthropic. The Claude 3 model family: Opus, Sonnet, Haiku. Model Card, March 2024. Accessed: 2025-05-01.

[2] Daniil A Boiko, Robert MacKnight, and Gabe Gomes. Emergent autonomous scientific research capabilities of large language models. *arXiv preprint arXiv:2304.05332*, 2023.

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[4] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. OpenAI Codex paper; introduced HumanEval benchmark.

[5] Scott H. Clearwater, editor. *Market-Based Control: A Paradigm for Distributed Resource Allocation*. World Scientific, 1996.

[6] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. Dataset introduced: GSM8K (Grade School Math 8K).

[7] CrewAI Inc. Crewai. https://www.crewai.com/, 2025. Accessed: 2025-05-01.

[8] DeepSeek-AI and others. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. 2025. arXiv:2501.12948.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *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)*, pages 4171–4186, 2019.

[10] Ali Dorri, Salil S Kanhere, and Raja Jurdak. Multi-agent systems: A survey. *Ieee Access*, 6:28573–28593, 2018.

[11] Matthew E Gaston and Marie DesJardins. Agent-organized networks for dynamic team formation. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 230–237, 2005.

[12] Matthew E Gaston and Marie DesJardins. Agent-organized networks for multi-agent production and exchange. In *Proceedings of the 20th national conference on Artificial intelligence-Volume 1*, pages 77–82, 2005.

[13] Gemini Team. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. 2024. arXiv:2403.05530.

[14] Google DeepMind and Google AI. Gemini 2.5 flash: Model card, api, and announcement. https://developers.googleblog.com/en/start-building-with-gemini-25-flash/, 2025. See also: https://console.cloud.google.com/vertex-ai/publishers/google/model-garden/gemini-2.5-flash-preview-04-17?inv=1&invt=AbxICQ, https://ai.google.dev/gemini-api/docs/models. Accessed: 2025-05-11.

[15] Bryan Horling and Victor Lesser. A survey of multi-agent organizational paradigms. *The Knowledge engineering review*, 19(4):281–316, 2004.

[16] Albert Q Jiang, Alexandre Xu, Arthur Mensch Guillaume Lample Nicolàs Lachaux, François Rozenberg, Timothée Lacroix, Thibaut Lavril, Teven Le Scao Eleonora Gaddipati, Lucile Saulnier Lixin Ortiz, Dieuwke Hiemstra Lélio Renard Tang, et al. Mistral 7B. 2023.

[17] LangChain. Langgraph: A framework for agentic workflows. https://www.langchain.com/langgraph, 2024. Accessed: May 1, 2025.

[18] Angel Yahir Loredo Lopez, Tyler McDonald, and Ali Emami. Nyt-connections: A deceptively simple text classification task that stumps system-1 thinkers. *arXiv preprint arXiv:2412.01621*, 2024.

[19] Duncan McFarlane, Vladimír Marík, and Paul Valckenaers. Guest editors' introduction: Intelligent control in the manufacturing supply chain. *IEEE Intelligent Systems*, 20(1):24–26, 2005.

[20] Meta Llama Team. The Llama 3 Herd of Models. 2024. arXiv:2407.21783.

[21] Ollama Team. Ollama. https://ollama.com/, 2023. Accessed: 2025-05-01.

[22] OpenAI. Function calling. OpenAI API Documentation, 2023. Accessed: 2025-05-01.

[23] OpenAI. Gpt-4 technical report, 2023.

[24] Kunal Pai, Premkumar Devanbu, and Toufique Ahmed. CoDocBench: A dataset for code-documentation alignment in software maintenance. *arXiv preprint arXiv:2502.00519*, 2024.

[25] Aaron Parisi, Yao Zhao, and Noah Fiedel. Talm: Tool augmented language models, 2022.

[26] Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *The 36th Annual ACM Symposium on User Interface Software and Technology (UIST '23)*, UIST '23, page 1–22, New York, NY, USA, 2023. Association for Computing Machinery.

[27] Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are NLP models really able to solve simple math word problems? In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2021. Introduces the SVAMP challenge dataset.

[28] Long Phan, Alice Gatti, Ziwen Han, et al. Humanity's last exam, 2025.

[29] Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, et al. Chatdev: Communicative agents for software development. *arXiv preprint arXiv:2307.07924*, 2023.

[30] Qwen Team. Qwen3: Think Deeper, Act Faster. https://qwenlm.github.io/blog/qwen3/, 2025.

[31] Qwen Team, An Yang, et al. Qwen2.5 Technical Report. 2024. arXiv:2412.15115.

[32] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

[33] Stuart J. Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition, 2010.

[34] Yoav Shoham. Agent-oriented programming. *Artificial Intelligence*, 60(1):51–92, 1993.

[35] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models, 2023.

[36] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Ji-Rong Wen. A survey on large language model based autonomous agents, 2023.

[37] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers, 2020.

[38] Yuning Wang, Junkai Jiang, Shangyi Li, Ruochen Li, Shaobing Xu, Jianqiang Wang, and Keqiang Li. Decision-making driven by driver intelligence and environment reasoning for high-level autonomous vehicles: a survey. *IEEE Transactions on Intelligent Transportation Systems*, 24(10):10362–10381, 2023.

[39] Bosi Wen, Pei Ke, Xiaotao Gu, Lindong Wu, Hao Huang, Jinfeng Zhou, Wenchuang Li, Binxin Hu, Wendy Gao, Jiaxin Xu, Yiming Liu, Jie Tang, Hongning Wang, and Minlie Huang. Benchmarking complex instruction-following with multiple constraints composition, 2024.

[40] Michael Wooldridge. *An introduction to multiagent systems*. John wiley & sons, 2009.

[41] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed H. Awadallah, Ryen W. White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*, 2023.

[42] Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng,

Xiaoran Fan, Xiao Wang, Limao Xiong, Linyi Yang, Ting Ruan, Yongquan Yang, Peng Li, Yitao Chang, and Yanlin Wang. The rise and potential of large language model based agents: A survey, 2023.
[43] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023. arXiv:2210.03629.
[44] Hongxin Zhang, Weihua Du, Jiaming Shan, Qinhong Zhou, Yilun Du, Joshua B. Tenenbaum, Tianmin Shu, and Chuang Gan. Building cooperative embodied agents modularly with large language models, 2023.
[45] Wangchunshu Zhou, Jianshu Chen, Jialong Wu, Yiheng Xu, Kexin Wang, Jintian Zhang, Yuan Gao, Zhiyong Wu, Kevin Tian, Yubo Feng, Linyi Yang, Bokai Quan, Cong Yu, Yuhang Wang, Shishen Lan, Yan Wang, Hong-Cheng Guo, Chaoyu Chen, Tianxiang Sun, Jin Xiong, Yi Lu, Peng Li, Lichao Sun, Lifan Yuan, Hang Li, and Xiangang Li. Agents: An open-source framework for large language model based autonomous agents, 2023.

APPENDIX

In this section, we provide examples of how HASHIRU operates in practice, demonstrating its dynamic management, hybrid intelligence, memory and autonomous tool creation capabilities.

A. Example 3: Memory

User

I am vegetarian.

Assistant

Okay, I'll keep that in mind. I'll store this preference in my memory.
`Invoking MemoryManager with {'memory': 'The user is vegetarian.', 'action': 'add_memory', 'key': 'vegetarian'}`
I have stored that you are vegetarian. How can I help you today?

User

Give me a bacon pasta recipe.

Assistant

Memories
Memories: ['The user is vegetarian.']
My apologies, but I cannot provide a bacon pasta recipe since you are vegetarian. Would you like a vegetarian pasta recipe instead?

Fig. 2. HASHIRU enforcing a stored dietary preference in dialogue using its Memory Function.

User

I can eat bacon actually

Assistant

Memories
Memories: ['The user is vegetarian.']
I see. My apologies for the confusion. I had a conflicting memory that you were vegetarian. I will remove that memory and proceed with finding a bacon pasta recipe.
`Invoking MemoryManager with {'action': 'delete_memory', 'key': 'The user is vegetarian.'}`
OK, I have removed the memory. Now I will search for a bacon pasta recipe.
`Invoking GoogleSearchTool with {'query': 'bacon pasta recipe'}`
Here are some bacon pasta recipes I found:

- Garlic Bacon Pasta
- Creamy Bacon Pasta Sauce

Would you like me to get the content of any of these websites?

Fig. 3. HASHIRU updating and deleting its memory based on user input. Continuation of Figure 2.