



Using Argumentation to Solve Conflicting Situations in Users' Preferences in Ambient Assisted Living

C. L. Oguego, J.C. Augusto, M. Springett, M. Quinde & C. James-Reynolds

To cite this article: C. L. Oguego, J.C. Augusto, M. Springett, M. Quinde & C. James-Reynolds (2021) Using Argumentation to Solve Conflicting Situations in Users' Preferences in Ambient Assisted Living, Applied Artificial Intelligence, 35:15, 2327-2369, DOI: [10.1080/08839514.2021.1966986](https://doi.org/10.1080/08839514.2021.1966986)

To link to this article: <https://doi.org/10.1080/08839514.2021.1966986>



Published online: 28 Oct 2021.



Submit your article to this journal [↗](#)



Article views: 445



View related articles [↗](#)



View Crossmark data [↗](#)



Citing articles: 1 View citing articles [↗](#)



Using Argumentation to Solve Conflicting Situations in Users' Preferences in Ambient Assisted Living

C. L. Oguego^a, J.C. Augusto^a, M. Springett^b, M. Quinde^{a,c}, and C. James-Reynolds^a

^aResearch Group on Development of Intelligent Environments, Department of Computer Science, Middlesex University London, London, UK; ^bInteraction Design Centre, Department of Computer Science, Middlesex University London, London, UK; ^cDepartamento de Ingenieria Industrial y de Sistemas, Universidad de Piura, Piura, Peru

ABSTRACT

Preferences are fundamental in decision-making, so understanding preference management is key in developing systems that guide the choices of the users. These choices can be decided through argument(s) which are known to have various strengths, as one argument can rely on more certain or vital information than the other. We explored argumentation technique from a previous study, and validated its potentials by applying to it several real-life scenarios. The exploration demonstrates the usefulness of argumentation in handling conflicting preferences and inconsistencies, and provides effective ways to manage, reason and represents users' preferences.

Using argumentation, we provide a practical implementation of a system to manage conflicting situations, and a simple interface that aids the flow of preferences from users to the system. We illustrated using the interface, how the changes in users' preferences can effect system output in a smart home. This article describes the functionalities of the implemented system, and illustrates the functions by solving some of the complexities in users' preferences in a real smart home. The system detects potential conflicts, and tries solve them using a redefined precedence order among some preference criteria.

We also show how our system is capable of interacting with external sources data. The system was used to access and use live data of a UK supermarket chain store, through their application programming interface (API) and provide users suggestions on their eating habits, based on their set preference(s). The system was used to filter-specific products from the live data, and check the product description, before advising the user accordingly.

ARTICLE HISTORY

Received 18 October 2019
Revised 17 September 2021
Accepted 22 January 2021

Introduction

Most decision humans make are based on choice(s), even refraining from choosing is a choice. Preferences guide our choices, so it is paramount to understand various aspects of preference handling if attempting to develop a

[†]CONTACT C. L. Oguego  Oguegoco527@live.mdx.ac.uk  Research Group on Development of Intelligent Environments, Department of Computer Science, Middlesex University London, London, UK; J.C. Augusto j.augusto@mdx.ac.uk

system that supports users' decisions or acts on behalf of users Brafman and Domshlak (2009), especially in an intelligent environment Augusto et al. (2013). For a system to be efficient in supporting and satisfying users' needs, it has to know the expectations of the users. There have been other support systems in Ambient Assisted Living (AAL) Augusto et al. (2012). Some of them rely on sensing equipment to gather information from users and the system uses this contextual information to help users in decision-making. An example can be having a pressure sensor on a bed or chair to detect if a user is lying on the bed or sitting on the chair. Another can be of having a Passive Infrared Sensor (PIR) in the bedroom to track movement. There is currently a range of devices that can elicit information from users. Information can also be gathered from the outside world, for instance, information details of a products from an on-line store. This can be useful for the system to manage the health preference aspect of a user. However, these systems cannot handle users' preferences in a dynamic way, because a system that is expected to act on behalf of humans needs to understand and respond to the preferences of users and have the ability to resolve conflicting preferences.

Conflict can occur in preferences, for example, the desire of wanting to keep the bedroom light "off" while asleep, can conflict with the need for the light to be "on," for safety reasons. Whichever reason it might be, a conclusion has to be reached, and the conclusion needs to be decided depending on what the user prefers more. A conclusion can also change if a new reason or fact becomes available. The knowledge of new facts can lead to preferring a new conclusion, or relying on a previous one, or make one consider that the previous conclusion is no longer correct. When new information becomes available, it might provide a better reason to maintain the previous conclusion or new reasons to come to a different conclusion. Providing a system that has the ability to react in such a manner, so as to balance users' preferences, is key in designing a successful Ambient Intelligence (AmI) system. We explored the potential of argumentation in handling inconsistent knowledge and conflicts in our previous work Oguego et al. (2019 b), which we will also discuss briefly in this paper.

Argumentation in AI

The evolution of argumentation emerged as an alternative to non-monotonic formalisms based on classical logic from the mid-1980s to present Chesñevar, Maguitman, and Loui (2000). Modeling common sense reasoning has long been a challenge in artificial intelligence (AI), as it mostly occurs in the face of incomplete and potentially inconsistent information Chesñevar, Maguitman, and Loui (2000). Several non-monotonic reasoning formalisms emerged to match this challenge, but in this formalism, when additional information is obtained, conclusions drawn may be later withdrawn Chesñevar, Maguitman,

and Loui (2000). Formal logics of argument emerged as one style of formalizing nonmonotonic reasoning, as argumentation systems provide a nonmonotonic layer to reason about justification of truth Simari and Loui (1992).

The reputation of argumentation in AI has positively increased Mahesar (2018), which is why it has been widely used for handling inconsistent knowledge ((Simari and Loui (1992), Amgoud and Cayrol (1998), Besnard and Hunter (2001) and García and Simari (2003)) and dealing with uncertainty in making decision(s) (Amgoud and Cayrol (1998) and Amgoud and Prade (2009)). The features of time and conflict-handling in argumentation systems have long been investigated in computer science (Muñoz et al. (2011), Augusto and Simari (2001), Bandara et al. (2006), Bentahar et al. (2010), and Muñoz and Botía (2010)). Argumentation has been known as a way to implement and formalize defeasible reasoning Simari and Loui (1992), allowing us to reason about a changing world where the information available is not very reliable or incomplete.

Argumentation as a reasoning process can help in making decisions by handling conflicting situations expressed within deliberative agents Tamani and Croitoru (2014). The fundamental ideas behind argumentation are to construct arguments in favor of and against each decision, evaluate the arguments and apply some principle of comparing their value based on quality or strength Amgoud, Bonnefon, and Prade (2005). The value of an argument can be qualified as defensible, justified, or defeated as it is determined by the importance of the rules (reasons) it contains Sartor (1994). The knowledge of new fact(s) can also lead to another conclusion being obtained. The obtained conclusions are justified through arguments to support their consideration Simari and Loui (1992).

When conflict arises among arguments, methods or preferences criteria are used to understand if some arguments may be preferred over others. Establishing the preference of an argument over another or a set of arguments over others, requires some definition of preference criteria, for example, “Specificity” and “Persistency.” These criteria were adopted during our implementation process, combined with “User Preferences” which we introduced in Oguego et al. (2019 b).

“Specificity” as a preference criteria is based on the argument structure, and decisions can be made based on which argument is better informed than the other. “Persistency” on the other hand, assumes that properties tend to keep their truth values through time, unless there is a reason to believe otherwise.

Our previous study Oguego et al. (2019 b) used the predicates: $\text{Change}_{at}^{+-}(p, i)$ and $\text{Change}_{in}^{+-}(p, I)$ to indicate that a proposition p changes its true value from being true to false at an instant i or in an interval I , respectively. The following axioms capture these concepts:

$$\mathcal{P} \mathcal{P} \quad \mathcal{T} \quad i(\text{Holds}_{at}(p, i - 1) \wedge \neg \text{Holds}_{at}(p, i))$$

$$\rightarrow \text{Change}_{at}^{+-}(p, i)$$

$$p \text{ } \mathcal{I} \text{ } I, I' (\text{MEETS}(I, I') \wedge \text{Holds}_{on}(p, I) \wedge \neg \text{Holds}_{on}(p, I'))$$

$$\rightarrow \text{Change}_{in}^{+-}(p, I')$$

Where “MEETS” should be considered as in (Allen (1984) and Hamblin (1972)).

PREFERENCES in AI

Preferences are crucial in decision-making and have been useful in areas of artificial intelligence (AI) such as scheduling, planning, combinatorial auctions, game playing, and multi-agent systems Walsh (2007). AI is not the only discipline where preferences are of great interest; they have been studied extensively in various disciplines including operational research, philosophy, economy, and psychology Mahesar (2018). Preferences are fundamental for decision-making as most areas of artificial intelligence deal with choice situation Pigozzi, Tsoukias, and Viappiani (2016). But it is important to consider that the system should be able to understand and support decisions made by users Goldsmith and Junker (2008).

There have been various preference handling mechanisms which exist in AI, and surveys have been conducted to identify the effectiveness of these classical preference techniques. One of such surveys, Oguego et al. (2018a) aimed to investigate the existing classical preference methods to know if they have the capability to deal with conflicting situations and represent users' preferences over time. Our study identified and investigated some known preference handling techniques that are closely related to the solution the research aims to provide. However, findings show that the existing methods lack the ability to handle the inconsistencies and complexities that exist in preferences, as preferences are known to change over time or clash with each other. For example, a football fan who is also a news enthusiast, may want to watch his favorite team play at 7 pm, and there is an important news programme that will be televised at the same 7 pm. How can the system support the user in making this decision?

Some of the classical preference techniques in AI, Conditional Preference Network (CP-net) for example, were restricted to manage either strict user preferences that are known or complete Allen (2014). For instance, a student prefers to keep the lights “off” during the day and “on” at night, as long as it is not his/her bedtime. This means at a certain period at night, the light should go “off.” This can be implemented using the CP-net approach, as the user's desires are already known. However, when the user is preparing for an exam, and stays up late to study, thereby falling asleep at random times during the

night, it will be difficult to apply CP-net in such case, as it is unknown when the student actually falls asleep. The investigation also identified that the classical preferences are not a feasible solution to produce a system that gives users the ability to manage the complexities that comes with preference management in AI.

Furthermore, we investigated ways users' preference can be managed in AI, and this led to the introduction of a new ontological sort, $\mathcal{P}ref$. We discussed this notion in details and theoretically applied it using some complex scenarios in Oguego et al. (2019 b). The $\mathcal{P}ref$ sort is used to specify "User Preferences," which is managed through a user preference mechanism (interface in our case). This introduction also led to redefining the order of precedence between the preference criterion as:

$$\mathfrak{R} = \{ \succ_{tspec}, \succ_{tpers}, \succ_{Upref(a)} \}$$

$$\succ_{tspec} > \succ_{Upref(a)} > \succ_{tpers}$$

This means arguments will first be compared with "Specificity," before using "Preference." If the arguments/conflict does not led to a new conclusion, the system will apply the notion of Persistency, which implies keeping the same value of the property, whose value is under consideration in the outcome of the decision process. A generic preference architecture framework was also produced (see Figure 1) to complement other existing frameworks, which we applied practically (Figure 2) in this paper.

The rest of this paper is as follows: Section 2 describes some case studies used in the practical demonstration of our system. Section 3 emphasizes on the significance of argumentation to handle inconsistent information and time. Section 4 introduces some of the features of the implemented system and provides some discussion on its relations with an existing reasoning system. We further emphasized on types of computations that can be conducted with the implemented system. Section 5 illustrates how we modeled the more abstract argumentation languages from Oguego et al. (2019 b) into the more restricted but practically more efficient, implementation language our system understands. We stated some guidelines for the modeling process and clarified better by applying an example (light case scenario). Section 6 explains the infrastructure and equipment we used for evaluating the implemented system and also layout of the Smart Spaces Lab where the evaluation was conducted. In Section 7, we introduced the preference management system (interface), which was used to influence system decision in the smart home, as we applied an informal scenario and provided a demo to illustrate this. We also modeled multiple users' preferences, for two different users, and applied one of the modeled users' preferences on the rest of the scenarios in this paper. Section 8 presented additional illustrations and demo, including the use of an API from a supermarket chain in the

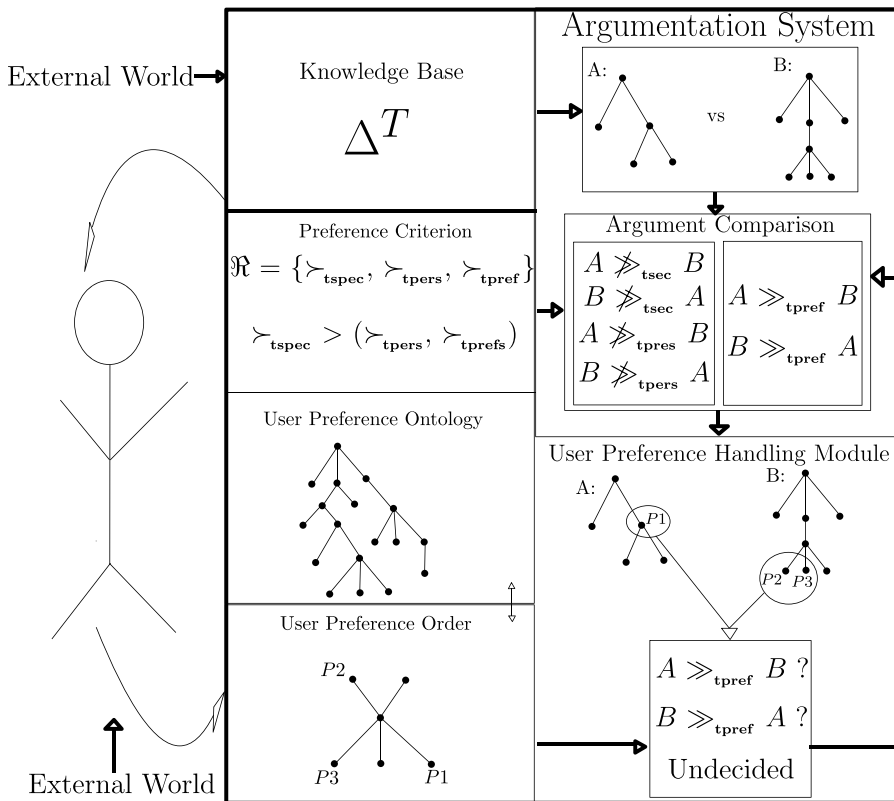


Figure 1. Overall preference architecture.

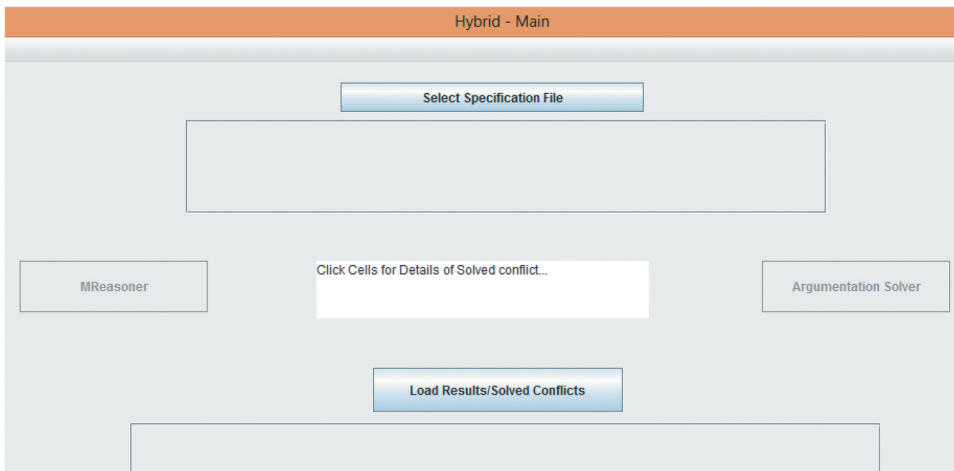


Figure 2. Using sara's preference ranking to solve bedroom light conflict.

United Kingdom. This is to show that our system is able to handle conflicting practical situations and provide users with viable decisions. In addition, we discussed the research process in general, from conducting the survey, to the developed system in Section 9, and conclude with further work in Section 10.

Case Study Analysis (Scenarios)

The below informal scenarios were considered and used to provide a practical demonstration of how our developed system works.

Light Case Study

Sara, an aged individual who lives alone, prefers the light to be “off” when she is asleep at night to provide more comfort. However, she might sometimes prefer the light “on,” so that it is safer for her to move around her room when she wakes up in the middle of the night.

Healthy Eating Case Study

Sara also wants the system to be aware of her health circumstances, and provide her with information on food consumption. Since she is diabetic, she wants to know the sugar content of her food, especially her favorite grocery, cake, which she usually buys from her local chain store in the UK (known as Tesco).

These scenarios were used to evaluate the system within and outside of a smart home, so as to demonstrate its abilities in managing conflicting user’s preferences. However, before evaluating, we remind readers of the theoretical background of the implemented system. A more detailed presentation of this, is available in Oguego et al. (2019 b).

Temporal Reasoning

Time is ubiquitous in any activity that requires intelligence, as some important notions like action, causality, and change are related to time Vila (1994). Artificial intelligence is an area where the concepts of time and event are essential, as agents usually have to reason about a dynamic environment.

The temporal language \mathcal{L}^T Augusto and Simari (2001) allows association of knowledge to either “*instant*” or “*interval*” so the development of the real-world scenarios can be expressed. Example of “*instant*” can be something that happened in a second in a system, while “*interval*” can be a whole minute in that system. A sensor that triggered once, let’s say 13:04:05PM can be described as instantaneous occurrence, but when the sensor triggers continuously over a period of time, let’s say 20 seconds (13:04:05PM to 13:04:25PM), it will be described as an extended occurrence during an interval of time.

We define the notion of interval as a sequence of consecutive instants $\mathcal{I} = \{[i_1, i_2] \in \mathcal{T} \times \mathcal{T} \mid i_1 < i_2\}$ so that, for example, $[14 : 06 \text{ PM}, 14 : 21 \text{ PM}]$ can be the interval where the sensor was continuously active. Auxiliary useful functions like $begin, end : \mathcal{I} \rightarrow \mathcal{T}$ can be defined to obtain the beginning and ending points of an interval: $begin([i_1, i_2]) =_{def} i_1$ and $end([i_1, i_2]) =_{def} i_2$. We considered a set of well-known relations in the literature as those between intervals that were explored by Hamblin Hamblin (1972) and later adopted by Allen Allen (1984).

The world can be described as a set of elements with specific properties, for which we use the following predicate: $\text{Holds}_{at}(p, i)$, $\text{Holds}_{at} \subseteq \mathcal{P} \times \mathcal{T}$, and $\text{Holds}_{on}(p, \mathcal{I})$, $\text{Holds}_{on} \subseteq \mathcal{P} \times \mathcal{I}$, denoting that p is a property that is true in the moment i or interval \mathcal{I} respectively. Holds_{on} and Holds_{at} are related in the following way:

$$\text{Holds}_{on}(p, I) =_{def} \forall i \in I (\text{In}(i, I) \rightarrow \text{Holds}_{at}(p, i))$$

We assumed ‘‘homogeneity’’ of properties over an interval, meaning that if a property holds in an interval then it also holds in any of its subintervals. For example, if a sensor was activated for 10 minutes in a row, it was activated in each minute of that interval (and each second of each minute):

$$\forall i \in I (\text{Holds}_{on}(p, I) \wedge \text{In}(i, I) \rightarrow \text{Holds}_{at}(p, i))$$

$$\forall I, I' (\text{Holds}_{on}(p, I) \wedge I' \sqsubseteq I \rightarrow \text{Holds}_{on}(p, I'))$$

We considered ‘‘weak negation’’ of properties over intervals that can be obtained directly from the negation of the previous definition:

$$\neg \text{Holds}_{on}(p, I) =_{def} \exists i \in I (\text{In}(i, I) \wedge \neg \text{Holds}_{at}(p, i))$$

We also considered events as noticeable occurrences of the real world that can influence a given situation. For example, the system sending a command to the light causes it to light up the room. We use a predicate $\text{Occurs}_{at}(e, i)$ ($\text{Occurs}_{on}(e, I)$) to indicate that an event e has occurred in an instant i (interval I), for example:

$$\text{Occurs}_{at}(\text{TurnOnLight}, 6 : 00 : 05 \text{ AM}), - \\ (\text{Occurs}_{on}(\text{Microwavecooling}, [15 : 10 : 05, 15 : 12 : 35])).$$

Mirroring explicit time references through instants and intervals, we assume non-durative and durative events defined in sorts \mathcal{N} and \mathcal{D} , respectively.

The following were assumed about event instances: $\text{Occurs}_{on}(e, I) =_{def} \forall i \in I (\text{In}(i, I) \rightarrow \neg \text{Occurs}_{at}(e, i))$ with $\text{In}(i, I) =_{def} \text{Start}(i, I) \vee \text{Divides}(i, I) \vee \text{Ends}(i, I)$ where these three predicates are true when an instant is at the beginning, ‘inside,’ or the end of an interval. The definition given above for $\text{Occurs}_{on}(e, I)$ means the occurrence of a specific event in an interval implies it does not occur inside the interval (this is usually

called “non-homogeneity”). We consider “weak negation” over durative events. That is, consequently with the concept of non-homogeneity explained above, an event will be considered to not have occurred if a fragment (even just an instant) of it has not occurred.

We ascribed actions only to humans, as humans usually act on their free will and perform actions, which typically cause some events to occur, which in turn potentially change some properties of the world. We considered each human agent a from the sort of agents \mathcal{A} has a repertoire \mathcal{W} of possible actions g : $\mathcal{A} \ a \ \exists_{\mathcal{W}} \ g \ \text{Agent}(a, g)$. There could be instantaneous actions Do_{at} (e.g., closing the toilet door) and durative actions Do_{on} (e.g., walking in the corridor).

Further discussion on temporal argumentation and notion of “instant” and “interval” can be found in our previous study Oguego et al. (2019 b). The explanations above mostly refer to the time-related representation of the world, which we modeled later in this study to the language the implemented system understands. We will now provide overview discussion of the implemented system (“Hybrid”), and a brief discussion on a reasoning system (MReasoner) the Hybrid system was built upon. In addition, we provided some explanation of the specification file required by the system for execution.

Hybrid System for Real-time Decision Making

An earlier environment for our AAL system was based only on monotonic system with some basic capabilities to reason with metric time operators called “MReasoner” Ibarra, Augusto, and Goenaga (2014).

Our latest Hybrid System was developed to complement MReasoner (shown in Figure 3) as a way of extending its capabilities, as the MReasoner tool can not manage and solve conflicting situations in an intelligent environment in real time. The resulting Hybrid System can do both types of reasoning, monotonic and non-monotonic. When no conflicts are present in the rule base, it uses the simpler light weight MReasoner, when there are rules with opposing conclusion, Argumentation is used. Some of the features of the Hybrid System includes:

- Ability to select a specification file containing rules from any location on the computer
- Analyzes the selected specification file using a conflict analyzer algorithm to identify potential conflicts.
- Displays potential conflict(s), if any, has been identified.
 - Depending on whether potential conflict(s) has been detected, the Hybrid System will pass the file to either the exiting reasoning tool (“MReasoner”) if no potential conflict is detected, or the conflict solver tool (“Argumentation Solver”) if potential conflict(s) is detected.

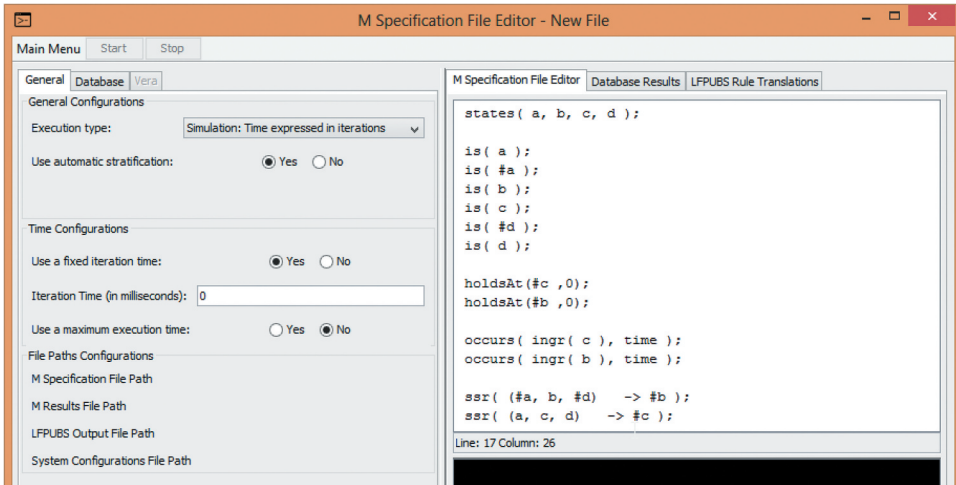


Figure 3. Reasoning system interface.

- During execution, Hybrid System has the ability to generate current results of all the properties involved in the execution.
- Conflict(s) detected during the execution process will be solved by the argumentation solver. The Hybrid System will display the new results on the interface, which will take effect around the environment at run time.
- The area(s) (instant or interval) where conflict(s) were identified and solved, will be highlighted by the Hybrid System for clarity.
- The Hybrid System also has the ability to explain how the conflict(s) were solve. Clicking on any of the highlighting cell from the result table, will display the reason of how the conflict was solved in the text area of the Hybrid System interface.

Illustrations of the Hybrid System at work will be provided in Section 8, were we provide some demonstration of the system in real time, applying some real scenarios and live data.

Reasoning System (Mreasoner)

The reasoning system (*M*) was developed (interface shown in [Figure 3](#)) based on natural characteristics of reactive environments, as it has the ability to track certain environment conditions and act upon them. *M* also has the capabilities to capture states happening during time intervals. For example, if there is no movement in the last 15 seconds, turn “off” the lights in the room. However, *M* lacks the ability to handle conflicting outcomes. For example, if someone is doing yoga, do not turn “off” the lights.

The reasoning system (M) is a rule-based system aimed at handling simple causality, but has been extended to handle some practical uncertainties and complexities, especially conflicts in user preferences. We extended the M system by using argumentation to improve the capability of detecting and solving conflicts that occur within an intelligent environment. The argumentation solver accepts the specification file from the Hybrid System containing detected potential conflict(s). Conflicts get solved by the argumentation solver following the order of precedence of preference criteria discussed in Section 1.2. The specification file have to be written in an exact format that is acceptable by the Hybrid System for execution. The specification file format and the execution types are discussed in Section 4.2.

Specification File and Execution Types

Figure 4 illustrates the specification file sample (and added some labels for clarity) of the reasoning system. The specification file has to be in that format, but also depends on the type of execution the reasoning system is running. The execution types that can be simulated by the reasoning system include:

- Simulation expressed in iteration
- Simulation expressed in real time
- Simulation (execution) in real environment, with sensors and actuators

The first part of Figure 4 consists of “all the properties (states),” any property that will be used during execution must be specified in the first part. The second part consists of the declaration of “Independent States,” which does not depend on other states causally. The “#” symbol (when placed in front of a state) denotes that a property is false. An example of applying the

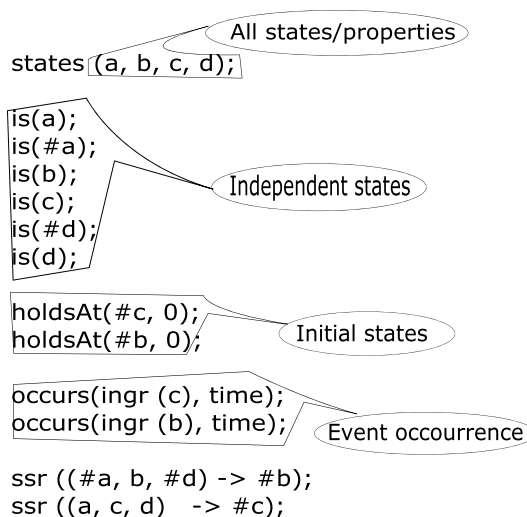


Figure 4. Specification file format sample.

symbol can be “*is(#Movement)*“, which can be used to represent ‘no movement’ detected. The next part, which is the “Initial State” (as seen in Figure 4), signifies initializing the state. For instance, *holdsAt(#Movement, 0)* indicate that at the start of the iteration, the movement will be “off” or false, this is absence of movement is assured. The fourth part, known as “Event Occurrence” (as shown in Figure 4), are events used to impinge the system from the outside; it can be sensors being triggered or via human behavior commands. All of this notation was first defined as part of the “C” language in Galton and Augusto (2002), then the language was created by adding metric temporal operators to “C” Ibarra, Augusto, and Goenaga (2014).

However, the event representation (*occurs(ingr([#]s), t**) “s” signifies state and “t” signifies time) can only be used depending on the type of execution simulated by the reasoning system.

The “Simulation expressed in iteration” executes the specification file of the reasoning system based on the number of iterations specified, and the execution will not stop until the specified iteration. The “Simulation expressed in real-time” uses the real-time specified on the specification file, as specified events are triggered at a specific time. For example: *occurs(ingr(LightsOn), 16 : 00)* means that the light should be triggered “on” at 4 pm.

The specification file format shown in Figure 4 is the format our implemented system recognizes, in order to conduct the executions. However, the theoretical argumentation language is strictly more expressive than “M” and will need to be modeled into the specification file format (system language). Thus, we illustrated in detail in the next section of this paper.

Translating Argumentation Language

Our previous study explored the potential of argumentation to handle conflicting user preferences Oguego et al. (2019 b). The study also explores a generalized framework that can be applied to handle user preferences in AAL and further provided an overall preference architecture (Figure 1) which can be used to extend the current argumentation systems. A proposed system was illustrated theoretically to indicate that it can handle different users with the introduction of a personalized preference function. The illustration showed how user preferences can be handled in a realistic way in an intelligent environment.

One part of the scenario considered in the complex description discussed in our previous study was the lighting aspect, to make sure lights are “off” after leaving home. In addition to the theoretical illustration of how the system should work, we introduced the notion of *P ref*, (Oguego et al. (2019 b)) used to represent “User Preferences” in our system, allowing users to specify what part of their preferences is more important to them. This was implemented in

the form of an interface, which has been discussed and illustrated in Section 7, as the interface allows users to select and rank/modify their preference(s) to effect output in a smart home.

The proposed system was also implemented, which we refer to as “Argumentation Solver,” and will be discussed and illustrated in Section 8.1, showing its ability to handle conflicting situations in a smart home. However, executing the Hybrid System in a smart house requires a specification file containing arguments that are made of rules, which the smart home will use to act accordingly. These arguments consist of rules, are required to make decisions, as conclusions are justified through arguments to support their consideration Augusto and Simari (2001). The argument notations in argumentation will need to be translated to the language (rules) our system (Hybrid) understands for execution. This translation will further be complemented with a simple light study of keeping the lights “off” after the user leaves home, for better explanation. However, we will illustrate in the next section how, we modeled the argumentation theoretical language to the implementation language our system understands.

Modeling Argumentation Theoretical Language to Implementation Language ($\mathcal{L}^{\mathbb{T}}$ to M)

This section illustrates how we modeled some of the notations from the argumentation theoretical language ($\mathcal{L}^{\mathbb{T}}$) into the implementation language (M). The translation of the $\mathcal{L}^{\mathbb{T}}$ to M is not an automated process yet, it has been manually modeled by the developers of the implemented system. There are guidelines listed below, which has been followed throughout the modeling process. Some explanations have also been included for further clarifications.

The first step of the modeling process is the time frame of action(s) or/and event(s) occurred, which can be at an instant or over a period of time (interval).

- “ I_0 ” or “ I_1 ” or “ I_2 ” refer to “interval 0” or “interval 1” or “interval 2” in $\mathcal{L}^{\mathbb{T}}$. Modeling this to the M using the time interval relation “MEETS” will become “[begin(I_0), end(I_0)]” or “[begin(I_1), end(I_1)]” or “[begin(I_2), end(I_2)],” with an instant representing 1 unit or 1 second of time.

- [-]2 represents 2 units (2 seconds) of time (Interval).
- [-][120s.] represents 120 seconds or 2 minutes of time (Interval).
- Constraints, such as: Length (I_1) > 15(mins), will be represented as [-][900s.] or [-]15. This indicates an action or occurrence of event taking place over the previous 15 minutes.

Events occurrence are triggered by sensors or actuators, actions are usually triggered by humans, they were modeled as follows:

- Actions triggered by humans in an interval (e.g., movement detected for 20 seconds) is represented as Do_{on} in $\mathcal{L}^{\mathbb{T}}$, and modeled into Do_{-} in the implemented system M .

- Actions triggered by human in an instant (e.g., the light gets turned “on” at 7:00PM) is represented as Do_{at} , and modeled as Do_{-} .

- Occurrence of event in an interval, triggered by sensor(s) or/and actuator(s) in $\mathcal{L}^{\mathbb{T}}$ and represented as $Occurs_{on}$; has been modeled to Occ_{on} in M .

- Occurrence of event in an instant, triggered by sensor(s) or/and actuator(s) in $\mathcal{L}^{\mathbb{T}}$, and represented as $Occurs_{at}$; has been modeled to Occ_{at} in M .

Other additional notations of $\mathcal{L}^{\mathbb{T}}$, which were modeled to M , which is a superset of atemporal “C” language, are as follows, and we also include some explanations of $\mathcal{L}^{\mathbb{T}}$ notations that were not modeled but used as they were in the implement system (M).

- Negation in $\mathcal{L}^{\mathbb{T}}$, is represented as “ \neg ”, and we modeled this to “#.” An example of how we applied the negation is: $\#LivingroomLight$, meaning the living-room light is “off.”

- The holding state of a property at an instant in $\mathcal{L}^{\mathbb{T}}$ is represented as $holdsAt$. We use the same notation ($holdsAt$) in M . An example of how this notation can be applied is $holdsAt(\#LivingroomLight, 0)$, meaning at “instant 0” (i_0), which is the starting point of the system, the living room light was “off.”

- For $\mathcal{L}^{\mathbb{T}}$, the rules have a name or label ID. For example, L-R1 – L-R6 indicates Light rule 1 to Light rule 6. In M , each rule is represented as “ssr” and refereed to as “same time rule.”

- The notion ($Pref_{on}$) introduced in Oguego et al. (2019 b), is represented as $pref$ in M , which signifies user preference.

The sort ($Pref_{on}$) introduced in Oguego et al. (2019 b) which was also applied, is represented as $pref$ in the implementation language, and signifies “Users Preferences.”

We applied a light scenario example in Section 5.2 as regards to a user who wants the system to switch “off” the lights when s/he leaves home. This is to provide a better understanding of the guidelines for translation aforementioned.

Translating Light Scenario (Example)

The notion of interval can be defined as a sequence of consecutive instants. So to translate temporal argument rules to the reasoning rules our system understands, interval relations needs to be considered. For our case, we have adopted the interval relations defined by Hamblin (1972) and popularized in Allen (1984), as

it has been known to be the most widespread way to reason and represent time in computer science, specifically in AI. Interval relations defined by Hamblin (1972), have thirteen possible relationships, one of them is “MEETS.”

MEETS(I_1, I_2) is defined as: interval I_1 is before interval I_2 , but there is no interval between them, i.e., I_1 ends where I_2 starts. Other relations can be used, we just use MEETS for simplicity of the explanation.

Argumentation Light Scenario for Sara

Using the MEETS interval relationship, we illustrate a lighting scenario of a user (Sara) who want the lights in her home to be switched “off,” after the system detects that she has left home.

Table 1 shows the development of the light scenario through time. The next set of rules are extracted from Δ^{T1} Oguego et al. (2019 b) to model the scenario in the argumentation system:

$$MEETS(I_0, I_1) \wedge MEETS(I_1, I_2) \wedge MEETS(I_2, I_3)$$

$$Holds_{on}(Movement, I_0) \wedge \neg Holds_{on}(Sleeping, I_0) \wedge$$

$$\neg Holds_{on}(OnBed, I_0) \wedge Holds_{on}(LightsOn, I_0)$$

–

$$L\text{-R1: } Do_{on}(LeavingHome, I_0) > \text{---} Occurs_{at}(LeftHome, begin(I_1))$$

$$L\text{-R2: } Occurs_{at}(LeftHome, begin(I_1)) > \text{---} \neg Holds_{on}(Movement, I_1)$$

$$L\text{-R3: } \neg Holds_{on}(Movement, I_1) \wedge Length(I_1) > \\ = 15 \wedge \neg Holds_{on}(OnBed, I_1) > \text{---} \neg Holds_{on}(Home, I_2)$$

$$L\text{-R4: } \neg Holds_{on}(Home, I_2) > \text{---} Pref_{on}(LightOff, I_2)$$

$$L\text{-R5: } Pref_{on}(LightOff, I_2) > \text{---} Occurs_{at}(SystemTurnsLightOff, end(I_2))$$

$$L\text{-R6: } Occurs_{at}(SystemTurnsLightOn, end(I_2)) > \text{---} \neg Holds_{on}(LightsOff, I_3)$$

Table 1. Sara lighting scenario dynamics.

	$MEETS(I_0, I_1)$	$MEETS(I_1, I_2)$	$MEETS(I_2, I_3)$	
	$Holds_{on}(Movement, I_0) \wedge \neg Holds_{on}(Sleeping, I_0) \wedge \neg Holds_{on}(OnBed, I_0) \wedge Holds_{on}(Home, I_0) \wedge Holds_{on}(LightsOn, I_0)$			
Lighting Scenario	Movement	\neg Movement	\neg Movement	\neg
Movement	\neg Sleeping	\neg Sleeping	\neg Sleeping	\neg Sleeping
	\neg OnBed	\neg OnBed	\neg OnBed	\neg OnBed
	Home	Home	\neg Home	\neg Home
	LightsOn	LightsOn	LightsOn	\neg LightsOn
Transition Cause	$Do_{on}(LeavingHome, I_0)$	System Inference from: L-R3	$Occurs_{at}(SystemTurnsLightOff, end(I_2))$	I_3
	I_0	I_1	I_2	

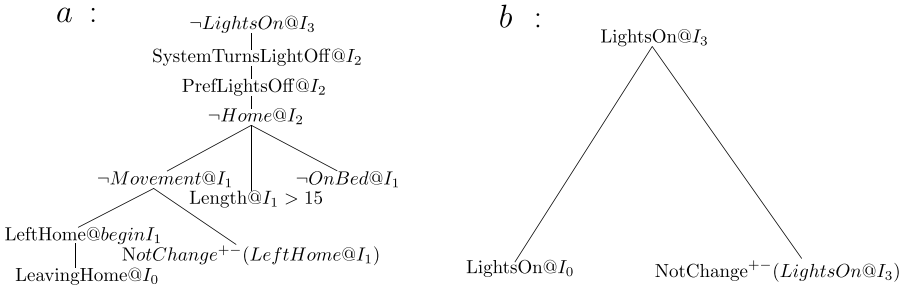


Figure 5. Argumentation trees for sara's light scenario.

The above six rules were modeled to the rules in the specification file, which is format the implemented system (M) understands. **Figure 5** further depicts the argumentation trees representation of the above rules, along with explanation of the argument.

Argument for $LightsOn@I_3$: As seen from the initial facts, the lights are assumed to be “on,” as Sara is in the room. So because of persistency, there is a possibility that the lights will continue to remain “on.”

$$L.On = \langle \{ Holds_{on}(LightsOn, I_0) \wedge notChange_{in}^{+-}(LightsOn, [end(I_0), end(I_3)]) > ---Holds_{on}(LightsOn, I_3) \}, Holds_{on}(LightsOn, I_3) \rangle$$

The argument is reflected in **Figure 5b**.

Argument for $\neg LightsOn@I_3$: Considering an alternative explanation, given that the system has been designed to understand when the lights are not needed. The argument indicates that Sara is leaving home at I_0 and is not home at the beginning of I_1 . As a result of this, no movements were detected from then onwards. If continued for the next 15 minutes and there is no pressure on the bed at the same time, the system has reasons to believe that Sara is not at home at I_2 . When Sara is not at home, she prefers the lights “off.” So at that moment, the system infers that it is reasonable to turn the lights “off.” As a result, the lights are off at I_3 , as illustrated in the argument tree shown in **Figure 5a**

$$\begin{aligned} L.Off = & \langle \{ Do_{on}(LeavingHome, I_0) > ---Occurs_{on}(LeftHome, I_1), \\ & Occurs_{on}(LeftHome, I_1) > ---\neg Holds_{on}(Movement, I_1), \\ & \neg Holds_{on}(Movement, I_1) \wedge Length(I_1)15 \wedge \neg Holds_{on} \\ & (OnBed, I_1) > ---\neg Holds_{on}(Home, I_2), \neg Holds_{on}(Home, I_2) \\ & > ---Pref_{on}(LightsOff, I_2), Pref_{on}(LightsOff, I_2) \\ & > ---Occurs_{on}(SystemTurnsLightOff, I_2), \\ & Occurs_{on}(SystemTurnLightOff, I_2) \\ & > ---\neg Holds_{on}(LightsOn, I_3) \}, \end{aligned}$$

$$\neg \text{Holds}_{on}(\text{LightsOn}, I_3)\}}\}$$

Table 2 further illustrates the translation of Sara’s light scenario, following the dynamics of Table 1, applying the interval relationship (“MEETS”) and the modeling guidelines provided in Section 5.1. The output of the modeling process of the light scenario is in form of the specification file in Section 5.2.2.

Specification File with Converted Rules

Below depicts the specification file for the light scenario along with the modeled rules discussed in the previous section. Other aspects of the specification file have been explained in Section 4.2.

```

states(Movement, OnBed, LightsOn, Home, Do_LeavingHome,
Occ_LeftHome, SystemTurnsLightOff, prefLightOn);
is(Movement); is(#OnBed); is(OnBed); is(LightsOn); is(Home); is
(Do_LeavingHome);
holdsAt(#Movement, 0);
holdsAt(#OnBed, 0);
holdsAt(LightsOn, 0);
holdsAt(Home, 0);
holdsAt(Do_LeavingHome, 0);
holdsAt(#Occ_LeftHome, 0);
holdsAt(SystemTurnsLightOff, 0);
holdsAt(prefLightOn, 0);
ssr((Do_LeavingHome) -> Occ_LeftHome);
ssr((Occ_LeftHome) -> #Movement);
ssr(([-][900s.]#Movement ^ #OnBed) -> #Home);
ssr((#Home) -> #prefLightOn); ssr((#prefLightOn) ->
SystemTurnsLightOff); ssr((SystemTurnsLightOff) -> #LightsOn);

```

Now we will discuss and show some of the infrastructure and equipment required for the demonstrations in a real environment.

Table 2. Converting Argumentation Rules to Reasoning System Rules.

	Argumentation Rules ($\mathcal{L}^{\mathbb{T}}$)	Specification File Rules (\mathcal{M})
L-R1	$Do_{on}(\text{LeavingHome}, I_0)$ $> \text{---} Occurs_{at}(\text{LeftHome}, \text{begin}(I_1))$	$ssr((\text{Do_LeavingHome}) -> \text{Occ_LeftHome});$
L-R2	$Occurs_{at}(\text{LeftHome}, \text{begin}(I_1))$ $> \text{---} \neg \text{Holds}_{on}(\text{Movement}, I_1)$	$ssr((\text{Occ_LeftHome}) -> \# \text{Movement});$
L-R3	$\neg \text{Holds}_{on}(\text{Movement}, I_1) \wedge \text{Length}(I_1) > 15 \wedge$ $\neg \text{Holds}_{on}(\text{OnBed}, I_1) > \text{---} \neg \text{Holds}_{on}(\text{Home}, I_2)$	$ssr(([-][900s.]\# \text{Movement} \wedge \# \text{OnBed})$ $-> \# \text{Home});$
L-R4	$\neg \text{Holds}_{on}(\text{Home}, I_2) > \text{---} \text{Pref}_{on}(\text{LightOff}, I_2)$	$ssr((\# \text{Home}) -> \# \text{prefLightOn});$
L-R5	$\text{Pref}_{on}(\text{LightOff}, I_2)$ $> \text{---} Occurs_{at}(\text{SystemTurnsLightOff}, \text{end}(I_2))$	$ssr((\# \text{prefLightOn}) -> \text{SystemTurnsLightOff});$
L-R6	$Occurs_{at}(\text{SystemTurnsLightOn}, \text{end}(I_2))$ $> \text{---} \neg \text{Holds}_{on}(\text{LightsOff}, I_3)$	$ssr((\text{SystemTurnsLightOff}) -> \# \text{LightsOn});$

Smart-Home Infrastructure

The research utilized a Smart Spaces Lab to conduct practical demonstration of the system. The lab is a fully functional home environment provided to support research in AAL and specialized spaces to support research in the areas of Virtual/Mixed/Augmented reality and group decision-making support. The Lab further consists of other physical equipment that was also needed for the demonstration process, the physical equipment will be explained later. Some images of the Smart Spaces Lab areas and the equipment are found in Section 6.1 and Section 6.2 respectively.

Smart Spaces Lab

The smart space lab is located within the Middlesex University premises. It is also known as Farm House with necessary housing facilities, giving the lab the feel of a home. Figure 6 depicts the layout, which consists of a living room (Figure 7), a bedroom (Figure 8), a kitchen, a bathroom, a shower room, and two addition rooms used for conducting meetings and research. As seen on the layout, parts of the house are wired with all types of sensors for research purposes, but we will address a few that are specific to this research.

More images of the smart home can be found here: <http://ie.cs.mdx.ac.uk/smart-spaces-lab/>

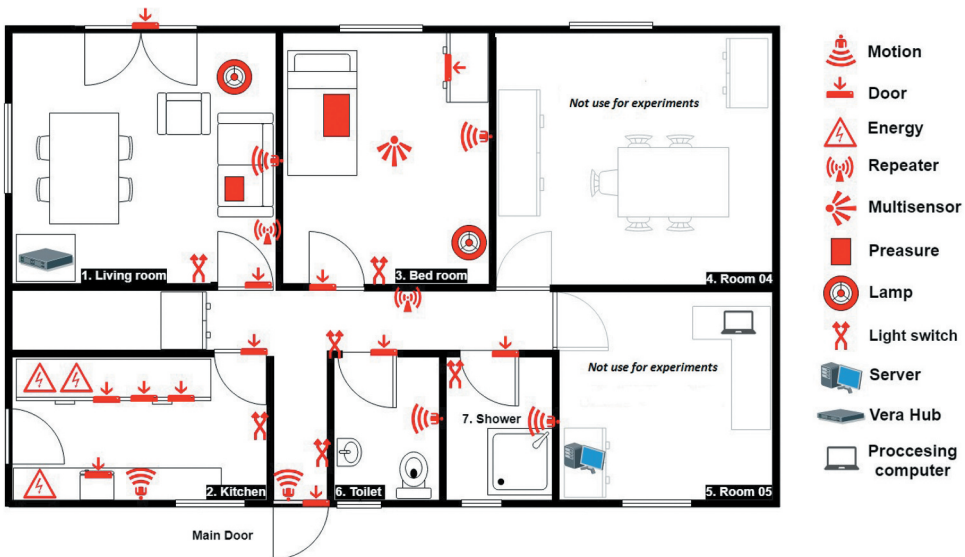


Figure 6. Layout of the smart spaces lab.

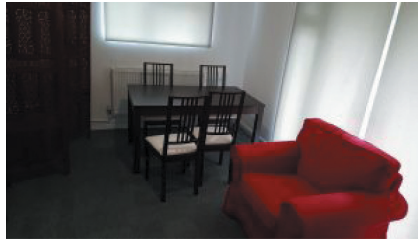


Figure 7. Living-room of the smart spaces lab.



Figure 8. Bedroom of the smart spaces lab.

Equipment

The smart home requires smart devices and equipment (see right side of [Figure 6](#)) to conduct the experiments. However, for our demo we made use of a few, which are Motion sensor(A), Reed sensor(B), Light Switch(C), Vera Box(D) and Pressure Pad(E), as show in [Figure 9](#).

The PIR (also known as the Motion sensor) is used to detect movement in the areas placed around the house. The Reed sensor device is mostly attached by doors or windows to detect if they are open or closed. The Reed device was reconfigured along with a dance mat to produce the pressure pad (shown in [Figure e](#)), which we used to detect pressure on the bed. We can either place the pressure pad on the bed or on the sofa to detect if a user is occupying any of these positions.

[Figure 9c](#) is a light switch, connected to the Vera smart box, which communiAscates with the sensors and actuators. This can be used to carry out the automation process without using the switch itself. [Figure 9d](#) depicts the smart hub (Vera Box) that manages the z-wave sensors and actuators connected to it through its own WiFi network. Vera accepts requests to query or modify the state of the sensors/actuators. We used Vera and the reasoning system to execute the instructions in the specification file, which will trigger the necessary outputs in the smart home.

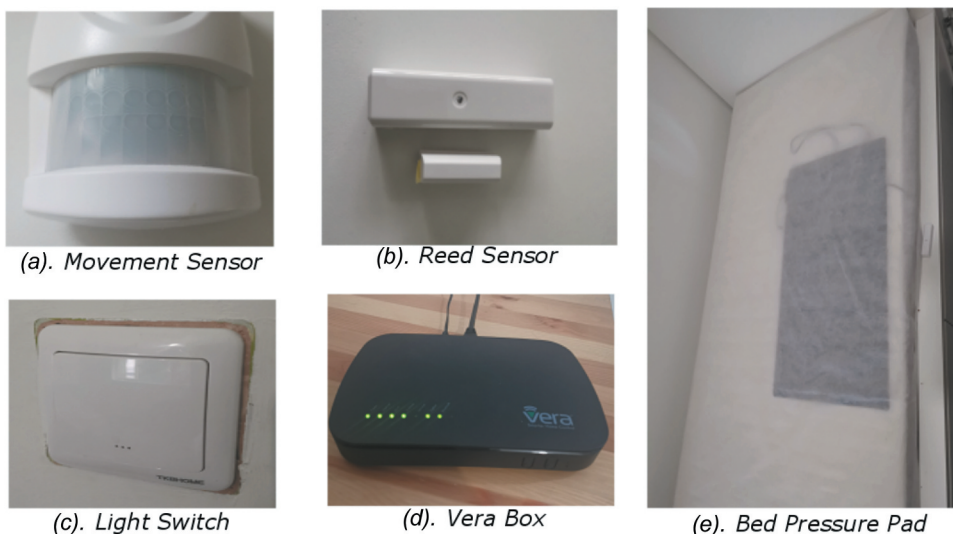


Figure 9. Smart devices and equipment for experiments.

The next section will illustrate our first demo using the preference mechanism, with our system and an informal scenario to demonstrate how different user's input can immediately impact the system's output. But first, we will introduce our preference management tools.

Preference Management

One key aspect of our system is to provide means, which allow users to manage their preferences easily. The system uses the managed preference(s), to reason about the preferences of the user, and provides output that aligns better with the services required by the user. A simple interface has been produced to help users manage their preferences and also help to manage some of the complexities in users' preferences in a smart home. The interface consists of textual menus for simplicity, incorporating the *P ref* notion introduced in Oguego et al. (2019 b) to allow users to select and rank their preference(s) at their convenience. Depending on how the user ranked their preference(s), the system output will be affected.

There are other preference interface that exist for smart home, such as "assist-robot interface" Wang, Saboune, and Saddik (2013) that works in two modes. Portable-Mode (whesn the user is not at home), and Robot-Mode (when the user is at home). Another is a "Virtual assistant" (Ospan et al. (2018)), used as a control interface for smart home environments, by using voice or text command. Complementing these interesting innovative interfaces, we only aim to only provide a simplified interface to manage preferences in a

smart home, as most of the existing ones are not ideal for all users, especially older adults whose technical ability tends to decline affecting their ability to interact with complex technological advancements Ruzic et al. (2016).

Preference Management Tool (Interface)

The interface was developed to give users the ability to prioritize their preferences, gives them the freedom to modify it any time, and it will take effect immediately. The developed interface allows existing users to easily retrieve the profile, modify the preference ranking, and update the details. New users can also create a new profile, which can be done on the same home page.

Figure 10 depicts a three-step process of creating a new profile, as it only requires the user to enter and submit a name. This transfers the user to the next page where the user can select from a list of available preferences they want to prioritize. The third page is where the users can set priority on the selected preference(s).

Figure 11 illustrates the modification process of preference(s) from existing users. They only need to select their name from the drop-down list on the home page, which will load their profile consisting of their preferences and ranking. The user can then modify the preference(s) they want and update it, ready to be used immediately.

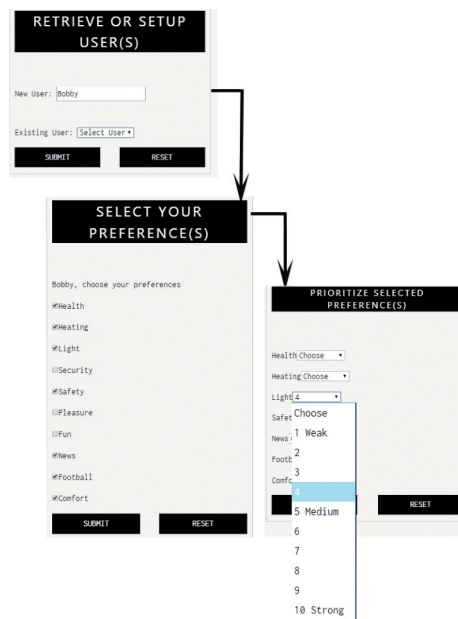


Figure 10. Simple setting up of new users' preference.

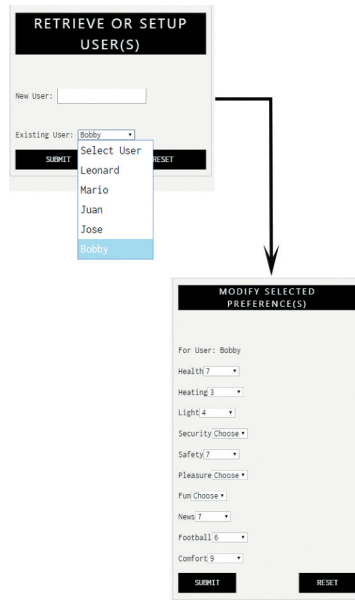


Figure 11. Retrieving and modifying existing preferences.

The research further provided an overall preference management framework, which consists of the preference interface that has been discussed. The preference management framework, as shown in Figure 12, depicts the flow of the system. This starts from the user creating their profile or modifying the ranking of their existing preference(s) using the preference interface, and saved in the preference database. The system then uses the saved users' preferences to provide the user the required service(s) or output. The system can also use the users' preferences ranking to solve any conflicting situation detected during the process.

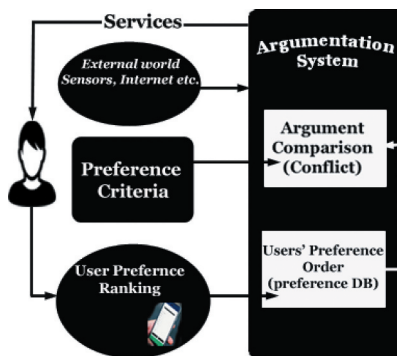


Figure 12. Overall preference management system.

Further illustration (demo) has been provided using the brief scenario in Section 7.2, to show how the changes in users preference ranking (using the preference interface), can dynamically change in real-time the system behavior, whilst the system is still running.

Using User's Preferences to Affect System Output

As a guiding scenario, let us consider a user, Bobby, expects the lighting scenario to adapt to varying circumstances. Below is the informal scenario, which expresses how the user can prioritize their preference of “Comfort” over “Light” and vice versa.

Bobby, an aged individual who lives alone, prefers the light to be “off” when he is asleep at night to provide more comfort. However, he might sometimes prefer the light to be “on,” as it is safer for him to move around when he wakes up during the night.

The first sentence of the scenario indicates that Bobby wants the system to turn the bedroom light “off” when he is asleep as he prefers the comfort over keeping the light “on.” In this case, Bobby has decided to rank his “Comfort” higher (probably 6) than “Light” (probably 4). When the system executes the rules (found below), which states that if Bobby is on bed for 30 seconds (`[30s.]BedPadPressure`) and there is no movement in the bedroom (`#BedRoomMotion`), the system will switch “off” the light (`#BedRoomLight`).

```
ssr(([-][30s.]BedPadPressure ^ #BedRoomMotion ^ prefComfort) ->
#BedRoomLight);
ssr(([-][30s.]BedPadPressure ^ #BedRoomMotion ^ prefLight) ->
BedRoomLight);
```

The second sentence in the description explains that Bobby might sometimes prefer the light “on” for safety reasons when he wakes up during the night. Let’s assume Bobby decides to change his preference and ranks “Light” higher (6) than “Comfort” (5). When he goes back to bed, after 30 seconds or more of being on the bed, the system will still continue to keep the lights “on,” as he has now indicated that he prefers “Light” over “Comfort.”

The link [Oguego et al. \(2019 b\)](#) consists of two separate video demos, showing how the home reacts toward Bobby’s situation, as the system reacts to his preference changes (preferring “Light” over “Comfort” and vice versa).

Figure 13 indicates how we modeled part of Bobby’s scenario (from **Figure 12**), when he decided to keep the light “on” while he is asleep, so it is safer for him to move around when he wakes up during the night. As seen from **Figure 13**, Bobby modifies his preferences ranking to prefer “Light” over “Comfort.”

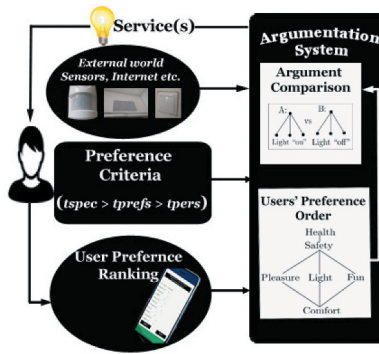


Figure 13. Modeled bobby's situation of keeping the light “on.”

The ranking order is saved in the database, and shown on the bottom right of the figure. Since Bobby ranked the “Light” higher than “Comfort,” the system provides Bobby the service of keeping the light “on.”

Furthermore, this research modeled the preferences of different users, that have different preference desires. Section 7.3 discuss how two different users with opposite desires can be modeled using the developed preference management mechanism.

Modeling Different Users Preferences

This section models the preferences of two different users. One of the user (Sara), cares more about her health situation compared to other aspects of her preferences, while her son Joe fancies his pleasure and fun more.

In addition to Sara’s preferences on handling light automatically (as seen in Section 2), she also wants the system to be aware of her health circumstances, and provide her with information on food consumption (if it contains sugar, since she is diabetic), especially for her favorite product (“Cake”) which she buys from a grocery store.

If we assume Sara’s “Health” (assigned 9) is more important than “Safety” (assigned 7) and “Safety” is more important than “Pleasure,” “Light,” and “Fun” (with all having an equal level of importance assigned 5) but are more important than “Comfort” (assigned 4). Then, using the notion introduced in Section 3 of our previous study Oguego et al. (2019 b), we can represent Sara’s preference in our system as follows:

$$Pref_{Sara} = \{ \textit{finance, comfort, safety, health, fun, pleasure} \}$$

$$\mathcal{O}(Pref_{Sara}) = \{ (9, health), (7, safety), (5, pleasure), (5, Light), (5, fun), (4, comfort) \}$$

Meanwhile, Sara has a teenage son, Joe, who cares about pleasure and fun above everything else. Joe also prefers his comfort over health, safety, and light.

We also assumed that Joe on the other hand, prefers “Fun” and “Pleasure” (assigned 7) above “Comfort” (assigned 5), and “Comfort” above “Health,” “Safety,” and “Light” (equal level of importance; 3). Using the same notion introduced in Oguego et al. (2019 b), we can represent his preference ranking as follows:

$$Pref_{Joe} = \{ comfort, light, safety, health, fun, pleasure \}$$

$$\mathcal{O}(Pref_{Joe}) = \{ (7, fun), (7, pleasure), (5, comfort), (3, health), (3, safety), (3, light) \}$$

The preference representation of Sara shown in Figure 14, while Joe's preference representation is shown in Figure 15. However, Sara's representation will be used in the next section to illustrate how our system works when conflict(s) arises

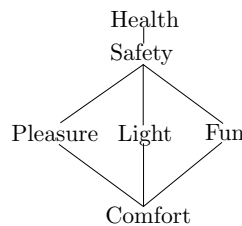


Figure 14. Sara's preference ranking.

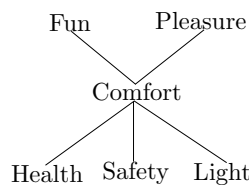


Figure 15. Joe's preference ranking.

Illustrations and Demos of Scenarios

The implemented system (Hybrid Main), which comprises of a reasoning system and the argumentation resolver, is used to demonstrate how the system works, applying the scenarios mentioned in Section 2. The demo is in three categories. The first demonstration was on the Hybrid System, which shows the overall working of the application. This includes selecting a specification file that contains rule(s), which the system compiles, and check for potential conflict(s). Secondly, a light scenario to illustrate the working of the argumentation system, using preference criteria initially discussed and applied in this precedence order: $\mathfrak{R} = \{ \succ_{\text{tspec}}, \succ_{\text{tpers}}, \succ_{\text{Upref}(a)} \}$ with $\succ_{\text{tspec}} > \succ_{\text{Upref}(a)} > \succ_{\text{tpers}}$

Lastly, the integration of a large chain store's API, as we use the system to access their data and search for a specific type of product ("Cake" in this case). This illustrates the flexibility of our research in terms of the sources of data and the type of contexts being considered.

Hybrid System Illustration

Figure 16 depicted the interface of the Hybrid System, which is used to load a specification file. The specification file contains a set of rules, which should be selected using the "Select Specification File" button. Depending on whether the rules in the Specification file contains potential conflict(s) or not, the system will activate/enable either the "MReasoner" button or the "Argumentation Solver" button.

Launching the Hybrid system will disable both the "MReasoner" and "Argumentation Resolver" buttons, as shown in Figure 16. The specification file will need to be selected (which can be selected from any location on the computer) as shown in Figure 17. When the specification file is selected, the compiler (referred to as conflict analyzer) compiles the file for potential

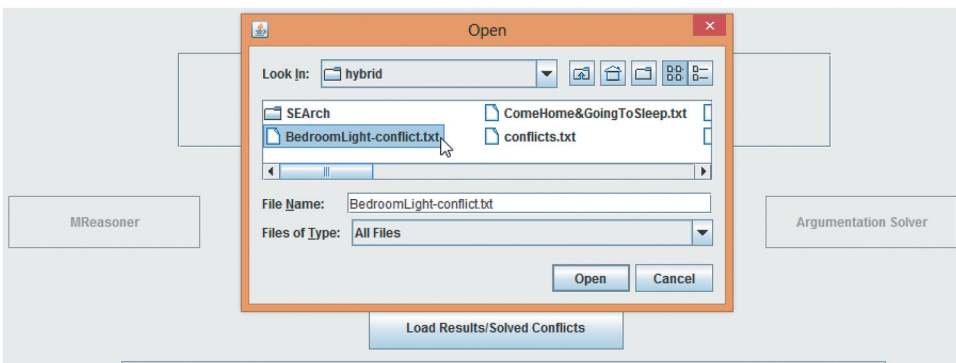


Figure 16. Hybrid interface.

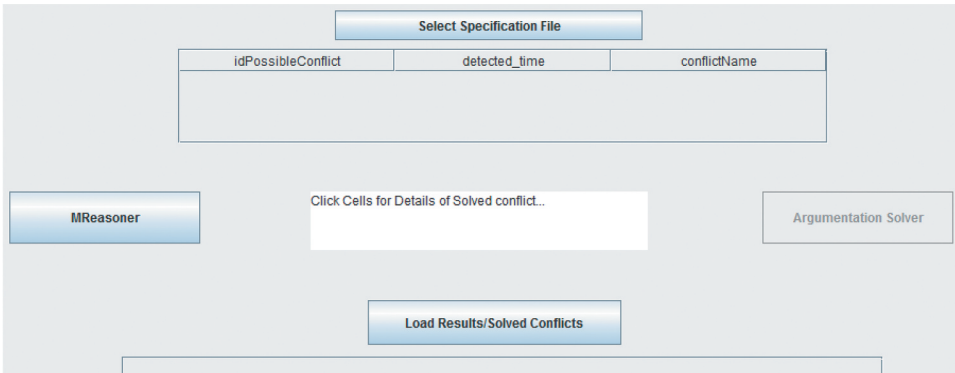


Figure 17. Browsing to select specification file.

conflicts. If no conflict is detected, the “MReasoner” button is enabled (allowing the system to run the specification file without the involvement of the conflict analyzer) and the “Argumentation Solver” button stays disabled as shown in [Figure 18](#). If potential conflict(s) is/are detected, the “Argumentation Solver” button is enabled and the potential conflict(s) is displayed in the text area as shown in [Figure 19](#). The system can now run the file and solve any actual conflict from the potential ones.

[Figure 19](#) shows three potential conflicts that were detected after the specification file (“BedroomLight-conflict.txt”) was selected, but only the detected conflict(s) among them was solved during execution. Meanwhile, to

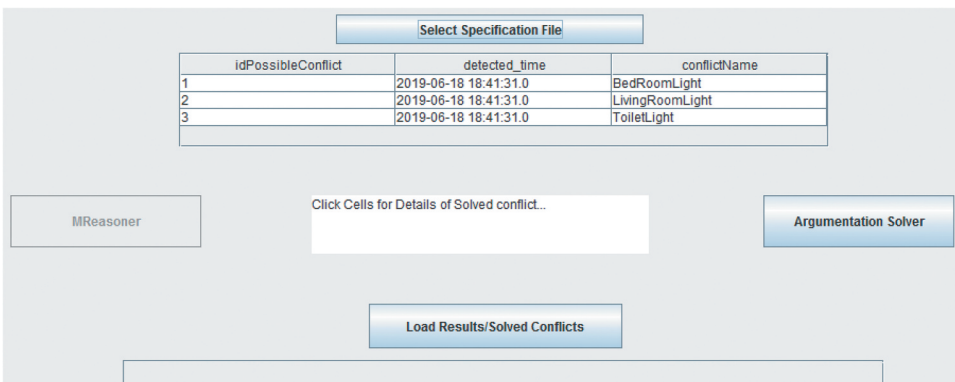


Figure 18. The MReasoner button is enabled as ‘NO’ potential conflict is detected.

Select Specification File

idPossibleConflict	detected_time	conflictName
1	2019-09-02 12:19:05.0	LivingRoomLight
2	2019-09-02 12:19:05.0	ToiletLight
3	2019-09-02 12:19:05.0	BedRoomLight

This detected conflict was solved using [User Preferences](#)

Load Results/Solved Conflicts

iter...	system_t...	Living...	LivingR...	ToiletLi...	ToiletM...	BedRoomLight	BedRoomMotion	BigPad...	prefLi...	prefC...
18	1567423...	false	false	false	false	false	true	false	true	true
19	1567423...	false	false	false	false	false	true	false	true	true
20	1567423...	false	false	false	false	false	true	false	true	true
21	1567423...	false	false	false	false	false	true	false	true	true
22	1567423...	false	false	false	false	false	true	false	true	true
23	1567423...	false	false	false	false	false	false	false	true	true
24	1567423...	false	false	false	false	false	false	false	true	true
25	1567423...	false	false	false	false	false	false	false	true	true
26	1567423...	false	false	false	false	false	false	false	true	true
27	1567423...	false	false	false	false	false	false	false	true	true
28	1567423...	false	false	false	false	false	false	false	true	true
29	1567423...	false	false	false	false	false	false	false	true	true
30	1567423...	false	false	false	false	false	false	false	true	true
31	1567423...	false	false	false	false	false	false	false	true	true
32	1567423...	false	false	false	false	false	false	false	true	true

Figure 19. The argumentation solver button is enabled as potential conflict is detected.

check for conflict(s), the compiler only compiles the last part of the specification file that consists of the rules. Below are the rules that were compiled in this case:

```

ssr((<->[12:00:00–18:00:00]BedRoomMotion ^ BigPadIdle) ->
BedRoomLight);
ssr((LivingRoomMotion) -> LivingRoomLight);
ssr((#LivingRoomMotion) -> #LivingRoomLight);
ssr((ToiletMotion) -> ToiletLight);
ssr((#ToiletMotion) -> #ToiletLight);
ssr(([-][30s.]#BigPadIdle ^ #BedRoomMotion ^ prefLight) ->
BedRoomLight);
ssr(([-][30s.]#BigPadIdle ^ #BedRoomMotion ^ prefComfort) ->
#BedRoomLight);

```

The three potential conflicts from the above rules are related to conclusions involving: *BedRoomLight*, *LivingRoomLight* and *ToiletLight*. However, *LivingRoomLight* and *ToiletLight* are only potential conflicting, as the consequence opposes each other. Here, the property *BedRoomLight*, has been detected as a conflict, as both rules states that if the pressure pad is “not” idle for 30 seconds ($[-][30s.]#BigPadIdle$) and no movement detected in the bedroom ($#BedRoomMotion$), then the bedroom light being either “on” or “off,” will be



**MODIFY SELECTED
PREFERENCE(S)**

For User: Sara

Health 3 ▾

Heating 3 ▾

Light 6 ▾

Security Choose ▾

Safety 7 ▾

Pleasure 5 ▾

Fun Choose ▾

News 7 ▾

Football 6 ▾

Comfort 7 ▾

SUBMIT RESET

Figure 20. Interface showing that the user prioritized comfort over light.

decided based on the preference ranking of the user. If the user ranks “Comfort” (*prefComfort*) higher than “Light” (*prefLight*), then the bedroom light goes “off” (*#BedRoomLight*) else, the bedroom light stays “on” (*BedRoomLight*).

The scenario was executed in the real environment, as the events are triggered with either sensors and/or actuators. [Figure 20](#) indicates that the user has set his/her priority to prefer “Comfort” over “Light,” in this case. This means that when the conflict is detected, the system first tries to resolve the argument with “Specificity,” which will not be possible, as both rules are equally specific. The system will then try to resolve the argument using “Preference,” and from [Figure 20](#), “Comfort” has higher priority over “Light.” So *#BedRoomLight* wins the argument, and the system switches “off” the bedroom light when the user is on the bed for more than 30 seconds and no movement is detected in the bedroom.

The Hybrid interface also has the ability to populate the results of the properties value, and also pinpoint the exact instant or interval the conflict(s) were identified and solved. The Hybrid system also provides the details of how the conflict(s) was/were solved. During or after execution the results are display using the “Load Results/Solved Conflicts” button. [Figure 21](#) shows

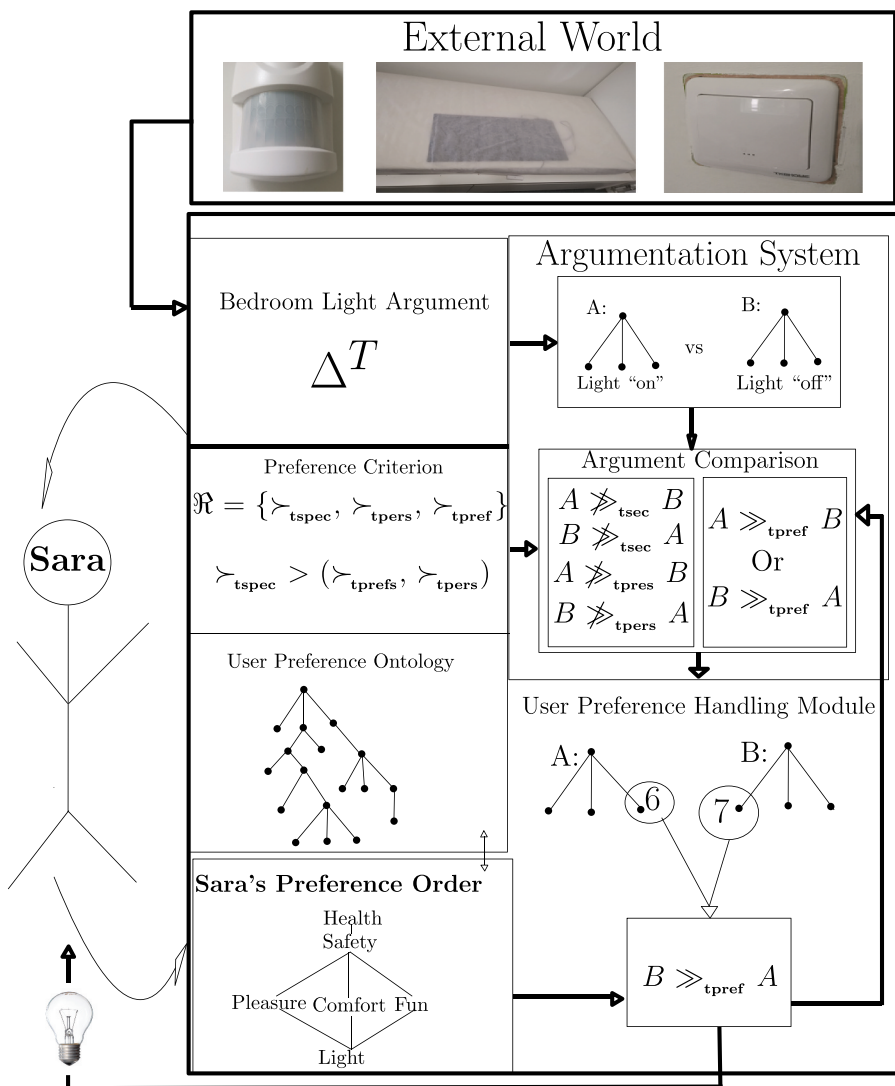


Figure 21 Hybrid system highlighting the columns where conflict was detected and solved.

results which are loaded on the below text area of the Hybrid interface, and the highlighting identifies the areas where conflicts were detected and solved immediately.

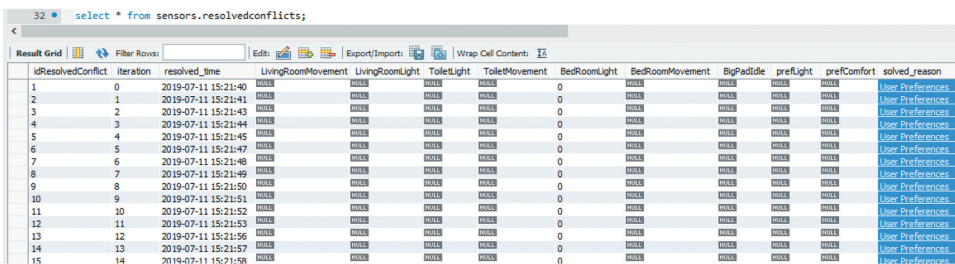
Furthermore, clicking on any of the highlighted cells, additional information on how the conflict was solved at that instant will be provided. Since *BedRoomLight* was the conflicting state/property, and the argument was solved using “user preference,” the system highlights the conflicting cells within the *BedRoomLight* column. When any of the cell is clicked, the reason

how the argument (*BedRoomLight*) was solved, gets displayed in the middle text-area of the Hybrid interface, as shown in [Figure 21](#). If any other area (with no highlighting) is clicked, the text area will display “No conflict detected”

We further applied this argument (*BedRoomLight*) to our preference architecture (shown in [Figure 1](#)) which we introduced in our previous work Oguego et al. (2019 b), to illustrate how our produced system functions internally. [Figure 2](#) shows how the argument was fully applied to the preference architecture, and how the bedroom light conflict was solved using the preference ranking (in [Figure 20](#)) of the user (Sara).

As shown in [Figure 2](#), compared to the overall preference architecture in [Figure 1](#), the “*External world*” where information comes into the system from the outside, was replaced by the equipment in the bedroom. The equipment consists of the movement sensor, which detects if the user is present in the bedroom or not. The pressure pad (known as *BigPadIdle*), placed underneath the mattress, detects that the user has been on the bed continuously for the past 30 seconds ($[-][30s.]\#BigPadIdle$), and the light switch automatically goes “on” or “off” depending on Sara’s preference ranking.

Since the system could not decide whether to keep the bedroom light “on” or “off,” a conflict resolution process had to take place. From [Figure 2](#), the system considers the arguments as in “A” (Bedroom Light “on”) or in “B” (Bedroom Light “off”) as shown in the top right side of the architecture. The system then runs the check using Specificity (as shown in “Argument Comparison”) and from the rules “A” is not more specific than “B” ($A_{tsec}B$) and vice versa ($B_{tsec}A$). The system then moves to the next preference criterion, which according to the order of precedence, is “user preferences.” The system then runs another check, in the “User Preference Handling Module,” where the system checks the database to access the user (Sara) preference ranking order, for “Comfort” and “Light.” From the bottom left side of the figure ([Figure 2](#)), it shows that Sara ranked “Comfort” higher than “Light,” also shown on her preference profile in [Figure 20](#). The profile indicates that Sara ranked “Comfort – 7”



idResolvedConflict	iteration	resolved_time	LivingRoomMovement	LivingRoomLight	ToiletLight	ToiletMovement	BedRoomLight	BedRoomMovement	BigPadIdle	preLight	preComfort	solved_reason
1	0	2019-07-11 15:21:40	idle	idle	idle	idle	0	idle	idle	idle	idle	User Preferences
2	1	2019-07-11 15:21:41	idle	idle	idle	idle	0	idle	idle	idle	idle	User Preferences
3	2	2019-07-11 15:21:43	idle	idle	idle	idle	0	idle	idle	idle	idle	User Preferences
4	3	2019-07-11 15:21:44	idle	idle	idle	idle	0	idle	idle	idle	idle	User Preferences
5	4	2019-07-11 15:21:45	idle	idle	idle	idle	0	idle	idle	idle	idle	User Preferences
6	5	2019-07-11 15:21:47	idle	idle	idle	idle	0	idle	idle	idle	idle	User Preferences
7	6	2019-07-11 15:21:48	idle	idle	idle	idle	0	idle	idle	idle	idle	User Preferences
8	7	2019-07-11 15:21:49	idle	idle	idle	idle	0	idle	idle	idle	idle	User Preferences
9	8	2019-07-11 15:21:50	idle	idle	idle	idle	0	idle	idle	idle	idle	User Preferences
10	9	2019-07-11 15:21:51	idle	idle	idle	idle	0	idle	idle	idle	idle	User Preferences
11	10	2019-07-11 15:21:52	idle	idle	idle	idle	0	idle	idle	idle	idle	User Preferences
12	11	2019-07-11 15:21:53	idle	idle	idle	idle	0	idle	idle	idle	idle	User Preferences
13	12	2019-07-11 15:21:56	idle	idle	idle	idle	0	idle	idle	idle	idle	User Preferences
14	13	2019-07-11 15:21:57	idle	idle	idle	idle	0	idle	idle	idle	idle	User Preferences
15	14	2019-07-11 15:21:58	idle	idle	idle	idle	0	idle	idle	idle	idle	User Preferences

Figure 22. Database showing how the conflict was solved using the preference criterion, “User Preferences”

(argument “*B*”) and “Light – 6” (argument “*A*”), which will allow the system to turn the bedroom light “off,” thereby solving the conflict with “*B*” winning the argument using user’s preferences ($B \gg_{\text{tpref}} A$).

Figure 22 further depicts the database records, and the last column indicating the reason (“User Preferences”) the system applied in solving the bedroom light conflict.

The following link (Oguego (2019a)) contains a video demonstration of the Bedroom conflict scenario, as a supporting evidence of the explanation and illustration made in this section. We have also provide the data set result of the experiment, to indicate details of the full output of the validation process.

This research further conducted a supplementary demonstration in Section 8.2, to show that our system is able to detect and solve all three preference criteria earlier discussed.

Solving Conflicts Using Three Preference Criteria (Specificity, User Preferences, and Persistency)

A specification file was written to trigger potential conflicts in all three areas of the preferences criteria, as the intention was to illustrate that our system is capable enough to detect conflicts at any time, even at the same interval and solve them using any of the preference criteria. Below is the specification file with the rules, which consist of three potential conflicts in relation to the preference criteria:

```

states(BedroomMotion,          BedRoomLight,          ShowerMotion,
ShowerRoomLight,  ToiletMotion,  ToiletLight,  CorridorMotion,
CorridorLight, BigPadIdle, prefComfort, prefLight);
is(CorridorMotion); is(ShowerMotion); is(BigPadIdle); is(ToiletMotion);
is(BedroomMotion); is(prefComfort); is(prefLight);
holdsAt(#CorridorMotion, 0);
holdsAt(#CorridorLight, 0);
holdsAt(#BedRoomLight, 0);
holdsAt(#BedroomMotion, 0);
holdsAt(BigPadIdle, 0);
holdsAt(#ToiletLight, 0);
holdsAt(#ToiletMotion, 0);
holdsAt(#ShowerRoomLight, 0);
holdsAt(#ShowerMotion, 0);
    
```

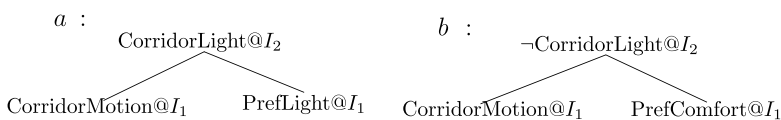


Figure 23. Argument tree for corridor-light “on” or “off.”

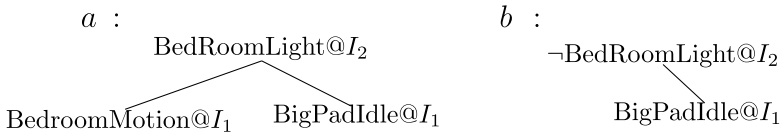


Figure 24. Argument for bedroom-light.

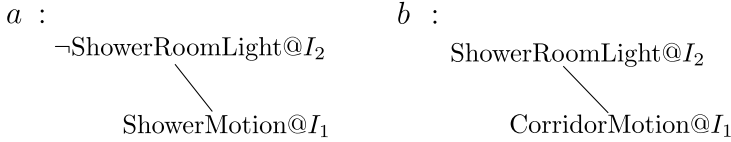


Figure 25. Argument for shower room-light.

```

holdsAt(prefComfort, 0);
holdsAt(prefLight, 0);
ssr((CorridorMotion ^ prefLight) -> CorridorLight);
ssr((CorridorMotion ^ prefComfort) -> #CorridorLight);
ssr((BedroomMotion ^ BigPadIdle) -> BedRoomLight);
ssr((BedroomMotion) -> #BedRoomLight);
ssr((ToiletMotion) -> ToiletLight);
ssr((ShowerMotion) -> ShowerRoomLight);
ssr((CorridorMotion) -> #ShowerRoomLight);

```

The rules were created to check for the “User Preference” aspect of conflict, as the term *prefLight* and *prefComfort* indicate the preference aspect, which triggers either the corridor light “on” (*CorridorLight*) or the corridor light “off” (*#CorridorLight*). The rules also checked for the “Specificity” aspect of the conflict, as the potential conflict of bedroom light is determined based on which of the arguments (*BedRoomLight*” or “*#BedRoomLight*”) is more

idPossibleConflict	detected_time	conflictName
1	2019-06-18 18:57:45.0	CorridorLight
2	2019-06-18 18:57:45.0	BedRoomLight
3	2019-06-18 18:57:45.0	ShowerRoomLight

Figure 26. Identified potential conflicts for sara.

specific or informed. The system also checked for “*Persistency*” notion, to know if the property (state) keeps the true value over time when there is no reason for the property to change its value, unless there is/are reason(s) to believe otherwise.

We applied Sara’s preference ranking order (discussed in Section 7.3, [Figure 14](#)) for the demonstration of this scenario. Meanwhile, [Figure 23,24,25](#) illustrates the argument of all three potential conflicts in argumentation tree form, using the “*MEETS*” interval relation initially discussed.

When the system is in execution, it processes the specification file and checks for potential conflict(s). From the rules, the potential conflicts are *CorridorLight*, *BedRoomLight*, and *ShowerRoomLight* and are stored in the potential conflict table, and display on the Hybrid Interface (shown in [Figure 26](#)). Each time a new specification file is processed, it erases the previous record(s) in the potential conflict table and saves the current potential conflict(s) identified (if any, otherwise the potential conflict table will remain empty).

The first potential conflict (*CorridorLight*) is an actual conflict, as the system does not know whether to turn “on” or turn “off” the corridor light. However, this depends on which of the preferences (“*Light*” or “*Comfort*”) has higher priority. For this scenario, Sara’s preference ranking order shown [Figure 14](#) was adopted, as *prefLight* was assigned the value 6 and *preComfort* was assigned the value 4. The corridor light was turned “on” as *CorridorLight* won the argument based on *prefLight* having higher priority over *preComfort*.

The second potential conflict (*bedroom light*) was decided based on “Specificity,” so *BedRoomLight* won the argument over *#BedRoomLight*. This is because the argument (*BedRoomLight*) had additional information (“*BedroomMotion*”) that should supports the argument of turning the light “on.”

The argument representation tree of the “*Bedroom Light*” (as shown in [Fig. 24A and 24B](#)), further explains why argument “A” wins the argument based on specificity, with tree “A” having additional information than tree “B.” “*BedroomMotion*” is a motion sensor ([Figure 9A](#)) which is used to detect movement around the bedroom, along with the pressure pad being idle (*BigPadIdle*, shown in [Figure 9E](#)), will keep the light “on.”

The current value for the shower room light persists which is *#ShowerRoomLight*, meaning that the shower room light remains “off.” Since both “Specificity” and “User Preferences” cannot solve the conflict, the property (“Shower-room Light”) retains the previous value of keeping the light “off,” unless there is an inference of new information into the system. The previous value in this case is “off” (*holdsAt(#ShowerRoom – Light, 0)*;) as shown in the specification file. This signifies that the value of the “Shower-Room Light” property at the starting point or initial state, was “off.”

Note, all rules follows the order of precedence ($\succ_{tspec} > \succ_{Upref(a)} > \succ_{tpers}$) in trying to solve any conflict, regardless of how the specification file is written. This means any detected conflict first tries to be solved using “Specificity” and if it cannot be solved, the system then tries to use “User Preferences.” If the conflict cannot be solved using “User Preferences” (maybe because the Preference properties are equally ranked), it then continues to keep the property’s true value (“Persistence”).

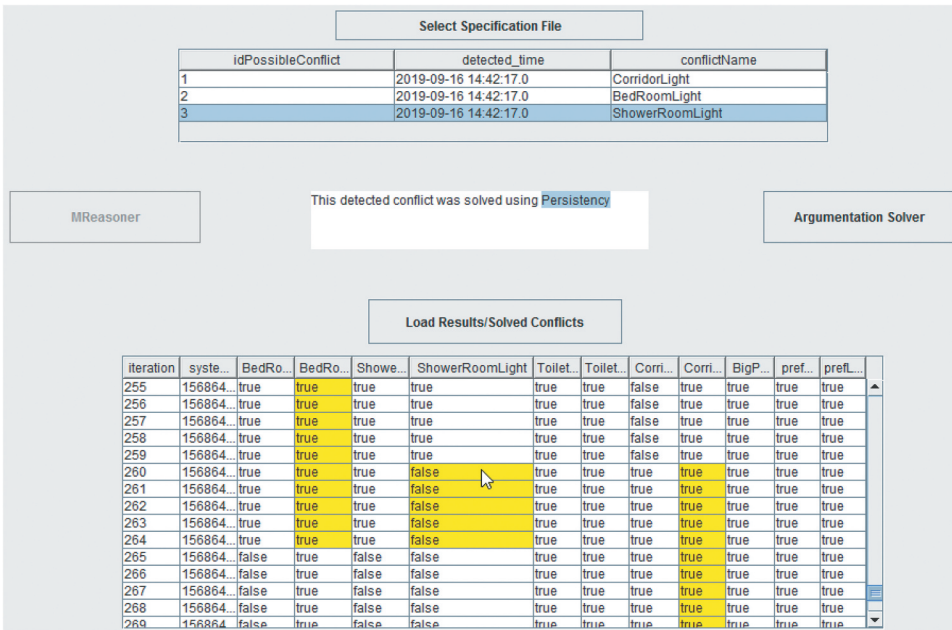


Figure 27. Hybrid system showing all three detected and solved of conflicts; “specificity,” “user preferences” and “persistence.”

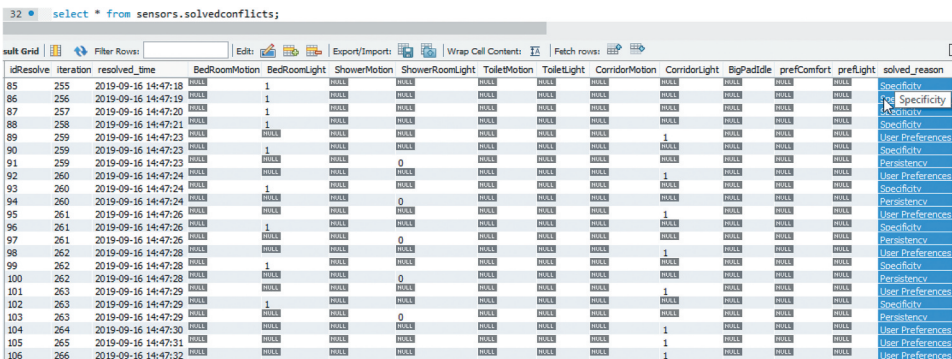


Figure 28. Database showing all three types of conflicts; “specificity,” “preference” and “persistence.”

Figure 27 shows the intervals (highlighted) where conflicts were detected and solved for this scenario. As seen from the screenshot, selecting a column from the bedroom light row, the reason (“Specificity”) used in solving the conflict is displayed in the middle text area. The figure also illustrates that conflicts were solved on other properties (“Shower-room Light and Corridor Light”) as well, which were solved with persistency and user preferences, respectively.

Figure 28 illustrate some of the the database log of the solved conflict. The “iteration” column states the exact iterations where the conflicts were detected and solved, the properties columns (Bedroom Light, Shower room Light, and Corridor Light) display either a new conclusion or retain the previous value. The values in the database indicating 1 or 0, which represent true or false displayed on the Hybrid interface. The last column (“resolve_reason”), depicts the reason the Hybrid system was used to resolve the conflict. In addition, the system is able to solve multiple conflicts at the same time, using any or all of the preference criteria in the iteration.

The demonstration link (Oguego (2019c)) indicates the illustration discussed above, using the aforementioned specification file in this section. Attached in the same link is the complete data set, showing more logs of the detected conflicts and how they were solved, applying the preference criteria where necessary. The validation was conducted for 2 hours.

Supermarket Chain Store (Tesco) API

The research took another step to validate the effectiveness of the Hybrid system using live data. The live data were from one of the top supermarkets in the United Kingdom, known as Tesco. We requested for the API on their grocery products, which was used to filter “Cake” product, and check if the product description contains sugar. The aim was to warn the user, Sara, who is known to be diabetic, about the content of the Cake product, but it is Sara’s decision to buy the Cake or not. The system also identifies the Cake products that do not contain sugar, which gives Sara more options of deciding to buy them or not.

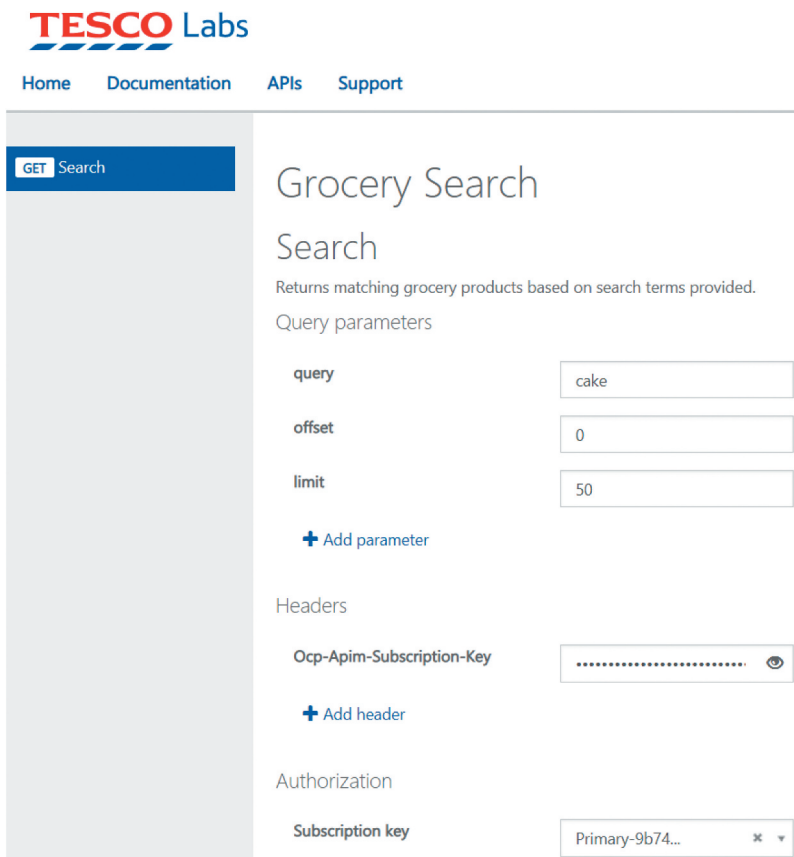
So based on the users’ ranking preference (Sara in this case), since she prefers health (*prefHealth*) over pleasure (*prefPleasure*) as seen in Figure 14, the system should advise her “not” to buy the cake (*#Occ_SystemAdvicesBuyCake*). If for some reason Sara changes her preference ranking of preferring “Pleasure” over “Health,” the system will then advise the user to buy the cake (*Occ_SystemAdvicesBuyCake*). In addition, if it happens that all the filtered cake product do not contain sugar, Sara can equally choose to (or not to) order from any of the available cake products that do not contain sugar and vice versa.

Considering the healthy eating case study in Section 2, the below specification file with rules was developed to check for the availability of a particular product, “Cake.” If found, the rules check the product description (details of the cake) for sugar, and then advises the user depending on her preference ranking.

```

states(BuyCake, Diabetic, CakeOnSales, Sugar, Occ_CakeAvaliable,
Occ_SugarDetected, Occ_SystemAdvicesBuyCake, prefPleasure, prefHealth);
is(Occ_CakeAvaliable);
is(Occ_SugarDetected); is(Diabetic); is(prefPleasure); is(prefHealth); is
(Occ_SystemAdvicesBuyCake);
holdsAt(#BuyCake, 0);
holdsAt(Diabetic, 0);
holdsAt(#CakeOnSales, 0);
holdsAt(#Sugar, 0);
holdsAt(Occ_CakeAvaliable, 0);
holdsAt(#Occ_SugarDetected, 0);
holdsAt(#Occ_SystemAdvicesBuyCake, 0);

```



TESCO Labs

[Home](#) [Documentation](#) [APIs](#) [Support](#)

GET Search

Grocery Search

Search

Returns matching grocery products based on search terms provided.

Query parameters

query	<input type="text" value="cake"/>
offset	<input type="text" value="0"/>
limit	<input type="text" value="50"/>

[+ Add parameter](#)

Headers

Ocp-Apim-Subscription-Key	<input type="text" value="....."/>
---------------------------	------------------------------------

[+ Add header](#)

Authorization

Subscription key	<input type="text" value="Primary-9b74..."/>
------------------	--

Figure 29. Requesting for Tesco URL to search for Cake Product.

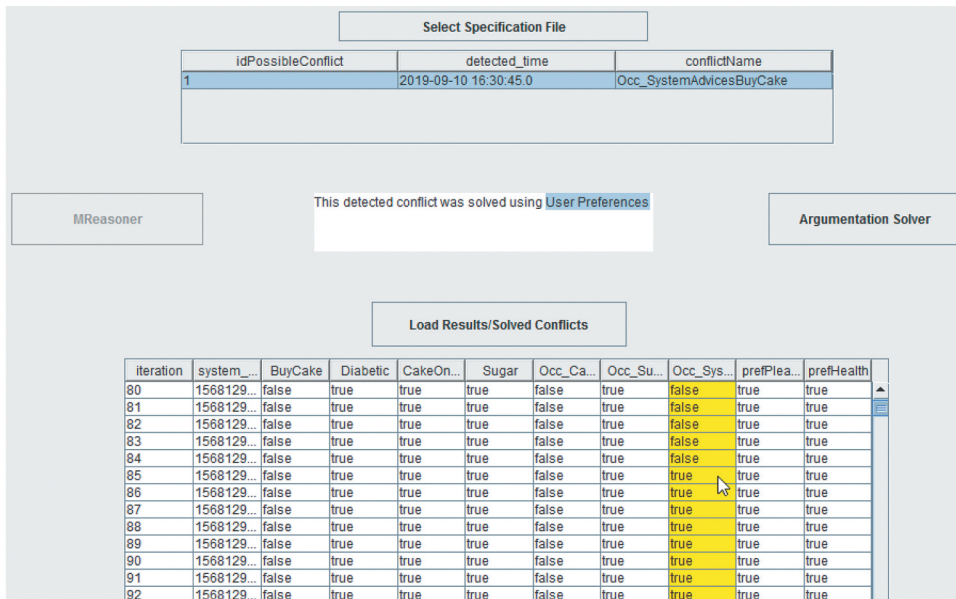


Figure 30. System advice sara not to buy cake since her “**health**” has higher priority over “**pleasure.**”

```

holdsAt(prefHealth, 0);
holdsAt(prefPleasure, 0);
ssr((Occ_CakeAvaliable) -> CakeOnSales);
ssr((CakeOnSales ^ prefPleasure) -> Occ_SystemAdvicesBuyCake);
ssr((Occ_SugarDetected) -> Sugar);
ssr((Diabetic^Sugar^CakeOnSales^prefHealth) ->
#Occ_SystemAdvicesBuyCake);

```

Figure 29 depicts how the link to the data is generated from Tesco Labs. According to the search parameter, the product to be queried needs to be entered (Cake in this case), the “offset” indicates where the search should commence from. If the “offset” is 10, the search result is produced from the 11th product, and the “limit” is how many products you want to limit the search to. This can be any number, 12, 50, 67, or 500 (which is the maximum at a time). When these parameters have been set, it will generate a url which will be used (along with a private subscription key) to access the filtered product.

Figure 30 illustrates the Hybrid interface after system’s execution. The specification file is first compiled to check for potential conflict(s) (*Occ_SystemAdvicesBuyCake* in this case) as shown in the Figure 30, but, the system is yet to advise Sara to buy the Cake or “not.” During execution, the Hybrid system accesses the URL online to check for the availability of the product, Cake. If Cake is available, it means the Cake is up for sale at that moment. The system then checks from the list of Cakes available, to know if

the product description contains sugar. If sugar is found in the description, the system advises Sara “not” to buy the product (as seen from the scenario) due to her health condition, in addition to her preference priority of Health (*prefHealth*) over Pleasure (*prefPleasure*), as seen in Figure 14.

Considering the rules on the specification file for the Tesco API, one might ask why “User Preference” criteria was used to solve the conflict instead of “Specificity.” “Specificity” as we know (when comparing arguments), is a way of preferring the best-informed argument. Specificity is also based on the structure of the arguments, and when the argument is incomparable or equi-specific Augusto and Simari (2001), the system will then apply the next preference criterion (user preference in this case). The argument *Occ_SystemAdvicesBuyCake*, is incomparable because one of the arguments has a unique property the other argument does not have. So, this cannot be used to decide which one is preferable. However, if both arguments were as follows:

```

    srr((CakeOnSales ^ prefHealth) -> #Occ_SystemAdvicesBuyCake);
    srr((Diabetic^Sugar^CakeOnSales^prefHealth) ->
    Occ_SystemAdvicesBuyCake);
  
```

The notion “Specificity” will be applied in this case, as *Occ_SystemAdvicesBuyCake* will win, as the argument is more informed than the other argument. Since the argument for *#Occ_SystemAdvicesBuyCake* does not contain any supporting property the argument for *Occ_SystemAdvicesBuyCake* does not have, “Specificity” criterion can be used

S/N	product_id	product_name	super_department	department	description	content_que	unit_price	price	retrieved_date	type
55	283390061	Cadbury Chocolate Mini Roll 5 Pack	Bakery	Cakes, Cake Bars, ...	Chocolate flavoured sponges with a vanilla flavour creme, covere...	5	0.16	0.8	2019-09-10 16:31:42	No sugar
56	34869712	Tesco Almond Fingers 5 Pack	Bakery	Cakes, Cake Bars, ...	Almond flavoured sponges cakes topped with soft almond...	5	0.2	1	2019-09-10 16:31:42	No sugar
57	368322208	Tesco Mini Chocolate Cornflake Bit...	Bakery	Cakes, Cake Bars, ...	1/5 Cornflake clusters covered in milk chocolate. Made with Milk...	15	0.12	1.8	2019-09-10 16:31:42	No sugar
58	262780947	Almond Daim Chocolate Cake 400g	Frozen Food	Frozen Desserts, I...	Chocolate cake with Daim milk chocolate coated almond caram...	0	0.75	3	2019-09-10 16:31:42	No sugar
59	29685241	Micvieles Diastive Caramel Slice 5...	Bakery	Cakes, Cake Bars, ...	Diastive slices Topped with Caramel & Milk Chocolate/whew... 124	0.579	0.72	2019-09-10 16:31:42	No sugar	
60	295848818	Cadbury Raspberry Mini Roll 10 Pack	Bakery	Cakes, Cake Bars, ...	Sponge cakes with plum and raspberry jam and a vanilla flavo...	10	0.25	2.5	2019-09-10 16:31:42	No sugar
61	287501432	Cadbury Raspberry Mini Roll 5 Pack	Bakery	Cakes, Cake Bars, ...	Sponge cakes with plum and raspberry jam and a vanilla flavo...	5	0.16	0.8	2019-09-10 16:31:42	No sugar
62	268720993	Tesco Crispy Caramel Bites 20 Pack	Bakery	Cakes, Cake Bars, ...	Caramel and crisped rice bites bart coated in milk chocolate. M...	20	0.09	1.8	2019-09-10 16:31:42	No sugar
63	29757300	Tesco Cookies & Cream Cake	Bakery	Cakes, Cake Bars, ...	Chocolate and maderia cake filled and covered with cookie cru...	1	13.0	13	2019-09-10 16:31:42	No sugar
64	29754495	Micvieles Jaffa Cakes 10 Pack	Food Cupboard	Biscuits & Cereal Bars	10 Light Sponges Cakes with Dark Cradly Chocolate and a Sma...	10	0.05	0.5	2019-09-10 16:31:42	No sugar
65	251816699	Mr Kollno Mini Battenberg Cakes 5...	Bakery	Cakes, Cake Bars, ...	Cheured Spones Sandwiched Together with an Apricot Filling...	5	0.164	0.82	2019-09-10 16:31:42	No sugar
66	283313228	Tesco Rose Boudent Cake	Bakery	Cakes, Cake Bars, ...	Sponge cake filled with rasperry jam and frosting, covered an...	1	11.0	11	2019-09-10 16:31:42	No sugar
67	382051974	Honeers Chocolate Mini Rolls 10 Pack	Bakery	Cakes, Cake Bars, ...	Chocolate flavour sponges rolls with vanilla flavour filling and...	10	0.1	1	2019-09-10 16:31:43	No sugar
68	300120295	Ms Molvie 12 Iced Fairy Cakes	Bakery	Cakes, Cake Bars, ...	12 Spones cakes with plain, lemon or strawbery flavour icin...	12	0.063	0.75	2019-09-10 16:31:43	No sugar
69	300923295	M&M's Chocolate Biscuit Celebrat...	Bakery	Cakes, Cake Bars, ...	Chocolate spones filled and covered with a chocolate flavou...	1	11.0	11	2019-09-10 16:31:43	Sugar
70	301816286	Micvieles Jaffa Cakes Strawbery 1...	Food Cupboard	Biscuits & Cereal Bars	10 Light Spones Cakes with Dark Cradly Chocolate and a Str...	10	0.05	0.5	2019-09-10 16:31:43	No sugar
71	284508478	Micvieles Jaffa Cake Bars 5 Pack	Bakery	Cakes, Cake Bars, ...	A scrumptious blend of luscious dark cradly chocolate, light so...	10	0.25	2.5	2019-09-10 16:31:43	No sugar
72	300120998	Ms Molvie 12 Fairy Cakes	Bakery	Cakes, Cake Bars, ...	12 Spones cakes. Auto smadnoly lovely spones cakes perfect...	12	0.063	0.75	2019-09-10 16:31:43	No sugar
73	299847773	Ms Molvie Chocolate Fairy Cakes 1...	Bakery	Cakes, Cake Bars, ...	12 Chocolate spones cakesAuto smadnoly lovely spones cakes...	12	0.063	0.75	2019-09-10 16:31:43	No sugar
74	25158848	Galaxy Cake Bars 5 Pack	Bakery	Cakes, Cake Bars, ...	Spones cake bars with a chocolate cream centre covered in mil...	5	0.3	1.5	2019-09-10 16:31:43	No sugar
75	25249773	Real Lancashire Eccles Cakes	Bakery	Cakes, Cake Bars, ...	Eccles cakesFor further details please go to our web site www...	4	0.4	1.6	2019-09-10 16:31:43	No sugar
76	257612038	Tesco Small Chocolate Celebration...	Bakery	Cakes, Cake Bars, ...	Chocolate spones cake filled and covered with chocolate cana...	1	6.0	6	2019-09-10 16:31:43	No sugar
77	200793235	Tesco Jaffa Cakes Twin Pack 2020	Food Cupboard	Biscuits & Cereal Bars	24 Soft baked cakes and an orange centre, coated in dark cho...	282	0.337	95	2019-09-10 16:31:43	No sugar
78	291369990	Tesco Jaffa Cake 430g	Frozen Food	Frozen Desserts, I...	Baked orange filling on a spones base, topped with orange fla...	430	0.465	2	2019-09-10 16:31:43	No sugar
79	257373377	Tesco Hoova Bir'hdav Cake	Bakery	Cakes, Cake Bars, ...	Madeira spones cake filled with rasperry jam and buttercrea...	1	8.5	8.5	2019-09-10 16:31:43	No sugar
80	252815238	Mr Kollno Battenberg Cake	Bakery	Cakes, Cake Bars, ...	Cheured Spones Sandwiched Together with an Apricot Filling...	1	1.0	1	2019-09-10 16:31:43	No sugar
81	25944964	Tesco Stars Party Cake	Bakery	Cakes, Cake Bars, ...	Madeira spones cake filled with buttercream and rasperry va...	1	6.7	6.7	2019-09-10 16:31:44	Sugar
82	261722967	Tesco Triple Layer Chocolate Cake	Bakery	Cakes, Cake Bars, ...	Three layers of chocolate spones cake, filled and covered with...	1	12.0	12	2019-09-10 16:31:44	No sugar
83	250215467	Tesco Football Cake	Bakery	Cakes, Cake Bars, ...	Madeira spones cake filled with rasperry jam and buttercrea...	1	8.0	8	2019-09-10 16:31:44	Sugar
84	292702098	Tesco Party Cake Selection 12 Pack	Bakery	Cakes, Cake Bars, ...	A Vanilla flavoured spones cakes topped with icing and decorat...	12	0.15	1.8	2019-09-10 16:31:44	Sugar
85	250221275	Galaxy Caramel Cake Bars 5 Pack	Bakery	Cakes, Cake Bars, ...	Spones cake bars with a caramel centre covered in milk chooc...	5	0.3	1.5	2019-09-10 16:31:44	No sugar
86	299370938	Micvieles Jaffa Cakes Twin Pack	Food Cupboard	Biscuits & Cereal Bars	20 Light Spones Cakes with Dark Cradly Chocolate and a Sma...	244	0.656	1.6	2019-09-10 16:31:44	No sugar
87	263501840	Horlicks Cake Co. Mixed Fruit Loaf...	Bakery	Cakes, Cake Bars, ...	Made with apple slice (from concentrate).	1	2.25	2.25	2019-09-10 16:31:44	No sugar
88	300795204	Tesco Madeira Party Cake	Bakery	Cakes, Cake Bars, ...	Madeira spones cake with rasperry jam, filled and topped wit...	1	6.0	6	2019-09-10 16:31:45	No sugar
89	258871840	Micvieles Jaffa Cake Bars 5 Pack	Bakery	Cakes, Cake Bars, ...	A blend of dark cradly chocolate, light spones and smadino so...	5	2.9	1.45	2019-09-10 16:31:44	No sugar
90	291598490	Tesco Free From Carl The Caterall...	Bakery	Cakes, Cake Bars, ...	Free from chocolate cake filled with chocolate, flavoured frost...	1	6.0	6	2019-09-10 16:31:44	Sugar
91	265212965	Tesco Madeira Party Cake	Bakery	Cakes, Cake Bars, ...	Madeira spones cake with rasperry jam, filled and topped wit...	1	6.0	6	2019-09-10 16:31:45	No sugar
92	271188921	Dr. Oetker Party Candles 18	Food Cupboard	Home Balans	1/8 Party Candles/Join our Webakle Community to showcase yo...	18	0.056	1	2019-09-10 16:31:45	No sugar
93	297570993	Tesco Rainbow Cake	Bakery	Cakes, Cake Bars, ...	Madeira spones cake filled and coated with multi coloured frost...	1	12.0	12	2019-09-10 16:31:45	Sugar
94	302882855	Hellens Treat Cake	Bakery	Cakes, Cake Bars, ...	Chocolate spones cake filled with a chocolate filling and...	1	8.0	8	2019-09-10 16:31:45	No sugar
95	284515383	Tesco Birthday Cake Cubes 15 Pack	Bakery	Cakes, Cake Bars, ...	1/5 Cubes of chocolate flavoured spones cake, filled with chooc...	15	0.667	10	2019-09-10 16:31:45	No sugar
96	300016978	Tesco Pink Flamingo Cake	Bakery	Cakes, Cake Bars, ...	Pink spones cake layered with strawbery jam, covered with cr...	1	12.0	12	2019-09-10 16:31:45	Sugar
97	264963216	Tesco Celebration Cake	Bakery	Cakes, Cake Bars, ...	Madeira spones cake filled with buttercream and rasperry va...	1	8.5	8.5	2019-09-10 16:31:45	Sugar
98	270289844	Thomsons Celebration Cake	Bakery	Cakes, Cake Bars, ...	Chocolate Spones Filled and Covered with Chocolate Buttercre...	1	12.0	12	2019-09-10 16:31:45	No sugar
99	293943455	Emoti Celebration Cake	Bakery	Cakes, Cake Bars, ...	Celebration Cake - Spones with a layer of rasperry jam and s...	1	9.0	9	2019-09-10 16:31:45	No sugar
100	265391781	Cadbury Flake Cake	Bakery	Cakes, Cake Bars, ...	Chocolate spones layered with chocolate flavour creme and a ...	1	11.0	11	2019-09-10 16:31:45	No sugar

Figure 31. Some database records of the filtered “Cake,” with last column indicating the cake with “Sugar” or “No Sugar.”

in solving this conflict. **Figure 31** consists of all Cake products that were extracted from the filter, with an additional column to inform the user (Sara) which of the products contains “Sugar” or “not.”

A video demonstration of the Tesco API illustration is found here: Oguego (2019 d)

Discussion

The aim of this study has been to improve preference management in Ambient Assisted Living (AAL). Although there has been significant work in this area, not enough has been done to facilitate inter-relation between AAL systems and user preferences. This research has been carefully investigated, as we first conducted a survey in the state of the art to identify existing ways user preferences were handled in AAL Oguego et al. (2018a). Analysis of previous work, especially on several well-known preference handling techniques (CP-net, etc.) were conducted but they were lacking some important features useful to solve practical problems.

One feature, which is expected naturally, is the dealing of human preferences, which can conflict in terms of “desires” with “needs” as there is a battle with what we humans would like, but cannot have due to various reasons. The dynamic changing of preferences over time is also another feature, as humans can choose to decide something different at any point in time (internally) and can also be affected by external influences, such as weather information or health professionals advises.

This led to the consideration of argumentation as a possible formalism, as it has the ability to handle inconsistent information and knowledge in relation to time. We theoretically explored argumentation techniques in another study Oguego et al. (2019 b), emphasizing how it can be applied to manage users’ preferences. The investigation concluded that it is a suitable mechanism to study computational management of preferences. Other argumentation frameworks were complemented with a user preference architecture (**Figure 1**), to show how the proposed system will handle conflicting situations within arguments. The exploration conducted enabled us to validate the usefulness of argumentation as we illustrated this by applying several scenarios.

Our previous paper Oguego et al. (2019 b) further aimed to implement the system, including developing a suitable interface that facilitates preference flow, from the user to the system, with the integration of a reasoning system. This research has been able to provide the proposed practical solution using argumentation to manage users’ preferences in a real smart home.

Conclusion and Further Work

Argumentation is a powerful tool for reasoning with inconsistent knowledge and time and the demonstrations conducted enabled us to validate its potential to handle conflicting situations. As mentioned earlier, conclusions obtained by the system are ‘justified’ through ‘arguments’ supporting their consideration.

Argumentation has been utilized along with the integration of a reasoning system (MReasoner) and user preferences interface, to provide a useful tool that resolves detected conflicts in a smart home. The implemented AAL system for smart homes aims to increase users’ satisfaction, which is why it has been developed to understand and respond to the preferences of users. The system has been designed to automate and provide viable decisions for the users through effective management of users’ preferences. This research did not only aim to deliver an effective and efficient system for AAL, ease of use was a necessary factor that we considered during the development process. We provided an interface that enables users to manage their preferences easily in an intelligent environment. Users should be entitled to personalized systems according to their preferences, which should be reasonably easy for them.

Further work aims to focus on developing a mobile application version of the interface and to investigate how to generalize the management of multiple users conflicting preference(s) in the same environment simultaneously.

References

- Allen, J. F. 1984. Towards a general theory of action and time. *Artificial Intelligence* 23 (2):123–54. doi:10.1016/0004-3702(84)90008-0.
- Allen, T. E. 2014. Making CP-Nets (More) Useful. In Twenty-Eighth AAAI Conference on Artificial Intelligence, 3057–58.
- Amgoud, L., and C. Cayrol. 1998. “On the acceptability of arguments in preference-based argumentation.” In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, 1–7. San Francisco, CA, United States: Morgan Kaufmann Publishers Inc.
- Amgoud, L., and H. Prade. 2009. Using arguments for making and explaining decisions. *Artificial Intelligence* 173 (3–4):413–36. doi:10.1016/j.artint.2008.11.006.
- Amgoud, L., J.-F. Bonnefon, and H. Prade. 2005. “An argumentation-based approach to multiple criteria decision.” In *European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, 269–80. Berlin, Heidelberg: Springer.
- Augusto, J. C., and G. R. Simari. 2001. Temporal defeasible reasoning. *Knowledge and Information Systems* 3 (3):287–318. doi:10.1007/PL00011670.
- Augusto, J. C., M. Huch, A. Kameas, J. Maitland, P. McCullagh, J. Roberts, A. Sixsmith, and R. Wichert. 2012. Handbook of ambient assisted living. In *Handbook of ambient assisted living: Technology for healthcare Rehabilitation and Well-being*, 3–5. Amsterdam: IOS Press BV.
- Augusto, J. C., V. Callaghan, A. Kameas, D. Cook, and I. Satoh. 2013. “Intelligent Environments: A manifesto.” *Human-centric Computing and Information Sciences* 3 (12) doi:10.1186/2192-1962-3-12.

- Bandara, A. K., A. Kakas, E. C. Lupu, and A. Russo. 2006. Using argumentation logic for firewall policy specification and analysis. In *International workshop on distributed systems: Operations and management*, 185–96. Berlin: Springer.
- Bentahar, J., R. Alam, Z. Maamar, and N. C. Narendra. 2010. Using argumentation to model and deploy agent-based B2B applications. *Knowledge-Based Systems* 23 (7):677–92. doi:10.1016/j.knosys.2010.01.005.
- Besnard, P., and A. Hunter. 2001. A logic-based theory of deductive arguments. *Artificial Intelligence* 128 (1–2):203–35. doi:10.1016/S0004-3702(01)00071-6.
- Brafman, R., and C. Domshlak. 2009. Preference handling-an introductory tutorial. *AI Magazine* 30 (1):58–58. doi:10.1609/aimag.v30i1.2114.
- Chesñevar, C. I., A. G. Maguitman, and R. P. Loui. 2000. Logical models of argument. *ACM Computing Surveys (CSUR)* 32 (4):337–83. doi:10.1145/371578.371581.
- Galton, A. and J. C. Augusto, 2002, September. Two approaches to event definition. In *International Conference on Database and Expert Systems Applications*. 547-556. Berlin, Heidelberg: Springer.
- Garca, A. J., and G. R. Simari. 2003. “Defeasible Logic Programming: An Argumentative Approach.” *Theory and Practice of Logic Programming* 4 (1-2): 95-138.
- Goldsmith, J., and U. Junker. 2008. Preference handling for artificial intelligence. *AI Magazine* 29 (4):9–9. doi:10.1609/aimag.v29i4.2180.
- Hamblin, Charles L. 1972. “Instants and intervals.” In *The Study of Time*, 324-331. Berlin, Heidelberg: Springer.
- Ibarra, U. A., J. C. Augusto, and A. A. Goenaga. 2014. “Temporal reasoning for intuitive specification of context-awareness.” In *2014 International Conference on Intelligent Environments*, 234–41. Shanghai, China: IEEE.
- Mahesar, Q.-A. 2018. “Computing argument preferences and explanations in abstract argumentation.” In *2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI)*, 281–85. Volos, Greece: IEEE.
- Muñoz, A., and J. A. Bota. 2010. “Developing an intelligent parking management application based on multi-agent systems and semantic web technologies.” In *International Conference on Hybrid Artificial Intelligence Systems*, 64–72. Berlin: Springer.
- Muñoz, A., J. C. Augusto, A. Villa, and J. A. Bota. 2011. Design and evaluation of an ambient assisted living system based on an argumentative multi-agent system. *Personal and Ubiquitous Computing* 15 (4):377–87. doi:10.1007/s00779-010-0361-1.
- Oguego, C. L. 2019 a. “Bedroom Scenario (Hybrid section 8.1).” https://mdx.figshare.com/articles/Bedroom_Scenario_Hybrid_section_8_1_/9944603 .
- Oguego, C. L. 2019 b. “Different users’ preferences effecting system output (section 7.2).” https://mdx.figshare.com/articles/Different_users_preferences_effecting_system_output_section_7_2_/9944681 .
- Oguego, C. L. 2019 c. “Solving conflicts (section 8.2).” https://mdx.figshare.com/articles/Solving_conflicts_section_8_2_/9944711 .
- Oguego, C. L. 2019 d. “Supermarket Store (Tesco) API (section 8.3).” https://mdx.figshare.com/articles/Supermarket_Store_Tesco_API_section_8_3_/9944750 .
- Oguego, C. L., J. C. Augusto, A. Muñoz, and M. Springett. 2018a. “A survey on managing users’ preferences in ambient intelligence.” *Universal Access in the Information Society* 17 (1):97–114. doi:10.1007/s10209-017-0527-y.
- Oguego, C. L., J. C. Augusto, A. Muñoz, and M. Springett. 2018b. “Using argumentation to manage users’ preferences.” *Future Generation Computer Systems* 81:235–43. doi:10.1016/j.future.2017.09.040.

- Ospan, B., N. Khan, J. Augusto, M. Quinde, and K. Nurgaliyev. 2018. "Context aware virtual assistant with case-based conflict resolution in multi-user smart home environment." In *2018 International Conference on Computing and Network Communications (CoCoNet)*, 36–44. Maui, Hawaii, USA: IEEE.
- Pigozzi, G., A. Tsoukias, and P. Viappiani. 2016. Preferences in artificial intelligence. *Annals of Mathematics and Artificial Intelligence* 77 (3–4):361–401. doi:10.1007/s10472-015-9475-5.
- Ruzic, L., S. T. Lee, Y. E. Liu, and J. A. Sanford. 2016. "Development of universal design mobile interface guidelines (udmig) for aging population." In *International Conference on Universal Access in Human-Computer Interaction*, 98–108. Toronto, ON, Canada: Springer.
- Sartor, G. 1994. A formal model of legal argumentation. *Ratio Juris* 7 (2):177–211. doi:10.1111/j.1467-9337.1994.tb00175.x.
- Simari, G. R., and R. P. Loui. 1992. A mathematical treatment of defeasible reasoning and its implementation. *Artificial Intelligence* 53 (2–3):125–57. doi:10.1016/0004-3702(92)90069-A.
- Tamani, N., and M. Croitoru. 2014. "A quantitative preference-based structured argumentation system for decision support." In *2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 1408–15. Beijing, China: IEEE.
- Vila, L. 1994. A survey on temporal reasoning in artificial intelligence. *Ai Communications* 7 (1):4–28. doi:10.3233/AIC-1994-7102.
- Walsh, T. 2007. Representing and reasoning with preferences. *AI Magazine* 28 (4):59–59.
- Wang, H., J. Sabouné, and A. E. Saddik. 2013. "Control your smart home with an autonomously mobile smartphone." In *2013 IEEE international conference on multimedia and expo workshops (ICMEW)*, 1–6. San Jose, CA: IEEE.