sequence-to-sequence models to synthesize IFTTT or Zapier routines from natural language descriptions [10]. Results were promising but somewhat limited in that the sequence models were able to generate the sequence of functions to call, but not the arguments to those functions.

3.2 Autonomous Agents

LLM-powered autonomous agents are designed to to perform complex and diverse tasks. Usually, this involves decomposing the task into multiple stages or subtasks. Several agent architecture designs have been proposed in the literature [1]. Chain-of-Thought (CoT) [11] is a well-known prompting technique that enables the agent to perform complex reasoning through step-by-step planing and acting. In the earlier CoT implementations, several CoT demonstrations are inserted in the prompt to guide the agent's reasoning process. Alternatively, zero-shot CoT [12] demonstrated the LLM reasoning capabilities by simply adding the sentence "think step by step" in the prompt. Another line of work extended CoT by adopting a tree-like reasoning structure where each intermediate step can have multiple subsequent steps (e.g., [13]). The aforementioned works did not consider feedback in the plan generation process which lead to the development of various agent designs where different feedback signals are considered. As an example, the ReAct agent [14] incorporates observations from the environment (e.g., outcomes of API calls or tools) received after taking an action. These observations are taken into consideration in the next reasoning step (i.e., thought). Human feedback can also help the agent adapt and refine its plan by asking for more details, preferences etc.

Another important part of the agent design is the use of external tools for the action execution. Tools enable the agent to go beyond its internal knowledge. APIs are the most common type of tools used in work such as Gorilla [15] and ToolLLM [16]. In addition to APIs, external knowledge bases can be used as a tool to acquire specific information or expert knowledge [17].

4 System Overview

The system described in this work is implemented as a hierarchical autonomous agent (illustrated in Figure 2), where large language models are used to coordinate the use of a multitude of tools to achieve their tasks. Each agent is comprised of an LLM prompt template, a collection of tools, a variable that summarizes the results of previous reasoning steps of the agent, and a parser capable of interpreting the outputs of the LLM. Each tool is comprised of a single function where the inputs and outputs are both strings and an accompanying textual description that describes to the agent how to use the tool. The hierarchical nature of the system arises from the fact that some of the tools are implemented as more specialized autonomous agents themselves.

We follow the prompt template proposed in the ReAct framework [14], where reasoning traces are generated in addition to task specifications. Since the output formats are conserved across all agents in this work, we make use of the standard ReAct output parser and history summarization function as implemented in LangChain [18].

We detail SAGE's main agent prompt in Figure 10, which includes a high-level description of the home automation task. The instructions are specialized for each sub-agent, as detailed further in Section 5. The prompt takes as arguments the human input, the tool names and descriptions, and the interaction history. The design parameters of the system include the behaviors of the tools, the descriptions of how to use the tools, the descriptions of the agents tasks, and the organization of the agent-tool hierarchy. The full agent-tool hierarchy used in this work is shown in Figure 1. The following section describes in detail the design of the individual tools.

5 Tools

In this section, we introduce a collection of novel tools developed for SAGE, see Table 1 for a comprehensive list and their descriptions. These tools span 4 categories: personalization, device interaction, device disambiguation, and persistent commands. SAGE also