# An Automation Script Generation Technique for the Smart Home

Jiayi Kuang, Gang Xue*, Zeming Yan and Jing Liu

*School of Software, Yunnan University, Kunming, Yunnan, China*
*E-mail: kuang041120@163.com; hill@ynu.edu.cn; guyingmoke@gmail.com; liujing@ynu.edu.cn*
**Corresponding Author*

Received 15 April 2022; Accepted 09 March 2023;
Publication 16 June 2023

## Abstract

A home automation system means monitoring and controlling various kinds of devices in the home remotely using the Internet of things (IoT). Technologies such as natural language processing techniques, user-friendly visual programming, and machine intelligence programming are already available for home automation. For such systems, the increase in the number of devices often makes users focused on the system's ability to perform complex or composing tasks. However, some existing natural language processing systems can only perform simple tasks and cannot meet users' needs. Thus, it is difficult for users to develop the home automation systems they need using visual programming systems because of the large amount of programming knowledge required. Meanwhile, automatic programming without user action can only write a few lines of code and implement little functionality. There are relatively few tools available for generating home automation scripting languages. To address this problem, we propose a practical method for generating executable home automation scripts using Chinese texts. Our method includes the following steps: it extracts critical information from the command sentences in Chinese; it uses first-order logic to check the validity of the extracted information; based on the validation, the correct sentences are mapped into the intermediate language scripts, which can interface with different home platforms. We conducted experiments on Home Assistant, converted intermediate scripts to Home Assistant, and collected 600 scenario descriptions. The experimental results show that the method can automatically generate executable scripts for the Home Assistant platform, and the correct rate was 93.66%.

**Keywords:** Home automation, automation script generation, first-order logic, natural language processing.

## 1 Introduction

Smart home systems are penetrating people's lives, and can monitor or control home attributes such as lighting, climate, entertainment systems, and home appliances, providing residents with a safe, comfortable, and energy-saving home environment. More and more people are choosing to use these systems to obtain a better residential experience [1]. When home automation systems are connected to the Internet, home devices become an important constituent of the IoT [2]. The IoT-enabled smart home applications have gained substantial attention nowadays [3, 4]. Employing wireless connections, such as Bluetooth, Zigbee, and WIFI, provides facilities to implement a wide range of smart home applications [5]. Therefore, these allow homeowners to securely control a large number of appliances regardless of whether they are inside or outside their homes.

Automation is a key technology for smart homes. All intelligent controls and services are built on a complete and mature smart home automation system, according to the end-user requirements, through automatic detection, information processing, analysis, judgment, and manipulation control to realize the process of corresponding control of the smart home [6–8].

Many smart home solutions aim to automate the basic operations of these home devices using various technologies [9]. The technologies currently available to achieve home automation include natural language processing technology [10] which refers to simple command processing, user-friendly visual programming [11], and machine intelligence programming [12]. However, along with the increase in end-user requirements, the tasks of home automation systems are becoming more and more complex. Relying solely on the existing technologies to fulfill the needs of all users is a time-consuming and unrealistic approach. What is more, previous works using natural language processing (NLP) only focus on the concept of "understanding" a simple sentence that can be handled by an instruction, for instance, such as turning on the light, turning off the television, playing media, etc. [13]. However, many tasks cannot be described by a single instruction, such as "When the humidity in the living room is high, the alarm will be turned on, and the dehumidification function of the air conditioner will be activated until it stabilizes." The conditional relations in such descriptions and entity logic relations in sentences are difficult to understand and handle.

Another way is users programming what they need by themselves [14]. Unfortunately, although some visual programming systems are user-friendly, they still require users to consciously think about the programming in a way similar to typical programmers, introducing variables and deciding when/how to use the conditionals, loops, or events [15, 16].

The technology of automatic script generation refers to the use of some techniques to automatically generate the source code for automatic programming. It can therefore improve the efficiency and quality of software development, enhance automation and reduce manual workloads. Script generation technology has matured in various fields, so it has promising applications for home automation [17, 18].

Motivated by this, in this paper, our proposed method solves the problem of generating the automation scripts based on implementing the Chinese text for home automation. In particular, the main contributions can be summarized as follows.

(1) This paper introduces an intermediate language specification that supports the generation of system-independent scripts.
(2) Based on the specification, we propose a method of script generation which achieves home automation. It differs from a simple implementation of commands and uses script generation techniques to implement complex scenarios containing multiple devices and actions.
(3) The proposed method was experimented with and validated on Home Assistant. The experiments results show that the method is able to express 93.66% of the examples in our experiment.

The rest of the paper is structured as follows. Section 2 summarizes the related work. The basic technologies are introduced in Section 3. Section 4 provides the details of how to generate the script. The experiments and the analysis of results are summarized in Section 5, while Section 6 concludes the presented work.

## 2 Related Work

Currently, some approaches focus on using natural language processing, visual programming, and machine programming for home automation. At the same time, the application of script generation techniques in other areas has encouraged our proposed method.

Introducing natural language processing technology into home automation is a significant choice for many studies. A work in the literature [19] implements three basic home appliances by using the ZigBee technology and cloud-based IoT. The user gives the voice command, and it is interpreted by the mobile using the NLP. The mobile acts as a central support, and it decides what type of operation should be fulfilled by appliances according to the requests. Similarly, considering that users can communicate with smart home automation by speaking their native language, allowing the devices to run the command-and-control mechanism more quickly, the study in [20] integrates a cloud-based speech recognition system into smart home automation. All of the methods mentioned before only concentrate on achieving function using a single natural language description, such as closing the light and opening the television, neglecting that end-users always have complex requirements that a single description cannot satisfy. Unlike these approaches, our proposed method automatically generates the executable scripts to complete more complex tasks based on the user's Chinese description, including various home devices, events, conditions, and actions. The complexity refers to the difference between a scenario and a normal command. The scenarios contain a series of devices linked together, and scenarios can be executed repeatedly. In contrast, a command is for a single device and can only be executed once.

Some programming tools focus on designing visual programming systems for homes in which users can synthesize some smart home system functions according to their complex and diverse requirements. Existing earlier work [21] builds a prototype of the home automation system with the visual programming solution and code generation module. Meanwhile, previous work [22] designs a smart home system that adopts message queuing telemetry transport (MQTT) as the communication protocol of the smart home system and introduces the component-oriented flow-based programming (FBP). Those systems are more friendly for the end-users to combine the tasks they need. However, they require a certain degree of complexity knowledge that most users have difficulty understanding and mastering. Moreover, they mainly work for specific phone or computer operating systems. In contrast, our proposed method automatically generates scripts based on the user's Chinese text without complex operations and knowledge, and the proposed intermediate language specification can support most of the home automation systems. There are many kinds of home automation systems. The compatibility between devices is a big problem. This paper solves the compatibility problem by introducing intermediate language specifications. The extracted Chinese content is not directly converted into scenario scripts but first converted into intermediate language specifications and then into executable scenario scripts under the target home automation system. For different home automation systems, only the step of converting the intermediate language specification to specific home automation differs in the processing flow. It is only necessary that the particular home automation platform can support the intermediate language specification. In this paper, we take the Home Assistant platform as an example. We do not experiment with many platforms. That is because only the conversion of the intermediate language specification is different. Simulating a series of devices under the home automation system is a lot of work. Home Assistant is chosen because it is a representative platform with good compatibility between devices.

The ability to implement complex scenario functionality and the degree of automation integration are the key metrics to consider when using home automation systems. Some existing studies use artificial intelligence (AI) methods to replace programmers to write programs [23]. Although some human intelligence technologies like machine learning are a breakthrough to automatic programming without any user operations, they could write a few lines of code and achieve a little function [24, 25]. Such a small amount of code cannot satisfy the growing demand for smart homes and a complex device environment of the home. In contrast to these works, the method proposed in this paper includes an automatic generation of scripts for home automation. Its automation process is easier and does not require any training process. At the same time, this method does not require a training model and can save computing resources. What is more, our proposed method uses cheap devices such as Raspberry Pi to provide a simple edge service that can reduce the pressure on the core network while protecting user privacy. Every command given by the user needs to be transmitted to the cloud in some existing smart home systems, which will leak the information of various devices in the user's home. As shown in Figure 1, since there is no need to train the model, the devices don't need to have strong performance to deploy the service. Because the method in this paper can deploy the text processing

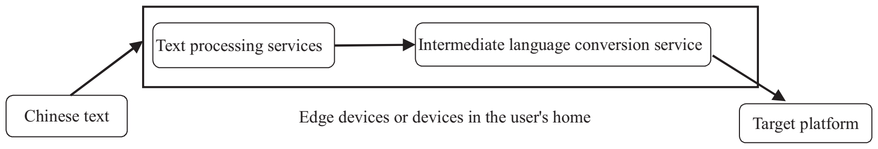

Text processing services
Intermediate language conversion service
Chinese text
Edge devices or devices in the user's home
Target platform

**Figure 1** The details of the architecture in this paper.

service and the intermediate language specification conversion service to the edge area or the user's home, there is no need to transfer information about the user's devices, etc., to the cloud. Therefore, user privacy can be protected.

Automation using script generation technology involves robots and smartphone fields. The existing works [26] propose a practical framework named Nerva, which can automatically synthesize robot applications using natural language descriptions. Other examples of the existing works [27] present Smart Synth, a novel end-to-end programming system for synthesizing smartphone automation scripts generated using natural language descriptions. Inspired by these papers, our proposed method aims to automatically generate home automation script directly utilizing user-natural language descriptions. In such a way, users can enter Chinese text containing the description of various complex device events, conditions, and actions to generate scripts.

## 3 Preliminaries

This section introduces basic technologies, including natural language processing, first-order logic, intermediate language specification, and related tools.

### 3.1 Natural Language Processing

Natural language processing (NLP) is a subfield of artificial intelligence (AI). This is a widely used technology for personal assistants that are used in various business fields/areas. This technology works on the speech provided by the user, breaks it down for proper understanding and processes accordingly. Natural language processing is an upcoming field where already many transitions such as compatibility with smart devices and interactive talks with a human have been made possible [28]. Knowledge representation, logical reasoning, and constraint satisfaction were the emphasis of AI applications in NLP. Here, first it was applied to semantics and later to the grammar.

### 3.1.1 Lexical semantics

Lexical semantics is a branch of linguistics which is concerned with the systematic study of word meanings. Two of the most fundamental questions addressed by lexical semanticists are the following ones: (a) how to describe the meanings of words, and (b) how to account for the variability of meaning from context to context. These two issues are necessarily connected, since an adequate description of meaning must be able to support our account of variation and our ability to interpret it. The study of contextual variation leads in two directions: on the one hand, to the processes of selection from a range of permanently available possibilities; and on the other hand, to the creation of new meanings from old, by such means as metaphor and metonymy, in response to contextual pressure [29].

The first part of the semantic analysis, studying the meaning of individual words is called lexical semantics. It includes words, sub-words, affixes (sub-units), compound words, as well as phrases. All these elements are collectively called lexical items [30]. In other words, lexical semantics reflects the relationship between the lexical items, the meaning of sentences, and the syntax of a sentence.

Lexical semantics include the following steps: Classification of lexical items like words, sub-words, and affixes, is performed in lexical semantics. Next, decomposition of lexical items like words, sub-words, and affixes, is performed in lexical semantics. Finally, differences and similarities between various lexical semantic structures is also analyzed [31].

### 3.1.2 Named entity recognition (NER)

The goal of the NER is to label names of people, places, organizations, and other entities of interest in text documents. There are three major approaches to NER: lexicon-based, rule-based, and machine learning-based. However, an NER system may combine more than one of these categories. Some approaches to NER rely on POS tagging [32]. NER is a preprocessing step for tasks such as information or relationship extraction.

## 3.2 First-order Logic

First-order logic known as predicate logic, quantificational logic, and first-order predicate calculus is a collection of formal systems used in mathematics, philosophy, linguistics, and computer science. The first-order logic keeps all the Boolean operators of propositional logic. However, it adds some important new mechanisms. The propositions are analyzed into predicates and arguments that take it a step closer to the structure of natural language [33]. The standard construction rules for the first-order logic recognize terms such as individual variables and individual constants, and predicates which take different numbers of arguments.

### 3.2.1 Lambda calculus

Lambda calculus (also written as $\lambda$-calculus) is a formal system in mathematical logic for expressing computation based on function abstraction and application using variable binding and substitution. Lambda calculus, as a widely used calculation model, can clearly define what a computable function is, and any computable function can be expressed and evaluated in this form [34].

The syntax of the lambda calculus defines some expressions as valid lambda calculus expressions and some as invalid. A valid lambda calculus expression is called a "lambda term". The following three theoretical rules give an inductive definition that can be applied to build all syntactically valid lambda terms: First of all, a variable, $x$, is itself a valid lambda term [35]. Next, if $t$ is a lambda term, and $x$ is a variable, then $(\lambda x, t)$ is a lambda term (called an abstraction). An abstraction $(\lambda x, t)$ is a definition of an anonymous function that is capable of taking a single input $x$ and substituting it into the expression $t$. It thus defines an anonymous function that takes $x$ and returns $t$; Finally, if $t$ and $s$ are lambda terms, then $(t, s)$ is a lambda term (called an application).

### 3.2.2 Alpha equivalence and beta reduction

There are two evaluation rules in first-order logic: Alpha equivalence (or conversion) and beta reduction [36].

A basic form of equivalence, definable on lambda terms, is alpha equivalence [37]. It captures the intuition that the particular choice of a bound variable, in an abstraction, does not (usually) matter [38].

The beta reduction rule states that an application of the form $\lambda(x.t)s$ reduces to the term $t[x := s]$. The $\lambda$-calculus may be seen as an idealized version of a functional programming language. Under this view, beta reduction corresponds to a computational step [39]. This step can be repeated continuously by the additional beta reduction in an iterative way, until there are no more applications left to reduce. In the untyped $\lambda$-calculus, as presented here, this reduction process may not terminate completely. That is, the term reduces to itself in a single beta reduction, and therefore the reduction process will never terminate [40]. Another aspect of the untyped $\lambda$-calculus is that it does not distinguish between the different kinds of data.

## 3.3 Intermediate Language Specification

An intermediate language specification with representative features of the current smart homes is introduced in this paper. Our proposed method can port the automation scripts to other smart home platforms using it. It is designed to represent a wide variety of smart home tasks, and keep the structure information from the NLP. The syntax of the intermediate language specification is shown in Figure 2.

In Figure 2, a script ***S*** represents a new scenario. ***P*** is a set of parameters. ***B*** is the main body of the module. ***B*** consists of a sequence of instructions that executed in order. A condition ***C*** consists of an event for a triggered scenario and ***I*** to be executed when the event is raised. Each instruction ***I*** contains the attribute set ***Attr*** and the action ***A***. The ***T*** represents the conversion function, which converts the intermediate instructions to target instructions, and it contains the instruction conversion function *Trans_I* and the attribute conversion function *Trans_Attr*.

The intermediate language specification mainly includes events, conditions and actions:

- Events: These represent events that occur in the daily home environment. For example, the event of people going home, or the event of high humidity in the living room. These events trigger scenario generation.
- Conditions: These denote the conditions on the state of the intelligent devices in the home. For example, the light is on or off, or the level of the current temperature.

```
Scenario S ::= P E C B
Parameter P ::= input($i_1, \dots, i_n$)
Event E ::= ($e_1, \dots e_n$) := when Event ()
Condition C ::= if ($\prod_1 \wedge \cdots \wedge \prod_n$) then
Body B ::= $I_1 \dots I_m$
Instruction I ::= A Attr | A
Attributes Attr ::= atrr($i_1, \dots, i_n$)
Action A ::= Action(Attr)
Conversion T ::= Convert(A, Attr):
        Target_I := Trans_I(A)
        foreach temp ∈ Attr
            do
                Target_I := Trans_Attr(Target_I, temp)
            od
```

**Figure 2** The syntax of the intermediate language specification.

- Actions: These are normal control home operations, such as turning off the television or light connection, identifying face, or speaking “Hello”. It executes a sequence of actions when its event is triggered and the conditions are satisfied.

An example is given to explain the intermediate language specification syntax in this part. The Chinese description is “当下午五点，打开热水器并且将其温度调到 45 度, 同时打开客厅的 LED 灯, 然后打开过道的灯。”. The message (At 5:00 p.m., the water heater is turned on and set to 45 degrees, and the LED lights in the living room are turned on, followed by the lights in the hallway.) corresponds to the intermediate instruction shown in Figure 3.

```
when (trigger, parameter): = Message Received: = time trigger
  if (trigger = "17:00") then
     instruction $i_1$ := Turn on the water heater ()
     instruction $i_2$ := Turn on the LED light (Living room)
     instruction $i_3$ := Turn on the light (Hallway)
       $target\_i_1$ := convert($i_1$, 45°)
       $target\_i_2$ := convert($i_2$, Living room)
       $target\_i_3$ := convert($i_3$, Hallway)
execute  $target\_i_1$ , $target\_i_2$ , $target\_i_3$
```

**Figure 3** Intermediate instruction corresponding to the example.

## 3.4 Related Technical Tools

The tools covered in this paper are Natural language Toolkit (NLTK) and Home Assistant. The NLTK is used for the first-order logic to discriminate between the true and false sentences. The Home Assistant is used to build home scenarios and deploy experiments.

### 3.4.1 NLTK

The NLTK is a complete toolkit for all the NLP techniques in which the method has been used. The NLTK is written in Python language, a simple yet powerful scripting language with excellent functionality for processing linguistic data. The NLTK defines a basic infrastructure that can be used to build the NLP programs in Python [41].

### 3.4.2 Home assistant

Home Assistant is a mature and complete Python-based smart home automation system.[1] Due to its advantages, Home Assistant was used in the experiment to build smart home scenarios. Then the experiments were conducted

[1]https://www.home-assistant.io/

by combining the collected data with our proposed method running on Home Assistant to verify its effectiveness.

## 4 The Script Generation Method

This chapter describes the technical details of the script generation method for home automation. In total, the steps for generating scripts contain text processing, validity check, and generating intermediate scripts. The main processes of the text processing include the scenario trigger condition extraction, operation extraction, and attribute extraction. The general steps of script generation are shown in Figure 4.

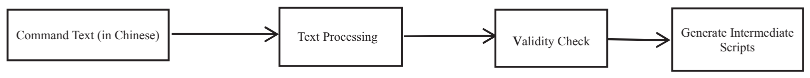


**Figure 4** Steps for script generation.

Scripts generated from natural language scenarios given by the user are called scenario scripts. In the scenario script, the processing flow can divide into the two parts, including the scenario recognition and command recognition. The related symbol descriptions of the algorithm are shown in Table 1.

**Table 1** Related symbol descriptions of the algorithm

| Symbol | Description |
|---|---|
| $\mathbf{F_i}$ | Information about the user's devices $\mathrm{F}_u$ |
| $\boldsymbol{R}_\mathbf{u}$ | User's home environment information |
| $\boldsymbol{D}_\mathbf{u}$ | User-defined dictionaries |
| $\mathbf{F_u}$ | The list of devices for user $u$ |
| $\mathbf{T_{text}}$ | The natural language text to be processed |
| $\mathbf{S}_{\boldsymbol{sentences}}$ | Sentences divided according to pause symbols in the text |
| $\boldsymbol{C'}$ | Conditions for scenario triggering |
| $\boldsymbol{S_{remains}}$ | The rest sentences of $\mathrm{S}_{sentences}$ |
| $\boldsymbol{F_{i,obj}}$ | The specific information of device *obj* |
| $\boldsymbol{attr_{obj}}$ | The attributes $attr_{obj}$ of device $obj$, and $attr_{ojb} \in F_i$ |

The steps consist of the following six algorithms. The first step is to perform a pre-load of the device data. Algorithm 1 is the overall processing flow. Algorithm 2 proposes the conditions for the scenario triggering. Algorithms 3 and 4 are used to extract the command to be executed and the attributes to be set. Algorithm 5 checks the validity of the crucial information. Algorithm 6 is used to generate a series of intermediate language scripts [42]. The relationship between the algorithms is shown in Figure 5.

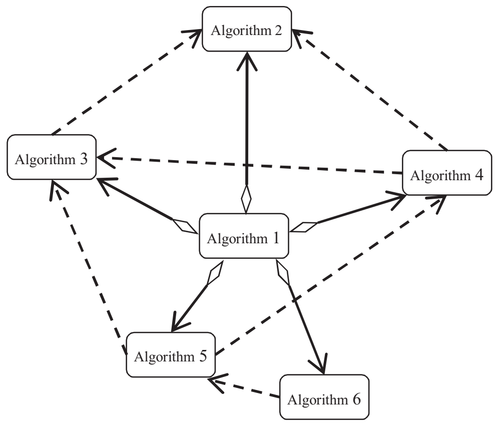


**Figure 5** The relationship between the algorithms.

In the pre-data loading process, the method needs to obtain information about the smart devices in the user's home. They contain some basic information such as which rooms are in the user's home and which devices are in these rooms. Moreover, it also needs to know the device information, including the properties supported by the device actions, such as lights supporting brightness settings. If the user provided the model number of the device, the properties of the device are easily available. Once this information is loaded, the algorithm can proceed to the next process.

Algorithm 1 is the steps for script generation. In Algorithm 1, lines 1 to 4 load the device information and the text information. Line 5 wraps the text in the sentences, and line 6 calls Algorithm 2 via the Trigger function to extract the conditions triggered by the scenario. Line 10 extracts the commands in the text sentence by sentence, line 12 to line 16 extracts a command, which includes the device body and the attributes to be set by calling Algorithms 3 and 4, respectively. Line 17 checks the validation of the extracted command and finally wraps the command and the trigger conditions.

The following scenario is used as an example to demonstrate the script generation process. Take a typical Chinese description "当下午五点时，打开热水器并且将其温度调到 45 度，同时打开客厅的 LED 灯，然后打开过道的灯。(At 5:00 p.m., turn on the water heater and set the temperature to 45 degrees while turning on the LED lights in the living room and then the lights in the corridor.)" as an example.

**Algorithm 1** Steps for script generation

```
Function: Scenario = Generating ($\mathrm{F}_u, \mathrm{T}_{\text{text}}$)
Input:
    $\mathrm{F}_u$ the list of devices for user $u$
    $\mathrm{T}_{\text{text}}$ the natural language text to be processed
Output:
    Scenario A scenario, including the conditions under which the scenario is triggered, the
    instructions that already need to be executed
1:  Load $\mathrm{F}_\mathrm{i}$
2:  Load $R_u$
3:  Load $D_u$
4:  The $\mathrm{T}_{\text{text}}$ is processed by word separation, encapsulating words and lexicalities into
    $T_{word}$
5:  Divide $T_{word}$ into sentence lists $S_{sentences}$ according to the stop sign
6:  $(C', S_{remains})$ = Trigger ($\mathrm{F}_u, \mathrm{S}_{sentences}, \mathrm{F}_\mathrm{i}, \mathrm{R}_\mathrm{u}$)
7:  Initialize a scenario object scenario
8:  $scenario.condition \leftarrow C'$
9:  Initialize a list L of the command class
10: FOR all sentences $sentence \in S_{remains}$
11:   Instantiate an object cmd of the command class
12:   FOR all words $word \in sentence$
13:     IF the operation data of cmd is not finished extracting
14:       Opera ($word,\ R_u,\ F_u,\ cmd$)
15:     ELSE
16:       Attribute ($word, F_i,\ cmd$)
17:   $cmd' =$ Check ($cmd,\ F_i,\ R_u$)
18:   IF $cmd'$ not null
19:     $L \leftarrow cmd$
20: $scenario.commands \leftarrow L$
21: return scenario
```

Algorithm 2 is mainly used to extract the trigger conditions of the scenario. Line 2 traverses the conditional statement and extracts the condition in line 3. Lines 4 and 9 determine the type of the condition, extract the trigger condition and encapsulate it in $S_{remains}$. As shown in Algorithm 2, it defines two modes of triggering the scenario, one is time-triggered (at a fixed point in time, such as when it is dark or at 6:00 pm), and the other is state triggered (when the state is satisfied, such as when I go home, or when the burglar device is triggered). In the above example, 5:00 PM would be extracted as a trigger time.

**Algorithm 2** Scenario trigger condition extraction

```
Function: $(C', S_{remains}) =$ Trigger $(\mathrm{F}_u,\ \mathrm{S}_{sentences}, \mathrm{F_i}, \mathrm{R_u})$
Input:
    $\mathrm{S}_{sentences}$ sentences divided according to pause symbols in the text
    $\mathrm{F}_u$ the list of devices for user $u$
    $\mathrm{F_i}$ information about the user's devices $\mathrm{F}_u$
    $\mathrm{R_u}$ user room information
Output:
    $C'$ Conditions for scenario triggering
    $S_{remains}$ The rest sentences of $\mathrm{S}_{sentences}$
1:  Load $\mathrm{F_i}$
2:  FOR all sentences $sentence \in S_{sentences}$
3:      Identify the temporal gerund $t$ and the temporal entity $e$
4:      IF both $t$ and $e$ satisfy the condition
5:          Create the time condition $time$
6:          $C' \leftarrow time$
7:          BREAK
8:      Identify the state clause $s$
9:      IF $s$ satisfies the state condition
10:         Extraction of object $o$ and the state condition $o \in \mathrm{F}_u$
11:         Create the state condition $state$
12:         $C' \leftarrow state$
13:         BREAK
14: $S_{remains} \leftarrow$ the rest of the sentences in $\mathrm{S}_{sentences}$
15: return $(C', S_{remains})$
```

Algorithm 3 extracts the operation of the instruction using the syntax of Chinese grammar. It includes the device that is the subject of the instruction, the action to be performed, and the location of the device, if it exists. Lines 3 to 4 determine if it is the location of the device based on the list of nouns and locations. Lines 5 to14 identify the device based on the noun and preposition relationship. Lines 15 to 22 identify the action to be performed by the device based on the verb-object structure. Lines 23 to 24 encapsulate the data. Based on Algorithm 3, the three entities in the example, water heater, LED light, and corridor light will be extracted, including the actions and known locations of the entities (e.g., the action "on" for the three devices and the locations of the two lights).

The function of Algorithm 4 is to extract the attributes of the instruction, in the case that they exist. Before this step, the algorithm has already obtained

**Algorithm 3** Operation extraction

```
Function: Opera ($word,\ R_u,\ F_u,\ cmd$)
Input:
    $word$ the word in the text
    $F_u$ the list of devices for user $u$
    $R_u$ user's room information
    $cmd$ the object of the command class
1:  Initialize a global list $name\_buf$ for caching data such as names
2:  Initialize a global action list $action\_buf$ to cache data for verb types
3:  IF $word.lexical \in n$ and $word \in R_u$
4:    $cmd.location \leftarrow word$
5:  ELIF $word.lexical \in (n, vn)$
6:    IF $word \in F_u$
7:      $cmd.obj \leftarrow word$
8:      Load the set of action functions $F_i$ for device $i$
9:    ELIF $\exists x \in (name\_buf + word)$ and $x \in R_u$
10:     $cmd.obj \leftarrow word$
11:     Load the set of action functions $f_i$ for device $i$
12:     $name\_buf \leftarrow empty$
13:   ELSE
14:     $name\_buf \leftarrow word$
15: IF $word.lexical \in (v, vn)$
16:   IF $word \in F_{cmd.obj}$
17:     $cmd.opera \leftarrow word$
18:   ELIF $\exists x \in (action\_buf + word)$ and $x \in F_{cmd.obj}$
19:     $cmd.opear \leftarrow x$
20:     $action\_buf \leftarrow empty$
21:   ELSE
22:     $action\_buf \leftarrow x$
23: IF $cmd.opear \neq null$ and $cmd.obj \neq null$
24:   the operation data of $cmd$ is not finished extracting
25: return $cmd$
```

the body of the device through Algorithm 3, and the subsequent attributes belong to the body of the device. There are two types of the attributes, one is a key–value pair, and the other is value only. Lines 4 to 6 perform a key–value pair determination. Lines 7 to 12 extract the values of the attributes. Lines 13 to 21 reorganize it according to the prepositional relationship. In the previously mentioned example, the temperature property 45° for the water heater will be extracted. The light has no attributes and is set to the default brightness in the script.

**Algorithm 4** Attribute extraction

```
Function: Attribute ($word,\ F_i,\ cmd$)
Input:
    $word$ the word in the text
    $\mathrm{F_i}$ information about the user's devices
    $cmd$ the object of the command class
1:  Initialize a global list $buf$ for caching data
2:  Initialize global variables $key$
3:  Load $F_{i,cmd.obj}$
4:  IF $word.lexical = m$
5:      IF $key \neq null$
6:          $cmd.attributes \leftarrow (key, word)$
7:  IF $word \in F_{i,cmd.obj}$ and $word.lexical \neq m$
8:      IF $key = null$
9:          $key \leftarrow word$
10:     ELSE
11:         $cmd.attributes \leftarrow key$
12:         $key \leftarrow word$
13: ELIF $\exists x \in (buf + word)$ and $x \in F_{i,cmd.obj}$
14:     IF $key = null$
15:         $key \leftarrow x$
16:     ELSE
17:         $cmd.attributes \leftarrow key$
18:         $key \leftarrow x$
19:     $buf \leftarrow empty$
20: ELSE
21:         $buf \leftarrow x$
```

Algorithm 5 performs a validation check to verify the validity of the extracted information. Lines 4 to 7 perform a check on the device location. Line 8 traverses the attributes of the instruction. Lines 10 to 12 check the validation of the attribute values. The invalid commands are eliminated. Algorithm 5 processes the example by checking the validation of the three devices extracted from the above algorithms. All three devices are in the correct position as well as the operation function. The temperature of the water heater is within the legal range.

Once the algorithm has the set of scenario trigger conditions and command objects, the method proposed in this paper converts the scenario into the intermediate language instruction obtained according to the method described in Section 3.3. Algorithm 6 represents the intermediate script generation process.

**Algorithm 5** Validity check

```
Function: $cmd' = \text{Check}(cmd,\ F_i,\ R_u)$
Input:
    $\mathrm{F_i}$ information about the user's devices $\mathrm{F}_u$
    $\mathrm{R_u}$ user room information
    $cmd$ a wrapper object for the command class
Output:
    $cmd'$ instruction object after first-order logic detection
1:  Initialize a command object $cmd'$
2:  Load attributes $attr_{obj}$ of device $cmd.obj$, $attr_{ojb} \in F_i$
3:  Get the corresponding devices $d$ in room $cmd.location$, $cmd.location \in R_u$
4:  IF $cmd.obj \in d$
5:      $cmd'.location \leftarrow cmd.location$
6:  ELSE
7:      return null
8:  FOR all attributes $attr \in cmd.attributes$
9:      Instantiate an object $cmd'$ of the command class
10:     IF $attr.key \in attr_{obj}$
11:         IF $attr.value$ is within legal limits
12:             $cmd'.attributes \leftarrow attr$
13: return $cmd'$
```

**Algorithm 6** Generation of scenario intermediate language scripts

```
1:  scenario name:
2:  Get the trigger condition $\rightarrow$ trigger
3:  Get instructions $\rightarrow$ instruction set
4:  Get command parameters $\rightarrow$ params
5:  when (trigger, parameter): = Message Received
6:  if (trigger = trigger) then
7:  for each in instruction set:
8:      Get and parse instruction each
9:      Get and set parameters $\rightarrow$ each
10:     Generate an intermediate instruction im_instruction
11:     Add im_instruction $\rightarrow$ intermediate instruction set
12:     scenario name:
13:   Get the trigger condition $\rightarrow$ trigger
```

Algorithm 6 converts the extracted scenario-correct crucial information into an intermediate language script. Lines 2 to 6 read the trigger conditions to set them and get the parameters and instruction set. Lines 7 to 13 iterate through each instruction in the instruction set and set the attributes and the values of the attributes. The corresponding results in the example are shown in Figure 6.

```
trigger:
  - type: time
time: '17:00:00'
equipments:
  - equipment: water_heater
      action: open
      attr:
        temperature: 45
  - equipment: LED_light
      action: open
      location: living room
  - equipment: light
      action: open
      location: hallway
```

**Figure 6** The intermediate script for Algorithm 6 corresponds to the example.

## 5 Evaluation

To validate the method proposed above, the experiments are designed on Home Assistant, and a home environment is built on this platform. In addition, the experimental part describes how to convert from the intermediate script to Home Assistant. The real data were collected to verify the effectiveness of the method based on the baseline data. Afterwards, the experiments were analyzed in terms of whether they could be run successfully and to assess the total running time of the generated results.

The experimental environment configurations are shown in Table 2. The two computers are used in the experiment, the Home Assistant Server is used to run Home Assistant, and the other Script Server is used to run the generated smart home automation scripts.

**Table 2** Experimental environment configurations

| Participant | Hardware Condition | Platform Environment | Network |
|---|---|---|---|
| Home assistant server | Intel Core i5–3770 CPU,8GB | Python v3.6.2rc2 | 100Mbps |
| Script Server | Intel Core i5-11320H,16GB | Python v3.8.10 | 100Mbps |

### 5.1 Convert From Intermediate Script to Home Assistant Automation Script

When the valid information is obtained, our proposed method mapped intermediate script to the target software language. As shown in Algorithm 7, this step aims to associate the intermediate script with the specific smart home platform. In the experiment, this platform refers to the Home Assistant.

Algorithm 7 converts intermediate scripts to the executable scripts. Line 1 loads specific smart home platform information and conversion rules. Lines

**Algorithm 7** Convert from the intermediate scripts to executable scripts

```
Function: file = Translate (scenario, name)
Input:
    scenario A smart home scenario
    name The name of the smart home platform
Output:
    file the deployment file corresponding to this scenario in a certain smart home platform
1:  Loading the transfer rules of platform name
2:  Initializing an empty file $file$
3:  IF $scenario.condition$ is the time type
4:      Generate the corresponding time class time
5:      $file \leftarrow time$
6:  IF $scenario.condition$ is the state type
7:      Generate the corresponding state conditions state
8:      $file \leftarrow state$
9:  Combination trigger conditions to condition
10: $file \leftarrow condition$
11: FOR all commands $cmd \in scenario.commands$
12:     Get the id and function of the device in the smart home platform
13:     FOR all attributes $attr \in cmd.attributes$
14:         Get the $attr$ field and set the value
15:         $file \leftarrow (attr, value)$
16: return file
```

3 to 8 set the trigger conditions of the script according to the type. Lines 10 to 15 iterate through all commands in the scenario and set the properties of the commands.

The experiment assumed that the smart home system platform is Home Assistant, which has a web client, and take Chinese description "打开浴室的热水器然后将温度调到 45 度 (Turn on the bathroom water heater then set the temperature to 45°.)" as an example. It will eventually call service in the Home Assistant that reads "Turn on the water heater and set the temperature property to 45°", with the specific code in the home assistant as shown in Figure 7.

## 5.2 The Experiment

We first built a home scenario on Home Assistant, including family members, home areas, and smart home components. Xiao Ming and Xiao Hong are family members. The home areas are divided into nine areas, including the living room, balcony, corridor, bathroom, kitchen, bedroom 1, bedroom 2, corridor, storage room, and a smart door. It is equipped with a smart door lock, a

```
id: '1641177232676'
alias: time_trigger
description: " "
trigger:
  - platform: time
    at: '05:00:00'
condition: []
action:
  - service: water_heater.turn_on
  - service: water_heater.set_temperature
    target:
      entity_id: water_heater.hot
    data:
      temperature: 45
  - service: light.turn_on
    target:
      area_id: hallway
      entity_id: hallway_light
   - service: light.turn_on
     target:
       area_id: living_room
       entity_id: living_room_LED_light
mode: single
```

**Figure 7** The specific code in the home assistant.

camera, a door magnetic alarm, and audio. The living room is equipped with a smart TV, an air conditioner, an electric curtain, a light, a microphone, an infrared detector, a temperature, and a humidity sensor. The balcony has a light, an electric drying rack, an infrared detector, a temperature, and a humidity sensor. There are two lights and an infrared detector in the corridor. The bathroom has a washing machine, a lamp, a ventilation fan, a showerhead, a faucet, an infrared detector, and a temperature, humidity sensor. The kitchen has a gas switch, a refrigerator, a water heater, a microwave oven, an oven, a pressure cooker, a hood, a faucet, and a gas alarm. There is a light and a fan in the dining room. The smart devices in the master bedroom 1 are a bedside lamp 1, a lamp 1, an electric curtain, a desktop computer, an air conditioner, a temperature and a humidity sensor, and a closet. Master bedroom 2 has a bedside lamp 2, a lamp 2, an electric curtain, an air conditioner, a temperature and a humidity sensor, and a closet. The storage room has a light and an infrared detector.

All the components of the home include virtual and actual devices, while they are also smart devices that can be properly connected to the Home Assistant via the Internet. The first-order logic process for loading data into a Python data structure is shown in Appendix A.2. The configuration of the home environment devices in Home Assistant is shown in Appendix A.3.

Based on the above home scenarios, 30 baseline data of the daily use environment of the smart home were set up. To collect the descriptions of the users, we conducted a user study at Yunnan University and provided a document to describe the tasks. Thirty students participated in this study and 600 different descriptions were given for 30 baseline data, correspondingly. The experiment used all these data to evaluate our method.

The success of the experiment is determined by checking whether the correct YAML generates, whether it can convert to the expected script file and whether the generated script file can run correctly in Home Assistant. According to these criteria, in these 600 hundred data, there was an error in 38 data entries. We found that our method was able to express 93.66% of the experimental environment data. In Table 3, the first column gives the descriptions of tasks. The statement *n* is used here to simplify the table. All these data are presented in English and Chinese in Appendix A.1; the latter two columns "success" and the "tms" shows whether the task can be successfully expressed

**Table 3** Characteristics of the baseline data

| Description | Success | tms |
|---|---|---|
| Sentence 1 | YES | 160.00 |
| Sentence 2 | YES | 130.99 |
| Sentence 3 | YES | 142.99 |
| Sentence 4 | YES | 104.00 |
| Sentence 5 | YES | 107.00 |
| Sentence 6 | YES | 141.98 |
| Sentence 7 | YES | 145.99 |
| Sentence 8 | YES | 116.00 |
| Sentence 9 | YES | 133.99 |
| Sentence 10 | YES | 105.00 |
| Sentence 11 | YES | 101.99 |
| Sentence 12 | YES | 116.00 |
| Sentence 13 | YES | 102.99 |
| Sentence 14 | YES | 111.00 |
| Sentence 15 | YES | 117.00 |
| Sentence 16 | YES | 118.00 |
| Sentence 17 | YES | 127.99 |
| Sentence 18 | YES | 129.99 |
| Sentence 19 | YES | 144.00 |
| Sentence 20 | YES | 122.99 |

(*Continued.*)

**Table 3** Continued

| Description | Success | tms |
|---|---|---|
| Sentence 21 | YES | 132.99 |
| Sentence 22 | YES | 101.99 |
| Sentence 23 | YES | 106.00 |
| Sentence 24 | YES | 101.99 |
| Sentence 25 | YES | 106.99 |
| Sentence 26 | YES | 100.00 |
| Sentence 27 | YES | 114.00 |
| Sentence 28 | YES | 137.00 |
| Sentence 29 | YES | 106.99 |
| Sentence 30 | YES | 128.99 |

and the total running time of the program. The time unit is milliseconds. What is used in the experiment is the Chinese language text.

In the experiment, some data occurred in error for the following reasons. The method proposed in this paper is mainly based on syntactic analysis to process the text. In the collected data, the results cannot be extracted correctly when there are syntactic errors in the sentences or when the overall format of the instructions is not followed. The algorithm in this paper bundles the device instructions and attribute settings together and thus does not allow multiple devices to be set consecutively in a single phrase. There are some limitations of the algorithms. First, the instructions that exceed the functional scope of the home device are not executed successfully. In addition, specific dictionaries must be set up for phrases that cannot be handled correctly by syntactic and lexical analyses.

## 6 Conclusion

The smart home automation system is one of the IoT applications that facilitates the control of home appliances over the Internet using an automation system. The complexity of the devices in the home IoT has increased the needs of users, specifically for the combined tasks of the multiple scenarios, conditions, and devices used simultaneously. However, the existing Chinese smart home automation systems and some research only provide limited applications that can perform a single task that is mostly a time-consuming and costly process. Users need a lot of professional programming knowledge and manual work to develop a system according to their own needs. To solve these problems and fill in the existing gap in technological development, we have proposed a method that generates smart home automation scripts using Chinese texts. Our proposed method has two main features. Firstly, it provides a smart home automation system independent intermediate language specification so that the method can easily adapt it to different systems. Secondly, based on the specification, it generates the smart home script automatedly. The experimental result demonstrated that 93.66% of the 600 Chinese texts from user research in the experiment can successfully generate the executable scripts.

## Acknowledgement

This work has been supported by the Science and Technology Plan in Key Fields of Yunnan Province under grant No. 202202AD080002, and the Open Foundation of Key Laboratory in Software Engineering of Yunnan Province under grant No. 2020SE318.

## Ethics

The authors strictly declare that there are no safety, environmental or ethical issues associated with this paper.

## Appendices

### A.1 Experiment Baseline Data

In the experiment, we provide the 30 representative Chinese texts. The baseline data that users always need and their explanations are shown in Table 4.

Based on the data mentioned above, 30 volunteers provided several of the 600 relevant descriptions used in the experiment.

**Table 4** Baseline data and corresponding explanation in the experiment

| Baseline data in the experiment | | |
|---|---|---|
| 1 | command | 打开感应门，打开卧室的空调然后将温度调到 26 度，打开热水器并且将其温度调到 45 度，然后打开烟雾报警器，关闭防盗感应器。 |
| | explanation | Turn on the induction door, turn on the bedroom air conditioner and set the temperature to 26 degrees, turn on the water heater and set the temperature to 45 degrees, then turn on the smoke alarm and turn off the burglar sensor. |
| 2 | command | 打开客厅的窗帘，打开空调将温度调到 26 度，关闭报警器，播放音乐。 |
| | explanation | Turn on the curtain in the living room, turn on the air conditioner to set the temperature to 26 degrees, turn off the alarm and play music. |
| 3 | command | 打开客厅的电视，然后关闭客厅的窗帘，打开灯光将亮度调到 50 然后将灯光颜色设置为橙色。 |
| | explanation | Turn on the TV in the living room, then close the curtains in the living room, turn on the lights and set the brightness to 50 then set the light color to orange. |
| 4 | command | 打开客厅的话筒，同时播放音乐。 |
| | explanation | Turn on the microphone in the living room while playing music. |
| 5 | command | 开启湿度报警器，启动空调设置为除湿模式 |
| | explanation | Turn on the humidity alarm and start the air conditioner set to dehumidification mode. |
| 6 | command | 开启卧室温度检测器，开启空调设置制冷模式然后将温度维持在 26 度。 |
| | explanation | Turn on bedroom temperature detector, turn on the air conditioner to set the cooling mode then maintain the temperature at 26 degrees. |
| 7 | command | 关闭客厅温度检测器，开启风扇并调整转速到三档。 |
| | explanation | Turn off living room temperature detector, turn on the fan and adjust the speed to third gear. |
| 8 | command | 打开空调设置为暖风模式然后将温度维持到 20 度。 |
| | explanation | Turn on the air conditioner and set it to warm air mode then maintain the temperature to 20 degrees. |
| 9 | command | 关闭卫生间的洗衣机，然后打开卧室的床头灯，关闭卧室的窗帘，开启报警器。 |
| | explanation | Turn off the washing machine in the bathroom, then turn on the bedside lamp in the bedroom, close the curtains in the bedroom, and turn on the alarm. |
| 10 | command | 打开卫生间的灯，然后将报警器关闭。 |
| | explanation | Turn on the bathroom light and turn the alarm off. |
| 11 | command | 关闭灯光，打开换气扇。 |
| | explanation | Turn off the lights and turn on the ventilation fan. |
| 12 | command | 打开卫生间的红外感应门，打开卧室的窗帘。 |
| | explanation | Turn on the infrared sensor door in the bathroom and open the curtains in the bedroom. |
| 13 | command | 将感应门打开，关闭报警器。 |
| | explanation | Open the induction door and turn off the alarm. |
| 14 | command | 关闭客厅的感应门，然后发送报警信息。 |
| | explanation | Close the sensor door in the living room and send an alarm message. |
| 15 | command | 打开报警器，关闭所有灯光，将其他电器关闭。 |
| | explanation | Close the sensor door in the living room and send an alarm message. |
| 16 | command | 关闭换气扇同时关闭天然气，然后打开厨房的烟雾报警器。 |
| | explanation | Turn off the ventilation fan and turn off the natural gas, then turn on the smoke alarm in the kitchen. |
| 17 | command | 当天黑时，打开客厅的 LED 灯，然后开启防盗报警器。 |
| | explanation | When it is dark, turn on the LED lights in the living room and then turn on the burglar alarm. |
| 18 | command | 每当下午八点时，打开热水器并且将温度调到 45 度。 |
| | explanation | Turn on the water heater and set the temperature to 45 degrees at 8 p.m. |
| 19 | command | 当报警器触发时，发送报警信息。 |
| | explanation | Send an alarm message when the alarm is triggered. |
| 20 | command | 当过道的灯打开时，打开主卧的灯。 |
| | explanation | When the hallway light is on, turn on the master bedroom light. |
| 21 | command | 当我回家时，打开热水器并且将其温度调到 45 度，然后打开烟雾报警器。 |
| | explanation | When I got home, I turned on the water heater and set it to 45 degrees, then turned on the smoke alarm. |

(*Continued*)

**Table 4** Continued

| | | |
|---|---|---|
| 22 | command | 打开厨房水龙头，同时打开热水器。 |
| | explanation | Turn on the faucet in the kitchen and turn on the water heater at the same time. |
| 23 | command | 打开洗衣机，打开主卧衣柜然后关闭电动晾衣架。 |
| | explanation | Turn on the washing machine, turn on the closet in the master bedroom and turn off the electric drying rack. |
| 24 | command | 打开高压锅，并打开燃气报警器。 |
| | explanation | Turn on the pressure cooker and turn on the gas alarm. |
| 25 | command | 打开天然气，开启燃气报警器，然后打开油烟机 |
| | explanation | Turn on the natural gas, turn on the gas alarm, and then turn on the range hood. |
| 26 | command | 关闭智能门锁，打开摄像头。 |
| | explanation | Turn off the smart door lock and turn on the camera. |
| 27 | command | 打开饭厅的灯然后调整合适灯光亮度，接着开启饭厅的风扇。 |
| | explanation | Turn on the light in the dining room and adjust the brightness of the light, then turn on the fan in the dining room. |
| 28 | command | 关闭厨房门，同时关闭厨房烟雾报警器；打开灯光将亮度调到 50 然后将灯光颜色设置为橙色，打开换气扇。 |
| | explanation | Close the kitchen door and turn off the kitchen smoke alarm at the same time; turn on the light and adjust the brightness to 50, then set the light color to orange, and turn on the ventilation fan. |
| 29 | command | 打开卫生间的换气扇，同时打开卫生间的窗户。 |
| | explanation | Turn on the ventilation fan in the bathroom and open the bathroom window at the same time. |
| 30 | command | 打开客厅的智能电视并且调至第 5 频道，然后打开空调并且将温度调到 20 度，然后打开湿度报警器。 |
| | explanation | Turn on the smart TV in the living room and adjust to Channel 5, then turn on the air conditioner and adjust the temperature to 20 degrees, then turn on the humidity alarm. |

## A.2 First-order Logical Home Environment Data Loading

The home environment data is loaded as follows.

```
def define (Fi, Ru):
  v = """
     客厅 => livroom
     阳台 => bolcony
     走廊 => corridor
     浴室 => bath
     厨房 => kitchen
     饭厅 => carteen
      卧室 1 => bedr1
      卧室 2 => bedr2
      储藏室 =>  stor
      智能门 => smdoor
```

智能电视 => smatv
空调 => aircondition
客厅电动窗帘 => curliv
麦克风 => mic
客厅红外探测器 => livdet
客厅 LED 灯 => LEDligliv
客厅温度湿度传感器 => livtempwet

阳台灯 => bolig
电动晾衣架 => rack
阳台红外线探测器 => boldector
阳台温湿度传感器 => boltemwet

走廊灯 1 => corlig1
走廊灯 2 => corlig2
走廊红外探测器 => boldect
防盗报警器 =>alarm
洗衣机 => washer
浴室灯 => balig
换气扇 => venti
花洒 => shower
浴室水龙头 => bafaucet
浴室红外线探测器 => badector
浴室温湿度传感器 => batemwet
燃气开关 => gasswitch
热水器 => watheater
烟雾报警器 => smalarm

冰箱 => refrig
厨房门 => kicdoor
饭厅灯 => eatlig
饭厅风扇 => fan

床头灯 1 => bedlig1
台灯 1 => tablig1
电动窗帘 => bed1cur
台式电脑 => computer
卧室 1 空调 => bed1air
卧室 1 温湿度传感器 => bed1tempwet
衣柜 1 => closet1

床头灯 2 => bedlig2
台灯 2 => tablig2
电动窗帘 => bed2cur
卧室 2 空调 => bed2air
卧室 2 温湿度传感器 => bed2tempwet
衣柜 2=> closet2

储藏室灯 => savelig
储藏室红外线探测器 => savedect

智能门锁 => smlock
摄像头 => camera
门磁报警器 => dooralram
音箱 => audio

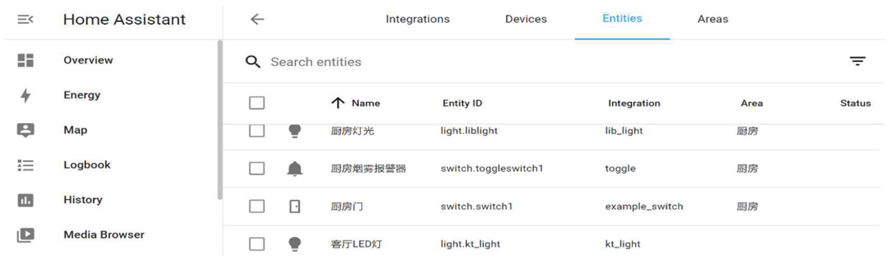
温度 => temp
亮度 => bright
湿度 => wet
打开 => on
关闭 => off
音量 => vol
颜色 => color
上升 => up
下降 => down
转速 => speed

has =>
{(livroom,LEDligliv), (livroom,smatv), (livroom,aircondition),
(livroom,curliv), (livroom,mic), (livroom,livdet),
(livroom,livtempwet),(corridor,corlig1), (corridor,corlig2),
(corridor,boldect), (kitchen,watheater),(kitchen,kicdoor),
(kitchen,smalarm), (kitchen,gasswitch), (kitchen,refrig),
(kitchen,oven), (kitchen,precooker), (kitchen,ranghood),
(kithchen,gaswith),(kitchen,kitfau), (kitchen,refrig),
(kitchen,fogalarm), (kitchen,kitdoor),(carteen,eatlig),
(carteen,fan), (bath,washer), (bath,balig), (bath,shower),
(bath, venti), (bath, bafaucet), (bath, badetor), (bath, batemwet),
(carteen,eatlig),(carteen,fan),(bedr1,bedlig1),(bedr1,tablig1),
(bedr1,bed1cur),(bedr1,computer),(bedr1,bed1tempwet),
(bedr1,closet1),(bedr1,bed1air),(bedr2,bedlig2),(bedr2,tablig2),
(bedr2,bed2cur), (bedr2,bed2air),(bedr2,bed2tempwet),(bedr2,closet2),

```
    (stor, savelig),(stor,savedect),(smdoor,smlock),
    (smdoor,camera), (smdoor,dooralarm),(smdoor,audio), (corridor,alarm)
}
attr =>
{
    (watheater,temp),(watheater,on),(LEDligliv,bright),(oven,temp)
    (LEDligliv,on),(LEDligliv,color),(ligcor,bright),(oven,off)
    (kicdoor,off),(smalarm,off),(venti,on),(ligcor,on),(oven,on)
    (smatv,on),(samtv,off),(aircondition,on),(aircondition,temp), (alarm,off)
    (aircondition,wet),(aircondition,off),(curliv,on),(curliv,off),(alarm,on)
    (mic,on),(mic,off),(mic,vol),(livdet,on),(livdet,off),(refrig,temp)
    (livtempwet,on), (livtempwet,off),(bolig,on),(bolig,off),(bolig,bright),
    (bolig,color), (rack,on),(rack,off), (rack,up),(rack,down),(boldector,on),
    (boldector,off),(boltemwet,on),(boltemwet,off),(corlig1,on), (corlig1,off),
    (corlig1,color),(corlig1,bright), (corlig2,on), (corlig2,off),(watheater,temp)
    (corlig2,color),(corlig2,bright), (boldect,on),(boldect,off),(smalarm,on),
    (smalarm,off),(gasswitch,on), (gasswitch,off),(refrig,on),(refrig,off),
    (precooker,on), (precooker,off), (precooker,temp), (ranghood,on),
    (ranghood,on), (ranghood,off),(kitfau,on),(kitfau,off),(kitfau,temp),
    (fogalarm,on),(fogalarm,off),(kitdoor,on),(kitdoor,off),(eating,on),
    (eatlig,off),(eatlig,bright),(eatlig,color),(fan,on),(fan,off),(fan,speed),
    (bedlig1,on),(bedlig1,off),(bedlig1,bright), (bedlig1,color),
    (bedlig2,on),(bedlig2,off),(bedlig2,bright), (bedlig2,color),
    (tablig1,on),(tablig1,off),(tablig1,bright), (tablig1,color),
    (tablig2,on),(tablig2,off),(tablig2,bright), (tablig2,color),
    (bed1cur,on),(bed1cur,off), (bed2cur,on),(bed2cur,off),
    (computer,on),(computer,off),(computer,bright),(bed1air,on)
    (bed1air,off), (bed1air,temp), (bed1air,wet), (bed2air,on)
    (bed2air,off), (bed2air,temp), (bed2air,wet),(bed1tempwet,on),
    (bed1tempwet,off), (bed2tempwet,on),(bed2tempwet,off),
    (closet1,on), (closet1,off), (closet2,on), (closet2,off),(smlock,on),
    (smlock,off),(camera,on),(camera,off),(camera,bright),(dooralarm,on),
    (dooralarm,off),(audio,on), (audio,off),(audio,vol),(savedect,on),
    (savedect,off),(savelig,on), (savelig,off), (savelig,color), (savelig,bright),
}
  """
  return v
```

## A.3 Home Devices Configuration

The configuration of the home environment devices that are part of the Home Assistant is shown in Figure 8 below.



| | Name | Entity ID | Integration | Area | Status |
|---|---|---|---|---|---|
| | 厨房灯光 | light.liblight | lib_light | 厨房 | |
| | 厨房烟雾报警器 | switch.toggleswitch1 | toggle | 厨房 | |
| | 厨房门 | switch.switch1 | example_switch | 厨房 | |
| | 客厅LED灯 | light.kt_light | kt_light | | |

**Figure 8** Devices in home assistant.

## References

[1] A. Cyril Jose and R. Malekian, "Smart Home Automation Security: A Literature Review," *Smart Comput. Rev.*, no. Rtdm, pp. 6–10, 2015, doi: 10.6029/smartcr.2015.04.004.

[2] C. Paul, A. Ganesh, C. Sunitha, "An overview of IoT based smart homes," *Proc. 2nd Int. Conf. Inven. Syst. Control. ICISC 2018*, no. Icisc, pp. 43–46, 2018, doi: 10.1109/ICISC.2018.8398858.

[3] M. Shahjalal, M. K. Hasan, M. M. Islam, M. M. Alam, M. F. Ahmed, Y. M. Jang, "An Overview of AI-Enabled Remote Smart- Home Monitoring System Using LoRa," *2020 Int. Conf. Artif. Intell. Inf. Commun. ICAIIC 2020*, 2020, pp. 510–513, doi: 10.1109/ICAIIC48513.2020.906 5199.

[4] A. Alhammadi, A. Alzaabi, B. Almarzooqi, S. Alneyadi, Z. Alhashmi, and M. Shatnawi, "Survey of IoT-Based Smart Home Approaches," *2019 Adv. Sci. Eng. Technol. Int. Conf. ASET 2019*, 2019, pp. 1–6, doi: 10.1109/ICASET.2019.8714572.

[5] J. A. Fadhil, K. Region, O. A. Omar, K. Region, Q. I. Sarhan, K. Region, "A Survey on the Applications of Smart Home Systems," 2020 International Conference on Computer Science and Software Engineering (CSASE), Duhok, Iraq, 2020, pp. 168–173, doi: 10.1109/CSASE48920 .2020.9142103.

[6] M. Asadullah and A. Raza, "An overview of home automation systems," *2016 2nd Int. Conf. Robot. Artif. Intell. ICRAI 2016*, 2016, pp. 27–31, doi: 10.1109/ICRAI.2016.7791223.

[7] K. Agarwal, A. Agarwal, G. Misra, “Review and performance analysis on wireless smart home and home automation using IoT,” *Proc. 3rd Int. Conf. I-SMAC IoT Soc. Mobile, Anal. Cloud, I-SMAC 2019*, 2019, pp. 629–633, doi: 10.1109/I-SMAC47947.2019.9032629.

[8] J. Jaihar, N. Lingayat, P. S. Vijaybhai, G. Venkatesh, K. P. Upla, “Smart home automation using machine learning algorithms,” *2020 Int. Conf. Emerg. Technol. INCET*, 2020, pp. 1–4, 2020, doi: 10.1109/INCET498 48.2020.9154007.

[9] M. Gamba, A. Gonella, C. E. Palazzi, “Design issues and solutions in a modern home automation system,” *2015 Int. Conf. Comput. Netw. Commun. ICNC 2015*, 2015, pp. 1111–1115, doi: 10.1109/ICCNC.2015 .7069505.

[10] C. J. Baby, F. A. Khan, J. N. Swathi, “Home automation using IoT and a chatbot using natural language processing,” *2017 Innov. Power Adv. Comput. Technol. i-PACT*, 2017, pp. 1–6, 2017, doi: 10.1109/IPACT.20 17.8245185.

[11] Y. Inayama and H. Hosobe, “Toward an efficient user interface for block-based visual programming,” *Proc. IEEE Symp. Vis. Lang. Human-Centric Comput. VL/HCC*, 2018, pp. 293–294, doi: 10.1109/VLHCC. 2018.8506530.

[12] T. Zeng, Y. Liu, X. Ma, X. Bao, J. Qiu, L. Zhan, “Auto-programming for numerical data based on remnant-standard-deviation-guided gene expression programming,” *2009 Fifth International Conference on Natural Computation*, Tianjian, China, 2009, pp. 124–128, doi: 10.1109/IC NC.2009.617.

[13] S. R. Swamy, K. S. Nandini Prasad, P. Tripathi, “Smart home lighting system,” *Proc. 2020 Int. Conf. Smart Innov. Des. Environ. Manag. Plan. Comput. ICSIDEMPC*, 2020, pp. 75–81, doi: 10.1109/ICSIDEMPC490 20.2020.9299585.

[14] C. Xie, H. Qi, L. Ma, J. Zhao, “DeepVisual: A visual programming tool for deep learning systems,” *IEEE Int. Conf. Progr. Compr.*, May 2019, pp. 130–134, doi: 10.1109/ICPC.2019.00028.

[15] H. Naito, T. Yokogawa, N. Igawa, S. Amasaki, H. Aman, K. Arimoto, “A node-style visual programming environment for the nuXmv model checker,” *2020 IEEE 9th Glob. Conf. Consum. Electron. GCCE*, 2020, pp. 71–75, doi: 10.1109/GCCE50665.2020.9291945.

[16] H. Kamada and K. Nishikawa, “The visual interactive programing learning system using image processing,” *2016 Third Int. Conf. Comput.*

*Meas. Control Sens. Netw.*, 2016, pp. 158–161, doi: 10.1109/CMCS N.2016.21.

[17] R. Anbunathan and A. Basu, "Automation framework for test script generation for android mobile," *Adv. Intell. Syst. Comput.*, vol. 731, pp. 571–584, 2019, doi: 10.1007/978-981-10-8848-3_55.

[18] H. Tanno and X. Zhang, "Test script generation based on design documents for web application testing," *Proc. Int. Comput. Softw. Appl. Conf.*, vol. 3, pp. 672–673, 2015, doi: 10.1109/COMPSAC.2015.74.

[19] S. Goel and R. Sharma, *Economic Analysis of Solar Water Pumping System for Irrigation*, in: Sharma, R., Mishra, M., Nayak, J., Naik, B., Pelusi, D. (eds) *Green Technology for Smart City and Society. Lecture Notes in Networks and Systems*, vol. 151, Singapore: Springer, https://doi.org/10.1007/978-981-15-8218-9_13.

[20] S. Oyucu, "Integration of cloud-based speech recognition system to the Internet of Things based smart home automation," *HORA 2021 – 3rd Int. Congr. Human-Computer Interact. Optim. Robot. Appl. Proc.*, 2021, pp. 30–32, doi: 10.1109/HORA52670.2021.9461360.

[21] A. Meliones and D. Giannakis, "Visual programming of an interactive smart home application using LabVIEW," *IEEE Int. Conf. Ind. Informatics*, 2013, pp. 655–660, doi: 10.1109/INDIN.2013.6622961.

[22] Z. Li, Y. Xiao, S. Liang, S. Wang, "Design of Smart home management system based on MQTT and FBP," *Proc. 2018 Chinese Autom. Congr. CAC 2018*, 2019, pp. 3086–3091, doi: 10.1109/CAC.2018.8623113.

[23] M. Campbell, "Automated coding:" *Computer*, Vol. 53, No. 2, pp. 2020–2022, 2020, doi: 10.1109/MC.2019.2957958.

[24] J. Yi, S. Fu, S. Cui, C. Zhao, "A deep contractive auto-encoding network for machinery fault diagnosis," *Isc. 2018 – 18th Int. Symp. Commun. Inf. Technol.*, 2018, pp. 85–89, doi: 10.1109/ISCIT.2018.8587983.

[25] X. Yang, H. Zhang, J. Cai, "Auto-encoding and distilling scene graphs for image captioning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8828, no. c, pp. 1–14, 2020, doi: 10.1109/TPAMI.2020.3042192.

[26] H. Li, Y.-P. Wang, T.-J. Mu, "Nerva: Automated application synthesis for humanoid robot from user natural language description," *Commun. Inf. Syst.*, vol. 17, no. 1, pp. 45–64, 2017, doi: 10.4310/cis.20 17.v17.n1.a3.

[27] V. Le, S. Gulwani, Z. Su, "SmartSynth," *Mobisys '13: Proceeding of the 11th Annual International Conference On Mobile Systems, Applications, And Services*, 2013, p. 193, doi: 10.1145/2462456.2464443.

[28] S. R. Joseph, H. Hloman, K. Letsholo, K. Sedimo, "Natural language processing: A review," *Int. J. Res. Eng. Appl. Sci.*, vol. 6, no. 3, pp. 1–8, 2016, available at: http://www.euroasiapub.org.

[29] S. J. Segalowitz and H. Chevalier, "Event-related potential (ERP) research in neurolinguistics: Part I. Techniques and applications to lexical access," *Handbook of Neurolinguistics*, Academic Press, 1998. https://doi.org/10.1016/B978-012666055-5/50009-5.

[30] D. Stringer, "Lexical semantics: Relativity and transfer," *Appl. Linguist. Teach. Cult. Linguist. Divers. Learn.*, pp. 180–203, 2019, doi: 10.4018/978-1-5225-8467-4.ch007.

[31] S. K. Joseph, "Natural language processing tutorial," *Tutorials Point Pvt. Ltd.*, pp. 1–13, 2019, available at: https://store.tutorialspoint.com.

[32] Y. Y. Hsu and H. Y. Kao, "Curatable named-entity recognition using semantic relations," *IEEE/ACM Trans. Comput. Biol. Bioinforma.*, vol. 12, no. 4, pp. 785–792, 2015, doi: 10.1109/TCBB.2014.2366770.

[33] A. Fern, "Lecture Notes: First-Order Logic: Syntax and Semantics Syntax of FO Logic," pp. 1–9, 2010, available at: https://web.engr.oregonstate.edu/~afern/classes/cs532/notes/fo-ss.pdf

[34] J. Laird, "A compositional cost model for the $\lambda$-calculus," *Proc. Symp. Log. Comput. Sci.*, vol. 2021-June, 2021, doi: 10.1109/LICS52264.2021.9470567.

[35] M. Biernacka, D. Biernacki, S. Lenglet, P. Polesiuk, D. Pous, A. Schmitt, "Fully abstract encodings of $\lambda$-calculus in HOcore through abstract machines," *Proc. Symp. Log. Comput. Sci.*, pp. 1–12, 2017, doi: 10.1109/LICS.2017.8005118.

[36] T. Lampert, "Minimizing disjunctive normal forms of pure first-order logic," *Log. J. IGPL*, vol. 25, no. 3, pp. 325–347, 2017, doi: 10.1093/jigpal/jzx003.

[37] A. Sernadas, "Fibring modal first-order logics: Completeness preservation," *Log. J. IGPL*, vol. 10, no. 4, pp. 413–451, 2002, doi: 10.1093/jigpal/10.4.413.

[38] C. Calvès and M. Fernández, "Matching and alpha-equivalence check for nominal terms," *J. Comput. Syst. Sci.*, vol. 76, no. 5, pp. 283–301, 2010, doi: 10.1016/j.jcss.2009.10.003.

[39] S. Guerrini, "Linear $\beta$-reduction," *Electron. Proc. Theor. Comput. Sci. EPTCS*, vol. 238, pp. 44–53, 2017, doi: 10.4204/EPTCS.238.5.

[40] P. H. Azevedo De Amorim, D. Kozen, R. Mardare, P. Panangaden, M. Roberts, "Universal semantics for the stochastic $\lambda$-calculus," *Proc.*

*Symp. Log. Comput. Sci.*, vol. 2021, June 2021, doi: 10.1109/LICS52264.2021.9470747.

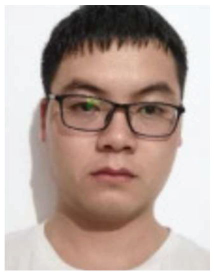
[41] S. Bird, E. Klein, E. Loper, "NLTK tutorial: Introduction to natural language processing," *English*, vol. 66, p. 22, 2005.

[42] V. Sinha, F. Doucet, C. Siska, R. Gupta, S. Liao, A. Ghosh, "YAML: A tool for hardware design visualization and capture," *Proc. Int. Symp. Syst. Synth.*, vol. January 2000, pp. 9–14, 2000, doi: 10.1109/ISSS.2000.874023.

## Biographies

**Jiayi Kuang** is a master student at the School of Software, Yunnan University, China. She received her B.Eng. degree from Zaozhuang University, China, in 2019. Her research interests include the Internet of Things and service computing.

**Gang Xue** received his B.Eng. degree from Wuhan Technical University of Surveying and Mapping in 2000. He received his M.Eng. and Ph.D. degrees from Yunnan University in 2006 and 2009, respectively. He is currently an associate professor at the School of Software, Yunnan University, China. His research interests include service computing, edge computing, and embedded systems.



**Zeming Yan** is a master student at the School of Software, Yunnan University, China. He received his B.Eng. degree from Jishou University, China, in 2019. His research interests include service computing and edge computing.

**Jing Liu** received the Ph.D. degree in computer application technology from the University of Electronic Science and Technology of China in 2003. From September 2003 to July 2005, he was with No. 30 Institute of China Electronics Technology Group Corporation as a postdoctoral fellow. From September 2005 to December 2012, he had been an assistant professor at Sun Yat-Sen University. Since January 2013, he has been an associate professor at Yunnan University. His current research interests include applied cryptography and network security.