Rule Conflict Prevention in Smart Homes through Hybrid Detection

Yaodong Zhang*
* School of Cyber Science and Technology, Beihang University,
Beijing 100191, China (e-mail: by2439213@buaa.edu.cn)

Abstract—hello world

Index Terms—Smart home, Rule Conflict, Conflict Detection, Conflict Resolution

1 Introduction

Smart home is an important application of the Internet of Things, and with the development of the Internet of Things, it has entered the field of more users. Smart homes include automation features that allow users to download or customize automation rules to make life easier and more comfortable. [1]

Automation rules typically consist of three parts: triggers, conditional judgments, and execution actions. For example, installing a temperature sensor and air conditioner in a room, and setting the automation rule ⟨ `indoor temperature is below 24 degrees Celsius ,air conditioner is off ,trun on the air conditioner and set the air conditioner temperature to 27 degrees Celsius` ⟩ means that when the indoor temperature is below 24 degrees Celsius, if the air conditioner is off, turn on the air conditioner and set the air conditioner temperature to 27 degrees Celsius to achieve the function of constant indoor temperature.

However, the execution of multiple automation rules may trigger unexpected operations for users, and the interaction of multiple rules may lead to rule conflicts, resulting in serious security threats. For example, Fig.1 shows two rules. Automation 1 states "When the smoke alarm is triggered, turn on the fire sprinkler," and Automation 2 states "When a water leak is detected, turn off the water valve." Executing the first automation rule may trigger the second rule, potentially creating a fire hazard.

There has been relevant research on the problem of rule conflicts in smart homes. Rule conflict detection methods are mainly divided into two categories: (1) defining some security policies, and if it is detected that the execution of the current automation rule violates the security policy, it is determined as a rule conflict; (2) judging according to the automation rule interaction mode, and if the rule interaction mode is satisfied, it means a hit, that is, there is a rule interaction threat.

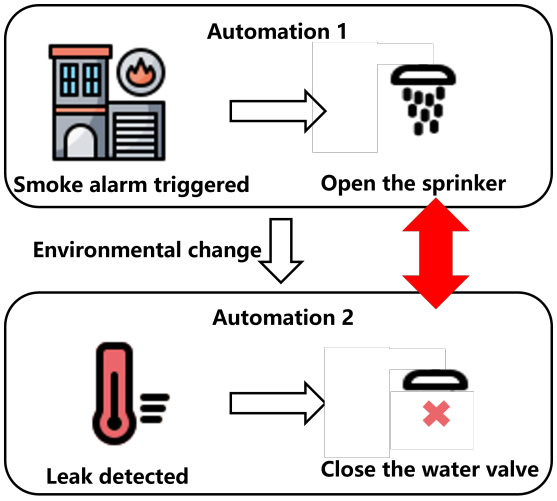

Fig. 1. Rule Conflict Example

The technologies for rule conflict detection mainly include three categories: (1) proxy analysis: analyzing rule files and reverse analysis to find possible rule conflicts; (2) model checking: detecting through model checking and other methods during the operation of automation rules; (3) graph analysis: generating a control flow graph for the application, and inferring whether a rule conflict occurs in the actual system according to the triggering, conditions, operations, and logical conditions of the detection control flow.

Rule conflict handling methods are mainly divided into the following three categories: (1) reconfiguring automation rules; (2) Set specific security policies, which ensures that the execution result does not violate the security policy when a rule conflict occurs; (3) customizing the handling method according to the rule conflict, which is essentially another predefined security policy, but provides multiple handling strategies for different rule conflicts to choose from.

The current rule conflict resolution scheme alleviates rule conflicts to some extent, but still has some defects.

In terms of rule conflict detection:

- Detecting whether smart home systems violate security policies to determine rule conflicts, but smart home systems are diverse, and security policies of

other systems cannot be directly applied to specific systems. Incomplete security policies indicate that this method cannot detect all possible rule conflicts. For example, one family sets "sprinkler valve opens when smoke is detected," but another family's rule sets "sprinkler valve opens after open flame detection alarm." The former family's rule may not apply to the latter family.

- Detecting whether there are rule interaction threats in smart home systems, i.e., whether a certain rule interaction pattern is satisfied. However, the boundary between rule interaction and rule conflict is blurred, which can easily lead to false positives. For example, there are two rules: "sunset, turn on the heating" and "when the temperature reaches 30 degrees Celsius, open the window and turn off the heating." These rules may be beneficial in an indoor garden, but the same rules applied to the bedroom of a first-floor resident will bring serious security risks.

In terms of rule conflict resolution:

- Modifying rule configuration may cause the rules to malfunction; modifying existing automation rules does not guarantee that no new conflicts will be introduced, and it cannot truly solve the problem.
- Using a general security strategy protects home safety to some extent, but the same security strategy does not apply to every household.

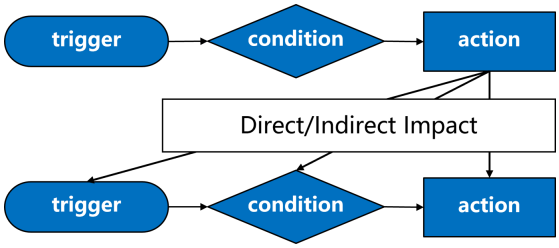

Fig. 2. Rule Interaction Pattern

For the automation rule interaction in smart home systems, the work of this paper is mainly based on the following observations: (1) As shown in Fig.2, the interaction between automation rules can be summarized into six types according to the following methods: the execution result of one rule has a direct or indirect impact on the trigger, condition, and execution of another rule; (2) Whether a rule interaction belongs to a rule conflict is subjective and depends on personal user preferences; (3) The smart home system uses the TCA model to implement the trigger judgment and execution of automation rules, and it is possible to consider using code instrumentation technology between systems to read the current and historical executed rule information, so as to obtain rule conflict judgment information and rule conflict processing.

This paper classifies rule interaction/conflict types based on the analysis of rule interaction patterns. To account for rule interactions/conflicts caused by indirect influences, it remodels rules by incorporating user environment configurations. Then, it uses formal analysis for model checking to detect all possible rule interactions. By combining user security configurations for entities, it enables basic rule conflict detection and recommends conflict resolution strategies, which users can also adjust according to their preferences. The smart home system executes automation rules based on the TCA model. When rule conflicts occur, it can selectively execute the two rules, thereby obtaining customized handling strategies for each pair of conflicting rules, rather than fixed, general security policies. Code instrumentation techniques can help acquire the system's device states and historical automation rule execution information, enabling real-time system detection and conflict resolution.

In summary, this paper makes the following contributions:

- Analysis of the rule execution mechanism and classification of rule conflicts: Classify rule conflicts according to the automated rule execution mechanism to cover all rule interaction types.
- Design and implementation of a solution combining static and dynamic conflict detection to achieve a widely applicable conflict detection mechanism suitable for smart home rules: Add region attributes to the rules and remodel them, combine with security entity configuration for static detection of automated rule conflicts, and use assertion detection methods to detect whether rule conflicts occur in the system in real-time operation.
- Design and implementation of conflict resolution solutions, customized conflict rule handling strategies, and implementation of conflict prevention measures: Implement various conflict handling strategies according to the automated rule execution mechanism, automatically select appropriate handling strategies for rule conflicts based on security entity configuration, and intercept rule conflicts before they occur to prevent them from happening.

2 Motivated Example

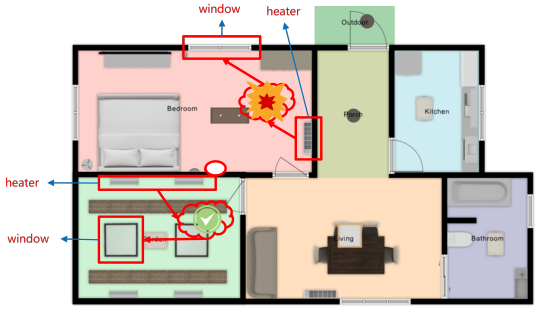

Fig. 3. Motivated Example

In this section, specific examples will be used to demonstrate some properties of rule conflicts, as well as the shortcomings of current rule conflict detection and mitigation methods. Fig. 3 shows a smart home floor plan, including seven areas: Outdoor (top center), porch (center), living room (bottom center), kitchen (bottom right), batch room (bottom right), bedroom (center left), and green house (garden, bottom left).

In the greenhouse, automation rule 1 is set with the trigger condition being `sunset`, and the action is `turn on the heater`. Simultaneously, automation rule 2 is set with the trigger condition being `temperature reaches thirty degrees Celsius`, and the action is `open the windows and turn off the heater`. As shown in the bottom left corner of Fig.3, when the sun sets in the greenhouse, rule 1 is automatically executed first, and the heater is turned on to continuously provide heat to the greenhouse. The temperature may continue to rise above 30 degrees Celsius. At this time, the execution result of automation rule 1 affects the indoor temperature, thereby triggering the execution of automation rule 2, which turns off the heater and opens the skylight. Opening the skylight in the greenhouse is often in line with user expectations and is safe. However, if these two rules are set in the user's bedroom, the situation will be significantly different: rule 1 triggers the execution of rule 2 by raising the bedroom temperature, opening the windows of the bedroom. The windows of the bedroom are often not skylights, and opening the bedroom windows unexpectedly can introduce security risks to the smart home system, and attackers may even maliciously use the rules to create execution paths to control security-sensitive devices such as windows and door locks.

The above examples show that rule interactions may not be directly influenced by the execution action of one rule on another (e.g., two rules executing simultaneously, one controlling the air conditioner to heat mode and the other to cool mode), but may also be generated through indirect channels (commonly temperature, humidity, brightness, sound, etc.). When detecting rule conflicts, it is necessary to consider rule conflicts originating from other channels. At the same time, it can be observed that rule conflicts have regionality and user subjectivity. The above rule interaction may not be regarded as a rule conflict if it occurs in a greenhouse, but it may be regarded as a rule conflict if it occurs in a bedroom. User subjectivity is reflected in the fact that for a pair of rule interactions, some users may regard it as a rule conflict, while others may regard it as a normal rule interaction (for example, the dehumidification mode of the bedroom air conditioner and the humidifier work alternately to maintain the air humidity at a suitable level, some users think it is a normal interaction, while others may think it is a waste of resources). In addition, the above examples also show that the detection of side channels in smart home systems should not be limited to temperature, humidity and other parameters, but should also have regional attributes. For example, the porch lights may affect the brightness of the kitchen and living room, but will not affect the brightness of the bedroom. If two brightness-related rules are executed at the same time, they may not interact with each other.

It requires a lot of effort for users to set all security policies using specific security strategies to detect potential risks, because different families need relevant professionals to set all security policies due to differences in housing types, equipment, etc., and security policies are likely to contain all rule conflicts. If the judgment is made based on the automatic rule interaction mode, the rule conflict that occurs in the bedroom can indeed be detected, but the rule interaction in the green house will also be regarded as a rule conflict. When the number of set automatic rules increases, a large number of false positives are likely to occur and need to be confirmed by users one by one, which will also lead to a large user's effort.

3 Design

3.1 Overview

A complete system is designed that detects rule conflicts hidden in rule interactions by combining static and dynamic approaches, and automatically executes customized conflict resolution strategies before conflicts occur to prevent actual conflicts. According to Fig.4, our method mainly includes two steps: 1) Static analysis; 2) Dynamic detection and conflict resolution.

The first step includes a rule modeling module, a formal analysis module, a static rule conflict detection module, and a resolution strategy generation module. The rules in the smart home system will be remodeled to achieve differentiation of environment zones. Afterwards, formal analysis will be based on the new rule model to extract all types of rule interactions along with corresponding rule interaction analysis data. The entity security configuration will be applied to the rule conflict detection module to extract rule conflicts from the set of rule interactions. Considering that the definition of whether a rule interaction constitutes a rule conflict is relatively subjective, system provides a user configuration interface through which the rule interaction analysis data extracted by formal analysis module will be displayed to the user in natural language to assist the user in determining rule conflicts. Finally, for each element in the formed set of rule conflicts, the resolution strategy generation module will generate a customized rule conflict resolution strategy.

In the second step, a rule event listener module, an assertion verification module, and a command generation module are included. In addition, the code instrumentation module in the smart home system for example Home Assistant will assist in dynamic rule conflict detection and prevention. The rule event listener module will monitor rule execution information in real time. When a rule is triggered, before and after condition checking, it will send the relevant information to the assertion checking module. The assertion check module will then perform assertion checks based on historical rule execution events, the information of the currently executing rule, and related device status information, to determine whether a rule conflict has actually occurred. If so, the command generation module will generate execution commands based on the current rule conflict information about to occur and the rule conflict resolution strategy from static detection, which will be enforced by the code instrumentation module.

3.2 Static Phase

3.2.1 Rule Modeling

Before formally introducing the static step, it is necessary to the classification of rule interactions and rule conflicts for this approach. Based on previous observations Fig.2, we summarize the interactions among rules using the following method: the execution result of one rule can have direct

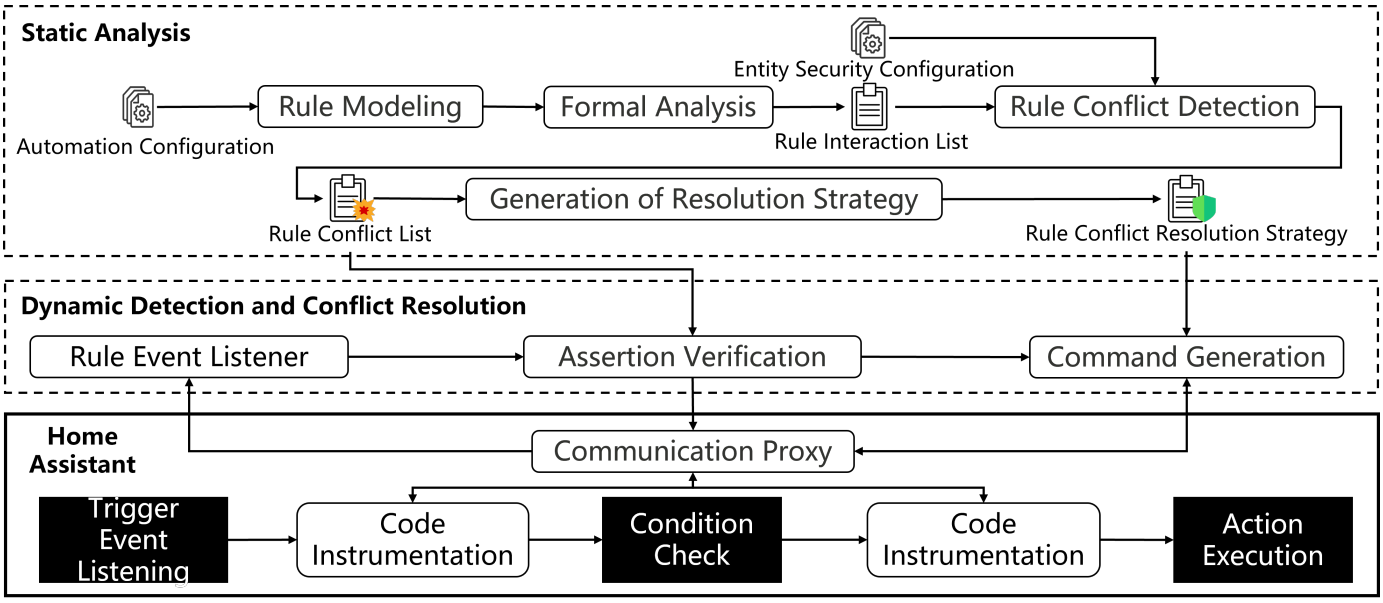

Fig. 4. Overall Architecture of Our Approach

and indirect impacts on the triggers, conditions, and actions of another rule. Thus, we can classify rule interactions as follows: Trigger Interaction, Condition Interaction, Action Interaction, Indirect Trigger Interaction, Indirect Condition Interaction, and Indirect Action Interaction. At the same time, six categories of rule conflicts can be identified: Trigger Conflict, Condition Conflict, Action Conflict, Indirect Trigger Conflict, Indirect Condition Conflict, and Indirect Action Conflict.

In smart home systems, rule configuration is usually stored as static files, from which the triggers, conditions, and action attributes of a rule can be extracted, and quickly modeled as a $\langle T, C, A\rangle$ model. However, the above data is insufficient to support the detection of the effects of side channels in different home areas within smart home systems. For example, rule interactions may be caused by different rules affecting the bedroom temperature, or by different rules turning on high-power devices, leading to an increase in the household's power consumption and subsequently triggering related rules to conserve electricity. Therefore, a new attribute E needs to be introduced. E is defined as$E = [e^1, e^2, \dots]$ and $e^i = (area, channel, trend)$, where $area$ indicates the spatial characteristics of the channel, such as the kitchen, living room, or the whole home; $channel$ indicates the name of the other channel which rules may impact each other, such as temperature, humidity, illuminance and so on; and $trend$ indicates the direction of the influence of the other channel, such as an increase and decrease. For the trigger/condition of a rule, if e is $\langle kitchen, temperature, increases\rangle$, it means that if the execution results of other rules may lead to an increase in kitchen temperature, then the trigger/condition of the current rule is more likely to be met. For the action execution of a rule, if e is $\langle kitchen, temperature, increases\rangle$, it means that the execution of the current rule may promote an increase in kitchen temperature. The attribute E will be determined based on the user's actual home situation and the related rules. For example, if there are no rules concerning "humidity" in a home, then there is no need to set the $channel$ attribute of e as "humidity". Thus, a rule can be re-modeled in the following form: $R = \langle T, C, A, E\rangle$ or $R = \langle T, C, A, E_T, E_C, E_A\rangle$ which called TCAE model.

3.2.2 Formal Analysis

According to the TCA model for rule conflict classification, let rule $R_1 = \langle T_1, C_1, A_1, E_{T_1}, E_{C_1}, E_{A_1}\rangle$ and rule $R_2 = \langle T_2, C_2, A_2, E_{T_2}, E_{C_2}, E_{A_2}\rangle$ be to represent two automation rules set in the same system, where T, C, A, and E respectively represent the corresponding attribute. This paper uses $\rightarrow$ to indicate that a trigger is activated or a condition is met, and $\nrightarrow$ to indicate that a condition is prohibited.$\perp$ denotes that the two channel attributes have the same area and channel but opposite trends (for example, rising kitchen temperature versus falling kitchen temperature), or it indicates that two actions are in conflict (such as turning on the air conditioner versus turning it off), while "=" indicates that the two channel attributes have the same area, channel, and trend, or that there are two identical triggers or conditions.

The formal analysis module will traverse each rule model and then expression validation. If an expression is satisfied, it indicates that the two rules meet the corresponding rule interaction classification. Rule conflict is a special case of rule interaction, and that expression can only filter out interacting rules. further detection is required to determine whether a rule interaction constitutes a rule conflict.The formal analysis module can analyze and obtain a specific list of rule interactions, which includes all rule interactions and interaction pattern information.

3.2.3 Rule Conflict Detection and Resolution Strategy Generation

Entity safety configuration is a set of safety states for a security-sensitive device entity, used to indicate the preferred state to maintain when a conflict occurs in a smart

TABLE 1
Formal Analysis Expression

Classification	Expression
Trigger Interaction	$(\exists a_1 \in A_1) \wedge (a_1 \rightarrow T_2)$
Condition Interaction	$((A_1 \nrightarrow C_2) \wedge ((T_1 \neq T_2) \wedge (R_1 \neq R_2))) \vee ((A_1 \rightarrow C_2) \wedge ((T_1 \neq T_2) \wedge (R_1 \neq R_2)))$
Action Interaction	$(\exists a_1 \in A_1) \wedge (\exists a_2 \in A_2) \wedge (a_1 \perp a_2)$
Indirect Trigger Interaction	$(\exists a_1 \in A_1) \wedge \left(E_{a_1} = E_{T_1}\right)$
Indirect Condition Interaction	$\left(\left(E_{A_1} \perp E_{C_2}\right) \vee \left(E_{A_1} = E_{C_2}\right)\right) \wedge (R_1 \neq R_2)$
Indirect Action Interaction	$(\exists a_1 \in A_1) \wedge (\exists a_2 \in A_2) \wedge (D_{a_1} \neq D_{a_2}) \wedge (E_{a_1} \perp E_{a_2})$

home system—for example, a door should be "closed" and a fire sprinkler should be "on." The rule conflict detection module performs an entity safety on all elements in the rule interaction list to determine whether a rule interaction violates the entity safety state, potentially causing a rule conflict.

The specific inspection method is as follows: a rule interaction includes two rules, and attention should be paid to whether the entity safety configuration requirements are violated after this rule interaction is completed. Therefore, each rule sets a safety value parameter, sf. If the execution result of a rule is more inclined to meet the entity safety configuration, the sf value is higher; otherwise, it is lower. The specific sf value can be calculated using a linear function by traversing all the entity safety configurations. For example, if the execution result of a rule is closing a door, which is more in line with the entity safety state "closed" for the entity "door", then the rule's sf is increased by one from the original value, Otherwise, it is decreased by one, with the default value being zero.

For different rule interactions, the method for determining rule conflicts varies.

- For Trigger Interaction and Indirect Trigger Interaction, the focus is on whether the safety value sf of the second rule is greater than zero; if it is less than zero, then it is determined to be a rule conflict.
- For Condition Interaction and Indirect Condition Interaction, if the first rule in a rule interaction prohibits the condition of the second rule while the second rule's sf is greater than zero, it is determined to be a rule conflict; or if the first rule satisfies the condition of the second rule while the second rule's sf is less than zero, it is determined to be a rule conflict. This indicates that due to the influence of the preceding rule, either the rule that is more inclined to the entity safety state was not executed for the latter rule, or the rule that contradicts the entity safety state was executed.
- For Action Interaction and Indirect Action Interaction, if either of the two rules has a non-zero safety value, it is determined to be a rule conflict. In this case, it should be noted that if both rules have safety values greater than zero, it will also be deemed a rule conflict, because in the determination of this type of rule interaction, the execution results of the two rules are contradictory, and typically only one rule needs to be executed.

Based on this, multiple resolution strategies can be set for each rule. The specific conflict resolution strategy can be selected according to Table.2.

3.3 Dynamic Phase

When the smart home system is running, rule events will be monitored in real-time. When a rule is triggered, assertion detection will be performed before and after the condition checking phase to determine whether a rule conflict is imminent. All information during the assertion detection process will be collected through the code instrumentation module, including the information of the currently executing rule, the information of the previously executed rule, the real-time status of related entities, and past state change information.

Assertion detection mainly includes the following functions:

- $obs()$: Observing the occurrence of an event (including the triggering of the trigger $obs(T)$,the passing of a condition check $obs(C)$, the failing of a condition check $obs(\neg C)$ and the execution of an action $obs(A)$
- $intime(X, Y, \delta)$: the time interval between events X and Y is less than δ, with δ defaulting to 0.1 seconds

Accordingly, for different types of rule conflicts, there are corresponding assertion detection methods, as shown in Table.3.

When the assertion detection indicates that a rule conflict corresponding to a certain type is indeed imminent, the system will immediately execute its conflict resolution strategy. The specific implementation is as follows: the system reviews the conflict resolution strategy associated with the rule conflict, then generates specific execution commands and transmit them over to the code instrumentation module. The code instrumentation module will cancel the original rule-triggered execution logic (that is, triggering, condition checking, and action) and forcefully execute the corresponding commands.

4 Evaluation

In this chapter, we mainly evaluate from the following perspectives: 1) The effectiveness of static detection of rule conflicts; 2) The viability of automated conflict resolution; 3) The performance of the system.

4.1 Smart Home Testbeds

This paper implements a complete system and conducts a virtual test. Home Assistant was chosen as the smart home platform for virtual test. It is an open-source smart home system based on Python that supports devices from

TABLE 2
Resolution Policy Decision (Need modification)

Classification	Decision	Options
Trigger Conflict and **Indirect Trigger Conflict**	$(sf_2 = 0) \vee (sf_2 > 0 \wedge sf_1 \geq 0)$	Not rule conflict
	$sf_1 < 0 \wedge sf_2 \geq 0$	Only execute R_2
	$sf_2 < 0$	Cancel execution of R_2
Condition Conflict and **Indirect Condition Conflict**	$(A_1 \rightarrow C_2 \wedge sf_2 > 0) \vee (A_1 \nrightarrow C_2 \wedge sf_2 < 0)$	Not rule conflict
	$A_1 \rightarrow C_2 \wedge sf_2 > 0$	Do not execute R_2
	$A_1 \nrightarrow C_2 \wedge sf_2 < 0$	Execute R_2
Action Conflict and **Indirect Action Conflict**	$sf_1 = 0 \wedge sf_2 = 0$	Not rule conflict
	$sf_1 > 0 \wedge sf_2 < 0$	Only execute R_1
	$sf_1 < 0 \wedge sf_2 > 0$	Only execute R_2
	$sf_1 > 0 \wedge sf_2 > 0 \wedge sf_1 > sf_2$	Both rules are executed, but ended with R_1
	$sf_1 > 0 \wedge sf_2 > 0 \wedge sf_1 < sf_2$	Both rules are executed, but ended with R_2
	$sf_1 < 0 \wedge sf_2 < 0$	Neither rule will be executed

TABLE 3
Assertion Verification Expression

Classification		Expression
Trigger Conflict		$obs(T_1), obs(C_1), obs(A_1)$ $obs(T_2), obs(C_2)$ $intime(A_1, T_2, delta)$
Condition Conflict	Make Conditions Forbidden	$obs(C_2)$ $obs(T_1), obs(C_1), obs(A_1)$ $obs(\neg C_2)$ $intime(A_1, \neg C_2, delta)$
	Make Conditions Satisfied	$obs(\neg C_2)$ $obs(T_1), obs(C_1), obs(A_1)$ $obs(C_2)$ $intime(A_1, C_2, delta)$
Action Conflict		$obs(T_1), obs(C_1), obs(A_1)$ $obs(T_2), obs(C_2)$
Indirect Trigger Conflict		$obs(T_1), obs(C_1), obs(A_1)$ $obs(T_2), obs(C_2)$
Indirect Condition Conflict	Make Conditions Forbidden	$obs(C_2)$ $obs(T_1), obs(C_1), obs(A_1)$ $obs(\neg C_2)$ $intime(A_1, \neg C_2, delta)$
	Make Conditions Satisfied	$obs(\neg C_2)$ $obs(T_1), obs(C_1), obs(A_1)$ $obs(C_2)$ $intime(A_1, C_2, delta)$
Indirect Action Conflict		$obs(T_1), obs(C_1), obs(A_1)$ $obs(T_2), obs(C_2)$

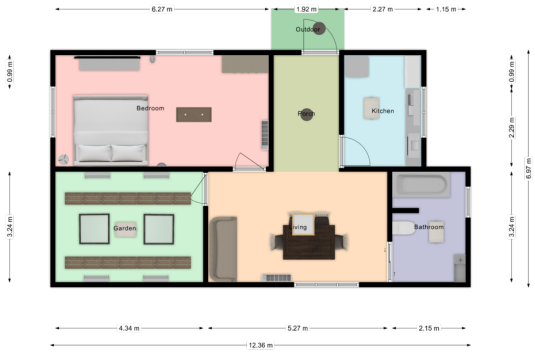

Fig. 5. Smart Home Floor Plan

different manufacturers and provides virtual devices to enable automation functions. On this platform, users can create custom automation rules through a visualization webpage or by writing configuration files. Home Assistant uses YAML configuration files to describe automation rules and adopts a trigger-condition-action (TCA) model to describe automation rules. Virtual tests use the smart home floor plan shown in Fig. 5. In the virtual environment of the current tested smart home system, there are a total of 7 areas: Outdoor, porch, living room, kitchen, bath room, bedroom, and greenhouse (garden). Based on observations, some channels in certain areas can affect each other, such as the brightness attributes of the porch and living room, while the brightness attributes of the living room and bedroom do not affect each other. Table.6 displays 17 automation rules configured in these 7 areas, along with the corresponding zones and devices. Table.4 shows the side channels configured in the smart home system, with each being composed of a channel, area, and trend. Table.5 outlines the entity security configurations in the smart home system, which include the state settings for three security-sensitive devices: valve on, sprinkler on, and bedroom window closed.

TABLE 4
Side Channel

Area	Channel	Trend
outdoor	brightness,	increase/decrease
porch	brightness,	increase/decrease
kitchen	brightness,humidity,temperature	increase/decrease
living room	brightness,humidity,temperature	increase/decrease
bedroom	brightness,temperature	increase/decrease
greenhouse	brightness,temperature	increase/decrease

TABLE 5
Entity Security Configuration

Entity Security	Status
valve	open
sprinkler	open
window (bedroom)	close

4.2 Effectiveness of Conflict Detection

In order to evaluate the effectiveness of rule conflict detection, all possible rule interactions were tested using manual triggering, and four rule conflicts were identified. Subsequently, the system implemented in this paper was used for both static and dynamic detection (also through manual triggering), while also comparing the detection results with those of IoTMEDIATOR.

The conflict results of 17 rules in all seven regions are shown in Table.7. Among them, there is a conflict between R10 and R11: R10 controls the water valve to close, causing R11 to fail execution when triggered. There is also a conflict between R11 and R10: when R11 is executed, it triggers the execution of R10, and the execution actions of R11 and R10 are mutually exclusive. Additionally, there is a conflict

TABLE 6
Testbeds

Rule ID	Content	Area	Devices
R1	When the door lock is opened, if the brightness sensor is below 50lux, turn on the outdoor light and porch light.	Outdoor	Smart door lock, light, porch light, brightness sensor
R2	When the door lock is closed, turn off the porch light.	Outdoor	Smart door lock, light, brightness sensor
R3	When motion is detected in the porch, if the brightness sensor is below 50lux, turn on the porch light.	Porch	Motion sensor, light, brightness sensor
R4	When motion is detected in the living room, if the brightness sensor is below 50lux, turn on the living room light.	Living Room	Motion sensor, light, brightness sensor
R5	When the living room humidity is below 40%, and the humidifier is off, turn on the living room humidifier.	Living Room	Humidity sensor, humidifier
R6	When the living room humidity is above 60%, turn off the living room humidifier.	Living Room	Humidity sensor, humidifier
R7	When the living room temperature is below 24°C, turn off the living room floor heating and turn on the underfloor heating.	Living Room	Themperature sensor, Underfloor heating
R8	When the living room temperature is above 30°C, turn off the living room air conditioner and set it to 27°C.	Living Room	Temperature sensor, air conditioner
R9	When motion is detected in the kitchen, if the brightness sensor is below 50lux, turn on the kitchen light.	Kitchen	Motion sensor, light, brightness sensor
R10	When a water leak is detected, close the main water valve.	Kitchen	Water leak sensor, water valve
R11	When the smoke detector is triggered, turn on the sprinkler.	Kitchen	Smoke detector, sprinkler
R12	When motion is detected in the bedroom, if the brightness sensor is below 50lux, turn on the bedroom light.	Bedroom	Motion sensor, light, brightness sensor
R13	At 7 PM, turn on the bedroom heating.	Bedroom	Heating
R14	When the bedroom temperature reaches 30°C, open the window and close the heating.	Bedroom	Temperature sensor, window, heating
R15	When motion is detected in the green house, if the brightness sensor is below 50lux, turn on the green house light.	Green house	Motion sensor, green house light, brightness sensor
R16	At 7 PM, turn on the green house heating.	Green house	Heating
R17	When the green house temperature reaches 32°C, open the window and close the heating.	Green house	Temperature sensor, window, heating

TABLE 7
Result of Rule Conflict Detection

	Classification	R1-R2	R1-R3	R1-R4	R1-R9	R2-R3	R2-R4	R2-R12	R3-R4	R3-R9	R4-R3	R5-R6	R6-R5	R7-R8	R8-R7	R9-R3	R10-R11	R11-R10	R13-R14	R14-R13	R15-R12	R16-R17
Ours	Trigger Conflict																					
	Condition Conflict																					
	Action Conflict																		✓	✓		
	Indirect Trigger Conflict																	✓	✓			
	Indirect Condition Conflict																					
	Indirect Action Conflict																✓	✓				
IoTMediator	Condition Enabling/Disabling		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			✓					✓	
	Race Condition																					
	Potential Race Condition	✓				✓						✓										✓
	Chained Execution											✓	✓	✓	✓			✓	✓			✓
	Action Revert											✓	✓						✓			✓
	Infinite Loop											✓	✓	✓	✓							
	Condition Bypass																					

between R13 and R14: the execution of R13 triggers the execution of R14, and the execution actions of both rules are mutually exclusive. There is also a conflict between R14 and R13: the execution result of R14 is mutually exclusive with that of R13.

The test results indicate that IoTMEDIATOR is able to detect the rule conflicts of R11-R10, R13 with R14, and R14-R13, but it fails to detect the rule conflict caused by the mutual influence through humidity between R10 and R11. Specifically, after the smart home system shuts off the water valve upon detecting a water leak, if the user does not respond in time and a fire is triggered, the water valve will not open, resulting in R11 not executing normally. Meanwhile, IoTMEDIATOR persists with many other rule interactions; however, whether they constitute rule conflicts is subject to the user's subjective preference. For example, R5 controls the living room humidifier to increase humidity, while R6 controls the humidifier to shut off in order to prevent excessive humidity. The interaction between these two rules conveniently maintains the living room's humidity and should not be regarded as a conflict.

All rule interactions can be precisely captured by our system, and it is capable of automatically converting any threatening rule interactions into rule conflicts, thereby detecting all four types of rule conflicts and identifying six possible ways in which conflicts might occur. In addition, our system effectively avoids false positives in identifying rule conflicts, greatly minimizing the chance that normal rule interactions are mistakenly classified as conflicts, while also allowing users to optionally include other rule interactions in the rule conflict list according to their personal preference.

4.3 Availability of Automated Rule Conflict Resolution

In order to assess the applicability of automated rule conflict resolution, this chapter will use specific examples to demonstrate how the system automatically processes a particular rule conflict and how AI speculatively handles the rule conflict information provided by the system.

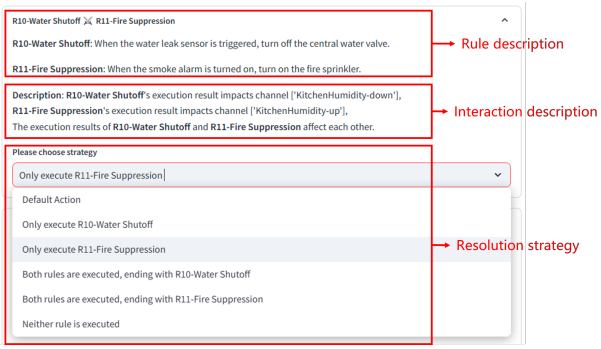

Fig. 6. Example of Automated Rule Conflict Resolution

Fig.6 displays the Indirect Action Conflict corresponding to rules R10 and R. The image shows the description of the rules (extracted from the automated rule configuration file) and the rule conflict description (generated by the Rule Conflict Detection Module based on contextual information) as: R10-Water Shutoff's execution result impacts channel ['KitchenHumidity-down'], R11-Fire Suppression's execution result impacts channel ['KitchenHumidity-up'], and the execution results of R10-Water Shutoff and R11-Fire Suppression affect each other. At the bottom is the resolution strategy, which consists of a default execution action and five rule conflict mitigation actions.

According to the entity security configuration, the execution action of rule R10 will close the water valve, and its evaluation is conducted using a linear function with the corresponding safety parameter $sf_{R10} = -1$, while the execution action of rule R11 will activate the fire sprinkler head, with its corresponding safety parameter $sf_{R11} = 1$.

Therefore, the Generation of Resolution Strategy Module will automatically recommend the "Only execute R11" strategy to prevent real conflicts between rules. In addition, the page provides options for resolution strategies, allowing users to choose a rule conflict resolution strategy that better fits their preferences.

To test the user-friendliness of the rule conflict description generated according to the rule model by the Rule Conflict Detection Module, this paper uses AI to test its strategy selection for conflict resolution. Still taking Fig.6 as an example, the model used is `gpt-4o-2024-11-20`, called via the OPENAI API, with other parameters remaining unchanged by default, and the system prompt will be shown in Appendix.A. For the above example, we obtained the following response: {

```
"policy": "2", // Only execute R11
"reason": "In this case of Indirect Rule
Action Conflict, the sprinkler system
activated by R11 is crucial for safety
during a fire, and its priority should
supersede the water cutoff triggered by
R10 to ensure effective fire suppression.
Therefore, only R11 should execute."
```

}

Through AI's understanding of rule conflict descriptions, the recommended results for rule conflict resolution strategies align with expectations, which to some extent indicates that the currently extracted rule conflict information can be well understood by users, enabling them to make correct and effective choices, and providing them with the option to use intelligent tools for recommending rule conflict resolution strategies.

4.4 Performance

In order to test the system performance, this paper divides the testing into performance evaluation of static analysis and performance evaluation of dynamic detection and conflict based on the system architecture.

In the performance evaluation of static analysis(Contains Formal Analysis Module and Rule Conflict Detection Module), the static analysis traverses all the automation rules and compares each pair of rules, including self-comparison. Therefore, for n rules, the time complexity of static detection is $O\left(n^2\right)$. For testing, the Testbeds rule set (17 rules), an AI-generated 100-rule set (100 rules), and a 1000-rule set (1000 rules) were used as inputs to evaluate the detection time multiple times, in seconds. To facilitate the display, three analysis results were selected and the average value was recorded. Table.8 shows the test results. It can be observed that the static detection of the 17 rules in Testbeds only takes $3.5ms$. Even with 1000 rules, static analysis only takes about $12s$, and the static analysis time of different data volumes meets the expected complexity.

In the two modules—the Assertion Detection Module for detecting rule conflicts dynamically and the Command Generation Module for handling rule conflict processing—the computational load is not high, and the time consumption mainly lies in the communication process. Therefore, the focus will next be on code instrumentation for communication performance evaluation, testing which includes randomly triggering rule conflicts and assessing the performance of all communication functions (covering the three stages of system sending messages, Home Assistant platform processing and returning messages, and system receiving messages). The introduction and performance evaluation results of the communication functions are shown in Table.9.

TABLE 8
Performance of Static Analysis

Number of Rules	Average Time Consumption
10	$1.337 \times 10^{-3}s$
17	$3.503 \times 10^{-3}s$
50	$2.907 \times 10^{-2}s$
100	$1.181 \times 10^{-1}s$
500	$2.991s$
1000	$1.195 \times 10^{1}s$

It can be observed that the average and maximum time consumption of the communication functions in the experimental environment are only at the millisecond level, and that only a limited number of calls are made to the communication functions during the two steps of assertion verification and rule conflict handling directive generation, with each call also taking only a few milliseconds.

5 Limitations and Discussion

6 Related Work

7 Conclusion

References

[1] someone, "This is an example," in *Network and Distributed System Security Symposium (NDSS)*, 2018.

Appendix A
System Prompt

```
You are a smart home expert who can handle
conflicts in automation rules in smart home
life. Rule conflicts are divided into the
following types: ['Rule Action Conflict',
'Rule Trigger Conflict', 'Rule Condition
Conflict', 'Indirect Rule Trigger Conflict',
'Indirect Rule Condition Conflict',
'Indirect Rule Action Conflict'];
There are several types of rule
conflicts:'Rule Action Conflict': 'The
execution actions of two rules conflict',
'Rule Trigger Conflict': 'The execution of
rule i causes the rule to be triggered',
'Rule Condition Conflict': 'The execution of
rule i prohibits or allows the conditions of
rule j', 'Indirect Rule Trigger Conflict':
'The impact of rule i on the other channel
unexpectedly causes rule j to be triggered',
'Indirect Rule Condition Conflict': 'The
impact of the execution of rule i on the
other channel causes the conditions of rule
j to be allowed or disabled', 'Indirect Rule
Action Conflict': 'Two rules interact with
```

TABLE 9
Performance of Communication Functions

Function Name	Description	Average Time Consumption	Maximum Time Consumption
get_entity_state	Get entity state	1.193×10^{-3}s	1.953×10^{-3}s
time_now	Get HomeAssistant system time	9.765×10^{-4}s	9.825×10^{-4}s
command_send	Send conflict handling command	9.643×10^{-4}s	9.787×10^{-4}s
communicate_finish	End communication phase	9.778×10^{-4}s	9.832×10^{-4}s

Note: Each function is tested 10 times.

two different devices that affect the same channel property';
For Rule Action Conflict, when multiple rules set different values for the same device at the same time, the device status will be updated according to the order in which the rules are executed. Pay attention to the index order starting from 0, `R1` represents the first rule in the rule pair, often indicating the rule that is executed first in the rule pair. `R2` represents the second rule in the rule pair, often indicating the rule that is executed later in the rule pair. `R1` and `R2` respectively refer to the chat rules given by the user in sequence.
For conflicts of type Rule Action Conflict and Indirect Rule Action Conflict, the conflict resolution strategies and their indexes are as follows: 'Default Action': 0, 'Only execute R1': 1, 'Only execute R2': 2, 'Both rules are executed, ending with R1': 3, 'Both rules are executed, ending with R2': 4, 'Neither rule is executed': 5;
For conflicts of type Rule Trigger Conflict and Indirect Rule Trigger Conflict, the conflict resolution strategies and their indexes are as follows: 'Default Action': 0, 'Only execute R2': 1, 'Default execution R2': 2, 'Cancel execution of R2': 3;
For conflicts of type Rule Condition Conflict and Indirect Rule Condition Conflict, the conflict resolution strategies and their indexes are as folloThe format of the data provided by the user is as follows:
``` Information about the first rule, including the rule name and id, rule content <trigger | condition | action>, and rule description
Information about the second rule, including the rule name and id, rule content <trigger | condition | action>, and rule description
Description of rule violation
``` The rules are given in the order of triggering execution, that is, the rules given first are executed first. The time interval between the execution of two rules may be very short, triggered almost at the same time, or it may be a long time, please consider the reasons for rule conflict and select an appropriate conflict resolution strategy based on the type of rule conflict and actual situation, thinking from the user's perspective while addressing resources and protecting the safety of the family, and give indicators and rationale for the appropriate resolution strategy. For example, to prevent unintended opening of doors and windows due to rule conflicts/rule interactions, and to water valves from being shut off during a fire.
Your response should be directly in the following JSON structure, not in markdown format, without any extra explanation and code block:
{
'policy': "index of policy",
'reason': "reason why you choose this policy"
}