***Agent++: A Language and Runtime for Agentic Automation with AI Safety***

**Abstract.**  Agent++ proposes a novel approach to enabling safe and efficient agentic automation through two core components. The first is a conceptual language specification designed to integrate seamlessly with large language models (LLMs). This programming/scripting language aims to be optimized for LLMs to learn, generate, and understand, making it ideal for a wide range of agentic tasks. Conceptually, the language draws inspiration from Python's flexibility and Bash's task-oriented syntax. The second component is an envisioned interpreter/runtime environment tailored to execute this language with a strong focus on safety. This environment would provide robust sandboxing capabilities, allowing users to configure safety settings based on their specific agent use cases.

Agent++ is designed with three primary objectives:

1. Adoptability: The proposed language specification can be integrated into LLMs using training data and tools to be developed.
2. Automation-First Design: The language is intended to support agentic capabilities and automation tasks effectively.
3. Safety and Security: The envisioned runtime environment addresses critical AI safety concerns, enabling customizable and secure execution of automated tasks.

By presenting a theoretical foundation for Agent++, this paper aims to outline a vision for combining innovative language design with a safety-centric runtime to bridge the gap between powerful automation and responsible AI deployment.

**1. Introduction.**

The rapid advancements in large language models (LLMs) have transformed artificial intelligence (AI) across domains, enabling capabilities that were once considered unattainable. Models like GPT-4, PaLM 2, and Claude have demonstrated unprecedented proficiency in understanding and generating human-like text, performing complex reasoning, and assisting in tasks ranging from coding to creative writing. However, as LLMs become more capable, the demand for leveraging them in agentic roles—autonomous entities that can execute tasks, orchestrate workflows, and interact with systems—has grown significantly.

Despite this progress, the landscape of agentic automation remains fragmented in 2024. There is currently no widely adopted standard for defining or executing agent tasks, which complicates the integration of LLMs into agentic workflows. Existing solutions often rely on custom approaches that lack scalability and interoperability. More critically, concerns about AI safety have escalated, particularly in scenarios where agents are granted significant autonomy. Without safeguards, such systems risk unintended behaviors, misuse, or lack of accountability.

Agent++ is proposed as a framework to address these challenges. At its core, Agent++ envisions a standardized language specification and a safety-first runtime environment that together create a foundation for agentic automation. The language is designed to be intuitive for both LLMs and human developers, drawing inspiration from popular programming languages like Python and Bash while focusing on automation and agentic tasks. The envisioned runtime environment prioritizes safety, allowing users to define constraints, monitor execution, and control the scope of agent behavior.

This paper presents the theoretical foundation of Agent++, describing its components, potential use cases, and the challenges it seeks to address. While the ideas outlined here are conceptual, they aim to inspire collaboration and provide a starting point for developing a practical framework for safe and efficient agentic automation in the era of LLMs.

**2. Agent++ Components**