Pervasive and Mobile Computing 12 (2014) 37–57

Contents lists available at ScienceDirect

Pervasive and Mobile Computing

journal homepage: www.elsevier.com/locate/pmc

Towards automatic conflict detection in home and building automation systems

Paulo Carreira[a,b,*], Sílvia Resendes[a], André C. Santos[a,b]

[a] IST – Technical University of Lisbon, Av. Prof. Dr. Aníbal Cavaco Silva, 2744-016 Porto Salvo, Portugal
[b] INESC-ID, Av. Prof. Dr. Aníbal Cavaco Silva, 2744-016 Porto Salvo, Portugal

ARTICLE INFO

Article history:
Received 23 February 2012
Received in revised form 31 May 2013
Accepted 3 June 2013
Available online 17 June 2013

Keywords:
Ambient intelligence
Home and building automation systems
Conflict detection
Conflict resolution

ABSTRACT

Home and Building Automation Systems (HBAS) are becoming of widespread adoption. When distinct users interact with such systems, their intentions are likely to be different, often resulting in conflicting situations, which the systems ought to recognize and resolve automatically. This work aims at investigating conflict in HBAS and creating a solution to detect and resolve them. Herein, we review the literature concerning conflict detection and resolution, and propose a formal framework based on constraint solving that enables detecting and solving conflict situations automatically.

1. Introduction

Home and building automation systems (HBAS) are systems that control electric equipment and devices in homes or buildings. These systems are knowing an increasing rate of adoption, mostly due to their role in energy management, helped by the dissemination of a large variety of intelligent consumer electronics equipment [1]. Still, massification of these systems is hampered by many reasons, ranging from the cost of the hardware, which is still high for the average home user (in the order of the thousands of euros), to difficulties of installation, configuration, and interaction. Nevertheless, it is safe to expect that these systems become ever more popular and offer a larger number of functionalities, some even not yet imagined. This should serve as an incentive to create truly flexible and intelligent systems, that are capable of anticipating users' intentions, and proactive in assisting users on their activities. However, this vision of intelligent home and building automation systems has been largely unmet, since many systems hardly go beyond scheduled sequences of actions and very simple sensor–actuator rules (e.g., [2]).

Another interesting aspect is that we are witnessing a shift in consumer's mentality with respect to their expectations towards electronic devices, which will eventually reach the HBA market. In consumer electronics, systems must be designed in a user-centric manner, with the needed flexibility for constant adaptation, since consumers search for more usable, easy to learn, customizable and adaptable products. If a device does not live up to this standard, the user eventually disposes of it in favour of a more sophisticated or flexible one. However, in home and building automation, this will not be the case. Consumer electronics can be disposed of and replaced easily, but replacing an HBA system is very difficult in practice. Also, users want to feel in control of their lives [3]. This evinces not being subject to wrongly working systems. Moreover, since

[*] Corresponding author at: IST – Technical University of Lisbon, Av. Prof. Dr. Aníbal Cavaco Silva, 2744-016 Porto Salvo, Portugal. Tel.: +351 966716670; fax: +351 21423508.
E-mail addresses: paulo.carreira@ist.utl.pt, pjcarreira@gmail.com (P. Carreira), silvia.resendes@tagus.ist.utl.pt (S. Resendes), acoelhosantos@ist.utl.pt (A.C. Santos).

1574-1192/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.pmcj.2013.06.001

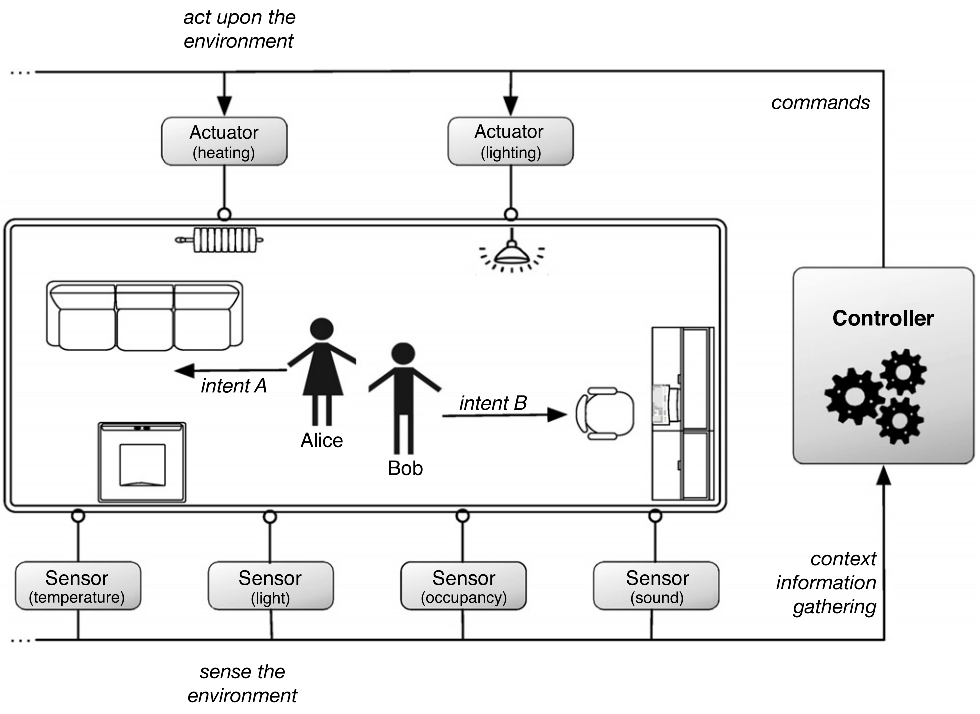

Fig. 1. Overview of an intelligent environment. The diagram depicts a smart-home with an HBAS consisting of sensors, actuators and controller modules. Within the home, users Alice and Bob have intents that enact distinct (and thus conflicting) actuations from the HBAS.

user needs and routines are constantly evolving, an HBA system should evolve along to remain useful. Therefore, built-in flexibility should be even higher.

Technological advancements in consumer electronics have also caused people to adapt and get used to a variety of equipment, cellphones being the most notable example, used in an almost unconscious way, such is the habitude and naturality with which this appliance is rooted in their daily lives. As Weiser stated, "*The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it*" [4]. Given their constant presence and participation in user's daily routines, it is of utmost importance that HBAS manages to be, as much as possible, non-intrusive and non-disruptive, i.e., these systems should not interfere or interpose in the normal course of their users' activities.

The main working blocks of an intelligent environment, controlled by an HBAS, are depicted in Fig. 1. Occupants interact with different objects and devices, within a space equipped with different sensors and actuators. An automatic controller system implements some sort of intelligence and controls these devices, interpreting the environment information gathered by sensors, determining what action to take, and commanding the actuators towards taking appropriate action upon the environment.

In order to be non-intrusive and non-disruptive, intelligent environments take action based on the state of devices, objects, places and people. This abstract notion of state is commonly referred to as *context*. Context detection refers to the collection and analysis of data from sensors placed on the user's environment, and to the system's inference of the current scenario. This inference is based on pattern detection of sensor readings from previous user actions and feedback, and even on mathematical predictions. Upon detecting a change in context, this system reacts based on a set of previously defined policies and rules. The system behaviour emerges from its context-awareness and from how it automatically responds to context changes.

In an intelligent environment, however, multiple contexts may coexist, entailing distinct reactions, which can *conflict* with each other. For example, one context may imply a lamp to be turned on while another may imply that it should be turned off. Ideally, these conflicting situations should be automatically resolved by the HBAS. With that aim, we will herein motivate and propose a solution consisting of a prototype system that, based on context information, is able to automatically detect and resolve conflicts on behalf of users.

1.1. Motivation

In their everyday life, both at home and at work, people will most likely share space and other resources, and thus the systems that control them. When considering a shared space, the actions taken by one user may affect others. At a given instant, each user's intentions can be different and, as a result, each user may need a different scenario setup. However, these different contexts require corresponding scenarios that frequently cannot be activated simultaneously on the same space. Therefore, we can say that we have a *conflict situation*. Interaction of applications in an intelligent environment creates room for further conflict. Consider the simple example scenario depicted in Fig. 1, that shows how conflict arises between two users, and at the same time between energy saving and comfort, in an HBA system: suppose that at the end of the day, Alice and Bob arrive home and enter their empty living room. The system is programmed to turn on the ceiling lamp, when placed sensors detect someone's presence, dimming it either at full capacity or at half capacity. This means that there are two illumination scenarios that are to be selected based on the inferred context. The first context is "*Alice wants to watch TV*"; the second context is "*Bob wants to do his homework*". In both contexts, half capacity is more energy-saving. However, while in the first context half capacity is also more comfortable, since it avoids glare; in the second context, full capacity is more comfortable for reading. This means that in the first case, energy saving and comfort are side to side, but in the second case they are conflicting. Moreover, these users' intended contexts are conflicting, since the two illumination scenarios cannot happen simultaneously, hence the system cannot fully fulfil the needs of both users with just one illumination configuration. In this scenario and faced with these possibilities, how should the system dim the lamp, and thus resolve the conflict?

This example shows conflict on comfort versus energy saving, and on concurrency over resources, therefore any system to succeed in resolving conflict must cope with the notions of *ownership* and *priority* among users, in order to appropriately adjust its behaviour. Moreover, conflicting scenarios may occur in the same physical space, and interfere with each other. A plethora of conflict types emerge, and systems must have the ability to resolve them on behalf of users, or otherwise acknowledge their own limitations and inform users that the conflict should be explicitly resolved. The fact that each person is essentially unique and ever-changing while subject to the influence of a large set of factors, results in different goals and mental models, thus less predictable and, in many cases, conflicting. In multi-user scenarios, context inference becomes even harder. It raises challenges such as distinguishing the preference of each user, as well as resolving the conflicts among different user preferences [5]. Moreover, the difficulty of tracking the intentions and thus the context of each user greatly limits the ability of the system to respond appropriately. As we can see, multi-user conflict resolution is more complex than a single user's. The more users, the harder are context inference and conflict resolution.

1.2. Contributions and research methodology

The main hypothesis explored in this paper is that it is possible to create an automatic system to detect and resolve conflicts in ambient intelligence systems (e.g., home and building automation systems). Our work will bring several contributions, including: a conflict taxonomy, a formal representation of an intelligent environments conditions and components, and the formal underpinnings of a prototype system for automatic conflict detection and resolution. Aiming at understanding how to build a system to automatically detect conflict situations in HBAS, this research work will undertake the following steps: (i) investigation of the nature of conflicts in ambient intelligence systems; (ii) development of a conflict classification, with the intent of understanding their sources and traits, to determine which are amenable to automatic resolution; (iii) development of a formal model that enables the specification of an intelligent environment's components, whose interaction can result in conflict, and subsequent development of a formal notion of conflict; (iv) design of a prototype system, capable of analysing environment models, determining the existence of conflicts and assessing their solvability.

1.3. Article organization

This paper is structured as follows. In Section 2, we introduce concepts and review the relevant aspects of ambient intelligence, namely context-awareness, multimodal interaction, and home and building automation systems. Section 3 discusses conflicts and, based on existing literature, a taxonomy of conflicts is suggested, along with strategies to deal with and resolve conflict. The details of a solution for automatic conflict detection and resolution in ambient intelligence systems, is motivated and further detailed in Section 4, and its validation described in Section 5. Finally, Section 6 presents the conclusions.

2. Ambient intelligence

Ambient intelligence (AmI) refers to the paradigm of creating environments that are able to, as silently and imperceptibly as possible, detect and respond to users' needs, without requiring any explicit orders from them. This paradigm envisions an environment consisting of a pervasive network of small electronic devices with minimum processing capability, that act as either sensors or actuators, and therefore obtain information or act upon the environment, thus changing it [6]. The effective commissioning and orchestration of these devices results in a smart environment, characterized by systems and technologies that incorporate the following features: (i) *embedded in the environment*, in the sense that technology is ubiquitous and as undistinguishable from the surroundings as possible; (ii) *context-aware*, i.e., capable of obtaining information about the users and ongoing activities to infer the current context and act accordingly; (iii) *personalizable*, in the sense that users can establish a set of preferences that tailor the system to their needs; (iv) *adaptive* to user and environment changes; and (v) *predictive*, i.e., capable of anticipating users' wishes based on their past behaviour, incorporating either implicit or explicit feedback. In other words, ambient intelligence aims at achieving an environment with "*machines that fit the human environment instead of forcing humans to enter theirs*" [4].

Nowadays, a large variety of disciplines can be grouped under the umbrella of AmI: distributed intelligence, data and information communication, software and hardware design, robotics, information fusion, computer vision, speech recognition, social sciences, ethics and law [7]. Moreover, AmI is continuously evolving, as new visions and theories appear, towards more aware and user-centred systems [8]. The awareness aspect of AmI has also evolved, partly due the increasing possibilities offered by the decurring technologic advancements. Three main phases can be considered, towards the actual level of user centeredness: (i) context-aware distributed systems, able to obtain relatively accurate context data; (ii) the ability to infer valuable information from the attained context data; (iii) providing social value.

Unfortunately, despite active research over the last two decades, only now the technological infrastructure required to support the ubiquitous computing world is starting to appear. Much work had to be done, and Weiser wisely pointed out the three main technology-related challenges that still had to be surpassed: (i) creating the needed mobile infrastructure for pervasive wireless networking, with enough bandwidth to support communication between hundreds of wireless computers; (ii) augmenting or enhancing networking protocols, otherwise unable to handle the desired infrastructure mobility; (iii) development of windows systems, aiming window mobility over a network [9].

In the remainder of this section, relevant aspects regarding context awareness (Section 2.1) and human–computer interaction (Section 2.2) will be detailed, followed by a careful survey on HBAS (Section 2.3), which is a practical example application of ambient intelligence and a focus of this work.

2.1. Context-awareness

Context information is defined by Dey as "*any information that can be used to characterize the situation of an entity*" [10], where an entity denotes any person, place or object that is considered relevant to the interaction between a user and an application, including the user and application themselves [11]. Herein, application stands for any sensor or actuator that can provide information or change the environment on behalf of the user.

However, the notion of context is slippery, as argued by Dourish [12]. He considers that the most broadly used notion of context, the representational notion, is based on wrong assumptions about context, namely its encodability, representability, delineability, stability and detachment from activities; thus misinterpreting the interactional role of context in daily activity. As such, he proposed an alternative model, *embodied interaction*, in which context is seen as the outcome of embodied practice and considered as a relational property between objects and activities, dynamic, particular and mutually constitutive to activities, arising from them and being enacted in their course.

A major difficulty regarding context is how implicit user orders and intentions are inferred from context changes, and how the system reacts to explicit user orders. User preferences change over time or based on situation [5]. On any given moment, these preferences are influenced by aspects such as mood, motivations, goals and needs. This kind of information is mostly subjective and inconstant, thus unattainable by the common and non-intrusive sensors. This poses several challenges when it comes to inferring context, e.g., problems regarding context validity [13] or quality-of-context (QoC) [14], and hence to determining the appropriate action. Therefore, there are two possible reasons for inappropriate context based system behaviour: (i) the system infers and acts upon a wrongly assumed context; (ii) the system does not detect a context and thus does not act at all. For both cases, even considering the fact that these systems are intended to automatically infer context and respond to implicit rather than explicit commands, it is essential that any system behaviour be overridable. This is especially true on a home control system.

Each actuator on a pervasive system may affect its surrounding physical environment, thus influencing the context. The definition of conflict varies from context-aware application to application [15]. Tuttlies defines conflict with respect to a user or application as: "…*a context change that leads to a state of the environment which is considered inadmissible by the application or user*" [16]. In this type of system (i.e., distributed), the possibility of conflict is higher than in other systems, mainly due to a number of contexts and services used, and the mobility of entities [17].

Context-aware systems' architecture is a complex topic, object of several studies. Approaches exist proposing central managers, while others follow the ubiquity notion into distributing system coordination among entities. The latter range from activity-based frameworks to bio-inspired architectures, which propose metaphors, for organism behaviour and interactions in terms of the laws that characterize them, e.g., physical, chemical, social or ecological.

For instance, Tentori et al. [18] propose a multi-agent framework that enacts a subjective coordination by enabling activity-awareness, through the incorporation of components that let autonomous agents announce activity events and gather context from activity recognition. Agents monitor changes in activity attributes, which are announced using a common language, and incorporate a context model for representing e-activities and supporting specialized queries. A perception component allows agents to perceive colleagues and artefacts in use, which leads to activity estimation by a reasoning component, and consequently to appropriate action.

In the bio-inspired line of thought, Viroli [19] proposes a chemical metaphor model for a distributed and self-organized spatial coordination of software services, which handles networked tuples by enabling their composition, aggregation, competition to serve requests, and ultimately extinction, through coordination laws structured as chemical-resembling reactions. Furthermore, services are matched with requests through a self-regulating dynamics inspired by tuple diffusion and the prey–predator model of population dynamics; enabling remote interaction and opportunistic service compositions.

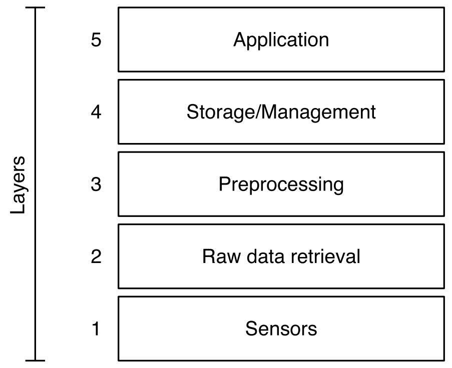

Fig. 2. Baldauf's conceptual framework for context-aware systems, displaying a layered organization of information processing components underlying context aware systems.
Source: Adapted from Baldauf et al., 2007 [22].

Tisato et al. [20] introduce an architectural model defining core concepts and laws for the design of augmented reality applications that realize virtual ecologies, along with a framework of software components corresponding to organisms. Their model supports space mapping to let organisms rely on their visible spaces, unaware of how they are mapped into spaces viewed by others, and includes a space-aware communication paradigm that allows organisms to indirectly interact by generating and observing information localized in the spaces they view. With this, the overall behaviour of a complex ecological system emerges as the result of interactions between organisms and environment, even if organisms are not mutually aware and view different ecology spaces. Moreover, as the model does not describe the internal behaviour of the virtual organisms, but only the basic concepts and laws allowing them to interact, it allows several bio-inspired kinds of interactions to coexist in the same ecology, provided that they rely on the core concepts of space and of spaces mapping, thus being a sound basis for the development of augmented reality applications with heterogeneous components.

Locatelli et al. [21] demonstrate the application of the Multilayered Multi-Agent Situated System (MMASS) model to represent and manage awareness information in Collaborative Ubiquitous Environments (CUEs). CUEs support collaboration in a context of ambient intelligence, by informing users about arising opportunities, such as the proximity and availability of an appliance or possible interaction partner. The model is structured as a graph where agents are located in sites (graph nodes). In this graph, moreover, weighted vertices represent the strength of (social) relationships between individuals, and edges between sites indicate the possibility of agents in each of them perceiving signals and agents in the other. Different interaction mechanisms are defined: (i) reaction, allowing agents from adjacent sites to synchronously change their state; (ii) field (signal) diffusion through the environment; (iii) field perception, depending on the agent's sensitivity threshold and receptiveness coefficient; and (iv) composition or comparison of fields of the same kind. Agent behaviour is specified in terms of reaction, field emission, transport operation (possibility to move) and trigger operation (change state upon perceiving an event in the local context), thus allowing a large range of agent types, potentially with completely different reactions.

A thorough survey and analysis of existing systems gave Baldauf [22] the foundations to depict a layered conceptual framework for such systems, shown in Fig. 2, in which the base layers are more hardware-oriented and the top ones correspond to the actual system logic and intelligence. It is important to note that this is merely a conceptual and functional separation, since systems usually aggregate some layers in their implementation.

Our choice of Baldauf's framework was based on the fact that his survey on context-awareness was the most referenced and the only one that presented a conceptual framework in addition to comparing existing systems. As this framework is of interest to our work, we point out some of the most relevant aspects of each layer. The first layer consists of *sensors*, that may be either hardware sensors that capture physical data; virtual sensors, that obtain context data from other sources, e.g., using mouse movements to infer user location; or logical, which combine these two. The *raw data retrieval* layer deals with obtaining raw data from sensors and offering a unified API for accessing sensor data. The *preprocessing* layer takes sensor raw data to infer the contextual data. At this level, distinct data sources may provide contradicting information about a context. These context sensing conflicts arise from data quality problems, which we will not pursue in this paper. The *storage and management* layer enables storing and querying context data. Problems in this layer are related to temporarily outdated context information between requests. Finally, the *application* layer is where conflict detection and resolution are performed, since it implements the actual system intelligence, reasons about the context information and reacts to context change events. This layer will be the focus of our work. However, as usual in layered systems, we must consider that some conflicts arisen in lower layers may propagate up, or pass undetected and unresolved as a result of the lower layers' limitations.

2.2. Human–computer interaction

The usual WIMP (Windows, Icons, Mouse and Point)-based interaction modes, popular in computer interfaces, are frequently inappropriate and inadequate to interact with ubiquitous systems, since they contradict the ideal of having technology vanish into the environment. In turn, multimodal interaction refers to a form of human–computer interaction that involves more than one interaction modality and therefore is better suited for ubiquity, as multimodality is applied and useful both in information input (human–computer) and output (computer–human). Moreover, depending on the system's goals, some interaction modalities are likely to be more effective than others.

As Gross [23] points out, there has been an evolution of foci towards interactive systems: from single-user WIMP, to cooperative systems, to single-user ubiquitous computing (ubicomp), to single user AmI, and most recently to cooperative AmI; the latter of which can be defined as environments that aim to improve users' work and private life by analysing and adapting to the current situation with a special focus on interaction among users. Therefore, AmI systems should ideally include the use of intelligent (natural) user interfaces [24].

2.2.1. *Acting as multimodal output and sensing as multimodal input*

In multimodal output, through which the computer can convey information to the user, a modality refers to the human senses employed to process incoming information [25]. To this end, the most commonly used modalities are vision and audition. Visual information can be received either in textual, graphical or mixed form, and through various types of displays. Auditive information ranges from simple beeps to more complex formats, such as speech synthesis.

In ambient intelligence systems, existing actuators take action upon the environment, therefore affecting nearby users. Nevertheless, their output is not limited to explicit visual or auditive informational content. Instead, actuators in a smart environment act on temperature, humidity or other environment parameters, which are often not immediately and consciously perceived by users. However, since the actions of these devices directly affect users' sensory receptors, they may well be considered as a slight extension of the usual notion of multimodal output.

Multimodal input has evolved both at the physical level (variety and ergonomy of used devices) and in depth (possible actions). This is partly due to the need for a better adaptation to the users and their ability to more effectively and naturally interact with available systems. In fact, it is expected that people will interact continuously with computation, in an ever-increasing range of forms, situations and locations [26]. As Markopoulos [26] states, the set of devices and services that a user might use to access a system is likely to be numerous, diverse and expanding over time. Therefore, contrarily to traditional interaction design, ubiquitous computer–human interaction (ubichi) should be designed to be extendable and to be combined with other, initially unknown, forms of interaction.

Users can provide information to the system either with or without directly manipulating input devices. Interaction through device manipulation can be divided according to the type of information provided: (i) text, through physical or virtual keyboards; (ii) spatial information, through pointing devices, such as mice, trackpads, light pens and others; (iii) audio, through microphones, MIDI keyboards or other digital musical instruments; (iv) imaging and video, through digital cameras, scanners and other imaging equipment. Some of these devices enable the input of more than one of the above information types. However, the richest and most interesting interaction forms are those that require no explicit device manipulation, thus making the interaction more natural. This is the case of gesture recognition, which allows users to interact with the system through a large range of pre-defined gestures, from simple pointing to more complex and composite ones, e.g., gestures used to play tennis on a Wii^{TM} console. Another interesting form is the application of eye gaze tracking systems to ease the input of spatial information, thus easing the interaction.

In ambient intelligence, to a large extent, explicit knowledge about the existence of the input devices by the users is not mandatory. We will consider interaction through sensors as *implicit interaction*, in the sense that it provides information input without requiring direct commands. The intended ubiquity and "invisibility" of HBAS brings new challenges to the interaction design of such systems, and can only win with the exploration of various forms of obtaining user information without requesting it.

Ideally, a conflict resolution system should aim at offering multimodal input and output, in order to take advantage of the synergies existing between the various interaction modes, providing the user a means to interact with the system in a more natural way, thus offering a smoother and more pleasant user experience.

2.2.2. *Social intelligence, perception and persuasion*

Markopoulos et al. argue in favour of considering intelligence beyond its narrow sense of problem solving, learning, and system adaptation; to cover the ability of a system to interact socially with people and become a socially competent agent in the group interactions it supports [27]. This notion is aligned with that of *social intelligence*, long defined by Vernon [28] as a person's ability to "...*get along with people in general, social technique or ease in society, knowledge of social matters, susceptibility to stimuli from other members of a group, as well as insight into the temporary moods or underlying personality traits of strangers*". Ideally, a system supporting effective conflict resolution should intervene in a way that is perceived as socially competent. Markopoulos, de Ruyter and Saini point out that this requirement of social intelligence presents challenging research problems to the Human–Computer Interaction (HCI) community. Although measuring the perceptiveness and value addition of one such system is difficult [29], experiments show that it is possible to build socially intelligent home dialogue systems that create a positive perception of technology and elicit a greater user acceptance regarding their social intelligence [30,31].

A good example of the use of AmI towards providing social value is the ALZ-MAS AmI-based multi-agent system proposed by Tapia et al. [32], which aims at enhancing the assistance, health-care and safety of Alzheimer patients living in geriatric residences. It uses complex reasoning and planning mechanisms to dynamically schedule the medical staff daily tasks, which can be done through an interface that also displays basic information about nurses, patients and the building. By taking advantage of the cooperation among autonomous agents and the use of context-aware and wireless technologies, ALZ-MAS obtains real-time environment information and allows users to control and manage existing physical services, thus providing a ubiquitous, non-invasive, high-level interaction among users, system and environment.

Another interesting aspect of social intelligence that has been of interest to researchers is *persuasion*. Captology (as the area of persuasive technology is called), explores the theory, design, and analysis of computers for planned persuasive effects. Fogg [33], laid the foundations for this area. In this book, he proposed that five primary types of social cues caused people to infer about social presence in a computing product: physical, psychological, language, social dynamics and social roles. In social sciences, Cialdini is well known for the six principles of persuasion: liking, reciprocity, social proof, consistency, authority and scarcity [34]. These principles are still a basis for other works on the area. Kaptein et al. [35] explore the idea that for persuasive technologies to be effective, their adaptivity to individual user susceptibility is necessary. Their results show not only that including persuasive cues increases user compliance to requests, but also that incorporating a user profile of susceptibility to specific cues, and adopting the right persuasive strategy, could greatly enhance a persuasive system's effectiveness.

The usefulness of these aspects for conflict resolution stems from the fact that some conflicts cannot be resolved without explicit user intervention. Therefore, a persuasive HBAS could be useful in two ways: (i) in persuading users to lightly resolve the conflict (and possibly suggest the most energy-efficient resolution); (ii) in occasionally persuading users to try environment conditions slightly different from their defined preferences, either to improve energy-saving or to minimize future conflicts.

2.3. Home and building automation systems (HBAS)

The basis of any HBAS is a network of sensors and actuators, connected to an intelligent automation control system [1]. Sensors periodically collect information from the surrounding environment and feed the control system, and may be of various types and purposes; e.g., sound, motion, temperature, humidity, luminosity, air flow. Conceptually, the information read by sensors is continuously aggregated and analysed, to determine what activities are taking place on a given place at a given point in time, i.e., to determine the contexts that apply to each entity. Using that information, an intelligent control system orchestrates all distributed devices, sending them commands, in order for the space to comply with a set of pre-defined and pre-programmed policies, thus carrying out users' preferences and demands. Actuators, as the name implies, receive instructions from the control system and act upon devices, thus producing an effect on the environment. Automotive doors and windows, motorized blinds and shades, lamps, Heating, Ventilation and Air Conditioning (HVAC) systems, multimedia and consumer electronics equipment are examples of generally controllable devices that can be used as actuators. It is important to note that an actuator is not always binary (on/off) nor bounded to the room division where it is installed. On the contrary, many devices influence adjacent spaces and the users therein. Therefore, in order to reason about conflict, it is important to specify the possible states and the effect of each on the surrounding environment. For example, audio equipment can be on or off, but each volume level has a different effect radius, and some levels may influence the comfort of users.

Having been initially meant for commercial and corporate ends, BAS have as one of its core goals resource management and optimization, not only energy related, but also other types of waste and access control. An interesting application of a BAS is to predict energy-related needs (for lighting and heating) based on occupancy patterns, thus allowing the creation and scheduling of useful scenarios with adequate configurations, adjusted to the energetic needs of each phase of a work day. A more concrete approach, that can be considered as a subset of a BAS, is room automation. When a single room has a particularly large number of devices, it is common to centralize their automation and make their control exclusive to that room. Good examples are corporate boardrooms, where projecting equipment and individual displays are only useful for that particular room (including for privacy and security reasons), or research labs, with specialized and often sensitive equipment.

Building Automation (BA) was also adapted to home environments, creating what we now know as Home Automation, Domotics or Smart-Homes. However, while in a commercial or corporate medium the main goals are productivity and profit, at the residential setting other priorities stand out, such as safety, leisure and comfort. Also, within a family there are usually multiple goals, which vary in importance, both among members and for each member through time. Moreover, at home people need the most to feel in control. While at a work environment, one is subject to hierarchically superior orders and decisions, it is only expectable that this hierarchical power extends to the control of the surrounding automated systems. Any discomfort caused by a system's incorrect context inference may be attributed to superior decisions or overrides, thus resignedly tolerated. However, at home this does not happen, therefore tolerance towards erroneous interference from an HAS is inferior to that of a BAS.

Despite the architectural similarities between HAS and BAS, their goals, priorities and requirements are different. Consequently, conflict situations that arise are also likely to be solved in differently.

There are two main visions regarding further development of HBAS systems: (i) evolution via *user approaching*: many advocate intelligent systems should evolve through the development of better human–computer interaction and the increase of user ability to tailor the system; (ii) evolution via *user understanding*: others argue evolution should be taken by

Table 1
A four-dimensional taxonomy of conflicts, organized according to their classification dimensions, possible types and respective meaning.

Classification dimension	Possible types	Meaning
Source	Resource	Concurrency over a resource
	Application	Concurrency over an application
	Policy	Conflicting policies in a context
	Role	Conflicting profiles in a context
Intervenients	Single user	Conflictuous user intentions
	User vs. user	Concurrency over a resource or application
	User vs. space	User conflicts with room rules
Time of detection	*A priori*	Detected before its occurrence (predicted)
	When it occurs	Detected during occurrence
	A posteriori	Detected after it has occurred
Solvability	Avoidance	Resolves conflict before its occurrence
	Resolution	Resolves conflict during its occurrence
	Recognize inability	Recognizes inability to resolve conflict
	Acknowledge occurrence	Acknowledges too late that a conflict occurred

increasing system's intelligence, through better profiling and machine learning, and more effective context detection and inference, thus reducing interaction to a minimum. Regarding user approaching, several authors propose more adequate, adaptable and innovative user interfaces and interaction modes for this kind of systems [36–38]; while others focus on the development of methods to allow the personalization of these systems' applications [39,40]. Regarding user understanding, an obvious path is the improvement of sensor data attainment, analysis and storage [41,42]. This sensor data is the main basis for user profiling [43] and profile processing and evolution throughout time [44].

There are also approaches studying backward-compatible solutions. Ruta et al. [45] propose semantic-based enhancements to one of the most widespread domotic standards: the EIB/KNX standard. Their solution uses knowledge representation, automated reasoning and annotations based on ontological formalisms to create a self-adapting framework for managing user profiles and device services, therefore supporting advanced resource/service discovery. These annotations enable not only exact matching, but also potential or intersection matches (where requests and supplied resources have something in common and no conflicting characteristics) and partial or disjoint matches (where requests and supplies have some conflicting features), which are useful in scenarios where nothing better exists. Loseto et al. [46] further explored this framework, through a case study that demonstrated its applicability in optimizing energy consumption and load scheduling in HBA, through a bilateral negotiation protocol. The protocol obtains a logic-based ranking of available services and resources, according to the current status of user, devices and environment, seeking to maximize user comfort and energy efficiency. In case of conflicting user and service "preferences", the requester and provider agents negotiate to find the service orchestration that best fulfils the request.

Making buildings greener has also been a subject of crescent interest. The use of occupancy sensors to increase energy saving [47,48] and the development of more efficient lighting and HVAC control techniques are the focus of most approaches [49–51]. Examples and estimates regarding the practical application of such techniques can be found in [52–54]. Reinisch et al. [55] propose an interesting and comprehensive multi-agent system concept for an intelligent home, addressing the reduction of energy consumption without neglecting user comfort.

Conflict detection and resolution will benefit from the developments of HBAS in the multiple areas mentioned above, since these systems will then be able to become closer and more knowledgeable of their users.

3. Conflict

Conflict is a natural disagreement between different attitudes, beliefs, values or needs [56]. Interpersonal conflict is something everyone learns to deal with, more or less effectively, from an early age. However, when people coexist in smart environments, with systems that perform automatic control over a variety of services and resources, it is only expectable that these systems end up having an intermediating role in the automatic resolution of any arising conflicts.

In this section, we present a conflict taxonomy and explanation regarding conflict in ambient intelligence systems, an overview on the topic of interpersonal conflict resolution, and a survey on related work on conflict resolution.

3.1. Classifying conflicts

As we have previously mentioned, Tuttlies defines conflict with respect to a user or application as "…*a context change that leads to a state of the environment which is considered inadmissible by the application or user*" [16]. To better understand the nature of conflicts in ambient intelligence systems, a brief explanation follows, regarding the various conflict types that will be considered throughout this paper. Our taxonomy classifies conflicts according to four different aspects: (i) source, (ii) intervenients, (iii) time of detection and (iv) solvability. A summary of this classification is presented in Table 1.

Relatively to the *source*, a conflict can happen either at: (i) resource level, i.e., when multiple users concur over a given resource, e.g., a radio; (ii) application level, when multiple users concur over an application, such as a display; (iii) policy level, when conflict arises due to conflicting policies in a given context, e.g., a user enters a library, and although his cellphone allows him to listen to music through his speakers, the room has a silence policy; or (iv) profile (or role) level, when there are conflicting user profiles and preferences in the same context, e.g., one user prefers the lights at full capacity for reading, and another user prefers the lights at half capacity to watch a movie.

Conflicts can also be classified regarding their *intervenients*. A conflict situation can be either: (i) single-user, in which one user has conflictuous intentions, for example comfort and energy saving; (ii) user vs. user, when more than one user concur over a given resource, application, or environment state; (iii) user vs. room, when user actions conflict with any established room policies, for example, a user's mobile phone ringing in a room with a silence policy.

The *time of detection* is another way of classifying conflicts. A conflict can be detected *a priori*, i.e., predicted to happen. In some literature, this is called a potential conflict. More particularly, it can be either a definite potential conflict, i.e. a conflict that will definitely occur if the user is in the right context; or a possible potential conflict, if the possibility of occurrence is less than that of the definite potential conflict, since it may still not happen, even if the user is in the right context and time. A conflict can also be detected while it is happening (also called actual conflict), either through the system context-awareness capabilities, or by receiving user feedback. The last case is when a conflict is only detected after a reasonable resolution time has passed, likely due to context awareness limitations or sensing delays, as we have mentioned before.

Finally, conflicts can be distinguished by their *solvability*. In the best case scenario, when detection happens before occurrence, a conflict is solved before it actually happens, thus avoided. More commonly, a conflict is detected during its actual occurrence, in which case the system attempts resolution, or otherwise recognizes its inability to deal with the conflict. Another possibility is that the system, not detecting a conflict soon enough to resolve it, realizes later that it happened (e.g., due to delayed sensor information) and can only acknowledges its occurrence, including it in the usual learning processes, and even informing system administrators of the situation, with the intent of future improvements.

As we have seen, conflicts may be of several types and arise from a number of dimensions, thus requiring different detection and resolution mechanisms. Given the predictive nature of HBAS, conflicts can sometimes be anticipated before they even happen. In that case, the system may take action against the actual occurrence of the conflict, thus resolving it. We will call this conflict resolution through avoidance, more commonly known as *conflict avoidance*. Another possibility is that the system only detects the conflict when a new context is identified, either by the application or through user feedback. In that case, action may be taken in order to eliminate the conflict. The system may: (i) restore the previous context; (ii) adjust the new context towards general desirability or tolerance; (iii) inform users that they should explicitly resolve the conflict. Several conflict resolution approaches for ambient intelligence have been developed so far, and will be described further in Section 3.3.

3.2. Interpersonal conflict resolution

Conflict occurs between people in all kinds of human relationships and in all social settings. Because of the wide range of potential differences amongst people, the absence of conflict usually signals the absence of meaningful interaction [57]. Conflict by itself is neither good nor bad. Interpersonal conflict arises when incompatible goals develop between persons, groups, or nations. However, the manner in which conflict is handled determines whether it is constructive or destructive [58].

Origins of conflict vary. The Circle of Conflict Model, originally developed by Moore [59] and adapted by Furlong [60], looks at conflicts in the perspective of their origins. This model postulates that five main underlying causes to conflict: (i) values: all values, morals, ethics and beliefs, etc.; (ii) relationships: negative past history or experiences; (iii) externals/moods: external factors not directly related to the situation, but that still contribute to the conflict; (iv) data: when parties have incorrect, incomplete or different data or interpretations of it; (v) structure: system structure problems, such as limited resources, authority problems, and organizational structures [60]. Moreover, unresolved or inadequately resolved conflicts tend to result in tension, which may trigger or intensify posterior conflicts.

Thomas [61] adapted a previously presented graphical view of conflict handling modes [62] following a two dimensional taxonomy: (i) assertiveness, the extent to which the person attempts to satisfy his own concerns; (ii) cooperativeness, the extent to which the person attempts to satisfy the other person's concerns. In this taxonomy, five modes are presented and described, and remain well accepted and studied. We present an adaptation of this taxonomy in Fig. 3. The Thomas–Kilmann Conflict Mode Instrument (TKI) is a broadly used tool to help identify which style one tends towards when conflict arises. The TKI has also been recently adapted to serve as basis for a computational simulation model for interpersonal conflict management in team building, using an agent-based modelling method [56], which shows both its currency and overall acceptance.

These two basic dimensions of behaviour define five different modes for responding to conflict situations: (i) Competition: power oriented mode, in which an individual pursues his own concerns at the other person's expense, resulting in a "win–lose" situation, because only one can win; (ii) Accommodation: an individual self-sacrifices by neglecting his own concerns to satisfy the other's, also resulting in a "win–lose" situation; (iii) Avoidance: the person neither pursues his own concerns nor the other's, thus the conflict is not actually dealt with, resulting in a "lose–lose" situation; (iv) Collaboration: involves an attempt to work with others to find some solution that fully satisfies their concerns, by digging into an issue and trying to find a solution to an interpersonal problem, therefore both parties win; (v) Compromise: attempt to find some

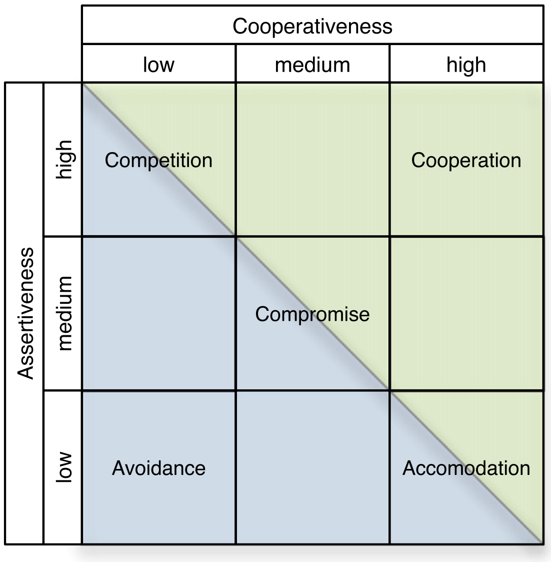

Fig. 3. Two-dimensional taxonomy, displaying the five types of interpersonal conflict handling modes in terms of assertiveness and cooperativeness levels. The diagonal represents the threshold between non-constructive or destructive (left) and constructive (right) interpersonal conflict resolution.
Source: Adapted from Thomas, 1992 [61].

Table 2
Summary of the related work on conflict resolution.

Conflict resolution topic	References
Conflict resolution in multi-agent systems	[63,64]
Interest/intention conflict resolution	[16,15]
Resource conflict resolution	[65–67]
Policy conflict resolution	[17,68–71]
Authorization conflict resolution	[11]

convenient and mutually acceptable "win–win" solution that partially satisfies both parties, possibly exchanging concessions, or seeking a quick middle-ground solution.

3.3. Conflict resolution in ambient intelligence

Although ambient intelligence and related areas have been the target of innumerable studies for decades, conflict resolution in smart environments is still a relatively new topic. Several works on intelligent environments and resource management describe their conflict detection and resolution methods (see Table 2).

3.3.1. Conflict resolution in multi-agent systems

Alshabi et al. [63] present a survey on agent cooperation models and conflict resolution methods in multi-agent systems, comparing three existing architectures, including HOPES [72] and HECODES [72]. The authors then propose their own multi-agent hierarchical architecture, in which agents independently choose their balance between cooperation and autonomy, and in case of conflict among agents, resolution follows one of the three negotiation techniques presented in [73].

Jacak and Pröll [64] present a heuristic approach for conflict management in an intelligent and cooperative multi-agent system. Each agent is able to perceive and react to the environment, to plan and execute an action, and to negotiate with other agents. The presented method allows the system to coordinate and negotiate agent actions in order to avoid conflicts and achieve a common global goal, which happens by a correct sequencing of each agent's local goals. A practical application of the method, regarding an intelligent multi-agent robotic system realizing complex transfer operations simultaneously, is shown as an example.

In [74], Kung and Lin propose and design the Context-Aware Embedded Multimedia Presentation System (CEMP), which provides context adaptation control and offers steady multimedia stream services according to the user's context information. It is based on a context vocabulary ontology, expressed with the Web Ontology Language (OWL), that provides the formal explicit description about the multimedia information domain and the formal user context. The system provides, among others, a mechanism for adaptation reasoning, which infers the best adaptation control when contexts change; and a mechanism for conflict resolution, which calculates the context's priority and the context weight to get the better context conflict resolution results and thus meet the multimedia service's quality requirements.

3.3.2. Interest/intention conflict resolution

With COMITY [16], Tuttlies focuses on *a priori* conflict detection and on resolution through avoidance, mainly of conflicts resulting from different user interests. The approach is based on the PCOM [75] component model, which uses contracts containing component's requirements and functionalities to select services and devices, and compare them to the current needs. COMITY requires applications to state how they affect other applications and users, thus alerting the system in case of conflict, and inducing a conflict resolution, possibly by adaptation of the conflicting applications. It includes a conflict manager component, connected to a database that stores a context model representing the environment state, while another database stores the situations that are considered conflicting. Based on this the conflict manager can detect conflicting situations at runtime.

Similar, but based on a different infrastructure, is the approach by Armac et al. [65]. These authors presented a classification of conflict types and developed a rule-based conflict detection mechanism for, which assumes that every resource and respective behaviour are specified in form of an w automaton, and paired to a set of rules a monitor component detects conflicts at runtime. However, it requires that services provide a semantic description of their behaviour, therefore only top-level services with rule-based implementations can be considered.

In [15], Park et al. propose a dynamic conflict resolution scheme for resolving conflicts between different context-aware applications in smart environments, which incorporates users' intentions and preferences. They model user intentions as the value assigned to a context attribute by the actions they requested from applications, and express user preferences as cost functions over the distance between user intentions and the resolved value. Based on this information, the resolved value is determined to the one which minimizes the cost of all users involved in conflicts.

In [76], Silva et al. define a collective conflict as an inconsistent state that a collective application may reach while evaluating collective contexts, in which the application becomes unable to satisfy, simultaneously, divergent individual interests. They propose a conflict detection and resolution methodology that uses a client–server architecture model to select and configure the current most appropriate conflict resolution algorithm available. This decision is made considering the application's demands for quality of services (QoS) and resources consumption. The QoS criteria considered in their work is the collective satisfaction of users regarding the attained resolution results. This framework is further detailed in [77], where the authors also present a case study regarding a collective tourist guide application to demonstrate the developed methodology's effectiveness.

3.3.3. Resource conflict resolution

In turn, Retkowitz et al. [66] do not address conflicts among users, arguing that such conflicts require manual resolution by the users in most real-life scenarios. Instead, they focus primarily on resource concurrency conflicts, and present an approach that attempts to avoid such conflicts through a more dynamic and effective resource sharing. Their approach considers dependencies between services that are realized as bindings. A configuration process tries to create a service composition that simultaneously matches user requirements, device environment and all service dependencies, respecting previously defined binding policies and constraints. Then, service method tagging and communication interception are used to monitor service bindings and handle access control, based on priority groups. This approach is not fully automatic, but provides a tool for users to visualize and manually modify the system state.

Huerta-Canepa et al. [67] also focus on resource conflicts, and propose a resource management scheme for smart spaces based on ad hoc interaction. They assume that only two jobs can be executed at each time, and explore conflict avoidance through area device control, as well as conflict resolution at the resource level, based on users' priorities, jobs' running time and other similar pre-defined metrics.

3.3.4. Policy conflict resolution

In [17], Syukur et al. investigated conflict detection and avoidance strategies based on context changes, assuming a policy-based application model in which policies are used to explicitly control the behaviour of an application. Even though policies, along with rules with priority schemes among entities are the most common techniques for resolving conflict [39], this paper succeeds in thoroughly classifying and discussing policy conflicts in terms of their sources, detection and resolution techniques, and different timing approaches for resolution. Moreover, they describe an experiment of using these techniques with their previously specified policy based framework, Mobile Hanging Services (MHS) [78]. Although they argue that each strategy's suitability depends on system details, their results pointed to a hybrid of reactive and proactive based conflict detection, and proactive immediate conflict resolution, to be the best working techniques.

Lupu et al. [68] also consider policy conflicts, and although assuming that some conflicts can only be detected at runtime, they rather focus on techniques for offline conflict detection and resolution, which assist the users in specifying policies, roles and relationships. They describe a conflict analysis tool which forms part of a Role Based Management framework, in which management policies are specified regarding object domains. They consider that conflicts potentially arise when overlaps occur between these domains, i.e. when $\langle$subject, action, target$\rangle$ tuples have different policies applying to them. Their approach uses roles and inter-role relationships to reduce the scope of policies that need to be examined, and applies a domain nesting based precedence strategy to reduce the number of overlaps presented to users.

Jiao et al. present another policy conflict detection algorithm in [69]. The authors focus on the particularity that it takes much time for a routine to search every rule in a policy rule set with conventional algorithms, to see if conflict occurs before a new rule is added to the rule set. To address this problem and thus increase the efficiency of policy conflict detection, they start by presenting a formalization of context, policy rules and policy conflicts; and constructing a formal concept lattice through formal concept analysis, based on which policies are later grouped into the associated formal concepts. These formalizations are then used to propose an algorithm for conflict detection, which was subject to a performance analysis and simulation. Results show that concept lattice effectively helps both in determining whether attribute values of policies overlap, and in reducing the number of policy rules to be detected against during conflict detection.

CARISMA [71] supports runtime dynamic policy conflict detection and resolution. Their authors define conflicts as the inconsistencies that arise when contradictory behaviours are requested by the same application as a reaction to a context change, or when cooperating applications do not agree on a common behaviour to be applied. They argue that conflicts cannot be resolved statically at the time applications are designed, but rather need to be resolved at execution time. This approach defines and handles two types of conflicts: intra-profile conflicts, in which a conflict exists inside the profile of an application running on a given device; and inter-profile conflicts, which exist between the profiles of applications running on different devices. Moreover, it uses a microeconomic technique that relies on a type of sealed-bid auction between applications in order to achieve a relatively fair conflict resolution method.

3.3.5. *Authorization conflict resolution*

Masoumzadeh et al. [11] refer to authorization conflicts, which are a particular case of policy conflicts, i.e. when two or more different policies both permit and forbid an access in a situation. As argued by the authors, the strength of their approach is that conflict detection is done almost statically and conflict resolution is left for run-time. They consider that a practical solution for this is establishing a precedence among conflicting policies. To that aim, they formalize the use of context constraints in a rule-based context-aware multi authority policy model, which allows the definition of precedence establishment principles. Moreover, they analyse timing strategies and resolution algorithms, and propose a comprehensive graph-based approach to enable precedence establishment among authorizations in a conflict situation. In the detection phase, a potential conflict graph is almost statically constructed, and using this graph in the actual conflict situation provides the resolution.

4. A solution for automatic conflict resolution

A system that performs automatic context-aware conflict resolution is one that takes information regarding the context of the environment as input, checks for conflicting situations and then produces a new context in the environment, in order to conciliate conflicting requirements or inform about the conflicting situation. In other words, the automatic resolution of conflicts consists of creating actuations to accommodate new requirements of new activities carried out by occupants.[1] The solution we envision, for automated conflict detection and resolution, is based on the idea of expressing requirements in terms of constraints on environment variables. For example, the requirements of occupants are said to be in conflict with one another if and only if no assignment to the environment variables can be found that satisfies all the constraints representing those requirements. Likewise, services are modelled as constraints that capture their possible effects on environment variables. Space zones also have constraints, which denote the limits (perhaps imposed by a policy) to environment variables, and actions have pre- and post-conditions, which are constraints on environment variables.

This section overviews our proposed solution for automatically detecting conflicts and determining adequate resolution methods, based on available context information. We begin with a motivation (Section 4.1), by presenting example scenarios and posing some questions that a system for automated conflict detection and resolution would be *useful* in answering. We proceed with an overview of the planned formal solution (Section 4.2), where the context domain model is explained. A formal treatment of the model with definitions of its most relevant aspects and components is given in Section 4.3. Section 4.4 describes the architecture of our proposed reasoning system and the general system working. Finally, we detail the technical framework of our proposed system (Section 4.5).

4.1. *Motivation*

Consider an intelligent building, occupied by several people and consisting of several rooms, with distinct purposes and capabilities in terms of the offered services. A number of interesting questions regarding context are likely to arise, which should be addressed by a system capable of handling conflict situations, for instance:

- ($Q1$) If an occupant intends to perform a given activity, will there be a conflict with the place's current conditions and occupant preferences?
- ($Q2$) What actuations could the intelligent system make, in order to conciliate several occupants' intentions and preferences on the conditions of a given space?
- ($Q3$) What adaptations are needed from an occupant (in terms of preferences) in order to fit in with a given place's current context?

1 We will not consider for the case conflicts with applications.

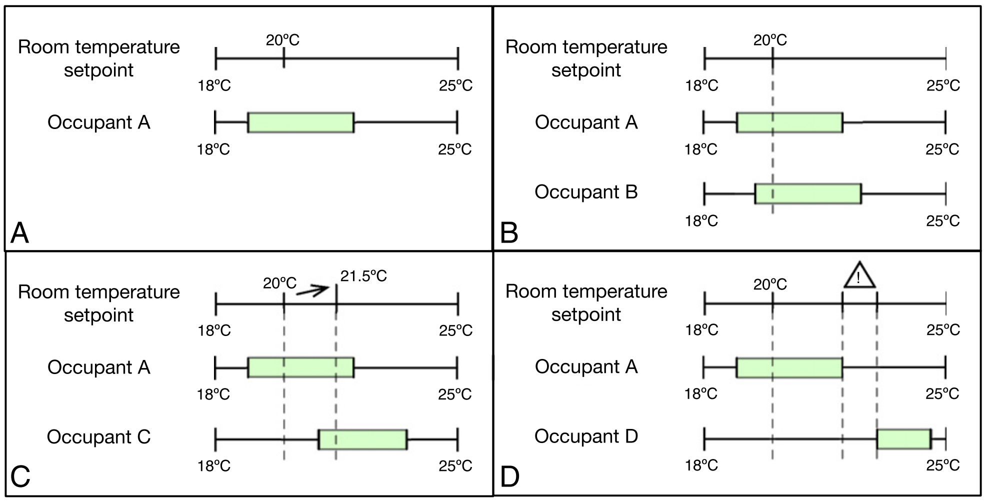

Fig. 4. Example scenarios, depicting the room temperature environment variable setting and the possible system responses to the entrance of a second occupant with a different preference for the temperature variable. The darkened bars represent occupant tolerances. In Scenario A, occupant A is alone in the room, with the temperature set to 20 °C. Scenarios B to D represent the entrance, respectively, of occupants B to D, along with the possible system responses.

These questions encompass occupants, activities, space, conditions and conflict. Analysing them according to the conflict framework presented in Section 3.1, it is clear that $Q1$ consists of conflict detection, while $Q2$ and $Q3$ present conflict resolution situations.

In a real-life scenario, consider that initially, occupant A is sitting alone in the room, and has adjusted the air temperature setpoint to his preference of 20 °C. However, he tolerates temperatures between 19 °C and 22 °C. Suppose that occupant A wants to have a meeting with another occupant. Considering that all occupants have the same priority level regarding the room's occupancy, when the second occupant enters the space, and has a different preference, how does the system adjust? In view of the described situation, Fig. 4 shows three possible scenarios, each corresponding to the entrance of a different occupant in the room, and each leading to a different system behaviour.

- *Occupant* B, with a tolerance of 19.5 °C to 22.5 °C, enters the room. There is no conflict, therefore no action is needed from the system.
- *Occupant* C, with a tolerance of 21 °C to 23.5 °C, enters the room. A conflict exists, since the temperature is set to a value below occupant C's tolerance. However, there is a possible system actuation that resolves it, since changing the setpoint to 21.5 °C accommodates the preferences of both occupants.
- *Occupant* D, with a tolerance of 23 °C to 24.5 °C, enters the room. A conflict exists, but in this case the system has no way of resolving it, since there is no temperature range that would accommodate the preferences of both occupants. Possible system actions would be: (i) maintaining its state, (ii) adjusting to an intermediate value, or (iii) informing occupants of its inability to solve the conflict.

4.2. Model overview

In order to answer the previously stated questions, and thus reason about possible conflict detection and resolution methods, we first need to establish a representation of services, activities occupants, zones and their corresponding relationships. To that aim, we start by creating the corresponding domain analysis using an Entity–Relationship (ER) model, since it enables a further modelling and understanding of the problem in hand, and according to Chen, "*adopts the more natural view that the real world consists of entities and relationships, and incorporates some of the important semantic information about it*" [79].

The ER model presented in Fig. 5 captures the relevant components having context information. From a high-level perspective, we can say that occupants perform certain activities in given zones. Depending on the activity, the occupants *prefer* having environment variables set to a certain range. In turn, each zone *offers* services which act on environment variables, ultimately satisfying its occupants' preferences. Moreover, we consider *pre-conditions* and *post-conditions*, respectively, as characterizing the acceptable inputs and the resulting outputs of an activity. Note that this is the most dynamic and ever-changing part of the model.

Regarding environment variables, our main focus will be the four most influencing of occupants' comfort levels in a given space, namely: sound, light, temperature and air flow. For simplicity, the following aspects will not be considered: (i) a time dimension; (ii) zone hierarchy and access control, since they are beyond the scope of our intended conflict analysis; (iii) hierarchy among occupants, since conflicts involving hierarchically different occupants are mostly resolved by a simple prioritization of the hierarchically superior occupant's preferences. However, they may be included in future work.

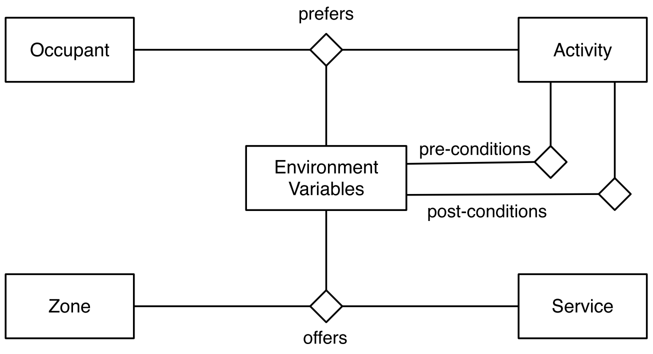

Fig. 5. Entity–Relationship representation of our domain model, representing the main components and the relationships among them.

Automatic conflict resolution requires a mechanism that performs automated reasoning. In order to perform automated reasoning we develop a formal framework around the notion of service. A service is described as a constraint on environment variables, which is then interpreted as denoting valid ranges for the environment variables that can be interpreted as the service being available. Services are available in zones, which abstract physical spaces, and therefore zones are defined as offering certain services. Activities are also defined in terms of constraints over environment variables. Activities have pre-conditions, which are constraints that denote the valid range of environment variables that enable performing the activity. In this way, the automated reasoner can determine what services are needed to perform a given activity, or what zones offer services where an activity can be undertaken. Activities also have post-conditions, which define what are the effects of an activity on the environment. This is used to enable the system to reason *a priori* about whether performing an action will lead to a conflicting situation. Some activities may have no pre- or post-conditions. Reading is an example of an activity with no post-conditions. It has enabling conditions, such as adequate lighting and noise level; however, the impact on the environment is virtually nonexistent. Finally, zone occupants are modelled according to their preferences with respect to the enabling conditions associated to the activities they perform. These preferences are modelled through conditions that also constrain the pre-conditions of a certain activity whenever the occupant intends to execute that activity.

At this point, a rough description of conflict is that a conflict occurs when the preferences of an occupant interfere with the preferences of another, or when the post-conditions of an activity that is starting interfere with the pre-conditions of an already decurring activity.

4.3. *Towards a formal model*

Based on the presented model, the next logical step was to find a mathematical definition and representation for the environment conditions and context components, enabling to formally reason about context and conflict resolution. A practical option is to define components according to the ER model presented in Section 4.2, and to define conditions and components as Boolean expressions.

4.3.1. *Syntax of environment conditions*

Aiming at a formal representation of the environment, we start by distinguishing two kinds of environment variables, namely *continuous* variables and *discrete* variables. Continuous variables are used to capture physical conditions such as temperature, luminosity or sound volume while discrete variables are used to capture equipment status. The sets $V_{\mathcal{C}}$ and $V_{\mathcal{D}}$ will denote, respectively, the set of continuous and discrete variable symbols. We assume sets $\mathcal{C}$ and $\mathcal{D}$ of continuous and discrete domains, respectively along with mappings $\mathrm{Dom}_{\mathcal{C}} : V_{\mathcal{C}} \to \mathcal{C}$ and $\mathrm{Dom}_{\mathcal{D}} : V_{\mathcal{D}} \to \mathcal{D}$ that associate their corresponding domains to variable symbols. Whenever the variable kind is understood, for simplicity, we will write $\mathrm{Dom}(x)$ to denote the domain of some variable symbol $x \in V_{\mathcal{C}} \cup V_{\mathcal{D}}$.

Let x_c and x_d be continuous and discrete variable symbols, respectively, and consider c and d to be constant values in their respective domains. An *environment condition* C is a logical formula over environment variables generated by the following grammar, where C_1 and C_2 are also environment conditions.

$$C ::= x_c \leq c \mid x_c \geq c \mid x_d = d \mid \neg C \mid C_1 \wedge C_2. \tag{1}$$

We could allow a more general form of formulae such as disjunction $C_1 \vee C_2$ which can be obtained syntactically as $\neg(\neg C_1 \wedge \neg C_2)$. However, this would not increase the expressive power of the language.

4.3.2. *Context components*

Model entities will be appropriately represented by corresponding predicates. Therefore, for a given model M, we will write $occupant(O)$ to represent that O is an instance of the occupant ER entity. Likewise, we will write $zone(Z)$, $service(S)$ and $activity(A)$.

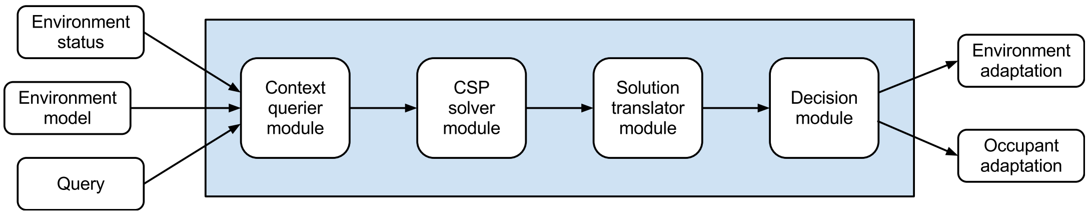

Fig. 6. Prototype framework of our conflict detection and resolution system, depicting its four constituent modules: (i) context querier module, (ii) CSP solver module, (iii) solution translator module and (iv) decision module. The system receives, as input, environment models described in our defined syntax, and then takes appropriate action upon the environment or leads the occupant into adapting.

Services are described by their effect on environment variables. Therefore, a service S is defined by an environment condition which denotes the valid ranges of environment variable values that the service is capable of offering. For simplicity, the condition on environment variables that corresponds to the capabilities of a service S in a zone Z will be represented by $cap(S, Z)$. Similarly, the conditions representing usage restrictions of a given zone Z will be represented by $restr(Z)$. The pre- and post-conditions of an activity A will be represented by $pre(A)$ and $post(A)$, respectively. The relation *prefers* is represented by $pref(O, A)$, which denotes the preferences of an occupant O regarding an activity A. Moreover, we assume that, if no capabilities are known for S in Z, $cap(S, Z)$ will be defined as *false*. In turn, $restr(Z)$, $pre(A)$ and $post(A)$ will be *true* if no restriction, pre- and post-condition are specified, respectively, for some zone Z and activity A. Similarly, when no preferences are specified for an occupant O with respect to an activity A, we will assume $pref(O, A)$ to be *true*.

4.4. The reasoning system

The defined environment conditions' syntax expresses a set of variables, that can have either a discrete or a continuous domain, and a set of constraints $C_1, \dots, C_n$, which specify restrictions to the possible values of these variables. A set $P = \{C_1, \dots, C_n\}$ is known as a *constraint system*. Moreover, an *assignment* is a function $\theta : V_{\mathfrak{C}} \cup V_{\mathfrak{D}} \rightarrow \mathrm{Dom}(V)$ that maps each variable to some value in its respective domain. When this mapping is such that C holds, θ is said to *satisfy* C, noted $\theta \models C$. An assignment θ that satisfies, at the same time, all the constraints of P is a *consistent assignment* or *solution* for P, noted $\theta \models P$. When no such solution can be found, then P is said to be *unsatisfiable*. In general, constraint systems, may have zero, one, or multiple solutions. In the case of multiple solutions, we may be interested in finding the *optimal solution*.

Answering the earlier posed questions $Q1$ to $Q3$ (see Section 4.1) consists, in fact, of determining if there is any assignment of values to some or all of the variables, for which no constraints are violated, thus resulting in a *consistent assignment*. In other words, we aim to determine whether the system context represented by the given set of variables and constraints is feasible, which implies solving the corresponding *constraint satisfaction problem* (abbreviated CSP). However, when no consistent assignment can be found, we will be interested in finding the assignment that validates the most constraints. This problem is known as MAX-CSP. In general, CSP problems are NP-hard, as discussed by Garey and Johnson [80]. Despite being NP complete, thus incurring in exponential resolution time, we conjecture that our problem can be solved in sensible time. To that end, we will apply CSP solvers, which have been long used to solve this type of problems, and hence determine for which values of the input variables and constraints conflict does not occur.

Essentially, a system for detecting and assessing the solvability of conflicts will start by taking: (i) a query, and translating into an input model of the CSP; (ii) environment conditions, that represent the environment model; and (iii) the environment status (see Fig. 6). Afterwards, it is possible to query the CSP solver about context aspects, and obtain a solution consisting of the values or value ranges (if existent) for which the system with the queried constraint is satisfiable. The system should then be able to analyse the output obtained from the CSP solver, and decide upon the adequate action. The architecture is depicted in Fig. 6, consisting of four main modules: the *context querier module*, the *CSP solver module*, the *solution translator module* and the *decision module*. The context querier module receives context information and any number of queries to that context, all described as environment conditions. It then combines the queries to the context as additional conjuncts, transformed into a constraint system to be given as input to the CSP solver module. The solution translator will receive the results from the CSP solver and translate them from the CSP solver output format to the environment model. Finally, the decision module analyses the information in order to determine: (i) whether there is a conflict; (ii) in case there is a conflict, if it is solvable or not; (iii) in case it is solvable, if its resolution involves environment adaptation, occupant adaptation, or both.

Although not represented in the diagram of Fig. 6, a complete working system would, depending on the decision module results, act upon the environment and/or present conflict information through adequate and available output modalities, ideally relying on two types of communication that resort to multimodal techniques outlined in Section 2.2.1: (i) *informative communication*, which is less disruptive and would consist of visual cues and/or background visual information; (ii) *interactive communication*, more disruptive, as it prompts for occupant attention and would involve manual, voice or gestural interaction.

4.5. Technical framework

Herein, we discuss the technological underpinnings for the implementation of our prototype system, in view of the proposed framework, depicted in Fig. 6.

In order to read the environment variables' values from sensors and other equipment, our solution requires a close integration with the building automation system. In turn, conflict resolution requires service activations that must be effected in the form of actuations on devices that involve contacting the BAS as well. In terms of implementation, we foresee that both sensing the environment variables and performing the needed service actuations can be achieved by using appropriated drivers. These drivers will then interface a specialized middleware such as OPC to contact with equipment and devices connected on diverse bus technologies such as BACNet, LonWorks or KNX. An automatic controller system, physically supported by adequate processing capability, with an embedded conflict detection and resolution system as suggested before, would gather information from these sensors, perform automated reasoning and command the actuators.

In terms of physical equipment, an environment capable of supporting our conflict detection and resolution approach would require, as a basis, a distributed network consisting of the following equipment: (i) several sensors, including: motion sensors (mainly used to determine occupancy); indoor and outdoor temperature, air flow, humidity and luminance sensors, used to ascertain and ideally use the outdoor environment factors to balance the building's conditions (e.g., opening windows to cool a room or opening blinds to heat it); and ambient noise sensors; (ii) a set of actuators, such as: remotely controllable wall sockets; dimmable lights, ideally being each luminaire in an individual circuit, to enable separate control; automated window blinds/shutters and HVAC systems.

Furthermore, to take full advantage of our approach and effectively communicate with occupants, assisting them in conflict resolution, a multimodal environment should be pursued, therefore requiring specific equipment and software. For informative communication, we envision: (i) visual cues, given through colour-changing LEDS that take a certain colour depending on the conflict state (e.g., green for no conflict, yellow for impending conflict and red for conflict); (ii) background visual information (either textual, graphical or mixed), presented through the displays of electronic devices carried by occupants or available in the room. In turn, interactive communication can be achieved through: (i) speech recognition for user voice commands, requiring microphones and appropriate software; (ii) speech synthesis for system replies or suggestions, requiring speech synthesizer software and conveyed by speakers; (iii) gestural interaction, through pre-defined user gestures (note that such pre-definition should involve users, for the sake of effectiveness and usability), requiring stereo or depth-aware cameras capable of gesture recognition. The combination of these modes would provide a flexible interaction and ease the systems disambiguation of received user commands.

5. Validation

Our approach is validated through a prototype system that implements the context querier, solution translator and decision modules in Java and delegates constraint solving to MiniZinc [81,82], a well-known CSP solver. In the following, we show how to encode queries such as $Q1$, $Q2$ and $Q3$, presented in Section 4.1, as generic CSP problems of conflict detection and resolution (Section 5.1). Afterwards, we explain how to apply these queries to different scenarios, like those depicted in Fig. 4, and how our prototype system generates MiniZinc model code (Section 5.2). We proceed by reporting on experiments that use our approach to detect and resolve conflict, and interpret its results output in the light of the model language (Section 5.3). Finally, the experiment results, the proposed solution and its further development, are analysed and discussed (Section 5.4).

5.1. Formal encoding

Each motivation scenario illustrates a situation where an occupant O needs to perform an activity A with occupants $O_1, \ldots, O_m$ in a zone Z (Section 4.1). This zone has a known state represented in terms of constraints over environment variables $E_1, \ldots, E_n$, modelled as $status(Z, E_1) \wedge \cdots \wedge status(Z, E_n)$. For each scenario, the system has to find out whether that will result in conflict with the zone's current conditions and/or with its current occupants' preferences regarding such conditions. When a conflict is detected, the system must either (i) actuate on the environment or (ii) assist occupants in resolving the arisen conflict.

Query $Q1$ corresponds to the conflict detection problem of determining whether there will be a conflict between the zone's current conditions and occupant preferences. This can be formulated as a constraint solving problem of determining whether there is an assignment θ such that $\theta \models [status(Z, E_1) \wedge \cdots \wedge status(Z, E_n)] \wedge [pref(O_1, A) \wedge \cdots \wedge pref(O_m, A)] \wedge pref(O, A)$. If no such assignment exists, the model is not satisfiable, which means that a conflict exists. In turn, queries $Q2$ and $Q3$ correspond to a conflict resolution problem of determining which actuations are needed to conciliate the occupants' preferences with the current zone status. This is a MAX-CSP problem, that consists of: (i) maximizing the number of occupant preference constraints satisfied; (ii) a reward function $r(E_1, \ldots, E_n, Z)$, which encapsulates the preferred actuation over environment variables $E_1, \ldots, E_n$ in zone Z, for example, one that maximizes energy efficiency or occupant comfort. Formally, this problem consists of finding an assignment θ that maximizes the number of constraints, such that $\theta \models pref(o, A)$ for $o \in \{O_1, \ldots, O_m, O\}$, along with $r(E_1, \ldots, E_n, Z)$. The output will consist of the appropriate assignments to the environment variables, and possibly a list of preference constraints that cannot be fulfilled and therefore must still

Table 3
Summary of formal encodings for the motivation queries presented in Section 4.1.

Query	Type	Encoding
$Q1$	CSP	$\theta \models [status(Z, E_1) \wedge \cdots \wedge status(Z, E_n)] \wedge [pref(O_1, A) \wedge \cdots \wedge pref(O_m, A)] \wedge pref(O, A)$
$Q2, Q3$	MAX-CSP	θ maximizing $w_1 \cdot \sum_{o \in \{O_1,\ldots,O_m,O\}} \begin{cases} 1 & \text{if } \theta \models pref(o, A) \\ 0 & \text{otherwise} \end{cases} + w_2 \cdot r(E_1, \ldots, E_n, Z)$

a
```
 1: var Dom(E_1):E_1 ;
 2: ...
 3: var Dom(E_n):E_n ;
 4:
 5: constraint status(Z,E_1) ;
 6: ...
 7: constraint status(Z,E_n) ;
 8:
 9: constraint pref(O_1,A) ;
10: ...
11: constraint pref(O_m,A) ;
12: constraint pref(O,A) ;
13:
14: solve satisfy;
```

b
```
 1: var Dom(E_1):E_1 ;
 2: ...
 3: var Dom(E_n):E_n ;
 4:
 5: set of int:PCs=1..(m + 1);
 6: var set of PCs:  vpcs
 7: int:pc1 = 1; ...; pcm=m; pc(m + 1)=(m + 1);
 8:
 9: constraint pref(O_1,A) <-> pc1 in vpcs;
10: ...
11: constraint pref(O_m,A) <-> pcm in vpcs;
12: constraint pref(O,A) <-> pc(m + 1) in vpcs;
13:
14: solve maximize w_1 * sum(c in PCs)(bool2int(c in vpcs))
15:    + w_2 * r(E_1, ..., E_n, Z);
16:
17: output["E_1",show(E_1),...,"E_n",show(E_n), "PCs", vpcs];
```

Fig. 7. Model code generation templates for the MiniZinc constraint solver. (a) Template for conflict detection as a CSP problem; and (b) Template for conflict resolution queries, encoded as MAX-CSP problem with a set of preference clauses.

be addressed. Table 3 presents a summary of the formal encodings for these queries. The MAX-CSP encoding uses weights w_1 and w_2 to balance the number of satisfied constraints with the reward function.

5.2. MiniZinc translation

Generally, a CSP problem is translated into a MiniZinc model consisting of variable declarations, constraints, a goal and an output specification. MAX-CSP problems further require specifying a set with an element corresponding to each clause. Fig. 7 demonstrates the translation templates for each of the cases described before. Fig. 7(a) corresponds to template of a conflict detection, starting with the declaration of the environment variables $E_1, \ldots, E_n$ and their respective domains (lines 1–3). Then, the constraints, i.e., the expressions that the variables must satisfy to validate the model, respectively zone Z's status (lines 5–7), and preferences of occupants $o \in \{O_1, \ldots, O_m, O\}$ (lines 9–12) with respect to the activity A, are declared. Finally, the *solve* expression indicates the constraint solver the kind of problem we intend to solve, in this case, a satisfaction problem, i.e., finding any value for which the variables satisfy the constraints (line 14).

Fig. 7(b) presents a conflict resolution template, which starts by declaring environment variable domains (lines 1–3) and occupant preference declarations (lines 9–12), as described above. The declarations section is further extended with the $m+1$ elements that represent each of the preference clauses (lines 5–7). This will enable using the number of satisfied clauses as a maximization goal. The solve expression follows, aimed at maximizing the number of satisfied occupant preference constraints and at the same time a reward function r, that encodes the preferred actuation over environment variables (lines 14–15). Finally, the output expression specifies the intended results, in this case, the assignment to environment variables that maximizes the solve expression, and the preference constraints that are fulfilled by such assignment (line 17).

Ultimately, with the resulting output we can answer query $Q2$, determining what changes the intelligent system could make, in order to conciliate the occupants' preferences and the conditions of a given space, i.e. to resolve any arising conflicts; and query $Q3$, determining what adaptations are needed from an occupant (in terms of preferences over environment conditions) in order to fit in with a given place's current context and other occupants' preferences. To demonstrate this, we will further refer to the 3 previously described scenarios.

5.3. Experiments

The experiments are based on the scenarios presented in the motivation, Section 4.1, and address the two main steps of the system's operation: (i) detecting conflict; (ii) in case of a conflict detection, determining if it is solvable by automatic resolution through system action. In each scenario, the occupant Alice is initially alone in zone Z_{room} with a temperature a

```
1: var 0..400: temp;
2: constraint temp = 200;
3: constraint 190 <= temp ∧ temp <= 220;
4: constraint 195 <= temp ∧ temp <= 225;
5: solve satisfy;
```

b

```
1: var 0..400: temp;
2: set of int: prefConstrs = 1..2;
3: var set of prefConstrs: vpcs;
4: int: pc1 = 1;
5: int: pc2 = 2;
6: constraint (190 <= temp ∧ temp <= 220) <-> pc1 in vpcs;
7: constraint (210 <= temp ∧ temp <= 235) <-> pc2 in vpcs;
8: solve maximize sum (c in prefConstrs)
   (bool2int (c in vpcs))*400 + (-abs(temp - 200))*1;
9: output[ "temp: 200->", show(temp),
   " SatPrefConstrs: ", show(vpcs)];
```

c

```
1: $ ----------
```

d

```
1: $ temp: 200->210,  SatPrefConstrs: 1..2
2: $ ----------
3: $ ==========
```

Fig. 8. MiniZinc example models for constraint solving, and respective outputs: (a) Executable model for conflict detection in Scenario B; (b) Executable model for conflict resolution in Scenario C; (c) Output obtained with executing model (a); (d) Output obtained with executing model (b).

setpoint of 20 °C, modelled as $status(Z_{room}, E_{temp}) = 20$. A meeting is an activity $A_{meeting}$ for which Alice has a preference with respect to the temperature environment variable between 19 °C and 22 °C, modelled as $pref(O_{Alice}, A_{meeting}) = (19 \le temp \wedge temp \le 22)$. Scenario B, presents an overlap of conditions. Alice wants to have a meeting with another occupant, Bob, whose preference for meetings is to have temperatures from 19.5 °C to 22.5 °C, i.e., $pref(O_{Bob}, A_{meeting}) = (19.5 \le temp \wedge temp \le 22.5)$. In scenario C, Alice wants to meet with Claire, who prefers temperatures from 21 °C to 23.5 °C, modelled as $pref(O_{Claire}, A_{meeting}) = (21 \le temp \wedge temp \le 23.5)$. Here, we have a partial intersection in the preference ranges, but a conflict with the current status. Finally, in scenario D, Alice will meet with Dave, who prefers to meet with temperatures from 23 °C to 24.5 °C, modelled as $pref(O_{Dave}, A_{meeting}) = (23 \le temp \wedge temp \le 24.5)$. This case presents a conflict between occupant preferences.

5.3.1. *Conflict detection*

Finding out, for each scenario, whether the occupants having a meeting in the given room will result in conflict with the current room's conditions or with each others' preferences, can be formulated as finding the appropriate θ such that $\theta \models status(Z_{room}, E_{temp}) \wedge pref(O_{Alice}, A_{meeting}) \wedge pref(O_X, A_{meeting})$, where X is the occupant Bob, Claire or Dave, depending on the scenario.

Giving the CSP encoding of scenario B as input to MiniZinc, we expect the result to indicate that the model is satisfiable, since the current temperature of 20 °C satisfies both occupants, therefore no conflict exists this situation. Fig. 8(a) presents the MiniZinc model for conflict detection of Scenario B. To simplify the encoding, we work with integer values and normalized temperature in the range of 0–400, where 400 corresponds to 40 °C (line 1). The encoding of the current status constraint (line 2) is followed by Alice's and Bob's temperature preferences for meeting. Finally, we tell the CSP solver that this is a satisfaction problem (line 5). The result obtained by the execution of this model indicates that the model was satisfiable, as we anticipated. Fig. 8(c) shows the output line of MiniZinc.

However, as one would expect, the output of executing the MiniZinc models for conflict detection of scenarios C and D, presents them as unsatisfiable. In such cases, the system should proceed by determining if there is a viable resolution, as will be detailed in the next section.

5.3.2. *Automatic conflict resolution*

As we have seen, no conflict exists in Scenario B, therefore no action is required. In the case of scenario C, the environment status constraint is not satisfied, and the detection's results show it is indeed unsatisfiable. Given the model depicted in 8(b), it is possible to determine if there is a setpoint that would rectify this situation. As before, the temperature variable is declared (line 1), as well as the same occupant preference constraints involved in the conflict detection (lines 6–7). Each preference is associated to a previously declared variable (lines 2–5), which allows us to treat preferences as parameters to the model. The solve expression of this model maximizes the number of satisfied preference constraints and a reward function $r(E_{temp}, Z) = -abs(\texttt{temp - 200})$, which, in case of multiple solutions, will select the one that together with the constants corresponding to the constraint, minimizes the needed adjustment towards the current status of 20 °C (line 8). Moreover, the choice of $w_1 = 400$ and $w_2 = 1$ expresses that priority should be given to satisfying the most preference clauses. Finally, the output expression yields the assignment to the temperature variable, and the preference constraints that are fulfilled by the obtained assignment (line 9).

The results of executing this model are presented in 8(d), which indicates that an actuation on the temperature from the current status of 20 °C to 21 °C would conciliate the preferences of both occupants, while minimizing the needed temperature adjustment. Therefore we conclude, as expected, that Scenario C simply requires an actuation from the system to automatically resolve the conflict.

In scenario D, however, the system detects a non-satisfiable set of preference constraints, and therefore, that no actuation could conciliate the preferences of both occupants, to resolve the conflict. In such case, the system's output is a maximal set of satisfiable constraints that is to be used in deciding between two alternatives: (i) use an alternative conciliation strategy – choosing environment variable assignments that comply to another user-defined or system-defined goal; or (ii) pursue assisted conflict resolution – assisting the occupants in resolving the conflict situation.

5.4. Discussion

The results obtained with the above experiments demonstrate the simplicity and viability of our approach. It is important to mention that all the experiments we performed so far, including the ones herein presented, had a runtime of less than 100 ms on a desktop PC, which is particularly relevant for conflict detection, since the fastest it is, the more time can be spent on determining and taking appropriate action.

Another noteworthy aspect is that the presented MAX-CSP problem of minimizing the variable adjustment, i.e., of favouring the solution closer to the system's current status, which can be viewed as a lazy approach, is merely a sample of what can be done. The expressive power of MiniZinc's language allows the definition of a large number of goal functions with little more effort and time. Such goal functions can then be made available natively by the system, as pre-definable parameter configurations. Examples of such include: (i) the average value of the occupant preferences, or other variations of this calculation; (ii) maximize comfort, possibly using Predicted Mean Vote (PMV) estimates [83]; (iii) minimizing cost; (iv) minimizing energy consumption. Note that these goal functions cannot only be used separately, but can also be combined, by assigning each a specified weight ponderation. Moreover, when several services are available in a given zone, the system can also determine the best service combination (in terms of a given goal), and take it into account when deciding upon an actuation.

Additionally, assisted conflict resolution has the potential to further help the system learn about their occupants and provide them with custom and thus more useful strategies. Moreover, resolution information (involved occupants, violated constraints, assisted resolution choices, reaction to previous automatic resolutions, most frequently unsatisfied constraints, among others) may provide useful insight of occupants and of the most common problems. After enough data has been gathered, this insight can be used to further pursue conflict resolution through avoidance. Such conflict avoidance can take several forms: (i) improving the pre-defined goal functions; (ii) persuading occupants to adjust their preferences towards pre-defined goals; (iii) changing or adapting the default variable setpoints; (iv) providing suggestions of new equipment or occupant relocation to system administrators and/or building managers.

An interesting application of our solution would be trying to solve conflict while at the same time optimizing energy usage. To that end, integration with existing energy management systems would be required, for obtaining information regarding expected energy consumption of services, which may vary by zone, activity, occupancy and schedule. This data would then be used to update the parameters amenable to optimization, thus allowing the system to chose service actuations based on the current and expected energy costs of each available option.

6. Conclusions

The development of usable, intelligent and highly adaptive HBAS, is a multi-domain problem that has been receiving increasing attention. Many gaps still remain regarding automatic adaptation to the user, one of which is the use of effective conflict detection and resolution mechanisms, able to respond automatically and appropriately to a variety of decision-demanding scenarios. Conflicts occur when a user or application changes the environment state, causing an undesired context. Conflict detection is based on this context analysis. We presented an overview of ambient intelligence systems (in which HBAS are included) with respect to relevant aspects on the areas of context awareness and human–computer interaction, followed by a survey on HBAS and its multidisciplinary developments. The potential of ambient intelligence is underscored by companies becoming increasingly interested in the subject. In fact, one of the European pioneers of this area is a vice-president of Philips Research.

Ideas similar to ours have already proven useful in other areas of ambient intelligence such as modelling device capabilities and interactions for enabling multimedia service search [84], optimization and validation of user intentions for e-retailing [85], and constraint solving to address conflict in context-aware activity recognition [86].

This paper brings several contributions to the aforementioned topics, namely: (i) a survey on conflict (Section 3), both in the interpersonal sense and with regard to existing conflict detection and resolution methods in ambient intelligence systems; (ii) a conflict taxonomy, regarding different classification dimensions, for clarity (Section 3); (iii) the definition of a formal representation for intelligent environments conditions and relevant components (Section 4); (iv) the outline of a prototype system for automatic conflict detection and resolution and its technological underpinnings (Section 4); (v) the validation of our envisioned solution, which included validating the formal model and the prototype system (Section 5).

Our efforts are now being directed to further developments of our approach to automatic conflict resolution, namely on improving the formal model and extending the presented reasoning system.

Acknowledgement

This work was supported by national funds through *Fundação para a Ciência e a Tecnologia* (FCT), under project PEst-OE/EEI/LA0021/2013, and grant number SFRH/BD/47409/2008.

References

[1] H. Merz, T. Hansemann, C. Hübner, Building Automation: Communication Systems with EIB/KNX, LON und BACnet, in: Signals and Communication Technology, Springer, 2009.
[2] R. Bucceri, Latest Technology in Automated Home Control: System Design Manual Using X-10 & Hardwired Protocols, Silent Servant Inc., 2006.
[3] S. Davidoff, M. Lee, C. Yiu, J. Zimmerman, A. Dey, Principles of smart home control, in: Ubiquitous Computing, UbiComp'06, in: Lecture Notes in Computer Science, vol. 4206, Springer, 2006, pp. 19–34.
[4] M. Weiser, The computer for the twenty-first century, Scientific American 3 (1991).
[5] M. Hasan, K. Anh, L. Mehedy, Y. Lee, S. Lee, Conflict resolution and preference learning in ubiquitous environment, in: Computational Intelligence, in: Lecture Notes in Computer Science, vol. 4114, Springer, 2006, pp. 355–366.
[6] D. Chalmers, Sensing and Systems in Pervasive Computing: Engineering Context Aware Systems, Springer, 2011.
[7] P. Remagnino, G. Foresti, Ambient intelligence: a new multidisciplinary paradigm, IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans 35 (2005) 1–6.
[8] H. Nakashima, H. Aghajan, J. Augusto (Eds.), Handbook of Ambient Intelligence and Smart Environments, Springer, 2010.
[9] M. Weiser, Hot topics: ubiquitous computing, IEEE Computer 26 (1993) 71–72.
[10] A. Dey, Understanding and using context, Personal and Ubiquitous Computing 5 (2001) 4–7.
[11] A. Masoumzadeh, M. Amini, R. Jalili, Conflict detection and resolution in context-aware authorization, in: 21st International Conference on Advanced Information Networking and Applications Workshops, AINAW'07, IEEE, 2007, pp. 505–511.
[12] P. Dourish, What we talk about when we talk about context, Personal and Ubiquitous Computing 8 (2004) 19–30.
[13] T. Zimmer, Towards a better understanding of context attributes, in: Proceedings of the 2nd Conference on Pervasive Computing and Communications Workshops, IEEE, 2004, pp. 23–27.
[14] R. Neisse, M. Wegdam, M. van Sinderen, Trustworthiness and quality of context information, in: Proceedings of the 9th International Conference for Young Computer Scientists, ICYCS'08, pp. 1925–1931.
[15] I. Park, D. Lee, S. Hyun, A dynamic context-conflict management scheme for group-aware ubiquitous computing environments, in: Proceedings of the 29th International Computer Software and Applications Conference, Vol. 1, COMPSAC'05, IEEE, 2005, pp. 359–364.
[16] V. Tuttlies, G. Schiele, C. Becker, COMITY: conflict avoidance in pervasive computing environments, in: On the Move to Meaningful Internet Systems: OTM 2007 Workshops, in: Lecture Notes in Computer Science, vol. 4806, Springer, 2007, pp. 763–772.
[17] E. Syukur, S. Loke, P. Stanski, Methods for policy conflict detection and resolution in pervasive computing environments, in: In Policy Management for Web Workshop in Conjunction with WWW'05 Conference, ACM, 2005, pp. 10–14.
[18] M. Tentori, M. Rodriguez, J. Favela, An agent-based middleware for the design of activity-aware applications, IEEE Intelligent Systems 26 (2011) 15–23.
[19] M. Viroli, On competitive self-composition in pervasive services, Science of Computer Programming 78 (2013) 556–568.
[20] F. Tisato, C. Simone, D. Bernini, M. Locatelli, D. Micucci, Grounding ecologies on multiple spaces, Pervasive and Mobile Computing 8 (2012) 575–596.
[21] M. Locatelli, G. Vizzari, Awareness in collaborative ubiquitous environments: the multilayered multi-agent situated system approach, ACM Transactions on Autonomous and Adaptive Systems 2 (2007).
[22] M. Baldauf, S. Dustdar, F. Rosenberg, A survey on context-aware systems, International Journal of Ad Hoc Ubiquitous Computing (IJAHUC) 2 (2007) 263–277.
[23] T. Gross, Cooperative ambient intelligence: towards autonomous and adaptive cooperative ubiquitous environments, International Journal of Autonomous and Adaptive Communications Systems 1 (2008) 270–278.
[24] M. Alcañiz, B. Rey, New technologies for ambient intelligence, Ambient Intelligence 6 (2005) 3–15.
[25] M. Maybury, W. Wahlster, Intelligent user interfaces: an introduction, in: Readings in Intelligent User Interfaces, Morgan Kaufmann Publishers Inc., 1998, pp. 1–13.
[26] P. Markopoulos, Designing ubiquitous computer human interaction: the case of the connected family, in: Future Interaction Design, Springer, 2005, pp. 125–149.
[27] P. Markopoulos, B. de Ruyter, S. Privender, A. van Breemen, Case study: bringing social intelligence into home dialogue systems, Interactions 12 (2005) 37–44.
[28] P. Vernon, Some characteristics of the good judge of personality, The Journal of Social Psychology 4 (1933) 42–57.
[29] V. Venkatesh, M. Morris, G. Davis, F. Davis, User acceptance of information technology: toward a unified view, MIS Quarterly 27 (2003) 425–478.
[30] B. de Ruyter, P. Saini, P. Markopoulos, A. van Breemen, Assessing the effects of building social intelligence in a robotic interface for the home, Interacting with Computers 17 (2005) 522–541.
[31] P. Saini, B. de Ruyter, P. Markopoulos, A. van Breemen, Benefits of social intelligence in home dialogue systems, in: Proceedings of the Human–Computer Interaction, INTERACT'05, in: Lecture Notes in Computer Science, vol. 3585, Springer, 2005, pp. 510–521.
[32] D. Tapia, S. Rodríguez, J. Corchado, A distributed ambient intelligence based multi-agent system for Alzheimer health care, in: Pervasive Computing, in: Computer Communications and Networks, Springer, 2010, pp. 181–199.
[33] B. Fogg, Persuasive Technology: Using Computers to Change What We Think and Do, Morgan Kaufmann Publishers, 2003.
[34] R. Cialdini, Harnessing the science of persuasion, Harvard Business Review 79 (2001) 72–81.
[35] M. Kaptein, P. Markopoulos, B. de Ruyter, E. Aarts, Can you be persuaded? Individual differences in susceptibility to persuasion, in: Proceedings of the 12th IFIP International Conference on Human–Computer Interaction, INTERACT'09, Springer, 2009, pp. 115–118.
[36] I. Satoh, Visual components for pervasive computing management, in: IEEE International Conference on Pervasive Services, pp. 19–28.
[37] T. Yamazaki, The ubiquitous home, International Journal of Smart Home (IJSH) 1 (2007) 17–22.
[38] R. Ballagas, M. Ringel, M. Stone, J. Borchers, iStuff: a physical user interface toolkit for ubiquitous computing environments, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI'03, ACM, 2003, pp. 537–544.
[39] F. Kawsar, T. Nakajima, Persona: a portable tool for augmenting proactive applications with multimodal personalization support, in: Proceedings of the 6th International Conference on Mobile and Ubiquitous Multimedia, MUM'07, ACM, 2007, pp. 160–168.
[40] X.A.M. García-Herranz, P. Haya, P. Martín, Easing the smart home: augmenting devices and defining scenario, in: 2nd International Symposium on Ubiquitous Computing & Ambient Intelligence, Thomson Editorial, 2007, pp. 67–74.
[41] J. Ledlie, C. Ng, D. Holland, Provenance-aware sensor data storage, in: 21st International Conference on Data Engineering Workshops.
[42] S. Szewcyzk, K. Dwan, B. Minor, B. Swedlove, D. Cook, Annotating smart environment sensor data for activity learning, Technology and Health Care 17 (2009) 161–169.

[43] R. Aipperspach, E. Cohen, J. Canny, Modeling human behavior from simple sensors in the home, in: Pervasive Computing, in: Lecture Notes in Computer Science, vol. 3968, Springer, 2006, pp. 337–348.
[44] R. Schaefer, W. Muller, J. Groppe, Profile processing and evolution for smart environments, in: Ubiquitous Intelligence and Computing, in: Lecture Notes in Computer Science, vol. 4159, Springer, 2006, pp. 746–755.
[45] M. Ruta, F. Scioscia, E.D. Sciascio, G. Loseto, Semantic-based enhancement of ISO/IEC 14543-3 EIB/KNX standard for building automation, IEEE Transactions on Industrial Informatics 7 (2011) 731–739.
[46] G. Loseto, F. Scioscia, M. Ruta, E.D. Sciascio, Semantic-based smart homes: a multi-agent approach, in: Proceedings of the 13th Workshop on Objects and Agents, Vol. 892, CEUR-WS, 2012.
[47] V. Garg, N. Bansal, Smart occupancy sensors to reduce energy consumption, Energy and Buildings 32 (2000) 81–87.
[48] Y. Agarwal, B. Balaji, R. Gupta, J. Lyles, M. Wei, T. Weng, Occupancy-driven energy management for smart building automation, in: Proceedings of the 2nd Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings, BuildSys'10, ACM, 2010, pp. 1–6.
[49] A. Cziker, M. Chindris, A. Miron, Implementation of fuzzy logic in daylighting control, in: 11th International Conference on Intelligent Engineering Systems, INES'07, pp. 195–200.
[50] M. Lah, B. Zupancic, J. Peternelj, A. Krainer, Daylight illuminance control with fuzzy logic, Solar Energy 80 (2006) 307–321.
[51] H. Zhou, M. Rao, K. Chuang, Artificial intelligence approach to energy management and control in the HVAC process: an evaluation, development and discussion, Developments in Chemical Engineering and Mineral Processing 1 (1993) 42–51.
[52] A. Galasiu, M. Atif, R. MacDonald, Impact of window blinds on daylight-linked dimming and automatic on/off lighting controls, Solar Energy 76 (2004) 523–544.
[53] E. Lee, D. DiBartolomeo, E. Vine, S. Selkowitz, Integrated performance of an automated Venetian blind/electric lighting system in a full-scale private office, in: Proceedings of the ASHRAE/DOE/BTECC Conference, Thermal Performance of the Exterior Envelopes of Buildings VII.
[54] P. Littlefair, J. Ortiz, C. Bhaumik, A simulation of solar shading control on UK office energy use, Building Research & Information 38 (2010) 638–646.
[55] C. Reinisch, M. Kofler, F. Iglesias, W. Kastner, ThinkHome energy efficiency in future smart homes, EURASIP Journal on Embedded Systems (2011).
[56] W. Wang, S. Ting, Development of a computational simulation model for conflict management in team building, International Journal of Engineering Business Management 3 (2011) 9–15.
[57] R. Fisher, Sources of conflict and methods of resolution, The American University, 2000.
[58] D. Sandole, S. Byrne, I. Sandole-Staroste, J. Senehi, Handbook of Conflict Analysis and Resolution, Routledge, 2009.
[59] C. Moore, The Mediation Process: Practical Strategies for Resolving Conflict, Jossey-Bass Publishers, 2003.
[60] G. Furlong, The Conflict Resolution Toolbox: Models and Maps for Analyzing, Diagnosing and Resolving Conflict, John Wiley & Sons Canada, Ltd., 2005.
[61] K. Thomas, Conflict and conflict management: reflections and update, Journal of Organizational Behavior 13 (1992) 265–274.
[62] K. Thomas, R. Kilmann, Comparison of four instruments for measuring conflict behavior, Psychological Reports 42 (1978).
[63] W. Alshabi, S. Ramaswamy, M. Itmi, H. Abdulrab, Coordination, cooperation and conflict resolution in multi-agent systems, in: Innovations and Advanced Techniques in Computer and Information Sciences and Engineering, Springer, 2007, pp. 495–500.
[64] W. Jacak, K. Pröll, Heuristic approach to conflict problem solving in an intelligent multiagent system, in: Computer Aided Systems Theory, EUROCAST'07, in: Lecture Notes in Computer Science, vol. 4739, Springer, 2007, pp. 772–779.
[65] I. Armac, M. Kirchhof, L. Manolescu, Modeling and analysis of functionality in ehome systems: dynamic rule-based conflict detection, in: 13th Annual IEEE International Symposium and Workshop on Engineering of Computer Based Systems, ECBS'06.
[66] D. Retkowitz, S. Kulle, Dependency management in smart homes, in: Distributed Applications and Interoperable Systems, in: Lecture Notes in Computer Science, vol. 5523, Springer, 2009, pp. 143–156.
[67] G. Huerta-Canepa, D. Lee, A multi-user ad-hoc resource manager for smart spaces, in: Proceedings of the International Symposium on a World of Wireless, Mobile and Multimedia Networks, pp. 1–6.
[68] E. Lupu, M. Sloman, Conflict analysis for management policies, in: Proceedings of the 5th IFIP/IEEE International Symposium on Integrated Network Management V: Integrated Management in a Virtual World, Chapman & Hall, Ltd., 1997, pp. 430–443.
[69] S. Jiao, Y. Liu, X. Qi, Y. Zhu, J. Wang, Detecting conflict policy rules with concept lattice, in: 5th International Conference on Wireless Communications, Networking and Mobile Computing, WiCom'09, pp. 1–4.
[70] S. Jiao, Y. Liu, H. Hu, D. Wei, Y. Zhang, Dynamic policy access model based on formal concept analysis, in: 4th International Conference on Wireless Communications, Networking and Mobile Computing, WiCOM'08, pp. 1–5.
[71] L. Capra, W. Emmerich, C. Mascolo, CARISMA: context-aware reflective middleware system for mobile applications, IEEE Transactions on Software Engineering 29 (2003) 929–945.
[72] D. Bell, J. Grimson, Distributed Database Systems, Addison-Wesley, 1992.
[73] V. Ramachandran, S. Ramaswamy, P. Rajan, Complex negotiation protocols for a distributed simulation, 2001.
[74] H. Kung, C. Lin, Application-layer context-aware services for pervasive computing environments, in: Proceedings of the 1st International Conference on Innovative Computing, Information and Control, Vol. 3, ICICIC'06, pp. 229–232.
[75] C. Becker, M. Handte, G. Schiele, K. Rothermel, PCOM—a component system for pervasive computing, in: Proceedings of the 2nd IEEE Annual Conference on Pervasive Computing and Communications Workshops, pp. 67–76.
[76] T. Silva, L. Ruiz, A. Loureiro, How to conciliate conflicting users' interests for different collective, ubiquitous and context-aware applications? in: IEEE 35th Conference on Local Computer Networks, LCN'10, pp. 288–291.
[77] T. Silva, L. Ruiz, A. Loureiro, Conflicts treatment for ubiquitous collective and context-aware applications, Journal of Applied Computing Research 1 (2011) 33–47.
[78] E. Syukur, D. Cooney, S. Loke, P. Stanski, Hanging services: an investigation of context-sensitivity and mobile code for localised services, in: Proceedings of the IEEE International Conference on Mobile Data Management, pp. 62–73.
[79] P. Chen, The entity-relationship model: toward a unified view of data, ACM Transactions on Database Systems 1 (1976) 9–36.
[80] M. Garey, D. Johnson, Computers and Intractability: A Guide to the Theory of NP-completeness, W.H. Freeman, 1979.
[81] N. Nethercote, P. Stuckey, R. Becket, S. Brand, G. Duck, G. Tack, MiniZinc: towards a standard CP modelling language, in: CP, pp. 529–543.
[82] K. Marriott, P. Stuckey, L. Koninck, H. Samulowitz, An introduction to MiniZinc, University of Melbourne/G12, 2012.
[83] ANSI/ASHRAE, ANSI/ ASHRAE Standard 55-2004, Thermal comfort conditions for human occupancy, Technical Report, American Society of Heating, Air-Conditioning, and Refrigeration Engineers, Inc., 2004.
[84] N. Roy, A. Roy, S. Das, Context-aware resource management in multi-inhabitant smart homes a Nash H-learning based approach, in: Proceedings of the 4th Pervasive Computing and Communications, PerCom'06, pp. 158–169.
[85] S. Bromuri, V. Urovi, K. Stathis, Game-based E-retailing in GOLEM agent environments, Pervasive and Mobile Computing 5 (2009) 623–638.
[86] C. Filippaki, G. Antoniou, I. Tsamardinos, Using constraint optimization for conflict resolution and detail control in activity recognition, in: Proceedings of the 2nd International Conference on Ambient Intelligence, AmI'11, Springer, 2011, pp. 51–60.