Available online at www.sciencedirect.com
ScienceDirect
Science of Computer Programming 67 (2007) 125–161
Science of Computer Programming
www.elsevier.com/locate/scico

Using semi-formal methods for detecting interactions among smart homes policies

Mohamed Shehata[a,c,*], Armin Eberlein[b], Abraham Fapojuwo[a]

[a] *Department of Electrical and Computer Engineering, University of Calgary, 2500 University Drive NW, Calgary, AB, Canada*
[b] *Department of Computer Engineering, American University of Sharjah, PO Box 26666, Sharjah, United Arab Emirates*
[c] *Department of Electrical and Computer Engineering, Shoubra Faculty of Engineering, Benha University, 108 Shoubra Street, Cairo, Egypt*

Received 2 October 2005; received in revised form 29 June 2006; accepted 27 November 2006
Available online 15 March 2007

Abstract

Feature Interaction is a problem mostly considered in the telecommunications domain. Many solutions for detecting interactions between telephony features have been reported. In this paper, we investigate the feature interaction problem beyond the traditional telecommunications domain and look at interactions between policies in other domains. We propose the use of semi-formal methods for detecting interactions between policies in the smart homes domain. The novelty of this research is threefold: firstly, a six step semi-formal approach, called IRIS (Identifying Requirements Interactions using Semi-formal methods), for detecting interactions is presented. A major component within IRIS, which is an interaction taxonomy, is also presented. Secondly, we extend the scope of the problem of feature interactions beyond telecommunication features and investigate interactions between policies in the smart homes domain. Thirdly, in order to show how IRIS is used to detect interactions between policies, a case study of the smart homes domain is conducted. A complete description of the results obtained is also provided. Our approach was successfully applied to the smart homes domain and was able to discover 83 interactions among 35 user policies using only 525 pairwise comparisons as opposed to 630 a human expert would have to do. These results support the paper's main claim of being able to use the semi-formal approach IRIS to detect interactions between policies. Furthermore, these results are to date the most complete publication of interactions between policies in the smart homes domain.
© 2007 Elsevier B.V. All rights reserved.

Keywords: Feature interaction; Policies; Semi-formal methods; Smart homes

1. Introduction

A common approach to system development is to build an initial system with a relatively small and basic core and then introduce additional features later on as needed. This type of development is often referred to as feature-based development. However, the addition of features is often very challenging due to the feature interaction problem [1, 2]. This problem has received a lot of attention from the telecommunications industry where many formal approaches have been developed [3–5]. Unfortunately, the development of formal models is very costly, and the suitability of

* Corresponding author at: Department of Electrical and Computer Engineering, University of Calgary, 2500 University Drive NW, Calgary, AB, Canada. Tel.: +1 403 2108196; fax: +1 403 2826855.
E-mail addresses: Msshehat@ucalgary.ca (M. Shehata), Eberlein@ucalgary.ca (A. Eberlein), Fapojuwo@ucalgary.ca (A. Fapojuwo).

formal methods for other domains, different from telecommunications, has been questioned [6,7]. A good summary of the current research status of feature interaction in telecommunications and software systems can be found in the proceedings of the feature interaction workshops [8–14].

Although feature-based systems are very attractive, many users find that they do not allow enough system customization. Furthermore, features are developed by system developers, not by users, and hence users have to use a feature as is and have very little control over its behavior.

In recent years, interest in policies has increased as a means to provide greater flexibility and customizability of systems. For example, a person might set a policy regarding incoming phone calls such that important calls from his/her boss are forwarded to the cell phone, private calls from the family to the spouse's number, and the remaining calls to another predefined number. A good summary of the current research status of the application of policies in distributed systems can be found in the proceedings of policy workshops [15–18].

However, due to the expected large number of policies within a system, their diversity, and the complexity that the human user will add, there is a high chance that policies will interact with each other, in a similar way as features do. Such interactions have been termed policy interactions. There is thus a need for approaches that help detect and resolve such interactions.

This paper proposes the use of a semi-formal approach, IRIS, to detect policy interactions in different domains. It also describes an interaction taxonomy that provides guidelines on when two policies are interacting. The paper then shows the application of IRIS to detect policy interactions in the smart homes domain. This domain was chosen because it extensively uses policies to express user preferences. A complete description of the obtained results is provided.

The obtained results from the smart homes case study show that the use of IRIS as a semi-formal approach was successful in detecting interactions between policies in smart homes. Further, these results serve as the first fully documented results of interactions between user policies in the smart homes domain.

This paper is structured as follows: Section 2 presents the concepts of features and policies and their relationship. Section 3 begins with a step-by-step description of the proposed IRIS approach and then followed by a presentation of the proposed domain-independent interaction taxonomy. Section 4 describes the use of IRIS to detect interactions between smart homes policies. Section 5 presents a prototype of a tool that supports the application of IRIS. Section 6 presents description of some related work to this paper. Finally Section 7 concludes the paper.

2. Features and policies

During the Feature Interaction Workshop (FIW VII) held in 2003 and the 8th International Conference on Feature Interactions (ICFI'05) held in 2005, it became obvious that there is a growing interest in policies and their interactions. However, the differences between policies and features as well as their interrelationship were still very unclear. In this section, a novel view on the relationship between features and policies is presented.

2.1. Understanding features and policies

A feature is defined as a coherent and identifiable bundle of system functionality that helps characterize the system from the user perspective [25]. Features are built by system developers as user-requested expansions of a base system. Features have been attractive as they allow the developers of long-lived systems to enrich system performance by adding features over time on top of the base system. An example of a feature in the telecommunications domain is Call Forward on Busy Line (CFBL). CFBL is a feature that, when active, will forward an incoming phone call to a busy subscriber to a predefined phone number.

A Policy is defined as information that is used to modify the behavior of the system [20]. Policies are created by different stakeholders (e.g., normal user, administrator, manager) to reflect personal, organizational or system goals. The attractiveness of policies stems from the fact that people can express their own preferences by setting their own policies to customize the system with greater flexibility. An example of a policy set by a user is: "If someone gets out of bed between 10 pm and 7 am then the lights in the bedroom and the hall switch on at initially 50% of illumination ramping up to 100% over 1 min and the bathroom fan is switched on. After leaving the bathroom, the bathroom light and the bathroom fan automatically switch off. After the person gets back into bed the bedroom light is dimmed from 100% to 50% over 1 min and then switched off".

However, there is a major and key difference between features and policies. The user has very limited control, if any, over the behavior of a feature. He can activate or deactivate the feature or supply a certain value for a parameter part of the feature. But s/he cannot customize the feature to work in a certain way to meet his/her needs. For example, consider the feature Teen Line (TL) from the telecommunications domain which restricts outgoing calls from the phone during a predefined time period unless a PIN is provided. The only control that a user has over this feature is to activate/deactivate it, specify the active time frame, and change the PIN. But the user cannot customize this feature to allow outgoing calls in case of emergencies, such as fire, or to allow any occupant to call 911. However, such customization is possible with policies. For instance, the user can set the policy: "The system shall override the Teen Line PIN restriction when the fire alarm is triggered".

2.2. Relationship between features and policies

As defined earlier, a feature is a bundle of system functionality. This means that each feature provides different functionalities to the system. For example the windows control feature is a feature that allows the control of windows within a smart home and contains the following functionalities:

- O_{w1}: The windows can be opened/closed at any time by the occupants using a remote control.
- O_{w2}: The system shall open/close the windows between time X1 and time X2.
- O_{w3}: The system shall open/close the windows when day/night begins

where O_{wi}: the operation i associated with feature w (windows).

Now, a policy is information that is used to allow the modification of system behavior. Policies achieve this modification and customization of system behavior through the invocation of one or more functionalities within one or more features. For example, consider the following policy set by an occupant in a smart home: "Open the windows between 5:00 pm and 6:30 pm". This policy will invoke only one functionality, O_{w2}, in the windows control feature and will execute the action of opening the windows when the clock of the system indicates 5:00 pm and closes them again at 6:30 pm.

The relationship between features and policies can be described from an object oriented perspective: A policy is a specific *Run* that the user wants the system to execute to exhibit a specific behavior using values that accommodate his/her special needs. Now, the system is composed of a set of features. Each feature can be thought of as an *Object*. Also, each feature will have different functionalities in it (e.g., the windows control feature). These functionalities can be considered as *Methods*.

A user can then set a *run* with specific values that invokes one or more *methods* from the same or different *objects* replacing the parameters in these methods with the values provided by the user in the run. A similar concept describes the relationship between features and policies. The user can set a policy (*a run*) with specific values to replace variables in the functionalities. This policy will invoke one or more functionalities (*methods*) from the same or different features (*objects*) replacing the variables within these functionalities with the values provided by the user in the policy to achieve its task. Fig. 1 shows a diagrammatic scheme to illustrate this point.

2.3. Features and policies in a smart home architecture

After defining features, policies, and their relationship, we now want to describe how features and policies look like in a smart home architecture. Fig. 2 presents an overall architecture showing policies, features and physical elements of smart homes. Physical elements are responsible for carrying out the physical actions of the different functionalities when triggered (e.g. actuators, appliances, air conditioning, heating, light bulbs…etc.).

The policy layer contains all the policies of the smart home including policies set by the occupants (user policy) or policies that are set by the system administrator and developer (system policy). The feature layer includes all the features within the smart home. Finally, the physical elements layer contains all the physical elements that are connected to the smart home network. Usually, all physical elements are connected to a central network where the master control software coordinates all the operations of the physical elements.

When a user has a certain preference for the behavior of one or more physical elements he defines a user policy in the policy layer describing his preference. This user policy then invokes the features controlling the behavior of the affected physical elements. The invoked features pass on the user preferences described in the user policy to the master

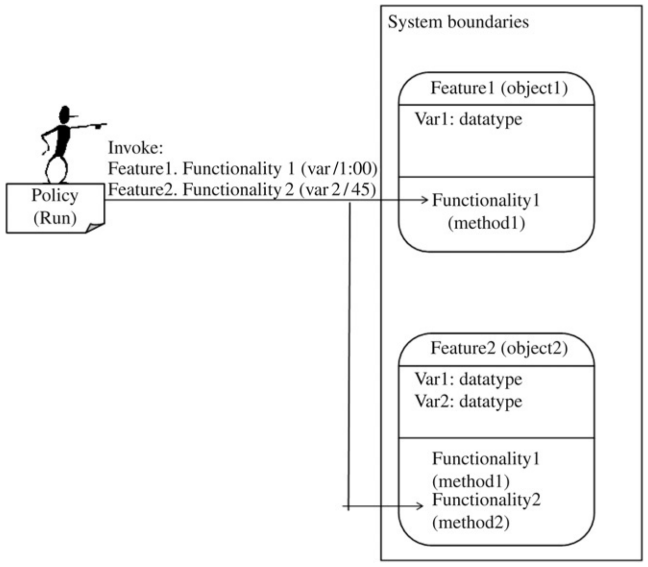


Fig. 1. Object oriented description of the relationship between features and policies.

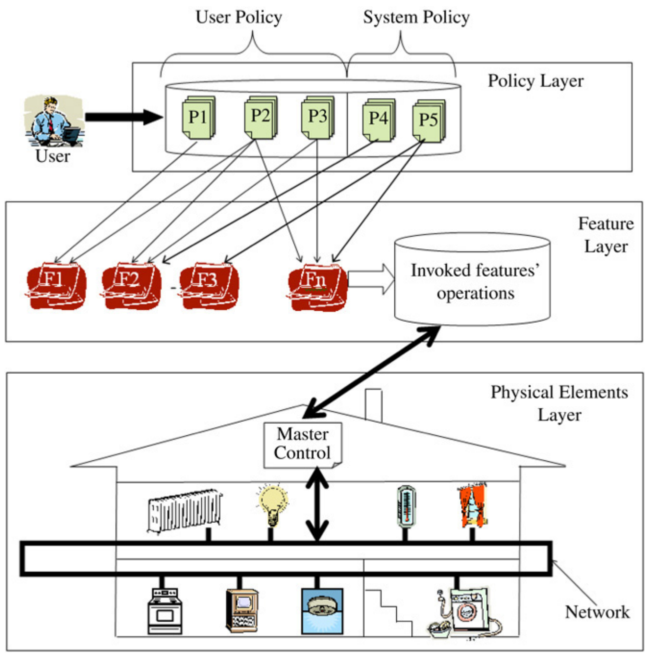


Fig. 2. Features, policies, and physical elements within smart homes.

control in the physical elements layer. Finally, the master control activates the required physical elements according to the user-defined behavior.

3. IRIS: Identifying Requirements Interactions using Semi-formal methods

This section describes the advantages of using IRIS as a semi-formal approach for detecting interactions. It then describes the internal structure of IRIS. Finally, it focuses on general interaction detection taxonomy which is an important component of IRIS.

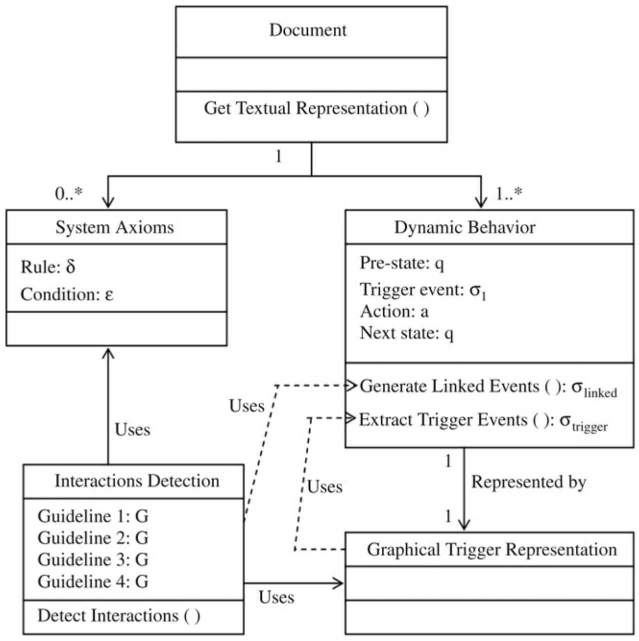


Fig. 3. Internal structure of IRIS.

3.1. *Internal structure of IRIS*

IRIS is the outcome of research into the application of lightweight semi-formal approaches to detect interactions in any domain in a cost effective manner. To validate IRIS, three case studies have been conducted. In the first case study, IRIS was applied to detect interactions in the lift control system. The results of this study are reported in [26]. In the second case study, IRIS was applied to the telecommunications domain (see [27] for the results of this case study.). One objective of this paper is to present the ability of IRIS to detect interactions between a set of policies in the new domain of smart homes. Another feature of IRIS is its flexibility in being applied at three levels of abstraction: requirements, features, and policies. In the lift control system, IRIS was used to detect interactions between a set of requirements. In the telecommunications domain, IRIS was used to detect interactions between a set of features. This paper shows the application of IRIS to policies.

IRIS is a systematic six step procedure that produces tables and graphs in the first five steps and then detects interactions in the last step using the tables and graphs created in the previous steps. The six step procedure of IRIS can be represented using the model shown in Fig. 3.

The "Interaction Detection" contains guidelines to detect policy interactions. The "Document" represents a textual document containing the system description and a textual definition of the different policies. The textual policies need to be translated into a tabular representation as well as an event-based graphical representation so that the guidelines for detecting interactions can be applied. The six steps of IRIS are ordered in such a manner that this translation of policies into graphical and tabular representations is gradually achieved. The objective of these representations is to facilitate the application of the interaction detection guidelines.

The following describes the six steps of IRIS:

- Step 1: Policies classification: The policies are classified into one of the following two categories: System Axiom policies and Dynamic Behavior policies. A policy is considered to be a system axiom policy if it describes a property that has to be preserved at all times. For example a system axiom might state "The system shall maintain the temperature of the water from the hot water tap at 45 °C". On the other hand, a policy is considered as a dynamic behavior policy if this policy specifies the reaction of the system when a certain event occurs. For example, a dynamic behavior might be "The system shall start the alarm clock and open the bedroom curtains at 7:00 am every morning". This step is shown in Fig. 3 as the classification of the Document into system axioms and dynamic behavior.

- Step 2: Policies attributes identification: This step is aimed at identifying different attributes within the system policies. To represent these attributes in a comprehensive manner, two tables are generated:
    - The first table is used to represent system axiom policies using the attributes Rules and Conditions (more attributes can be added when needed)
    - The second table is used to represent system dynamic behavior policies using the four attributes Pre-state, Trigger event, Action, and Next state (more attributes can be added if needed).

  This step is done by identifying the different attributes of the system policies and recording them in the "system axiom" and "dynamic behavior" shown in Fig. 3.
- Step 3: Trigger events extraction: This step identifies and extracts all the different trigger events from the dynamic system behavior policies. The step is shown as the process Generate Trigger Events() in the "dynamic behavior" in Fig. 3 and applies only to the dynamic behavior policies.
- Step 4: Linked events identification: During this step a table is developed that contains all linked trigger events. A trigger event is called a linked trigger event if it can lead to other trigger events (see Section 4.4.4). This step is shown as the process Generate Linked Events() in the "dynamic behavior" in Fig. 3 and applies only to the dynamic behavior policies.
- Step 5: Graphical representation: In this step an event-based graphical representation is used to link each event with the policies it triggers. This graphical representation will later facilitate the detection of interactions between the policies. The step is represented by the "Graphical Trigger Representation" in Fig. 3 and applies only to the dynamic behavior policies.
- Step 6: Interactions detection: During this step, the developer detects interactions between policies using the IRIS interaction detection guidelines (see Section 3.3). This step is shown in Fig. 3 by the "interaction detection" and applies to all policies, including system axiom policies and dynamic behavior policies, as it is required to detect interactions between any two policies regardless of their classification.

3.2. Interaction taxonomy

The actual detection of interaction takes place during the last step of IRIS. A human developer uses the tables and graphs generated during the earlier steps to detect interactions using specific detection guidelines. Although these tables and graphs make the policies clearer and more understandable, detailed guidelines on when two policies interact are necessary. These guidelines are designed for use by non-experts thus reducing the interaction detection cost. These guidelines are domain independent, which means that they can be applied in any domain including the smart homes domain selected as case study in this paper. They are applicable for detecting interactions between requirements, features, or policies. As this paper focuses on interaction detection between policies therefore the term policies will be used during the presentation of the taxonomy.

In order to develop interaction detection guidelines, a general interaction taxonomy had first to be established. Therefore, an extensive review of currently existing definitions of interactions was carried out. Based on this review, a three layer taxonomy was developed as shown in Fig. 4. The first layer of the taxonomy contains the main types of interactions. In the second layer of the taxonomy these main types are decomposed into subcategories. The third layer of the taxonomy associates each subcategory of layer 2 with one or more types which in turn map in the fourth layer of the taxonomy to an interaction scenario that describe an interaction situation. Each of these scenarios generated a guideline on when two policies will interact.

The first layer contains nine main types of interactions from which three are considered as main types of interactions and we use and describe in this paper:

1. Interactions between two system axiom policies.
2. Interactions between a dynamic behavior policy and a system axiom policy.
3. Interactions between two dynamic behavior policies.

From these three main types of interactions, more subcategories are derived in the second layer. Finally in the third layer, detailed scenarios for each of these subcategories are presented.

Four guidelines from the taxonomy were used in the smart home case study. The following contains a description of the guidelines used in the smart homes case study:

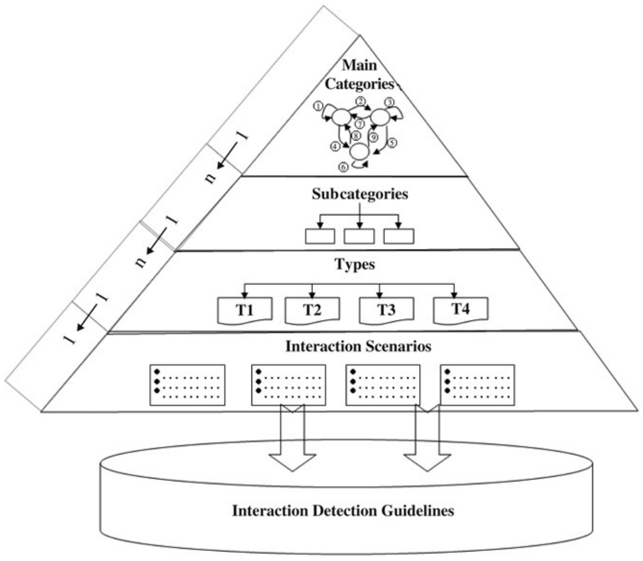


Fig. 4. General interaction taxonomy.

1. Interaction between two system axioms policies: Guideline 1 states that "*Two system axiom policies interact when the rule attribute (see* Table 1 *in Section* 4.4.2 *for an example) of one system axiom policy contradicts the rule attribute of a second system axiom policy*". For example, assume the following scenario: A system axiom policy states "Web pages that are used to submit confidential information will always require secure logon and secure communication with the web server" Another system axiom policy states "The transition time from any page to the transactions page should be minimal and should at all times be <10 s". If the security of the logon to the transaction page is done using a username/password, then these two system axiom policies are interacting.
2. Interaction between a dynamic behavior policy and a system axiom policy: Guideline 2 states that "*a system axiom policy interacts with a dynamic behavior policy when the action attribute of the dynamic behavior policy (see* Table 2 *in Section* 4.4.2 *for an example) contradict the rule attribute of the system axiom policy (see* Table 1 *in Section* 4.4.2 *for an example)*". For example: A dynamic behavior policy states "When the lift is overloaded, doors will not close". Also consider the following system axiom policy "The lift shall always serve unserved calls". Obviously the first dynamic policy interacts with the system axiom policy leading to an interaction.
3. Interaction between two dynamic behavior policies
   3.1 Guideline 3 states that "*two dynamic behavior policies interact if:*
      1. *The Previous State attributes (see* Table 2 *in Section* 4.4.2*) of both policies are the same, AND*
      2. *The Trigger Event attributes (see* Table 2 *in Section* 4.4.2*) of both policies are the same, AND*
      3. *The Action attributes (see* Table 2 *in Section* 4.4.2*) of both policies contradict each other"*.

      To illustrate this interaction assume the following scenario: one dynamic policy requires the system to activate a remote access module if there is an incoming call with no answer for 6 rings while the second dynamic policy requires the system to activate an answer machine if there is an incoming call with no answer for 6 rings. Obviously the system will not be able to do both actions at the same time leading to an interaction.

   3.2 Guideline 4 states that "*two dynamic policies will interact when they are triggered by linked events (see* Table 4*) and the action attribute of one policy will cancel or contradict the action attribute of the other policy (see* Table 2*)*". For example, assume the following scenario: one policy lets a CD player play music for three hours starting 7:00 pm and the other policy shuts down all audio/video devices after 9:00 pm. These two policies might have been set by two different people who live in the same house (e.g., a parent and child or a husband and wife). Obviously the second policy will cancel the action of the first policy before its completion.

In all three guidelines the word "contradict" is of significance. It can mean one of the following:

1. Negative impact: This means that the two attributes are still preserved, but one attribute will negatively affect the second one reducing its performance and efficiency.
2. Override: This means that the one attribute will override and cancel the other attribute.

The smart homes case study presented in this paper shows the application of IRIS and its guidelines for detecting interactions between policies.

3.3. *Advantages of detecting interactions using the proposed semi-formal approach IRIS*

Semi-formal approaches are well recognized in different research communities. For example, the requirements engineering community extensively uses the Unified Modeling Language (UML) [28] to model the requirements of a system. Different UML notations (e.g., static structure diagrams, use case diagrams, sequence diagrams, activity diagrams) are used to represent the set of requirements at hand. Another example is the User Requirements Notation (URN) which uses graphical representations to represent system requirements [29]. The system modeling community uses Structured Analysis and Structured Design (SASD) as a modeling approach that heavily relies on representing and modeling the system with different graphical notations [30,31].

The proposed IRIS approach is not just another graphical modeling approach but it rather models the system using different graphs and tables with the ultimate aim of detecting interactions between a set of policies:

1. Tables allow a comprehensive representation and visualization of the policies of the system in a structured format.
2. The creation of these tables requires a good understanding of the policies forcing the developer to clearly think about policies which will likely improve them.
3. Using a graphical representation (see Section 4.4.5) makes detection of interactions easier. For instance, policies that are triggered by the same event are grouped together in the graphical representation. Therefore detecting interactions can be easily done by examining the different actions and states of the policies.
4. Instead of having to perform pairwise comparisons between all policies in textual form, the developer can focus on interactions between system axiom policies, between a system axiom and a dynamic behavior policy, or between dynamic behavior policies that are triggered by the same or by linked events. This can result in a drastic reduction in the number of comparisons. One might argue that these savings will result in likely missed interactions. This is, to some extent, true; however, the created interaction detection guidelines will help the user to detect most of the critical interactions. Moreover, it must be noted that there is a tradeoff between cost and efficiency of interaction detection. Examples of domain where IRIS is recommended might be commercial PC software, smart homes or other non-critical systems. However, formal methods are recommended for life critical systems.

4. Case study: Detecting policy interactions in smart homes

4.1. *Smart homes*

Smart homes are residences that use networking technology to integrate various home automation systems. This integration allows systems (e.g., house appliances) to communicate with each other, either directly or through a master control, thus building a comprehensive environment that the user can control [32–34]. In order for different devices to correctly communicate and understand commands, a communication protocol is needed as a common language for all these different devices. Various communication protocols are available in the market, such as LonWorks [35], Konnex [36], X10 [37].

Smart homes are controlled by users setting different policies according to their preferences. However, the complexity of these policies can vary greatly. Therefore, we introduce the concepts of *simple policy* and *compound policy.* A simple policy is a policy that causes a direct invocation of only one functionality in only one feature. A compound policy is a policy that causes an invocation of more than one functionality within the same or different features. In other words, a compound policy can be seen as the concatenation of two or more simple policies. Consider a policy that states “Close the water tap when the water level reaches 75% of the sink in the kitchen”. This policy is considered a simple policy because it directly invokes only one functionality (P11.1) in the water overflow control features (see Section 4.3). On the other hand, the policy “Close the water tap when the water level reaches 75% of the sink in the kitchen and call the main occupant of the house on his cell phone to inform him”, is considered a compound policy because it invokes two functionalities, namely: functionality “Close the water tap when the water level reaches 75% of the sink in the kitchen”; and functionality “call the main occupant of the house on his cell phone” in the communication feature.

According to the above discussion, detecting policy interactions can be achieved by detecting simple policy interactions. This even can provide more precise results as it will detect interactions within a compound policy by detecting interactions between two simple policies in the body of this compound policy.

Since by definition a simple policy invokes only one functionality in one feature, detecting interactions between functionalities is equivalent to detecting interactions between simple policies. Therefore, the remainder of this paper uses the terms functionality and simple policy interchangeably.

4.2. Assumptions used in the smart homes case study

In the case study described in this paper, the following assumptions were made:

1. The case study detects interaction between functionalities because the interaction detection between functionalities is equivalent to interaction detection between user policies (see Section 4.1).
2. Throughout the case study the term simple policy is used for the term functionality (see Section 4.1).
3. The vacation control feature is assumed to turn on/off TV and lights at predefined time settings. This limitation imposed to use the TV and lights was done for simplicity. However, in other settings other devices can be used in the vacation control.
4. Only the answer machine feature from the set of traditional telecommunications features was separately used and listed in the case study because answer machine can be installed without having to install a more comprehensive set of features.
5. The features used in the case study were defined by the authors based on different resources (see Section 4.3).
6. All devices and sensors are connected to a central network controlled by a master control software.
7. States are described using state variables. Within a certain state only variables of interest to the policy under investigation are listed. This simplification is possible since other state variables have no effect on the outcome of the interaction detection process.

4.3. Smart homes simple policies identification

Before IRIS can be applied to the smart homes case study, the simple policies of a smart home have to be defined. To do this, the smart homes features first have to be identified and then the different functionalities contained in each feature have to be extracted from the features textual definition. These functionalities can be treated as simple user-defined policies to which IRIS can be applied in order to detect interactions (see assumption 1).

Smart homes can contain many different features, some of which are not very common due to their high cost and technical difficulties. Furthermore, many features are designed to help people with specific disabilities and therefore are not installed in all smart homes. This case study will investigate only the common features that are likely to be used in smart homes. Fig. 5 shows an overview of these features.

In order to identify the different functionalities contained in each feature, the textual description of each feature is decomposed into one or more functionalities or so-called simple policies (see assumption 2). Each simple policy is given a unique ID that starts with a P then the number of the feature and finally the number of the simple policy (e.g. P3.2 stands for simple policy number 2 in feature 3).

Feature 1: Intruder Alarm Feature

• **Feature Textual Definition**

This feature sends an alarm signal to the security warden when triggered. The occupants can activate/deactivate the intruder alarm from inside the house using the alarm switch. The intruder alarm feature, when active, can be triggered by a magnetic reeds sensor indicating that a window has been opened, by the main door lock sensor indicating that the main door lock has been opened, by a Passive Infra-Red (PIR) sensor indicating movement in some areas, or by pressure pads indicating that a person stepped on a predefined area.

• **Functionalities (simple policies) in the Intruder Alarm Feature**

P1.1 Activated/deactivated by a switch from inside the house called alarm switch.
P1.2 Alarm is triggered when the feature is active and a magnetic reed sensor indicates that a window is being opened
P1.3 Alarm is triggered when the feature is active and the main door lock sensor indicates that the main door lock is being opened.

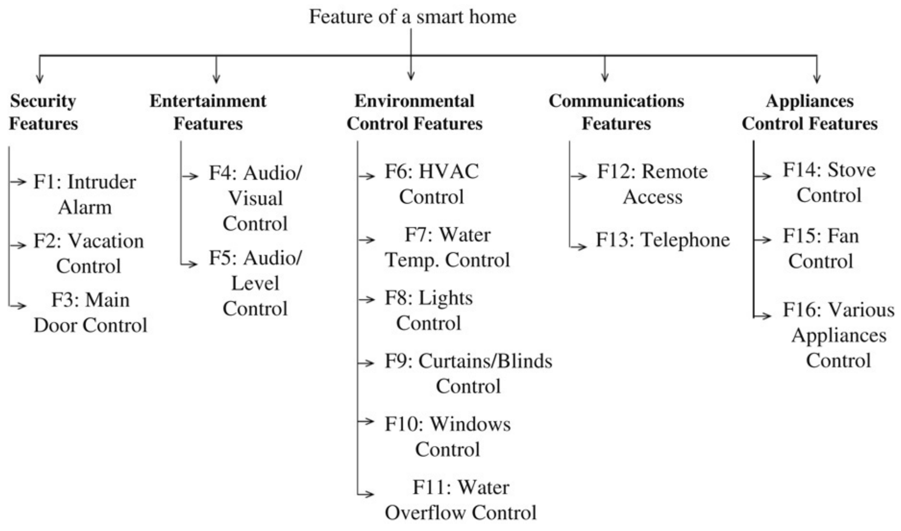


Fig. 5. Overview of smart homes features.

P1.4 Alarm is triggered when the feature is active and a PIR sensor indicates movement in X1, where X1: Location, X1 = {Living room, Bedrooms, Hallway, Kitchen}.
P1.5 Alarm is triggered when the feature is active and pressure pads indicate the presence of a person in X2, where X2: location, X2 = {Living room, Bedrooms, Hallway}.

Feature 2: Vacation Control Feature
- **Feature Textual Definition**

A feature that can be used when the occupants are on vacation for an extended period of time. This feature uses predefined time settings to automatically turn on TV and lights for 60 min in predefined areas. The feature is activated/deactivated by a switch from the interior of the house.
- **Functionalities (simple policies) in the Vacation Control Feature**

P2.1 Activated/deactivated by a switch from inside the house called vacation switch.
P2.2 Turns on TV for 60 min at X3, where X3: Time, X3 = 00:00–23:59.
P2.3 Turns on lights for 60 min at X4 in X5, where X4: Time, X4 = 00:00–23:59 and X5: Location, X5 = {Living room, Bedrooms}.

Feature 3: Main Door Control Feature
- **Feature Textual Definition**

A feature that locks the main door lock of the house using an electronic lock when the main door is shut. The occupants can use an interior switch to unlock the main door from the inside. For safety purposes the electronic lock automatically unlocks when the Gas/Heat/Smoke sensor is triggered.
- **Functionalities (simple policies) in the Main Door Feature**

P3.1 Locks the main door lock of the house when the main door is shut.
P3.2 Occupants can unlock the main door from inside by interior switch.
P3.3 Unlocks the main door when the Gas/Heat/Smoke sensor is triggered.

Feature 4: Audio/Visual Control Feature
- **Feature Textual Definition**

A feature that allows the occupants to control A/V devices through remote controls or to ask the system to turn certain A/V devices on/off at predefined time settings.
- **Functionalities (simple policies) in the Audio/Visual Control Feature**

P4.1 Occupants can control all A/V devices through remote controls.
P4.2 Turns on/off X6 A/V device at X7, where X6: A/V device, X6 = {TV, CD, DVD} and X7: Time, X7 = 00:00–23:59.

Feature 5: Audio Level Control Feature

• **Feature Textual Definition**

A feature that allows the occupants to preset the audio level of different A/V devices to certain levels when they are turned on during the day or night. It also allows the occupants to set a maximum audio level throughout the house that cannot be exceeded. This maximum audio level is chosen by the occupant to avoid loud noise or disturbance during the day/night.

• **Functionalities (simple policies) in the Audio Level Control Feature**

P5.1 Presets the audio level of audio device X8 to X9 when turned on, where X8: A/V device, X8 = {TV, CD, DVD} and X9: Audio level, X9 = {1..63}.

P5.2 Occupants can set X10 as a maximum audio level throughout the house, where X10: Audio level, X10 = {1..63}.

Feature 6: Heating, Ventilation and Air Conditioning Control Feature

• **Feature Textual Definition**

The Heating, Ventilation and Air Conditioning (HVAC) control feature controls the temperature of the house. This feature increases/decreases the temperature inside the house to a user preset temperature when the thermostats' readings are different from this preset temperature. This feature also allows the occupants to define a program to increase/decrease the temperature of the house at predefined time intervals.

• **Functionalities (simple policies) in the Heating, Ventilation and Air Conditioning Control Feature**

P6.1 Increases/decreases the ambient temperature inside the house to X11 when the readings from the thermostats are different from this preset temperature, where X11: Temperature, X11 = {15..35}.

P6.2 Increases/decreases the temperature of the house to X12 at X13, where X12: Temperature, X12 = {15..35} and X13: Time, X13 = 00:00–23:59.

Feature 7: Water Temperature Control Feature

• **Feature Textual Definition**

This feature controls the temperature of the hot water in the house. It maintains the temperature of the hot water from the hot water tap in the kitchen at 45 °C and that of the hot water of the hot water tap of the bathroom at a temperature of 40 °C.

• **Functionalities (simple policies) in the Water Temperature Control Feature**

P7.1 Maintains the temperature of the hot water from the hot water tap in the kitchen at 45 °C.

P7.2 Maintains the temperature of the hot water from the hot water tap of the bathroom at 40 °C.

Feature 8: Lights Control Feature

• **Feature Textual Definition**

This feature controls the intensity of light inside the house. The feature increases/decreases light intensity to correspond to the increase/decrease of a light dimmer. During the night, this feature increases the light intensity in a certain part of the house to the maximum within 2 min when a positive PIR signal is received from that part. When the PIR signal is negative for 15 min, the lights are automatically switched off. Finally, this feature can be set to automatically turn on the lights according to a daylight sensor when the night begins.

• **Functionalities (simple policies) in the Lights Control Feature**

P8.1 Increases/decreases light intensity to correspond to the increase/decrease of a light dimmer.

P8.2 Increases the light intensity during night in X14 to the maximum within 2 min when a positive PIR signal is received from X14, where X14: Location, X14 = {Living room, Bedrooms, bathroom}.

P8.3 Automatically shuts down the lights during night in X15 when a PIR signal is negative for 15 min from X15, where X15: Location, X15 = {Living room, Bedrooms, bathroom, hallway}.

P8.4 Automatically turns on the lights according to a daylight sensor when the night begins.

Feature 9: Curtains and Blinds Control Feature

• **Feature Textual Definition**

This feature can be used to automatically open/close the curtains and blinds in a certain area at predetermined time settings. It can also be set to automatically open/close the curtains and blinds in a certain area according to a daylight sensor.

• **Functionalities (simple policies) in the Curtains and Blinds Control Feature**

P9.1 Automatically opens/closes the curtains and blinds in X16 at X17, where X16: Location, X16 = {Living room, Bedroom} and X17: Time, X17 = 00:00–23:59.

P9.2 Automatically opens/closes the curtains/blinds in X18 according to daylight sensor, where X18: Location, X18 = {Living room, Bedroom}.

Feature 10: Windows Control Feature

• **Feature Textual definition**

This feature opens/closes the windows in predefined areas based on predefined time settings.

• **Functionalities (simple policies) in the Windows Control Feature**

P10.1 Opens/closes the windows in X19 at X20, where X19: Location, X19 = {Living room, Bedroom} and X20: Time, X20 = 00:00–23:59.

Feature 11: Water Overflow Control Feature

• **Feature Textual Definition**

This safety feature shuts down the water tap when the water reaches or exceeds 75% of the total volume of the sink or the tub in the kitchen or in the bathroom.

• **Functionalities (simple policies) in the Water Overflow Control Feature**

P11.1 Closes the water tap when the water reaches or exceeds 75% of the total volume of the sink or the tub either in the kitchen or in the bathroom.

Feature 12: Remote Access Feature

• **Feature Textual Definition**

This feature allows the occupants to remotely activate any feature within the smart home from a remote location via the telephone. The occupants call the home phone number, and when there is no answer after a user-defined number of rings a remote access module is activated asking for a PIN to allow the remote control of home features.

• **Functionalities (simple policies) in the Remote Access Feature**

P12.1 Activates a remote access module when an incoming telephone call has not been answered within X21 rings, where X21: number of phone rings, X21 = {2..8}.

Feature 13: Telephone Feature

• **Feature Textual Definition**

This feature enforces the presence of a Plan Old Telephone Service (POTS) or Voice over Internet Protocol (VoIP) telephone line. It can also have an answer machine installed which records messages when receiving a phone call with no answer for a certain number of rings.

• **Functionalities (simple policies) in the Telephone Feature**

P13.1 Enforces the presence of a telephone line with either standard POTS or VOIP.

P13.2 Activates an answer machine to record messages when receiving a call with no answer for X21 rings, where X21: number of phone rings, X21 = {2..8}.

Feature 14: Stove Control Feature

• **Feature Textual Definition**

This safety feature can be used to shut down and prevent any activation of the stove during predefined time periods. This feature is also used to shut down the stove when the Gas/Heat/Smoke sensor is triggered.

• **Functionalities (simple policies) in the Stove Control Feature**

P14.1 Shut down and prevent any activation of the stove during X22 and X23, where X22 and X23: Time, X22 and X23 = 00:00–23:59.

Feature 15: Fan Control Feature

• **Feature Textual Definition**

This feature automatically turns on the kitchen fan when the humidity sensor is triggered. When the sensor signal is lost for 20 min while the fan is on, this feature automatically switches off the fan.

• **Functionalities (simple policies) in the Fan Control Feature**

P15.1 Automatically turns on the kitchen fan when the humidity sensor is triggered.

P15.2 Automatically switches off the kitchen fan when the humidity signal is lost for 20 min while the fan is on.

Feature 16: Control of Various Appliances Feature

• **Feature Textual Definition**

This feature allows the occupants of the house to control various appliances like the food processor, water boiler. . . etc. using remote controls.

• **Functionalities (simple policies) in the Control of Various Appliances Feature**

P16.1 Occupants can control various appliances like the food processor, water boiler, etc. using remote controls.

4.4. Applying IRIS to detect interactions in smart homes

This section presents the results of applying IRIS to the smart home case study. Each subsection first names and shows the execution of the IRIS step and then presents results obtained from applying the step.

4.4.1. Results of step 1: Simple policies classification

The first step is used to organize the simple policies into system axiom simple policies and dynamic behavior simple policies. A simple policy is considered a dynamic behavior simple policy if it represents an action that is executed when an event triggers a transition of the system from one state to another state. On the other hand, a simple policy is considered a system axiom simple policy if it represents a rule or a property that has to be preserved at all times independent of the occurrence of any trigger events.

The results of the application of the first step are as follows:

System axiom simple policies: P4.1, P5.2, P7.1, P7.2, P13.1, P16.1

Dynamic behavior simple policies: P1.1, P1.2, P1.3, P1.4, P1.5, P2.1, P2.2, P2.3, P3.1, P3.2, P3.3, P4.2, P5.1, P6.1, P6.2, P8.1, P8.2, P8.3, P8.4, P9.1, P9.2, P10.1, P11.1, P12.1, P13.2, P14.1, P14.2, P15.1, P15.2.

4.4.2. Results of step 2: Simple policies attributes identification

This step identifies different attributes within the smart homes policies. To represent these attributes in a comprehensive view, two tables are generated (see Tables 1 and 2). Table 1 contains system axiom simple policies and

Table 1
System axioms simple policies table

ID	Description	Rule	Condition	Variables	Variables value range
P4.1	Occupants can control all A/V devices through remote controls	Control all A/V devices through remote controls	True	–	–
P5.2	P5.2 Occupants can set X10 as a maximum audio level through house	Set X10 as a maximum audio level throughout the house	True	X10:Audio level	{1..63}
P7.1	Maintains the temperature of the hot water from the hot water tap in the kitchen to 45 °C	Maintains the temperature of the hot water of the hot water tap in the kitchen to 45 °C	True	–	–
P7.2	Maintains the temperature of the hot water from the hot water tap of the bathroom to 40 °C.	Maintains the temperature of the hot water of the hot water tap of the bathroom to 40 °C.	True	–	–
P13.1	Enforces the presence of a telephone line with either standard POTS or VOIP	A telephone line is always present.	True	–	–
P16.1	occupants can control various appliances like the food processor, water boiler. . . etc. by remote controls	Control various appliances by remote control	True	–	–

Table 2
Dynamic behavior simple policies table

ID	Description	Pre-state	Triggering event	Action	Next state	Variables	Variables value range
P1.1	Security Alarm is Activated/deactivated by a switch from inside the house called alarm switch.	SecurityAlarm=off/on	Activate/ deactivate security alarm switch is pressed	Activate/ deactivate security alarm	SecurityAlarm =on/off	–	–
P1.2	Alarm is triggered when the feature is active and a magnetic reeds sensor indicates that a window is being opened	SecurityAlarm =on, Alarm=not_set, Windows(_)= closed	Window is opened	Set security alarm	SecurityAlarm =on, Alarm=set, windows(_)=open	–	–
P1.3	Alarm is triggered when the feature is active and the main door lock sensor indicates that the main door lock is being opened	SecurityAlarm =on, Alarm=not_set, MainDoorLock=close	Main door lock is opened	Set security alarm	SecurityAlarm =on, Alarm=set, MainDoorLock=open	–	–
P1.4	Alarm is triggered when the feature is active and a PIR sensor Indicates movement in X1	SecurityAlarm =on, Alarm=not_set, PIR=negative	Movements in X1	Set security alarm	SecurityAlarm =on, Alarm=set, PIR=positive	X1: Location	{LivRm, BdRm, Hall, kitch}
P1.5	Alarm is triggered when the feature is active and pressure pads indicate the presence of person in X2	SecurityAlarm =on, Alarm=not_set, PressurePad= negative	Pressure pad in X2 is pressed	Set security alarm	SecurityAlarm =on, Alarm=set, PressurePad=positive	X2: location	{LivRm, BdRm, Hall}
P2.1	Vacation Control is Activated/deactivated by a switch from inside the house called vacation switch	VacationControl =off/on	Activate/ deactivate vacation control switch is pressed	Activate/ deactivate vacation control	VacationControl=on/off	–	–
P2.2	Vacation control Turns on TV for 60 min at X3	VacationControl =on, TV=off	Time=X3	Turn on TV for 60 min	VacationControl=on, TV=on	X3: Time	{00:00–23:59}
P2.3	Vacation control turns on lights for 60 min at X4 in X5	*VacationControl =on, Lights(X5)=off*	*Time=X4*	*Turn on lights for 60 min in X5*	*VacationControl=on, Lights(X5)=on*	*X4: Time, X5: Location*	*X4={00:00 –23:59} X5 = {LivRm, BdRm}*
P3.1	Main Door lock feature will lock the main door lock of the house when main door shut.	*MainDoor=open MainDoorLock=open*	*Main door is shut*	*Lock the main door lock*	*MainDoor =closed MainDoorLock =closed*	–	–
P3.2	Occupants can unlock the main door from inside by interior switch	*MainDoor =closed MainDoorLock =closed*	*Unlock main door switch is pressed*	*Unlock the main door lock and open the main door*	*MainDoor=open MainDoorLock =open*	–	–
P3.3	Unlocks the main door when the Gas/ Heat/ Smoke sensor triggers.	*MainDoor =closed MainDoorLock =closed*	*Gas/heat/ Smoke sensor is triggered*	*Unlock the main door lock and open the main door*	*MainDoor=open MainDoorLock =open*	–	–

Table 2 *(continued)*

ID	Description	Pre-state	Triggering event	Action	Next state	Variables	Variables value range
P4.2	Turns on/off X6 A/V device at X7	*X6On=false/true*	*Time=X7*	*Turn on/off the A/V device X6*	*X6On=True/false*	*X6: A/V device, X7: Time,*	*X6 = {TV, CD, DVD} X7 = {00:00–23:59}*
P5.1	Presets the audio level of audio device X8 to X9 when turned on	*X8On=False X8Audio_ level =DxR*	*X8 is turned on*	*Preset the audio level of X8 to X9*	*X8On=True, X8Audio_level=X10*	*X8:A/V device, X9:Audio level*	*X8 = {TV, CD, DVD}, X9 = {1..63}*
P6.1	Increases/Decreases the temp. inside the house to X11 when the reading from the thermostats are different from this preset temp.	*Temp=DxR*	*Thermostats ≠ X11*	*Increases/ Decreases the temp. inside the house to X11*	*Temp=X11*	*X11: Temp.*	*{15..35}*
P6.2	Increases/decreases the temperature of the house to X12 at X13.	Temp=DxR	Time=X13	Increase/ decrease temp of the house to X12	Temp=X12	X12: Temp. X13: Time	X12 = {15..35} X13 = {00:00–23:59}
P8.1	Increases/decreases light intensity to correspond to the increase/decrease of a light dimmer slider	LightIntens(_) =DxR	Increase/ decrease of the dimmer slider	Increase/ decrease light intensity to match increase/ decrease of the slider	LightIntens.(_) =DimmerSlider	–	–
P8.2	Increases the light intensity during night in X14 to a maximum within 2 min when a positive PIR signal is received from X14.	Daylight=false Lights(X14)=off LightIntens(X14)=0	Movement in X14	Increase the light intensity in X14 to a max within 2 min	Daylight=false Lights(X14)=on LightIntens.(X14) =max	X14: Location	{LivRm, BdRm, bathRm}
P8.3	Automatically shuts down the lights during night in X15 when a PIR signal is negative for 15 min from X15.	Daylight=false Lights(X15)=on LightIntens (X15)=DxR	No movements in X15 for 15 min	Shut down the lights	Daylight=false Lights(X15)=off LightIntens.(X15) =0	X15: Location	{LivRm, BdRm, bathRm, hall}
P8.4	Automatically turns on the lights according to a daylight sensor when the night begins.	Daylight=True Lights(_)=off	Night begins	Turn lights on automatically	Daylight=false Lights(_)=on	–	–
P9.1	Automatically opens/ closes the curtains and blinds in X16 at X17.	CurtainsBlinds (X16)=close /open	Time=X17	Open/close the blinds and curtains in X16	CurtainsBlinds (x16)=open/ close	X16: Location, X17: Time	X16 = {LivRm, BdRm} X17 = {00:00–23:59}

(continued on next page)

Table 2 *(continued)*

ID	Description	Pre-state	Triggering event	Action	Next state	Variables	Variables value range
P9.2	Automatically open/close the curtains and blinds in X18 according to daylight sensor	Daylight= false /true, CurtainsBlinds (X18)=close /open	Day/night begins	Open/close curtains and blinds in X18	Daylight=true/ false, CurtainsBlinds (x18)=open/ close	X18: Location	{LivRm, BdRm}
P10.1	Opens/closes the windows in X19 at X20	Windows(X19) =close/open	Time=X20	Open/close windows in X19	Windows(X19)= open/close	X19: Location, X20: Time	X19 = {LivRm, BdRm} X20 = {00:00–23:59}
P11.1	Shuts down the water tap when the water reaches or exceeds 75% of the total size of the sink or the tub either in the kitchen or the bathroom	Tap/shower-Valve=open	Water level ≥75%	Shutdown water	Tap/showerValve =closed	–	–
P12.1	Activates a remote access module when receives a telephone call for X21 rings with no answer	Telephone=idle RemoteAccess=idle	Receive a call request, and no answer for X21 rings.	Activate remote access module	Telephone=busy, RemoteAccess =Active	X21: number of phone rings	{2..8}
P13.2	Activates an answer machine to record messages when receiving a call with no answer for X21 rings	Telephone=idle AnswerMachine =idle	Receive a call request, and no answer for X21 rings.	Activate answer machine	Telephone=busy, AnswerMachine=on	X21: number of phone rings	{2..8}
P14.1	Shut down and prevent any activation of the stove during X22 and X23	Stove=DxR	Time=X22	Shutdown and prevent any activation of the stove till X23	Stove=off	X22, X23: Time	{00:00–23:59}
P14.2	Shutdown the stove when the Gas/Heat/Smoke sensor is triggered	Stove=DxR	Gas/heat/ Smoke sensor is triggered	Shutdown the stove	Stove=off	–	–
P15.1	Automatically turns on the kitchen fan when the humidity sensor is triggered	KitchenFan=off	Humidity sensor is triggered	Turn on kitchen fan	KitchenFan=on	–	–
P15.2	Automatically shut off the kitchen fan when the humidity signal is lost for 20 min while fan is on	KitchenFan=on	Humidity sensor is negative for 20 min	Turn off the kitchen fan	KitchenFan=off	–	–

has the following columns: *ID*: contains a unique identifier for each simple policy. This ID will be used to refer to this simple policy later on. *Description* contains an informal description of the simple policy. *Rule* describes the required property to be valid at all times. *Condition* contains any specific conditions that may be relevant to this simple policy and if no conditions apply, then it is given the value True. Table 2 contains dynamic behavior simple policies and has the following columns: *ID* and *description*: both of them are used in the same way as in Table 1. *Pre-state* describes the state of the system before the trigger event occurs. *Triggering event* describes the event that requires the system to take a certain action as described in the simple policy. *Action* describes the system response to the trigger event in order to fulfill the simple policy. *Next state* describes the state that the system will reach after executing the specified action. In order to accommodate domains that include variables, such as the smart homes domain, the tables include two more columns which contain the variables in the corresponding simple policy and their value range. These are represented by *Variables* and *Variables Value Range*.

State variables were used to describe the pre-state and next state of the system. For example, MainDoorLock=closed states that the main door lock is in a closed state. Not every state will have a value assigned to it. The Value DxR in the table corresponds to a don't care value. The don't care value is necessary to represent certain cases as in P6.2 where it does not matter what the value of the previous temperature is because the objective of P6.2 is to increase/decrease the temperature to the predefined setting regardless of the previous temperature.

4.4.3. *Results of step 3: Trigger events extraction*

In this step the developer identifies and extracts all the different trigger events from Table 2. The idea behind this step is to identify different simple policies that are triggered by the same trigger event. The results of this step are shown in Table 3.

Table 3
Trigger events and the simple policies they trigger

Event ID	Informal event description	Simple policies triggered by this event
E1	Activate/deactivate security alarm switch is pressed	P1.1
E2	A Window is opened	P1.2
E3	Main door lock is opened	P1.3
E4	Movements	P1.4, P8.2
E5	Pressure pad is pressed	P1.5
E6	Activate/deactivate vacation control switch is pressed	P2.1
E7	Time	P2.2, P2.3, P4.2, P6.2, P9.1, P10.1, P14.1
E8	Main door is shut	P3.1
E9	Unlock main door switch is pressed	P3.2
E10	Gas/heat/Smoke sensor is triggered	P3.3, P14.2
E11	A/V device is turned on	P5.1
E12	Thermostats $\neq$ X11	P6.1
E13	Increase/decrease of the dimmer slider	P8.1
E14	No movements in X15 for 15 min	P8.3
E15	Day begins	P8.4, P9.2
E16	Night begins	P9.2
E17	Water level $\geq$75%	P11.1
E18	Receive a call request, and no answer for X21 rings.	P12.1, P13.2
E19	Humidity sensor is triggered	P15.1
E20	Humidity sensor is negative for 20 min	P15.2

4.4.4. *Results of step 4: Linked events identification*

Linked events can best be described using an example. Consider the event E2 stating that "A window is opened". The occurrence of this event will most likely cause a change in the temperature of the house and therefore the event

Table 4
Linked events

ID	Informal event description	Linked to	Mathematical representation
E1	Activate/deactivate security alarm switch is pressed	E2, E3, E4, E5, E6, E8, E9, E13, E18	E1<∼>E2, E1<∼>E3, E1<∼>E4, E1<∼>E5, E1<∼>E6, E1<∼> E8, E1∼>E9, E1<∼>E13, E1∼>E18
E2	A window is opened	E9, E12, E18, E20	E2<∼>E9, E2∼>E12, E2∼>E18, E2∼>E20
E3	Main door lock is opened	E2, E4, E5, E6, E11, E13, E14, E18, E20	E3∼>E2, E3<∼>E4, E3<∼>E5, E3<∼>E6, E3∼>E11, E3<∼>E13, E3∼>E14, E3∼>E18, E3∼>E20
E4	Movements	E2, E5, E6, E8, E9, E11, E13, E18	E4∼>E2, E4<∼>E5, E4∼>E6, E4∼>E8, E4∼>E9, E4∼>E11, E4∼>E13, E4∼>E18
E5	Pressure pad is pressed	E2, E6, E8, E9, E11, E13, E18	E5∼>E2, E5∼>E6, E5<∼>E8, E5∼>E9, E5∼>E11, E5∼>E13, E5∼>E18
E6	Activate/deactivate vacation control switch is pressed	E8, E9, E13, E14, E18	E6<∼>E8, E6∼>E9, E6∼>E13, E6∼>E14, E6∼>E18
E7	Time	Ei, where $i = 1..20$	E7∼>Ei, where $i = 1..20$
E8	Main door is shut	E2, E9, E10, E11, E13, E14, E18	E8∼>E2, E8∼>E9, E8∼>E10, E8∼>E11, E8<∼>E13, E8∼>E14, E8∼>E18
E9	Unlock main door switch is pressed	E3, E12, E13, E14, E18	E9∼>E3, E9∼>E12, E9∼>E13, E9∼>E14, E9∼>E18
E10	Gas/heat/smoke sensor is triggered	E2, E3, E4, E5, E9, E12, E18, E19	E10∼>E2, E10∼>E3, E10∼>E4, E10∼>E5, E10∼>E9, E10<∼>E12, E10∼>E18, E10∼>E19
E11	A/V device is turned on	E18	E11∼>E18
E12	Thermostats ≠ X11	E18, E19, E20	E12∼>E18, E12∼>E19, E12∼>E20
E13	Increase/decrease of the dimmer slider	E18	E13∼>E18
E14	No movements in X15 for 15 min	E18	E14∼>E18
E15	Day begins	E1, E2, E3, E4, E5, E9, E11, W12, E13, E18	E15∼>E1, E15∼>E2, E15∼>E3, E15∼>E4, E15∼>E5, E15∼>E9, E15∼>E11, E15∼>E12, E15∼>E13, E15∼>E18
E16	Night begins	E1, E2, E3, E4, E5, E9, E11, E12, E13, E14, E18	E16∼>E1, E16∼>E2, E16∼>E3, E16∼>E4, E16∼>E5, E16∼>E9, E16∼>E11, E16∼>E12, E16∼>E13, E16∼>E14, E16∼>E18
E17	Water level=75%	E4, E5, E10, E18, E19	E17∼>E4, E17∼>E5, E17∼>E10, E17∼>E18, E17∼>E19
E19	Humidity sensor is triggered	E18	E19∼>E18
E20	Humidity sensor is negative for 20 min	E18	E20∼>E18

E12 "Thermostats ≠ X11" will also be triggered. This means that whenever event E2 occurs then event E12 will also occur as a logical consequence after a short time period. This means that event E2 leads to event E12 (or event E12 is linked to E2) and this is expressed as E2 $\sim>$ E12 where the curly arrow ($\sim>$) indicates that the occurrence of first event will most likely lead to the occurrence of second event.

Linked events can be formally defined as follows: "For events that can be initiated by a user (U) or a system (S): Event $\sigma 2$ is said to be linked to event $\sigma 1$ ($\sigma 1 \sim> \sigma 2$) if event $\sigma 1$ can be logically followed by event $\sigma 2$" or formally: $(\sigma 1 \sim> \sigma 2) \leftrightarrow (@\ \sigma 1 \rightarrow (\sigma 2 = \text{S.createEvent( )} \vee \quad \sigma 2 = \text{U.createEvent( )}))$. This latter formula can be read as follows "(event $\sigma 1$ leads to event $\sigma 2$) is equivalent to (at the occurrence of $\sigma 1$ this will lead to ($\sigma 2$ is created by the system S OR $\sigma 2$ is created by the user U))".

The importance of linked events is to examine the actions of the policies that might be triggered sequentially by sequential events and the effects of later actions on actions that are already being executed (refer to Section 3.3, guideline 4 for an example). The results of this step are shown in Table 4. It is worth mentioning that when two events can lead to each other (i.e. E1 $\sim>$ E2 and E2 $\sim>$ E1) then this can be written as E1 $<\sim>$ E2. In our case study, this is only listed once in the row of the first event E1 in Table 4.

Linked events can also be used to examine interactions that arise when there is a chain of requirements actions due to their linked events, which is called infinite looping. Although this will make the interaction analysis more thorough, but it will require more formalism put into the IRIS approach which degrades its semi-formality and make it tends towards using formal semantics such as state charts.

4.4.5. Results of step 5: Graphical representation

In this step, an event-based graphical representation is used to link each event from Table 4 with the simple policies it triggers. This graphical representation facilitates the detection of interactions between the simple policies. Each simple policy has the following attributes: *Pre-state*, *Action*, and *Next state.* The result of this step is shown in Fig. 6.

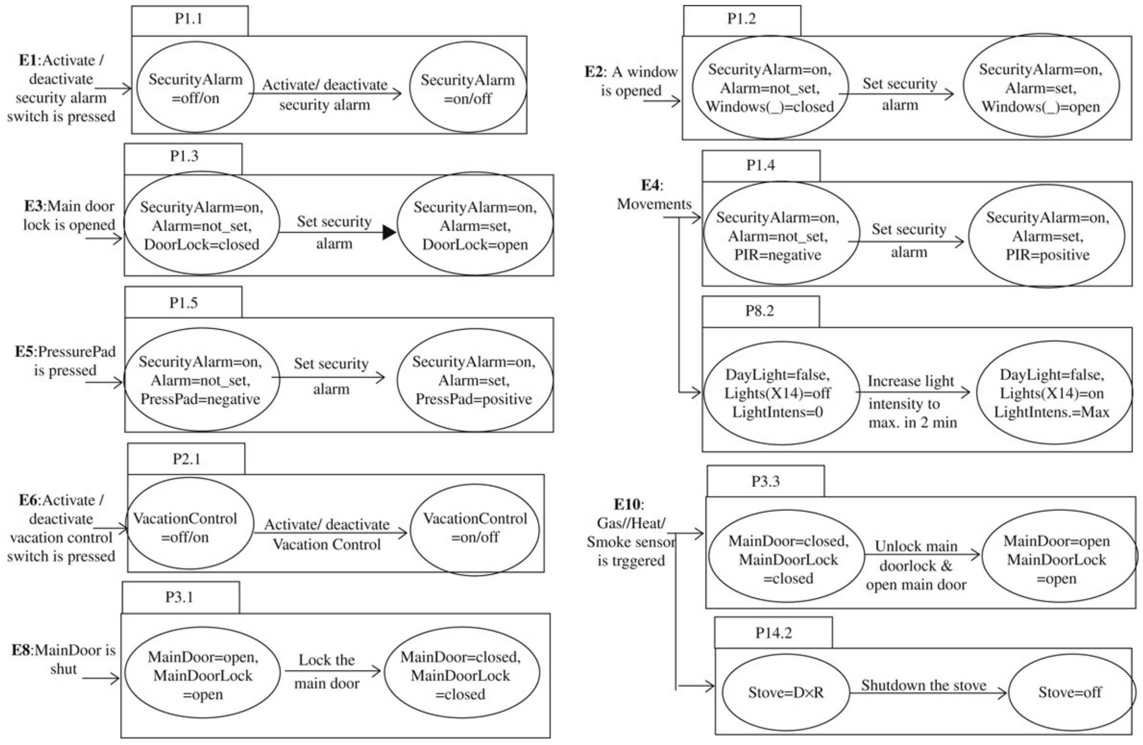

P1.1
E1:Activate / deactivate security alarm switch is pressed
SecurityAlarm =off/on
Activate/ deactivate security alarm
SecurityAlarm =on/off
P1.2
E2: A window is opened
SecurityAlarm=on, Alarm=not_set, Windows(_)=closed
Set security alarm
SecurityAlarm=on, Alarm=set, Windows(_)=open
P1.3
E3:Main door lock is opened
SecurityAlarm=on, Alarm=not_set, DoorLock=closed
Set security alarm
SecurityAlarm=on, Alarm=set, DoorLock=open
P1.4
E4: Movements
SecurityAlarm=on, Alarm=not_set, PIR=negative
Set security alarm
SecurityAlarm=on, Alarm=set, PIR=positive
P1.5
E5:PressurePad is pressed
SecurityAlarm=on, Alarm=not_set, PressPad=negative
Set security alarm
SecurityAlarm=on, Alarm=set, PressPad=positive
P8.2
DayLight=false, Lights(X14)=off LightIntens=0
Increase light intensity to max. in 2 min
DayLight=false, Lights(X14)=on LightIntens.=Max
P2.1
E6:Activate / deactivate vacation control switch is pressed
VacationControl =off/on
Activate/ deactivate Vacation Control
VacationControl =on/off
P3.3
E10: Gas//Heat/ Smoke sensor is trggered
MainDoor=closed, MainDoorLock =closed
Unlock main doorlock & open main door
MainDoor=open MainDoorLock =open
P3.1
E8:MainDoor is shut
MainDoor=open, MainDoorLock =open
Lock the main door
MainDoor=closed, MainDoorLock =closed
P14.2
Stove=D×R
Shutdown the stove
Stove=off

Fig. 6. Events and the simple policies they trigger.

4.4.6. Results of step 6: Interactions detection

In this step, the developer detects interactions between simple policies using the guidelines introduced in Section 3.3. The detection is subjective which means a human developer will use the different tables and graphs developed along with the provided guidelines to determine if there exists an interaction between two simple policies or not.

According to the presented interaction taxonomy (Section 3.3), there are three types of interactions: Interactions between two system axiom simple policies, interactions between a system axiom simple policy and a dynamic behavior simple policy, and interactions between two dynamic behavior simple policies. The developer now tries to find interactions within these three categories. Accordingly, the developer starts by looking at each of these

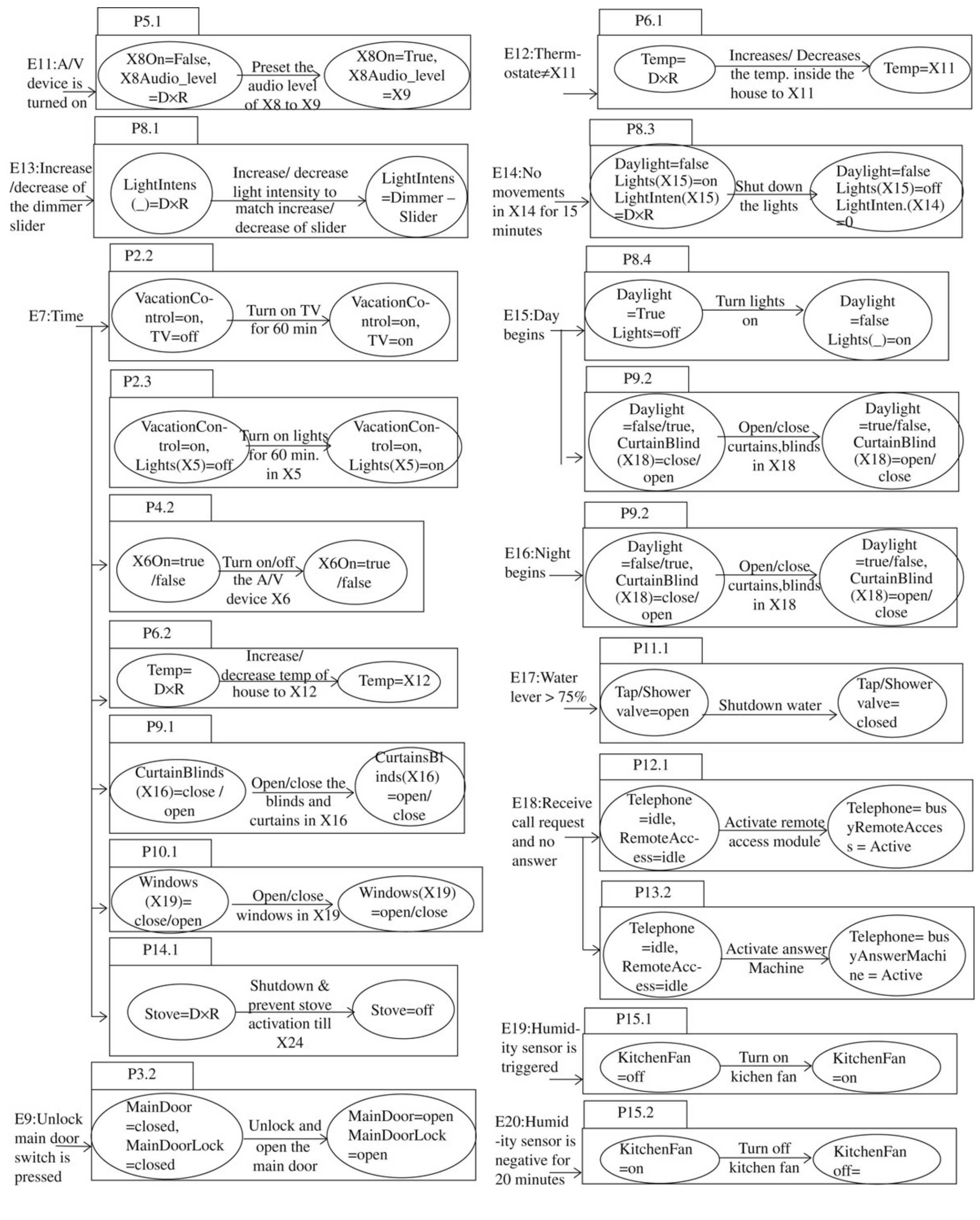

P5.1
E11:A/V device is turned on
X8On=False, X8Audio_level =D×R
Preset the audio level of X8 to X9
X8On=True, X8Audio_level =X9
P6.1
E12:Thermostate≠X11
Temp= D×R
Increases/ Decreases the temp. inside the house to X11
Temp=X11
P8.1
E13:Increase /decrease of the dimmer slider
LightIntens (_)=D×R
Increase/ decrease light intensity to match increase/ decrease of slider
LightIntens =Dimmer – Slider
P8.3
E14:No movements in X14 for 15 minutes
Daylight=false Lights(X15)=on LightInten(X15) =D×R
Shut down the lights
Daylight=false Lights(X15)=off LightInten.(X14) =0
P2.2
E7:Time
VacationControl=on, TV=off
Turn on TV for 60 min
VacationControl=on, TV=on
P8.4
E15:Day begins
Daylight =True Lights=off
Turn lights on
Daylight =false Lights(_)=on
P2.3
VacationControl=on, Lights(X5)=off
Turn on lights for 60 min. in X5
VacationControl=on, Lights(X5)=on
P9.2
Daylight =false/true, CurtainBlind (X18)=close/ open
Open/close curtains,blinds in X18
Daylight =true/false, CurtainBlind (X18)=open/ close
P4.2
X6On=true /false
Turn on/off the A/V device X6
X6On=true /false
P9.2
E16:Night begins
Daylight =false/true, CurtainBlind (X18)=close/ open
Open/close curtains,blinds in X18
Daylight =true/false, CurtainBlind (X18)=open/ close
P6.2
Temp= D×R
Increase/ decrease temp of house to X12
Temp=X12
P11.1
E17:Water lever > 75%
Tap/Shower valve=open
Shutdown water
Tap/Shower valve= closed
P9.1
CurtainBlinds (X16)=close / open
Open/close the blinds and curtains in X16
CurtainsBlinds(X16) =open/ close
P12.1
E18:Receive call request and no answer
Telephone =idle, RemoteAccess=idle
Activate remote access module
Telephone= busyRemoteAccess = Active
P10.1
Windows (X19)= close/open
Open/close windows in X19
Windows(X19) =open/close
P13.2
Telephone =idle, RemoteAccess=idle
Activate answer Machine
Telephone= busyAnswerMachine = Active
P14.1
Stove=D×R
Shutdown & prevent stove activation till X24
Stove=off
P15.1
E19:Humidity sensor is triggered
KitchenFan =off
Turn on kichen fan
KitchenFan =on
P3.2
E9:Unlock main door switch is pressed
MainDoor =closed, MainDoorLock =closed
Unlock and open the main door
MainDoor=open MainDoorLock =open
P15.2
E20:Humid-ity sensor is negative for 20 minutes
KitchenFan =on
Turn off kitchen fan
KitchenFan off=

Fig. 6. *(continued)*

categories independently and then applies all the guidelines provided under this category to detect interactions between simple policies. This section presents samples of the interactions detected in each interactions type along with a full description of how these interactions were detected. A summary of all the detected results is presented in Table 9 to give an overview of the overall results of applying IRIS to smart homes. Also, to provide a comprehensive coverage

Table 5
Example of interaction between two system axiom simple policies using guideline 1

Interaction ID	I52
Type of interaction	Interaction between two system axiom simple policies
Interacting simple policies	P4.1 and P5.2
Guideline used	Guideline 1
Explanation and scenario of interaction	There is a contradiction between the two values of the *Rule* attribute for the two system axiom simple policies. The rule of P4.1 can override the rule of P5.2 or vice versa. An interaction scenario can be "what happens when the user tries to use the remotes to go beyond the max audio level of the house?" If the system allows the user to use the remote control to exceed the maximum audio level then the P4.1 rule has overridden the P5.2 rule. But if the system will not allow the user to use the remote to go beyond the maximum audio level then the P5.2 rule has overridden P4.1 rule. It is worth mentioning that both simple policies are of equal priorities

of the obtained results, all detected interactions are presented in Appendix A along with a full description of each interaction and a possible interaction scenario for the two interacting simple policies.

As mentioned earlier, a key issue on detecting interactions between simple policies is the application of the interaction detection guidelines described in Section 3.3. This is very important because these guidelines serve as the experience factor for a regular human developer. Also, the developed tables and graphs from the previous steps serve as a clear presentation of all the information that will facilitate the detection process. Therefore, with these guidelines being applied on the developed tables and graphs, the subjectivity part of the approach is heavily reduced. In the following, we illustrate how each of the guidelines is applied to the case study along with examples of detected interactions. Each example of interactions will be presented in a standard table form as shown in Table 5.

4.4.6.1. Interactions between two system axiom simple policies using guideline 1. This type of interactions occurs between two system axiom simple policies. Guideline 1 is used to detect the interaction: "*Two system axiom simple policies interact when the rule attribute (see Table 1) of one system axiom simple policy contradicts the rule attribute of a second system axiom simple policy*". Based on this guideline, the developer is required to examine the system axiom simple policies table (*see Table 1*) developed in step 2. The developer has to pairwise compare all system axiom simple policies (Table 1) with the aim of finding contradictions between the *Rule* attributes of the two simple policies under investigation. Such a contradiction is an indication of an interaction between the two system axiom simple policies.

According to Table 1, there are 15 comparisons necessary in order to examine all system axiom simple policies in the smart homes case study. Table 5 provides an example of such an interaction that was detected using guideline 1.

4.4.6.2. Interaction between a system axiom simple policy and a dynamic behavior simple policy using guideline 2. The second type of interactions occurs between a system axiom simple policy and a dynamic behavior simple policy. This type of interactions is detected using guideline 2 which states that "*an interaction between a system axiom simple policy and a dynamic behavior simple policy occurs when the action attribute of the dynamic behavior simple policy (see Table 2) contradicts the rule attribute of the system axiom simple policy (see Table 1)*". Based on this guideline, the developer is required to examine the dynamic behavior simple policies (Table 2) and system axiom simple policies tables (Table 1) developed in step 2. The developer has to compare pairwise every dynamic behavior simple policy and every system axiom simple policy with the objective of finding contradictions between the value of the *Action* attribute of the dynamic behavior simple policy and the *Rule* attribute of the system axiom simple policy. An interaction is detected if the *Action* attribute of the dynamic behavior simple policy contradicts the *Rule* attribute of the system axiom simple policy. Although this might seem to be a very long process, the tool presented in Section 5 will extract the *Action* attribute of the dynamic behavior simple policy and the *Rule* attribute of the system axiom simple policy, display them on the screen along with guideline 2 and ask the developer to look for contradictions. Then the tool support extracts the next comparison.

There are 174 comparisons necessary in order to detect all possible interactions between a system axiom simple policy and a dynamic behavior simple policy in the case of the smart homes case study. Table 6 provides an example of an interaction between a system axiom simple policy and a dynamic behavior simple policy that was detected using guideline 2.

Table 6
Example of interaction between a system axiom simple policy and a dynamic behavior simple policy using guideline 2

Interaction ID	I24
Type of Interaction	Interaction between a system axiom simple policy and a dynamic behavior simple policy
Interacting simple policies	P2.2 and P4.1
Guideline used	Guideline 2
Explanation and scenario of interaction	There is a contradiction between the value of the *Action* attribute of the dynamic behavior simple policy (P2.2) and the value of the *Rule* attribute for the system axiom simple policy (P4.1). The action of P2.2 violates the rule of P4.1. A possible interaction scenario could be the following: A user gets home while the vacation control feature is active and the action of P2.2 is being executed and the user tries to use the remote control to switch off the TV. According to the definition of the vacation control feature, the control of the TV is now exclusively done by the feature. When the user now tries to use the remote and the remote does not work then the action of P2.2 has broken the rule of P4.1. On the other hand, if the remote works then P4.1 rule (which gave control of the TV to the user) has cancelled P2.2 action before its completion. It is again worth mentioning that both simple policies are of equal priority.

4.4.6.3. Interaction between two dynamic behavior simple policies using guideline 3. The third type of interactions occurs between two dynamic behavior simple policies. Guidelines 3 and 4 are used to detect interactions of this type. First we consider the interactions according to guideline 3 which states that "an interaction between two dynamic behavior simple policies occurs if the *previous state* attributes of both dynamic behavior simple policies are the same AND the *trigger event* attributes of both dynamic behavior simple policies are the same AND the *Action* attributes of both dynamic behavior simple policies contradict each other".

Based on this guideline, the developer is required to examine Fig. 6 which was developed in step 5. S/he looks for trigger events that trigger more than one dynamic behavior simple policy. With each one of these trigger events, the developer compares every two dynamic behavior simple policies. The focus in this comparison is on finding identical pre-states for both dynamic behavior simple policies under investigation but with contradicting actions. With the aid of Fig. 6 developed in step 5 and the tool support that automatically finds events that trigger more than two dynamic behavior simple policies and displays them, this becomes a trivial task.

There are 25 comparisons necessary in order to detect all possible interactions between two dynamic behavior simple policies using guideline 3 in the case of the smart homes case study. Table 7 provides an example of a detected interaction of this type.

Table 7
Example of interaction between two dynamic behavior simple policies using guideline 3

Interaction ID	I63
Type of interaction	Interaction between two dynamic behavior simple policies
Interacting simple policies	P6.2 and P10.1
Guideline used	Guideline 3
Explanation and scenario of interaction	Both simple policies are triggered by E7 AND they can be considered to have the same pre-states AND there is a contradiction between the two actions of the two simple policies. The action of P10.1 has a negative impact on the action of P6.2. An example of an interaction scenario: "The system opens the windows and at the same time tries to raise the temperature of the house". It is obvious that if the temperature outside the house is cold then action of P10.1 has negative impact on action of P6.2.

4.4.6.4. Interaction between two dynamic behavior simple policies using guideline 4. We now consider the interactions between two dynamic behavior simple policies according to guideline 4 which states that "two dynamic behavior simple policies interact when they are triggered by linked events and the action of one dynamic behavior simple policy cancels or contradicts the action of the other dynamic behavior simple policy".

Based on this guideline, the developer is required to examine Fig. 6 which was developed in step 5 using the linked events table from step 4. The developer now has to focus on the dynamic behavior simple policies that are triggered

Table 8
Example of interaction between two dynamic behavior simple policies using guideline 4

Interaction ID	I65
Type of interaction	Interaction between two dynamic behavior simple policies
Interacting simple policies	P8.1 and P8.2
Guideline used	Guideline 4
Explanation and scenario of interaction	These two simple policies are triggered by the linked events E4 and E13 where E4 $\sim>$ E13. The action of P8.1 can override and cancel the action of P8.2 before its completion. An example of an interaction scenario is the situation when someone wakes at night and tries to increase the light through the light dimmer. According to P8.2 a person that wakes at night and walks into a specified part of the house causes the lights to increase in that part to a maximum over the period of two minutes. But the user can turn the light dimmer after 30 s to increase/decrease the light intensity. In such a situation, P8.2 was triggered first by E13, i.e., the system starts increasing the light over a period of two minutes. But then that person changes the light dimmer to increase/decrease the lights and thus triggering P8.1 which in turn cancels the action of P8.1 before its completion.

by linked trigger events. This means that the developer looks at the linked trigger events table (*see Table* 4) and selects the first two linked events. Then the developer examines the dynamic behavior simple policies that are triggered by these two linked events with the focus on finding an action of one dynamic behavior simple policy that will cancel or contradict the action of the other dynamic behavior simple policy. Then the developer goes back to select the next two linked events and goes through the same comparison again. With the aid of the linked event table developed in step 4, Fig. 6 which was developed in step 5, and the tool support that automatically extracts every 2 linked events and displays the dynamic behavior simple policies they trigger on the screen using step 5 graphical representation, the interaction detection becomes a manageable task.

Guideline 4 results in 311 comparisons in the smart homes case study. Table 8 provides an example of a detected interaction of this type.

4.4.6.5. Summary of the detected interactions. Table 9 presents the summary of all the obtained results (see Appendix A for a possible interaction scenario of each of the detected interactions). The simple policy column contains the simple policy under investigation while the interacting simple policies column lists the simple policies that interact with the simple policy under investigation. Note that when an interaction is detected between two simple policies (e.g. P1.1 and P1.2), then this interaction is listed in the row of the first policy (P1.1) only and will not be repeated as part of the interactions of the second policy (P1.2). The total number of detected unique interactions is 83 interactions.

4.5. Discussion of the obtained results

Section 4.4 presented the use of IRIS to detect interactions among policies of smart homes. The case study had the following number of features and simple policies as presented in Table 10.

This section evaluates and discusses the obtained results using the three measures of *Suitability, Accuracy,* and *Reduction.* Suitability measures how suitable the approach was for detecting interactions between the 35 simple policies. This procedure is most beneficial for systems that have many policies describing the dynamic behavior versus a few policies describing system axioms as in such a case there is a significant reduction of comparisons (see below). Since the smart home case study had only 6 system axioms versus 29 dynamic behavior simple policies, the procedure was very suitable for this particular case study. Moreover, the tool support described in the next section made the application of the procedure easier.

Accuracy shows how precise the process of detecting interactions was and if any feature interactions were missed. Unfortunately, there are no fully documented results in the literature with which we could have compared our results. However, Kolberg et al. in [38] lists an overview of the interactions that arise between services supporting networked appliances in a smart home environment. This overview included some interaction examples. All interaction examples mentioned in [38] were detected using IRIS. Moreover, to support the accuracy claim of our case study, IRIS was successfully applied on different other case studies for which there were results reported by other researchers. Our previous work in [39] lists these case studies and compares the results obtained through the application of IRIS with those obtained by other researchers. IRIS was among those approaches that detected the highest number of interactions.

Table 9
Summary of interaction detection results

Policy	Interacting policies	Policy	Interacting policies
P1.1	P1.2, P1.3, P1.4, P1.5, P3.2, P10.1, P12.1	P1.2	P3.2, P10.1, P12.1
P1.3	P3.2, P3.3, P12.1	P1.4	P3.2, P8.2, P12.1
P1.5	P3.2, P8.2, P12.1	P2.1	P2.2, P2.3, P10.1, P12.1
P2.2	P4.1, P4.2, P5.2, P9.1, P9.2, P10.1, P12.1	P2.3	P6.1, P6.2, P8.1, P8.2, P8.3, P8.4, P9.1, P9.2, P10.1, P12.1
P3.1	P3.3, P12.1	P3.2	P6.1, P6.2, P12.1
P3.3	P6.1, P6.2, P12.1	P4.1	P4.2, P5.1, P5.2, P12.1
P4.2	P5.2, P12.1	P5.1	P5.2, P12.1
P5.2	P12.1, P16.1	P6.1	P6.2, P10.1, P12.1
P6.2	P10.1, P12.1	P7.1	Nothing
P7.2	Nothing	P8.1	P8.2, P8.4, P12.1
P8.2	P12.1	P8.3	P12.1
P8.4	P12.1	P9.1	P9.2, P12.1
P9.2	P12.1	P10.1	P12.1
P11.1	P12.1	P12.1	P13.1, P13.2, P14.1, P14.2, P15.1, P16.1
P13.1	Nothing	P13.2	Nothing
P14.1	P16.1	P14.2	P16.1
P15.1	P16.1		

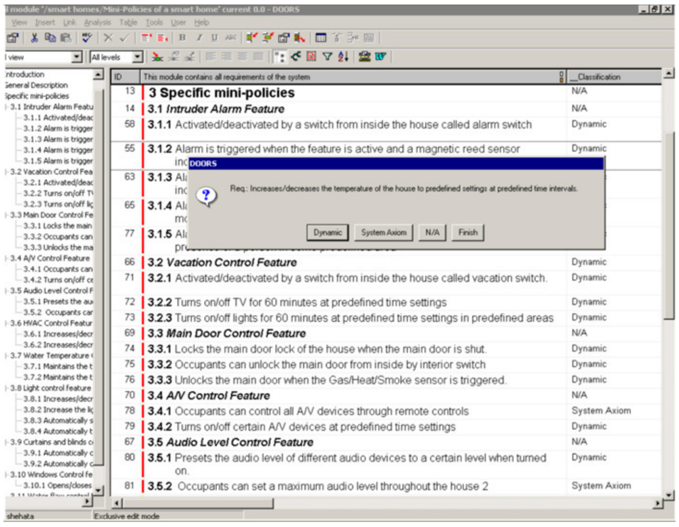
Table 10
Statistics on the smart homes case study

Number of features	16
Number of simple policies	35
Number of detected interactions using IRIS	83

Reduction is a measure that indicates how much the necessary number of pairwise comparisons could be reduced due to the application of IRIS. The number of comparisons that were done using IRIS compared to the number of comparisons that an expert would have to do was 525 comparisons as opposed to 630. Thus we have achieved a 16.7% reduction in the number of pairwise comparisons. Although this reduction cannot be directly translated into the same percentage reduction in time and effort due to the fact that the application of IRIS will cause an overhead in time and effort, but we claim that IRIS, as a structured approach, is likely to increase the number of detected interactions. Hopefully, the increase in the number of detected interactions will compensate for the additional time and effort overhead of applying IRIS.

5. Tool support

Tool support is required to make the application of IRIS easier. Especially the transfer of information from one table to another, the extraction of the triggering events and the graphical representation of step 5 could greatly benefit from tool support. It was therefore decided to build an extension to the DOORS requirements management tool [40] from Telelogic. This would allow developers who have their specification developed with the DOORS tool to easily apply the six steps of IRIS. DOORS was chosen because of its wide use in the industry. It offers an application programming interface (API) that can be programmed using the DOORS Extension Language (DXL) [41]. This language gives the programmer access to the internal DOORS functionality and allows scripts to be written. DXL is very similar to the C-language but has some additional commands to accommodate the DOORS environment.

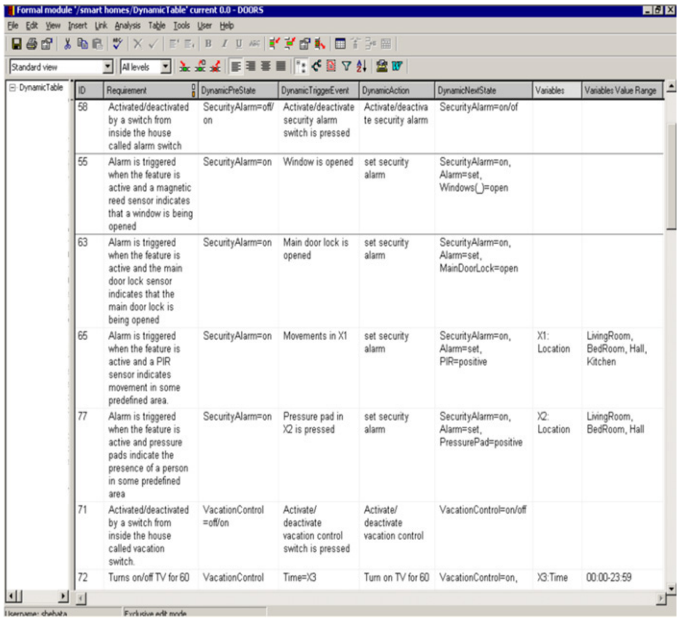


ID	This module contains all requirements of the system	_Classification
13	**3 Specific mini-policies**	N/A
14	**3.1 *Intruder Alarm Feature***	N/A
58	**3.1.1** Activated/deactivated by a switch from inside the house called alarm switch	Dynamic
55	**3.1.2** Alarm is triggered when the feature is active and a magnetic reed sensor inc	Dynamic
63	**3.1.3** Al inc	
65	**3.1.4** Al mc	
77	**3.1.5** Al pr	
66	**3.2 *Vacation Control Feature***	Dynamic
71	**3.2.1** Activated/deactivated by a switch from inside the house called vacation switch.	Dynamic
72	**3.2.2** Turns on/off TV for 60 minutes at predefined time settings	Dynamic
73	**3.2.3** Turns on/off lights for 60 minutes at predefined time settings in predefined areas	Dynamic
69	**3.3 *Main Door Control Feature***	N/A
74	**3.3.1** Locks the main door lock of the house when the main door is shut.	Dynamic
75	**3.3.2** Occupants can unlock the main door from inside by interior switch	Dynamic
76	**3.3.3** Unlocks the main door when the Gas/Heat/Smoke sensor is triggered.	Dynamic
70	**3.4 *A/V Control Feature***	N/A
78	**3.4.1** Occupants can control all A/V devices through remote controls	System Axiom
79	**3.4.2** Turns on/off certain A/V devices at predefined time settings	Dynamic
67	**3.5 *Audio Level Control Feature***	N/A
80	**3.5.1** Presets the audio level of different audio devices to a certain level when turned on.	Dynamic
81	**3.5.2** Occupants can set a maximum audio level throughout the house 2	System Axiom

DOORS

Req.: Increases/decreases the temperature of the house to predefined settings at predefined time intervals.

Fig. 7. A screen shot for the interactive screen for classifying policies.

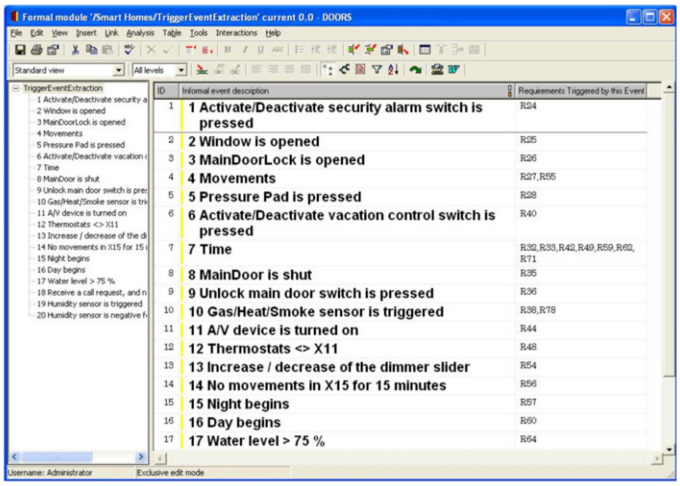


Formal module '/smart homes/DynamicTable' current 0.0 - DOORS

ID	Requirement	DynamicPreState	DynamicTriggerEvent	DynamicAction	DynamicNextState	Variables	Variables Value Range
58	Activated/deactivated by a switch from inside the house called alarm switch	SecurityAlarm=off/on	Activate/deactivate security alarm switch is pressed	Activate/deactivate security alarm	SecurityAlarm=on/of		
55	Alarm is triggered when the feature is active and a magnetic reed sensor indicates that a window is being opened	SecurityAlarm=on	Window is opened	set security alarm	SecurityAlarm=on, Alarm=set, Windows(_)=open		
63	Alarm is triggered when the feature is active and the main door lock sensor indicates that the main door lock is being opened	SecurityAlarm=on	Main door lock is opened	set security alarm	SecurityAlarm=on, Alarm=set, MainDoorLock=open		
65	Alarm is triggered when the feature is active and a PIR sensor indicates movement in some predefined area.	SecurityAlarm=on	Movements in X1	set security alarm	SecurityAlarm=on, Alarm=set, PIR=positive	X1: Location	LivingRoom, BedRoom, Hall, Kitchen
77	Alarm is triggered when the feature is active and pressure pads indicate the presence of a person in some predefined area	SecurityAlarm=on	Pressure pad in X2 is pressed	set security alarm	SecurityAlarm=on, Alarm=set, PressurePad=positive	X2: Location	LivingRoom, BedRoom, Hall
71	Activated/deactivated by a switch from inside the house called vacation switch.	VacationControl =off/on	Activate/ deactivate vacation control switch is pressed	Activate/ deactivate vacation control	VacationControl=on/off		
72	Turns on/off TV for 60	VacationControl	Time=X3	Turn on TV for 60	VacationControl=on,	X3:Time	00:00-23:59

Fig. 8. A screen shot for the created dynamic behavior table.

The current prototype of the tool allows the automation of the six steps as described in Section 4.4. It gradually goes from the first step to the sixth step with user interactive screens while creating the required tables and graphs. The tool support starts with the first step of IRIS by automatically extracting the relevant requirements from all the text stored in the project and then displays them one by one asking the user to classify each one as shown in Fig. 7. Then the tool creates a table that has the classification of the requirements. The second step of IRIS is carried out in the tool support by automatically extracting the requirements and asking the user to input the values for the attributes corresponding to each requirement (e.g., Rule and Condition in the case of a system axiom and Pre-state, Trigger Event, Action, and Next state in the case of a dynamic requirement). Once finished, the tool support automatically creates tables and stores all the data collected from the user in their perspective places (see Fig. 8). The third step


Formal module '/Smart Homes/TriggerEventExtraction' current 0.0 - DOORS
File Edit View Insert Link Analysis Table Tools Interactions Help
Standard view
All levels
TriggerEventExtraction
1 Activate/Deactivate security a
2 Window is opened
3 MainDoorLock is opened
4 Movements
5 Pressure Pad is pressed
6 Activate/Deactivate vacation
7 Time
8 MainDoor is shut
9 Unlock main door switch is pres
10 Gas/Heat/Smoke sensor is tri
11 A/V device is turned on
12 Thermostats <> X11
13 Increase / decrease of the di
14 No movements in X15 for 15
15 Night begins
16 Day begins
17 Water level > 75 %
18 Receive a call request, and n
19 Humidity sensor is triggered
20 Humidity sensor is negative f
Username: Administrator
Exclusive edit mode

ID	Informal event description	Requirements Triggered by this Event
1	**1 Activate/Deactivate security alarm switch is pressed**	R24
2	**2 Window is opened**	R25
3	**3 MainDoorLock is opened**	R26
4	**4 Movements**	R27,R55
5	**5 Pressure Pad is pressed**	R28
6	**6 Activate/Deactivate vacation control switch is pressed**	R40
7	**7 Time**	R32,R33,R42,R49,R59,R62, R71
8	**8 MainDoor is shut**	R35
9	**9 Unlock main door switch is pressed**	R36
10	**10 Gas/Heat/Smoke sensor is triggered**	R38,R78
11	**11 A/V device is turned on**	R44
12	**12 Thermostats <> X11**	R48
13	**13 Increase / decrease of the dimmer slider**	R54
14	**14 No movements in X15 for 15 minutes**	R56
15	**15 Night begins**	R57
16	**16 Day begins**	R60
17	**17 Water level > 75 %**	R64

Fig. 9. A screen shot for Trigger events extraction.

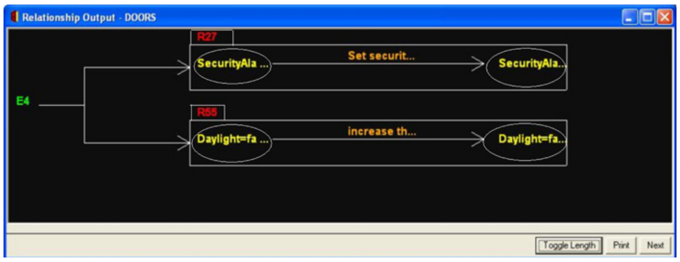


Fig. 10. A screen shot for the created graphical representation.

is carried out automatically where the tool support extracts unique trigger events and creates a table to store these trigger events along with the requirements they trigger as seen in Fig. 9. The linked events extraction is done by the tool support displaying each event separately along with a list of other events. The user will be prompted to select if the event under consideration is linked to another event. Once finished, a table is automatically created that has linked events information. The graphical representations are then automatically generated for all dynamic requirements as shown in Fig. 10. This step will include generating all graphical representation for all requirements triggered by same events. Finally the tool support asks the user to detect interactions based on the provided set of interaction detection guidelines.

The tool support has proven to be an important aid when performing the smart home case study. With the help of such a tool the process of detecting interactions has become much easier and faster. The full details of how the tool support was created in DOORS can be found in [46].

6. Related work

The policy research community has recognized that there are interaction issues between policies and has referred to it as policy conflicts. However, so far there has been very little research done to address the problem of policy conflicts. For example, [19] defined policies in a hierarchical way to prevent policy conflicts. But even with such an approach, if a policy in the hierarchy changes then policies can still conflict. Sloman et al. [20,21] promote the use of meta-policies, i.e., policies about creating policies, as a way to prevent conflicts. The work in [22] acknowledges the inevitability of policy conflicts and suggests a negotiation approach for their resolution. The work in [23] describes the use of policies in the telecommunications domain. It suggested the use of a feature interaction manager where policies are used to control the composition of services and features in telephony features and therefore avoiding the problem of feature interactions. The work in [24] proposed a policy architecture for enhancing telephony features and even promoted the use of policies as the features of the future.

Most of the work done so far have not comprehensively addressed the problem of policy interactions. For example, the work in [23] and [24] has been limited towards the use of policies in the traditional telecommunications domain. The work in [20] and [22] only look at the prevention of policy interactions and not their detection. However, so far no prevention technique can guarantee that no interactions will occur. Furthermore, there has not been so far a precise definition of when two policies are considered interacting. Even though policies are heavily used in defining user preferences in smart homes, there is so far no work done on investigating the policy interactions in this domain.

Some other recent research efforts have tried to address the problem of policy conflicts. Bandara et al. in [42] proposed the use of event calculus to transform both policy and system behaviour into formal notation and hence be able to use a priori analysis to detect conflicts between policies. However, this work uses formal notation and it is not clear if it can be easily applied in the smart homes domain. In [43], Blair and Turner uses an architecture and its realization for distributed and hierarchical policies within the telecommunications domain. However, this work is specific to the telecommunication domain and there were no indication of its applicability in other domains.

From the industry perspective, there has been several attempts by companies to build smart homes technologies such as the Philips ambient intelligence in HomeLab [44]. This homelab is designed to serve as a prototype for novel technologies that require interconnection between home systems and hence provide smart behaviour of a home to its occupants. Another example is the Aware Home Research Initiative (AHRI) [45] which is an interdisciplinary research endeavor at the Georgia Institute of Technology that addresses challenges facing the future of technologies in building an intelligent home aware of the needs of its occupants. The purpose for the Aware Home Research Initiative is to investigate what kinds of services can be built on top of an environment that is aware of the activities of its occupants especially elder people who are in need of assistive technology to lead a normal life.

7. Conclusions

This paper proposed the use of a semi-formal approach, IRIS, to detect interactions in any domain. A comprehensive view of the integration of policies and features was presented. A case study was carried out for detecting interactions among policies in smart homes using the proposed semi-formal approach and was presented in this paper. The proposed approach was successfully customized and applied in the smart homes domain. It was able to discover 83 interactions among 35 user policies using only 525 pairwise comparisons as apposed to 630 a human expert would have to do. These results support the paper's main claim of being able to use our semi-formal approach, IRIS, successfully to detect interactions between policies. Further, these results serve as the first fully documented results of interactions between policies in the smart homes domain.

Appendix A

This appendix presents all the detected interactions in the proposed case study. The aim is to provide clarification on how each interaction was detected and an example scenario of interaction for illustration purposes. Table A.1 uses the following abbreviations:

- ID: A unique interaction ID for each interaction
- Policies: Identify the two simple policies that interact
- Analysis: Lists what analysis procedure was used to detect the interaction
- Interaction: Lists an example scenario of a possible interaction, and suggests a resolution. The scenario listed might not be the only possible interaction scenario and there might be other situations in which the two policies interact. Similarly, the solution is only a suggestion and the manufacturer and occupants might prefer other solutions.

The procedure described in Section 4.4.6 explains in details how interactions are detected. This procedure was also used to detect the interactions listed in this appendix. Equal priorities of all simple policies were assumed during the detection process.

Table A.1
Detected interactions and suggested solution in the smart homes domain using IRIS

ID	Policies	Analysis	Interaction
I1	P1.1, P1.2	Linked events E1, E2	**Type:** The action of P1.1 overrides the action of P1.2. **Scenario:** A thief opens the window and once he is in, he quickly deactivates the alarm using the alarm switch thus making the alarm appear as a system glitch. **Solution:** Freeze the security alarm control panel as soon as alarm is triggered until a PIN is provided.
I2	P1.1, P1.3	Linked events E1, E3	**Type:** The action of P1.1 overrides the action of P1.3. **Scenario:** A thief opens the door and once he is in, he quickly deactivates the alarm using the alarm switch thus making the alarm appear as a system glitch. **Solution:** Freeze the security alarm control panel when alarm triggers until a PIN is provided.
I3	P1.1, P1.4	Linked events E1, E4	**Type:** The action of P1.1 overrides the action of P1.4. **Scenario:** A thief is in the house and once he sees the PIR, he quickly deactivates the alarm using the alarm switch thus making the alarm appear as a system glitch. **Solution:** Freeze the security alarm control panel when alarm triggers until a PIN is provided.
I4	P1.1, P1.5	Linked events E1, E5	**Type:** The action of P1.1 overrides the action of P1.5. **Scenario:** A thief is in the house, once he feels the alarm is triggered by the pressure pads, he quickly deactivates the alarm using the alarm switch thus making the alarm appear as a system glitch. **Solution:** Freeze the security alarm control panel when alarm triggers until a PIN is provided.
I5	P1.1, P3.2	Linked events E1, E9	**Type:** The action of P3.2 overrides the action P1.1. **Scenario:** P1.1 activates the security alarm to secure the house, while P3.2 can still override P1.1 and the door can be opened. For example, if a thief is inside the house and wants to get out, he can just press the open door switch to get out. Another example, if an occupant opens the doors using P3.2 then it will falsely trigger the alarm. **Solution**: If alarm is activated using P1.1, a PIN is required before executing P3.2.
I6	P1.1, P10.1	Linked events E1, E7	**Type:** The action of P10.1 negatively impacts the action of P1.1. **Scenario:** Occupant schedules windows to open in a certain place at a certain time and while this action is executing, occupant without knowing that windows are opening, activates alarm and hence falsely triggers the alarm. **Solution**: Ask the occupant to secure open doors and windows before executing P1.1.
I7	P1.1, P12.1	Linked events E1, E18	**Type:** The action of P12.1 negatively impacts the action of P1.1. **Scenario:** Occupant A comes home and deactivates alarm. Occupant B at work can’t remember if he activated alarm or not, so he calls and activates alarm using remote access module. Occupant A opens a window and alarm triggers. **Solution:** Information about the last deactivation of the security alarm is provided over the phone to the person who wants to use the remote access module to set the alarm.
I8	P1.2, P3.2	Linked events E2, E9	**Type:** The action of P3.2 overrides the action of P1.2. **Scenario:** An occupant unlocks the main door using P3.2 while the alarm system is active. If this disables the alarm, to avoid a false trigger, and the main door is opened, then a thief could open a window in the basement at the same time and the alarm will not be triggered. Thus P1.2 action is overridden by P3.2 action. **Solution**: When P3.2 opens the door, then only the door is excluded from triggering the alarm.
I9	P1.2, P10.1	Linked events E2, E7	**Type:** The action P10.1 negatively impacts the action of P1.2. **Scenario:** If the occupant sets a time when the windows open automatically while the security alarm is active then the action of P10.1 will falsely trigger the alarm. **Solution:** The occupant is notified to cancel scheduled windows opening times before alarm is allowed to be active.
I10	P1.2, P12.1	Linked events E2, E18	**Type:** The action of P12.1 overrides the action of P2.1. **Scenario:** A thief breaks into the house through a window. One second later, an occupant uses the remote access module to cancel the alarm as he is on his way home. The alarm trigger looks like a system glitch and thief escapes. **Solution:** If the alarm is triggered then the remote access module cannot disable the alarm system.

Table A.1 *(continued)*

ID	Policies	Analysis	Interaction
I11	P1.3, P3.2	Linked events E3, E9	**Type:** The action of P3.2 negatively impacts the action of P1.3. **Scenario:** Occupants open the main door using the interior main door switch while the alarm is active. This falsely triggers an alarm. **Solution:** The system shall ask for a PIN before opening the door if alarm is active.
I12	P1.3, P3.3	Linked events E3, E10	**Type:** The action of P3.3 negatively impacts the action of P1.3. **Scenario:** Steam and smoke from cooking causes the G/H/S detector to trigger while the alarm is active. The system opens the main door according to P3.3. Thus the intruder alarm triggers falsely. **Solution:** Intruder alarm is disabled when the G/H/S detector triggers.
I13	P1.3, P12.1	Linked events E3, E18	**Type:** The action of P12.1 overrides the action of P1.3. **Scenario:** A thief breaks into the house through the door and the alarm is triggered. One second later, an occupant uses the remote access module to deactivate the alarm as he is on his way home. Then the thief can get away as the alarm trigger looks like a system glitch. **Solution:** If alarm is triggered then the remote access module cannot disable the security alarm.
I14	P1.4, P3.2	Linked events E4, E9	**Type:** The action of P3.2 overrides the action of P1.4. **Scenario:** An occupant unlocks the main door using P3.2 while the alarm system is active. To avoid false triggering of alarm, the system disables the alarm then opens main door. If during this time a thief breaks in from the upper floor and his movements are detected by a PIR, then alarm will not be triggered because it is temporarily disabled. **Solution:** When P3.2 opens the main door, then only the main door is excluded from triggering the alarm.
I15	P1.4, P8.2	Same trigger event E4	**Type:** The action of P8.2 negatively impacts the action of P1.4. **Scenario:** An occupant sets the same part of the house (e.g. hallway) for both lights (in P8.2) and security (P1.4) to check for a positive PIR signal. The occupant gets up at night and the PIR sensor detects his movements thus the system activates lights but also activates the alarm which in this case is triggered falsely by the occupant. **Solution:** Don not allow occupants to set the same area for lights increase and security check.
I16	P1.4, P12.1	Linked events E4, E18	**Type:** The action of P12.1 overrides the action of P1.4. **Scenario:** A thief breaks into the house and triggers the alarm by a positive PIR signal. But an occupant uses the remote access module to deactivate the alarm as he is on his way home. Then the thief can get away as the alarm trigger looks like a system glitch. **Solution:** If alarm is triggered then the remote access module cannot disable the security alarm.
I17	P1.5, P3.2	Linked events E5, E9	**Type:** The action of P3.2 overrides the action of P1.5. **Scenario:** An occupant unlocks the main door using P3.2 while the alarm system is active. To avoid false triggering of alarm, the system disables alarm then opens main door. If during this time a thief breaks in from upper floor and his movements are detected by pressure pads, then alarm will not be triggered because it is temporarily disabled. **Solution:** When P3.2 opens the main door, then only the main door is excluded from triggering the alarm.
I18	P1.5, P8.2	Linked events E4, E5	**Type:** The action of P8.2 negatively impacts the action of P1.4. **Scenario:** An occupant sets the same part of the house (e.g. hallway) for both lights and security to check for PIR and pressure pads signals. The occupant gets up at night and activates lights by PIR but also walks on the pressure pads in the hallway thus triggers alarm by himself falsely. **Solution:** Do not allow occupant to set the same area for lights increase and pressure pads security check.
I19	P1.5, P12.1	Linked events E5, E18	**Type:** The action of P12.1 overrides the action of P1.5 **Scenario:** A thief breaks into the house and triggers the alarm by the pressure pads. But an occupant uses the remote access module to deactivate the alarm as he is on his way home. Then the thief can get away as the alarm trigger looks like a system glitch. **Solution:** If alarm is triggered then the remote access module cannot disable the security alarm.
I20	P2.1, P2.2	Linked events E6, E7	**Type:** The action of P2.1 overrides the action of P2.2. **Scenario:** If an occupant presses the deactivate switch while P2.2 is executing, then P2.1 will cancel the action of P2.2 before completion. Note that both functionalities are of equal priorities. This might be important if a child is the one who deactivated the vacation control. **Solution:** P2.1 is of higher priority and a PIN is required before the actual deactivation of the vacation control.

(continued on next page)

Table A.1 *(continued)*

ID	Policies	Analysis	Interaction
I21	P2.1, P2.3	Linked events E6, E7	**Type:** The action of P2.1 overrides the action of P2.3. **Scenario:** If an occupant presses the deactivate switch while P2.3 is executing, then P2.1 will cancel the action of P2.3 before completion. Note that both functionalities are of equal priorities. This might be important if a child is the one who deactivated the vacation control. **Solution:** P2.1 is of higher priority and a PIN is required before the actual deactivation of the vacation control.
I22	P2.1, P10.1	Linked events E6, E7	**Type:** P10.1 negatively impacts the action of P2.1. **Scenario**: P10.1 is a security hole that negatively impacts the intended purpose of P2.1 which is to keep the house safe during extended periods of absence. For example, an occupant activates vacation control (P2.1) as a protection of the house while away but P10.1 can still open windows and thus negatively impact the intended purpose of P2.1 **Solution:** Ask the occupant to cancel scheduled window opening before activating vacation control.
I23	P2.2, P4.1	P2.2 interacts with system axiom P4.1	**Type:** The rule of P4.1 overrides the action of P2.2 **Scenario:** The vacation control is on and P2.2 is executing. The occupant comes home and forgets to turn the vacation control off. The occupant then tries to use the remote control to turn the TV off. If the TV is turned off, then the rule P4.1 overrode the action of P2.2. If not then P4.1 is violated by P2.2. **Solution:** The occupant must turn the vacation control off first before being able to use the remote control.
I24	P2.2, P4.2	Same trigger event E7	**Type:** The action of P4.2 overrides the action of P2.2. **Scenario:** An occupant sets the action of P2.2 to turn on the TV at time X for 60 min while the action of P4.2 was set to turn off the TV at the same time X. A similar scenario can also occur if the defined times overlap. **Solution:** Manual settings by occupants for TV, such as P4.2, are cancelled when the vacation control is active.
I25	P2.2, P5.2	P2.2 interacts with system axiom P5.2	**Type:** The action of P2.2 overrides the rule of P5.2. **Scenario:** An occupant sets the TV to Y volume. The next day he lowers the max audio level in P5.2 below Y. Then he activates the vacation control (thus activating P2.2) and leaves. When P2.2 starts the TV with the last setting of volume which is Y, it violates P5.2. **Solution:** The vacation control starts the TV with a volume below the allowed max audio level.
I26	P2.2, P9.1	Same trigger event E7	**Type:** The action of P9.1 negatively impacts the action of P2.2. **Scenario:** P9.1 will be a security hole that negatively impacts the intended purpose of P2.2. For example, if the predefined areas of curtain and blinds are the same as the location of TV and the predefined time is the same for P2.2 and P9.1, then this will enable passers-by to see that no one is watching TV. **Solution:** Disable the opening of curtains and blinds when the vacation control is opening the TV. However when the TV is not on then curtains and blinds can be opened/closed to give impression that occupants are home.
I27	P2.2, P9.2	Linked events E7, E15	**Type:** The action of P9.2 negatively impacts the action of P2.2. **Scenario:** P9.2 will be a security hole that negatively impacts the intended purpose of P2.2. For example, if the predefined area of curtains and blinds are the same as the location of TV and the predefined time is the same for P2.2 and P9.1, then this will let anyone looking from windows know that no one home watching TV **Solution:** Disable the opening of curtains and blinds when the vacation control is opening the TV.
I28	P2.2, P10.1	Same trigger event E7	**Type:** The action of P10.1 negatively impacts the action of P2.2. **Scenario:** The action of P10.1 will be a security hole that counteracts the intended purpose of vacation control P2.2 which is to keep the house safe during extended periods of absence. For example, an occupant activates vacation control (P2.2) and windows control (P10.1) then leaves. **Solution:** Disable the opening of windows when the vacation control is activated.
I29	P2.2, P12.1	Linked events E7, E18	**Type:** The action of P12.1 overrides the action of P2.2 **Scenario**: Occupant A activates vacation control (and thus P2.2) and leaves. Occupant B calls and use remote access module (P12.1) to turn off the TV or even cut off the power to it. Hence overriding the action of P2.2 **Solution:** Remote access module cannot control TV when vacation control is active

Table A.1 *(continued)*

ID	Policies	Analysis	Interaction
I30	P2.3, P6.1	Linked events E7, E12	**Type:** The action of P2.3 negatively impacts the action of P6.1. **Scenario:** An occupant chooses low Temperature for P6.1 and also chooses to turn on all lights with high watts. The heat radiated from the light bulbs swill affect the decrease of temperature and requires more power for the HVAC. **Solution:** With low temperature settings, use medium light intensity or low power lamps.
I31	P2.3, P6.2	Same trigger event E7	**Type:** The action of P2.3 negatively impacts the action of P6.2. **Scenario:** An occupant chooses low Temperature for P6.2 and also chooses to turn on all lights with high watts. The heat radiated will affect the decrease of temperature and requires more power for the HVAC. **Solution:** With low temperature settings, use medium light intensity or low power lamps.
I32	P2.3, P8.1	Linked events E7,E13	**Type:** The action of P8.1 overrides the action of P2.3. **Scenario:** Vacation control is on and occupant comes home. P2.3 has finished (after 60 min) and starts shutting off lights while occupant uses light dimmer to increase the light intensity and the lights might not respond to this request. **Solution:** Occupant must turn the vacation control off once s/he enters the house before getting control over lights.
I33	P2.3, P8.2	Linked events E4, E7	**Type:** The action of P2.3 overrides the action of P8.2. **Scenario:** The occupant comes home and forgets to deactivate vacation control (P2.3). At night P2.3 triggers and switches on the lights for 59 min. Then occupant gets up to go to bathroom hence triggering P8.2 which will increase light to the max over 2 min. But P2.3, after first minute, shuts down lights because the 60 min have elapsed. **Solution:** Occupant must turn off the vacation control once s/he gets home before other policies are active again.
I34	P2.3, P8.3	Linked events E7, E14	**Type:** The action of P8.3 overrides the action of P2.3. **Scenario:** An occupant has both P2.3 and P8.3 active. P2.3 triggers and switches the lights on. After 15 min P8.2 switches off lights because no one is home. Thus P8.3 switches off lights after 15 min (not 60 min as in P2.3). **Solution:** Disable other light control policies when vacation control is activated.
I35	P2.3, P8.4	Linked events E7, E15	**Type:** The action of P8.4 overrides the action of P2.3. **Scenario:** P2.3 is triggered and switches on the lights for 60 min. At the end of the 60 min, the system starts shutting off the lights. At the same instant, the night begins and P8.4 starts to open lights while they are being shut off. Therefore the lights are not shut off and P3.2 action is not completed **Solution:** Disable other light control policies when vacation control is activated.
I36	P2.3, P9.1	Same trigger event E7	**Type:** The action of P9.1 negatively impacts the action of P2.3. **Scenario:** P9.1 will be a security hole that negatively impacts the intended purpose of P2.3. For example, if the predefined area is the same for curtains and lights and the predefined time is also the same for P2.3 and P9.1, then anyone looking from windows know that no one home opening/closing the lights. **Solution:** Close curtains/blinds during times when vacation control is active and lights are about to be turned on/off.
I37	P2.3, P9.2	Linked events E7, E15	**Type:** The action of P9.2 negatively impacts the action of P2.3. **Scenario:** P9.2 will be a security hole that negatively impacts the intended purpose of P2.3. For example, If the predefined area is the same for curtains and lights and the predefined time is also the same for P2.3 and P9.2. Then anyone looking from windows know that no one home opening/closing the lights. **Solution:** Close curtains and blinds during times when vacation control is active and the lights are about to be turned on/off. However, curtains and blinds can be opened during other times to give impression that occupants are home.
I38	P2.1, P12.1	Linked events E6, E18	**Type:** The action of P12.1 overrides the action of P2.1 **Scenario:** Occupant A deactivates vacation control when she comes home. Occupant B is away and cannot remember if he activated the vacation control or not, so he calls and uses remote access module (P12.1) to turn on vacation control. Occupant A at home looses control over lights and TV. **Solution:** State last activation/deactivation information of security alarm over the phone to the person who wants to use the remote access module to set the alarm.

(continued on next page)

Table A.1 *(continued)*

ID	Policies	Analysis	Interaction
I39	P2.3, P10.1	Same trigger event E7	**Type:** The action of P10.1 negatively impacts the action of P2.3. **Scenario:** P 10.1 will be a security hole that counteracts the intended purpose of vacation control P2.3 which is to keep the house safe while extended periods of absence. For example, an occupant activates vacation control (P2.3) and windows control (P10.1) then leaves. **Solution:** Disable the opening of windows when the vacation control is activated.
I40	P2.3, P12.1	Linked events E7, E18	**Type:** The action of P12.1 overrides the action of P2.3. **Scenario:** Occupant A activates vacation control (and thus P2.3) and leaves. Occupant B calls and uses the remote access module (P12.1) to turn off all lights in the home, hence overriding the action of P2.3. **Solution:** Remote access module cannot control lights when vacation control is active.
I41	P3.1, P3.3	Linked events E8, E10	**Type:** The action of P3.3 overrides the action of P3.1. **Scenario:** The only occupant at home shuts the main door expecting it to lock automatically according to P3.1 and leaves. One minute later, G/H/S triggers and opens door according to P3.3, thus leaving house vulnerable to anyone. **Solution:** Only open the main door when there is someone inside (movements can be detected using PIR).
I42	P3.1, P12.1	Linked events E8, E18	**Type:** The action of P12.1 is used to override the action of P3.1. **Scenario:** Occupant A is leaving and shuts the doors behind her expecting it to lock automatically. While shutting, occupant B calls and opens the main door lock. Thus house is vulnerable. **Solution:** Critical parts of the house like the main door cannot be controlled by remote access module.
I43	P3.2, P6.1	Linked events E9, E12	**Type:** The action of P3.2 negatively impacts the action of P6.1. **Scenario:** When occupants use P3.2 to open the main door while P6.1 is triggered trying to raise the temperature of the house. The open door will affect the increase of temperature if left open for a long time. **Solution:** Close the door after some time units to maintain the temperature of the home.
I44	P3.2, P6.2	Linked events E7, E9	**Type:** The action of P3.2 negatively impacts the action of P6.2. **Scenario:** When occupants use P3.2 to open the main door while P6.2 is triggered trying to raise the temperature of the house. The open door will affect the increase of temperature if left open for a long time. **Solution:** Close the door after some time units to maintain the temperature of the home.
I45	P3.2, P12.1	Linked events E9, E18	**Type:** The action of P12.1 overrides the action of P3.2 **Scenario:** Occupant A presses unlock door interior switch to unlock door. Occupant B calls and uses remote access module to lock door, as he suspects it was left open. Occupant A tries to open the door but it does not respond. **Solution:** The interior switch has higher priority and still opens the house's main door lock.
I46	P3.3, P6.1	Linked events E10, E12	**Type:** The action P3.3 negatively impacts the action of P6.1. **Scenario:** P6.1 tries to raise the house's temperature. G/H/S is triggered by mistake (e.g. battery fault or short circuit) and opens the door according to P3.3. If left for a long time, it will affect the ability of P6.1 to increase temperature. **Solution:** Close door after some time units, but only if the G/H/S alarm has stopped.
I47	P3.3, P6.2	Linked events E7, E12	**Type:** The action P3.3 negatively impacts the action of P6.2. **Scenario:** P6.2 tries to raise the house's temperature. G/H/S is triggered by mistake (e.g. battery fault or short circuit) and opens the door according to P3.3. If left for long time, it will affect the ability of P6.2 to increase temperature. **Solution:** Close door after some time units, but only if the G/H/S alarm has stopped.
I48	P3.3, P12.1	Linked events E10, E18	**Type:** The action of P12.1 overrides the action of P3.3. **Scenario:** G/H/S triggers because of a fire and opens the main door and hence occupant A tries to get out. Occupant B calls and uses the remote access module to close the main door (as he suspects it might have accidentally been left open). Occupant A is then stuck inside. **Solution:** Remote access module is disabled in case of emergencies, such as fire.
I49	P4.1, P4.2	P4.2 interacts with system axiom P4.1	**Type:** The action of P4.2 overrides the rule of P4.1 **Scenario:** Occupant A uses the remote control to switch on the TV while at the same time P4.2 is scheduled by occupant B to turn off the TV. The TV shuts according to the action of P4.2 and hence P4.1 rule has been violated. **Solution:** Assign higher priority for the system axiom P4.1.

Table A.1 *(continued)*

ID	Policies	Analysis	Interaction
I50	P4.1, P5.1	P5.1 interacts with system axiom P4.1	**Type:** P4.1 rule overrides the action of P5.1 **Scenario:** Some A/V devices (like TVs) take several seconds before they are actually on. If during this time, an occupant uses the remote control to raise volume very high (using P4.1), then the device will have different volume setting when it is actually on than the preset sound level defined in P5.1. For example, a parent presets TV volume to start at volume X but a child increases volume (using P4.1) to the maximum before the actual start of the TV device. **Solution:** Assign higher priority for the action of P5.1.
I51	P4.1, P5.2	Two interacting system axiom P4.1, P5.2	**Type:** The rule of P4.1 overrides the rule of P5.2. **Scenario:** An occupant tries to use the remote control of an A/V device to go beyond the maximum preset audio level of the house. Note that this might be a multi-user environment like parents and children. **Solution:** Assign higher priority for the rule of P5.2.
I52	P4.1, P12.1	P12.1 interacts with system axiom P4.1	**Type:** The action of P12.1 overrides the rule of P4.1. **Scenario:** Occupant A uses the remote control to turn on the TV (using P4.1 rule). Occupant B calls and uses the remote access module (P12.1) to turn off all A/V devices (as he suspects he forgot to switch them off when he left). **Solution:** State last activation information over phone before turning off any A/V device.
I53	P4.2, P5.2	P4.2 interacts with system axiom P5.2	**Type:** The action of P4.2 overrides the rule of P5.2. **Scenario:** Occupant A (e.g. parent) uses P5.2 to set a relatively low maximum audio level for the house. Every family member uses P4.2 to turn on an A/V device at overlapping time settings. The combined volume of the several audio devices will exceed the max volume allowed for the home. **Solution:** Assign higher priority of P5.2 and do not allow combined volumes of A/V devices to exceed the max. limit.
I54	P4.2, P12.1	Linked events E7, E18	**Type:** The action of P12.1 overrides the action of P4.2. **Scenario:** Occupant A sets the VCR to turn on and record a show. Occupant B calls from work to completely shut down all A/V devices as he suspects that he left one of them on. This prevents P4.2 action from ever executing. **Solution:** State to user over phone if there are any A/V devices affected by power cut off or scheduled to work later.
I55	P5.1, P5.2	P5.1 interacts with system axiom P5.2	**Type:** The action of P5.1 overrides the rule of P5.2. **Scenario:** Occupant A (e.g. parent) sets the maximum audio level of the house to X. Occupant B (e.g. child) presets audio of TV and CD to levels that when combined (i.e. added to each other) will exceed max audio level of house X. **Solution:** Assign higher priority to P5.2 and do not allow combined volumes of A/V devices to exceed the max. limit.
I56	P5.1, P12.1	Linked events E11, E18	**Type:** The action of P12.1 overrides the action of P5.1. **Scenario:** Occupant A (e.g. parent) receives a phone call from a neighbor that TV is too loud. The parent calls and uses remote access module to lower volume of all A/V devices. At the same time, occupant B (e.g. child) turns on a CD recorder expecting a certain audio level, as specified in P5.1, but the parent has lowered volume using P12.1. **Solution:** State over the phone if there are A/V devices affected by lowering the volume. In all cases, a priority assignment is needed in case parent still proceed with lowering the volume.
I57	P5.2, P12.1	P12.1 interacts with system axiom P5.2	**Type:** The action of P12.1 overrides the rule of P5.2. **Scenario:** Occupant A uses P5.2 to set a low maximum audio level Y for the house. Occupant B calls and uses the remote access module to activate several A/V devices that have audio levels which when combined (i.e. added to each other) will be greater than Y. **Solution:** Assign higher priority for P5.2 and do not allow combined volumes of A/V devices to exceed the max limit.
I58	P5.2, P16.1	P16.1 interacts with system axiom P5.2	**Type:** The action of P16.1 overrides the rule of P5.2. **Scenario:** The already existing audio level of the house is almost at maximum. Occupant A activates various loud appliances like the food processor and blender. Although appliances are not A/V devices they increase the noise level of the house and violates the intended purpose of P5.2 which is to keep the house below a certain noise level. **Solution:** Reduce the volume of an A/V device to compensate for the additional noise of appliances.

(continued on next page)

Table A.1 *(continued)*

ID	Policies	Analysis	Interaction
I59	P6.1, P6.2	Linked events E7, E12	**Type:** Action of P6.2 overrides the action of P6.1. **Scenario:** Occupant A uses P6.1 to preset X as the temperature of the house for the whole day. Occupant B is not aware of the preset value of X and sets another temperature Y during day using P6.2. If X is different from Y then later one might override the prior temperature. **Solution:** Do not allow different temperature settings in overlapping time intervals.
I60	P6.1, P10.1	Linked events E7, E12	**Type:** The action of P10.1 negatively impacts the action of P6.1. **Scenario:** P10.1 opens the windows. If the outside temperature is low then the opened windows will negatively impact P6.1 and prevent the HVAC unit from keeping the room temperature at the predefined temperature setting. **Solution:** The system checks for the outside temperature and if the temperature affect the room temperature if the windows are open then the occupant is prompted to choose between either one of the policies.
I61	P6.1, P12.1	Linked events E12, E18	**Type:** The action of P12.1 overrides the action of P6.1. **Scenario:** The temperature inside the house gets low and P6.1 triggers. An occupant calls and uses the remote access module to shutdown the HVAC unit before completing its work. A similar scenario can occur when the occupant calls and uses the remote access module to open the windows. **Solution:** User is informed over phone if temperature is affected by the remote access module action.
I62	P6.2, P10.1	Same trigger event E7	**Type:** The action of P10.1 negatively impacts the action of P6.2. **Scenario:** P10.1 opens the windows. If the outside temperature is low then the opened windows will negatively impact P6.2 and prevent the HVAC unit from getting the room temperature to the predefined temperature setting. **Solution:** The system checks for the outside temperature and if the temperature affect the room temperature if the windows are open then the occupant is prompted to choose between either one of the policies.
I63	P6.2, P12.1	Linked events E7, E18	**Type:** The action of P12.1 overrides the action of P6.2. **Scenario:** P6.2 triggers to increase/decrease the temperature to the predefined settings. The occupant calls and uses the remote access module to shutdown the HVAC unit before the predefined temperature has been reached. Or the occupant calls and uses the remote access module to open the windows. **Solution:** The occupant is informed over the phone if the temperature is affected by the action of the remote access module and is asked to confirm the action.
I64	P8.1, P8.2	Linked events E4, E13	**Type:** The action of P8.1 overrides the action of P8.2. **Scenario:** An occupant wakes up at night and P8.2 triggers to increase the light to a maximum over 2 min. The occupant uses light dimmer to decrease the lights intensity (P8.1). Thus both are not able to execute at same time. **Solution:** Assign higher priority to manual light dimmer and terminate the action of P8.2 if light dimmer is used.
I65	P8.1, P8.4	Linked events E13, E15	**Type:** The action of P8.1 overrides the action of P8.4. **Scenario:** An occupant uses the light dimmer to decrease the light's intensity (P8.1) but at the same time the night starts and P8.4 tries to turn on and increase the intensity of lights to the specified maximum. **Solution:** Assign higher priority to manual light dimmer and terminate the action of P8.4 if light dimmer is used.
I66	P8.1, P12.1	Linked E13, E18	**Type:** The action of P12.1 overrides the action of P8.1. **Scenario:** Occupant A gets home and uses the light dimmer to increase the light intensity (P8.1). Occupant B calls and uses the remote access module to switch off all lights as he suspects he left them on. **Solution:** Assign higher priority to manual light dimmer and terminate the action of P12.1 if light dimmer is used.
I67	P8.2, P12.1	Linked events E4, E18	**Type:** The action of P12.1 is used to cancel the action of P8.2. **Scenario:** Occupant A gets up at night and thus triggering P8.2 to increase light intensity. Occupant B calls and uses the remote access module to switch off all lights as he suspects he left them on not knowing that occupant A is home **Solution:** Inform the user over the phone that someone is at home and ask for confirmation before executing actions.

Table A.1 *(continued)*

ID	Policies	Analysis	Interaction
I68	P8.3, P12.1	Linked events E14, E18	**Type:** The action of P8.3 overrides the action of P12.1 **Scenario:** An occupant uses P12.1 to switch on the lights in the garage just before his arrival. For some reason he is 15 min late (P8.3 has now switched off the lights) and when he gets into the garage the lights are off. **Solution:** The occupant is informed over the phone if he wants to double the time before P8.3 switches off the lights. Then the occupant can decide to accept or not.
I69	P8.4, P12.1	Linked events E15, E18	**Type:** The action of P12.1 overrides the action of P8.4. **Scenario:** Occupant A activates P8.4 when she gets home and takes a shower. Night begins and P8.4 switches on the lights including the bathroom light. Occupant B calls and uses the remote access module (P12.1) to switch off all lights not knowing that occupant A is home. **Solution:** Inform user over the phone that someone is at home and ask for confirmation before executing the actions.
I70	P9.1, P9.2	Linked events E7, E15	**Type:** The action of P9.1 overrides the action of P9.2. **Scenario:** Occupant A uses P9.1 to set the curtains/blinds to open at 6 pm (same time night starts). Occupant B sets curtains/blinds to close when night begins using P9.2. **Solution:** Assign higher priority to either one of them.
I71	P9.1, P12.1	Linked events E7, E18	**Type:** The action of P12.1 overrides the action of P9.1. **Scenario:** Occupant A sets the curtains to open at time X using P9.1. When time X comes, the curtains start opening. Occupant B calls and uses the remote access module to close curtains thus canceling action of P9.1. **Solution:** Inform user over phone of affected policies actions (P9.1) and ask for confirmation before execution.
I72	P9.2, P12.1	Linked events E15, E18	**Type:** The action of P12.1 overrides the action of P9.2 **Scenario:** Occupant A sets the curtains to open in the early morning using P9.2 and when the day begins the curtains open. Occupant B, who spent the night at work, calls and uses the remote access module to close curtains (as he suspects he might have left them open) thus canceling action of P9.2. **Solution:** Inform the user over phone of affected policies actions (P9.2) and that there is someone at home who had set a new policy that uses P9.2. Then user is asked for confirmation before proceeding to execute any commands.
I73	P10.1, P12.1	Linked events E7, E18	**Type:** The action of P12.1 overrides the action of P10.1. **Scenario:** Occupant A sets the windows to open at time X using P10.1. At time X, the windows starts opening. Occupant B calls and uses the remote access module to close all windows thus canceling action of P10.1. **Solution:** Inform the user over the phone of affected policies actions (P10.1) and that there is someone at home who set a new policy that uses P10.1. Then user is asked for confirmation before proceeding to execute any commands.
I74	P11.1, P12.1	Linked events E17, E18	**Type:** The action of P12.1 overrides the action of P11.1. **Scenario:** Occupant A takes a shower and leaves without tightly closing the water tap. The water starts filling the tub till it reaches 75%. P11.1 triggers and starts closing the water tap. Occupant B calls and uses the remote access module to open the water tap in shower for 10 min to fill the tub before his arrival and thus flooding bathroom. **Solution:** Assign higher priority to P11.1.
I75	P12.1, P13.1	P12.1 interacts with system axiom P13.1	**Type:** The action of P12.1 overrides the rule of P13.1. **Scenario:** When one occupant calls and uses the remote access module (P12.1) for a long time this violates the presence of a telephone line enforced by P13.1 as they both use the same telephone line. **Solution:** Do not allow extended use of the remote access module beyond a certain time limit.
I76	P12.1, P13.2	Same trigger event E18	**Type:** Next state non-determinism between P12.1 and P13.2. **Scenario:** The system will have a next state non-determinism if the occupant assigns the number of rings to activate the remote access module and the answer machine to be the same. The system does not know which next state it should transfer to: the answer machine or the remote access module. **Solution:** Assign higher priority to either one of them.

(continued on next page)

Table A.1 *(continued)*

ID	Policies	Analysis	Interaction
I77	P12.1, P14.1	Linked events E7, E18	**Type:** The action of P12.1 overrides the rule of P14.1. **Scenario:** A parent uses P14.1 to prevent any activation of the stove while s/he is out. A child uses a cell phone to call the home phone number and uses the remote access module to activate the stove. **Solution:** Assign higher priority to P14.1.
I78	P12.1, P14.2	Linked events E10, E18	**Type:** The action of P12.1 overrides the action of P14.2. **Scenario:** The G/H/S triggers and shuts down the stove to prevent any fire (P14.1). Occupant A .calls and uses the remote access module (P12.1) to turn on the oven to be heated until he gets home. **Solution:** Assign higher priority to P14.2.
I79	P12.1, P15.1	Linked events E18, E19	**Type:** The action of P12.1 overrides the action of P15.1. **Scenario:** Occupant A is cooking and the humidity sensor triggers and turns on the kitchen fan (P15.1). Occupant B calls, not knowing that occupant A is home, and uses the remote access module to shutdown all kitchen appliances including the kitchen fan as he suspects he might have accidentally left them on. **Solution:** Inform user over phone of affected policies actions (P15.1) and that there is someone at home.
I80	P12.1, P16.1	P12.1 interacts with system axiom P16.1	**Type:** The action of P12.1 overrides the action of P16.1. **Scenario:** Occupant A at home uses the remote control to run the food processor (P16.1). Occupant B, not knowing that occupant A is home, calls and uses the remote access module to switch off all kitchen appliances as he suspects he might has left something switched on. **Solution:** Inform the occupant over the phone of affected policies actions (P16.1) and that there is someone at home. Then the occupant is asked for confirmation before executing any commands.
I81	P14.1, P16.1	P14.1 interacts with system axiom P16.1	**Type:** The rule of P16.1 overrides the action of P14.1 **Scenario:** P16.1 enforces that any appliances including the stove can be controlled by the remote control. What happens if a parent uses P14.1 to switch off the stove while being away and a child finds the remote control and use it to turn on the stove? If the stove turns on then P14.1 was canceled; if not, then P16.1 was violated. **Solution:** Assign higher priority to P14.1.
I82	P14.2, P16.1	P14.2 interacts with system axiom P16.1	**Type:** The rule of P16.1 overrides the action of P14.2. **Scenario:** P16.1 enforces that any appliances including the stove can be controlled by the remote control. What happens if the G/H/S triggers (maybe falsely) P14.2 to switch off the stove and a child finds the remote control and insists on using it to turn on stove. If the stove turns on then P14.2 was overridden. If not, then P16.1 was violated. **Solution:** Assign higher priority to P14.2.
I83	P15.1, P16.1	P15.1 interacts with system axiom P16.1	**Type:** The action of P15.1 overrides the rule of P16.2. **Scenario:** The humidity sensor is triggered and P15.1 turns on kitchen fan while. At the same time, occupant uses remote control to switch off kitchen fan. The fan will switch off for a second but then turns on again because the humidity sensor is still triggered. P14.2 canceled remote control action although occupant wants to switch off the fan. **Solution:** Assign human control a higher priority.

References

[1] M. Calder, E. Magill, M. Kolberg, S. Reiff-Marganiec, Feature interaction: A critical review and considered forecast, Computer Networks 41 (1) (2003) 115–141.

[2] E.J. Cameron, et al., A feature interaction benchmark for IN and beyond, IEEE Communications Magazine (March) (1993) 64–67.

[3] D. Jackson, J. Wing, Lightweight formal methods, Computer (April) (1996) 21–22.

[4] P. Zave, Formal description of telecommunication services in Promela and Z, in: Proceedings of the Nineteenth International NATO Summer School, 1999.

[5] M. Plath, M.D. Ryan, The feature construct for SMV: Semantics, in: Sixth International Workshop on Feature Interactions in Telecommunications and Software Systems, IOS Press, 2000, 16 pp.

[6] Q. Fu, P. Harnois, L. Logrippo, J. Sincennes, Feature interaction detection: A LOTOS-based approach, Computer Networks 32 (4) (2000) 433–448.

[7] M. Shehata, A. Eberlein, Requirements interaction management: A multi-level framework, in: 6th IASTED International Conference on Software Engineering and Applications, MIT, USA, 2002.

[8] H. Velthuijsen, N. Griffeth, Y.-J. Lin (Eds.), International Workshop on Feature Interactions in Telecommunications Software Systems, IOS Press, Amsterdam, Florida, 1992.

[9] L.G. Bouma, H. Velthuijsen (Eds.), Feature Interactions in Telecommunications Systems, IOS Press, Amsterdam, 1994.

[10] K.E. Cheng, T. Ohta (Eds.), Feature Interactions in Telecommunications Systems III, IOS Press, Amsterdam, 1995.
[11] P. Dini, R. Boutaba, L. Logrippo (Eds.), Feature Interactions in Telecommunication Networks IV, IOS Press, Amsterdam, 1997.
[12] K. Kimbler, L.G. Bouma (Eds.), Feature Interactions in Telecommunications and Software Systems V, IOS Press, Amsterdam, 1998.
[13] M. Calder, E. Magill (Eds.), Feature Interactions in Telecommunications and Software Systems VI, IOS Press, Amsterdam, 2000.
[14] D. Amyot, L. Logrippo (Eds.), Feature Interactions in Telecommunications and Software Systems VII, IOS Press, Amsterdam, 2003.
[15] M. Sloman, J. Lobo, E.C. Lupu (Eds.), Proceedings of the 2nd International Workshop on Policies for Distributed Systems and Networks, in: Lecture Notes in Computer Science, Springer-Verlag, 2001.
[16] J.B. Michael, J. Lobo, N. Dulay (Eds.), Proc. 3rd. International Workshop on Policies for Distributed Systems and Networks, IEEE Computer Society, Los Alamitos, California, USA, June 2002.
[17] H.L. Lutfiyya, J. Moffett, F. Garcia (Eds.), Proceedings of 4th IEEE International Workshop on Policies for Distributed Systems and Networks, IEEE Computer Society, Italy, 2003.
[18] D. Verma, M. Devarakonda, E. Lupu, M. Kohli (Eds.), Proceedings of the 5th IEEE International Workshop on Policies for Distributed Systems and Networks, IEEE Computer Society, New York, USA, 2004.
[19] M. Amer, A. Karmouch, T. Gray, S. Mankovskii, Feature interaction resolution using fuzzy policies, in: M. Calder, E. Magill (Eds.), Feature Interactions in Telecommunications and Software Systems VI, IOS Press, Amsterdam, 2000, pp. 94–112.
[20] E.C. Lupu, M. Sloman, Conflicts in policy-based distributed systems management, IEEE Transactions on Software Engineering 25 (6) (1999) 852–869.
[21] N. Damianou, N. Dulay, E. Lupu, M. Sloman, Ponder: A language specifying security and managements policies for distributed systems, Research Report, Imperial College, London, 2000.
[22] G. Yee, L. Korba, Feature interactions in policy driven management, in: D. Amyot, L. Logrippo (Eds.), Feature Interactions in Telecommunications and Software Systems VII, IOS Press, Amsterdam, 2003, pp. 231–238.
[23] A. De Marco, F. Khendek, eSERL: Feature interaction in Parlay/OSA using composition constraints and configuration rules, in: D. Amyot, L. Logrippo (Eds.), Feature Interactions in Telecommunications and Software Systems VII, IOS Press, Amsterdam, June 2003, pp. 247–254.
[24] S. Reiff-Marganiec, K.J. Turner, A policy architecture for enhancing and controlling features, in: D. Amyot, L. Logrippo (Eds.), Feature Interactions in Telecommunications and Software Systems VII, IOS Press, Amsterdam, 2003, pp. 239–246.
[25] M. Heisel, J. Souquières, A heuristic algorithm to detect feature interactions in requirements, in: S. Gilmore, M. Ryan (Eds.), Language Constructs for Describing Features, Springer-Verlag, 2000.
[26] M. Shehata, A. Eberlein, Requirements interaction detection using semi-formal methods, in: ECBS 2003, 10th IEEE Symposium and Workshops on Engineering of Computer Based Systems Huntsville Alabama, USA, April 2003.
[27] M. Shehata, A. Eberlein, A. Fapojuwo, The use of semi-formal methods for detecting requirements interactions, in: SE 2004, The IASTED International Conference on Software Engineering, 17–19 February 2004, Innsbruck, Austria.
[28] G. Booch, J. Rumbaugh, I. Jacobson, The Unified Modeling Language User Guide, Addison-Wesley Inc, 1999, ISBN: 0201571864.
[29] D. Amyot, G. Mussbacher, URN: Towards a new standard for the visual description of requirements, in: Third SDL and MSC Workshop, SAM02, UK, in: LNCS, vol. 2599, June 2002.
[30] T. Demarco, P.J. Plauger, Structured Analysis and System Specification, Yourdon Press, Englewood Cliffs, NJ, 1978.
[31] E. Yourdon, Modern Structured Analysis, Yourdon Press, Englewood Cliffs, NJ, 1989.
[32] M. Edge, B. Taylor, G. Dewsbury, M. Groves, The Potential for Smart Home systems in meeting the care needs of older persons and people with disabilities, Seniors' Housing Update 10 (1) (2000).
[33] R.L. Smith, Smart House: The Coming Revolution in Housing, GP Publishing Inc, ISBN: 0876839189, 1988.
[34] D. Briere, P. Hurley, Smart Homes for Dummies, 2nd edition, Wiley Publishing Inc, ISBN: 0764525395, 2003.
[35] Lonworks, http://www.echelon.com/products/Core/default.htm, last viewed on July 13, 2004.
[36] Konnex, http://www.konnex.org/, last viewed on July 13, 2004.
[37] X10, http://www.x10.org/, last viewed on July 13, 2004.
[38] M. Kolberg, E.H. Magill, M. Wilson, Compatibility issues between services supporting networked appliances, IEEE Communications Magazine 41 (11) (2003).
[39] M. Shehata, A. Eberlein, A. Fapojuwo, IRIS: A semi formal approach for detecting requirements interactions, in: ECBS 2004, 11th IEEE International Conference and Workshop on the Engineering of Computer Based Systems, 24–27 May 2004, Brno, Czech Republic.
[40] Telelogic DOORS, http://www.telelogic.com/products/doorsers/doors/index.cfm/, last viewed on July 13, 2004.
[41] I. Alexander, Get More from DOORS with DXL Links, Telelogic News Byte Magazine, May 2001.
[42] A.K. Bandara, E.C. Lupu, A. Russo, Using event calculus to formalise policy specification and analysis, in: 4th IEEE Workshop on Policies for Networks and Distributed Systems (Policy 2003), Lake Como, Italy, 2003.
[43] L. Blair, K.J. Turner, Handling policy conflicts in call control, in: S. Reiff-Marganiec, M.D. Ryan (Eds.), Proc. 8th. International Conference on Feature Interaction VIII, IOS Press, Amsterdam, June 2005, pp. 39–57.
[44] Philips Ambient Intelligence In HomeLab, http://www.research.philips.com/, last viewed on June 1st 2006.
[45] Georgia Home Tech, http://www.gatech.edu/innovations/futurehome/index.php, last visited June 1st 2006.
[46] M. Shehata, Detecting requirements interactions using semi-formal methods, Ph.D. Thesis, 2005, University of Calgary, Alberta, Canada.