Speculative Orientation and Tracking System

João Ramos[1], **Paulo Novais**[1], **Ken Satoh**[2],
Tiago Oliveira[1] **and José Neves**[1]

[1]CCTC/Department of Informatics
University of Minho
Braga, Portugal
{jramos,pjon,toliveira,jneves}@di.uminho.pt

[2]National Institute of Informatics
Hitotsubashi, Chyoda-ku
Tokyo, Japan
ksatoh@nii.ac.jp

ABSTRACT

The current progresses at the intersection of computer science and health care have the potential of greatly improving the living conditions of people with disabilities by removing obstacles that impair the normal unfolding of their everyday lives. Assistive technologies, as an application of scientific knowledge, aim to help users with their diminished capacities and, usually, imply a small adaptation from individuals so that they can use the devices that convey assistive functionalities. One of the most commonly diminished capabilities is that of spatial orientation. This is mirrored by several research works whose goal is to help human beings to travel between locations. Once set up, most of the systems featured in these research works requires changes in the configurations to be made manually in order to achieve a better adjustment to the user. In order to overcome this drawback, the work presented herein features a framework of Speculative Computation to set up the computation of the next step of a user using default values. The consequence of the application of the framework is a faster reaction to user stimuli, which may result in issuing warnings when he is likely to choose the wrong direction.

Keywords: Cognitive disabilities, Mobile application, Guidance, Person tracking, Ambient intelligence, Logic programming.

2000 Mathematics Subject Classification: 68N17, 68T27.

1 Introduction

Technology plays an important role in peoples lives in so many different domains such as business, education, communication, social relationships, and so on. With this ever increasing penetration of technology in everyday life, it was only a matter of time until the development of mainstream technology started having an impact in the lives of people with disabilities. The technology developed specifically for the purpose of improving the lives of people with diminished capacities received the designation of assistive technology. The concept was first legally

established in the United States in 1988, by the Technology-Related Assistance for Individuals with Disabilities Act, also referred to as Tech Act. This act was extended in 1998 by the Assistive Technology Act (Alper and Raharinirina, 2006; Scherer, Hart, Kirsch and Schulthesis, 2005). In it, it is stated that "assistive technology is any item, piece of equipment, or product system. . . that is used to increase, maintain, or improve functional capabilities of individuals with disabilities". Passing this act was the first step in seeking to disseminate assistive technologies and provide them to people with disabilities. Indeed, since the first form of the act was passed, the attention towards this kind of technology has been increasing, which led, up until now, to the development of devices to be embedded in a home environment, in order to assist users and monitor their activities, movements, and health on a daily basis (Stefanov, Bien and Bang, 2004).

The existence of these smart home environments, also called smarthouses (Sadri, 2007), minimizes the loss of independence from users with disabilities. This technology also enables caregivers and physicians to have access to information collected from the environment (Carneiro, Novais, Costa, Gomes, Neves, Tscheligi, De Ruyter, Markopoulus, Wichert, Mirlacher, Meschterjakov and Reitberger, 2009), which, in turn, leads to letting the user (*e.g.*, an elder or a person with cognitive disabilities) stay at home longer with increased independence, rather than resorting to admission to a health care institution. However, users can only benefit from the advantages of smarthouses indoors. So, it is necessary to develop means through which people with disabilities can have an active role in the outside world, participate in their community, and feel perfectly integrated in society.

The focus of this work is placed on people with cognitive disabilities. According to the Diagnostic and Statistical Manual for Mental Disorders (Spitzer, Gibbon, Skodol and First, 1994), cognitive disabilities are a medical condition in which individuals have more difficulties in performing mental tasks. These tasks include self-care, communication, use of community resources, work, leisure, and so on. There is a classification for cognitive disabilities according to their severity, which includes the following phases: mild, moderate, severe, and extreme. The main criterion for a diagnosis is the extent to which the mental functions of an individual are compromised, which is also related to the cause of the disabilities. They can be brought about by events such as strokes, Alzheimers disease, mental handicaps, and traumatic brain injuries.

Orientation is one of the mental tasks that are most affected in people with cognitive disabilities. If someone is unable to go to where activities take place, it becomes impossible to participate, which may cause a greater social isolation and bring about, or accentuate, feelings of depression. To address this issue, researchers have been working on new ways of using the available technology to increase the outdoor independence of users, namely in helping them to travel alone between locations. In order to accompany users in their everyday lives, the developed systems have to be available in devices that users can carry around, they have to be small, lightweight, and resistant (Dawe, 2006; Dawe, 2007). These physical features can be found in modern smartphones and tablets, which serve has the support for many of the research works now in progress. At the same time, the interfaces provided by these systems play an important role in their acceptance (Dawe, 2006; Dawe, 2007; Friedman and Bryen, 2007). They have

to deliver all the important functionalities, yet be simple enough to allow people with cognitive disabilities to easily use them. Any distractive element should be removed and the instructions should be provided in simple texts.

After analysing the current state of the art in the area of orientation systems for people with cognitive disabilities and identifying the limitations of current systems, a system (Ramos, Anacleto, Costa, Novais, Figueiredo and Almeida, 2012) was developed for the purpose of guiding the user to different locations using simple interfaces, with the possibility of his caregiver monitoring his movements. This increases the freedom of both parts. The user can travel alone while the caregiver is engaged in other activities. The differentiating factor, when compared to other existing systems, is that this system learns the most frequent behaviours of a user and analyses his activity patterns in order to generate default values for Speculative Computation. In sensible decision points, when a user has to choose a direction to follow, Speculative Computation provides a way of managing the lack of real information by providing a tentative computation for the path the user should follow, based on the extracted defaults. This tentative computation can be used to issue warnings in order to alert the user in the event of his following the wrong direction.

This paper is organized as follows. Section 2 contains the main related work on orientation systems for people with cognitive disabilities and reasoning under incomplete information in the context of assistive technologies. The criterion used to gather the current research works was that the orientation systems should be hosted by smartphones. Section 4 provides a description of the system in development, with its latest improvements. The framework for Speculative Computation is formulated in section 5. Finally, conclusions are drawn and future work considerations are made in section 6.

2 Related Work

The work presented herein focuses on orientation systems and ways of anticipating the next step of a user in the event that he makes a mistake in his path and a warning has to be issued. In the current orientation systems aimed for people with cognitive disabilities, this kind of safeguard has not been extensively explored. To demonstrate this, the following subsections provide related work on the topics of orientation systems and reasoning under incomplete information.

2.1 Orientation Systems

The technological development experienced in recent years increased the availability of small, portable, handheld devices with high processing power. Nowadays, purchasing a smartphone is within reach of most people. These devices are capable of running applications with various purposes, but one of particular interest for the theme of this work is that of orientation. The incorporation of a Global Positioning System (GPS) module in a smartphone made possible the development of navigation applications. However, these applications serve a general purpose, they were not developed to be used by people with cognitive disabilities. Recognizing the importance of orientation for people with disabilities and the pivotal role smartphones could have

in the development of cognitive assistants, researchers started the development of systems fit for the characteristics of these individuals.

The work of (Carmien, Dawe, Fischer, Gorman, Kintsch and Sullivan, 2005) features a system that enables a user to travel between locations using a public transportation system (*e.g.*, the bus). They devised an interface that enabled caregivers to create lists of activities that should be carried out by people with cognitive disabilities. The role of their system is to deliver those tasks in a simple and comprehensible way while assisting the user in his travels. As such, the components of the system include: a personal travel assistant that uses real-time GPS data from the bus fleet to deliver just-in-time prompts to the user; a mobile prompting client and a prompting script configuration tool for caregivers; and a monitoring system that collects real-time task status from the mobile client and alerts the support community of potential problems.

Liu, Hile, Borriello, Kautz, Brown, Harniss and Johnson 2009 make use of static images with overlaid arrows, audio messages, and text messages in their prototype for guiding a user. The applicability of the system was tested with several real-world scenarios. The underlying principle of this work is to find models to produce tailored pedestrian wayfinding directions for people with cognitive impairment. They developed a framework based on a Markov Decision Process (MDP), which enables the representation of user preferences as costs in a network. Another feature is the use of recognizable landmarks near the user to facilitate guidance.

AlzNav (Fraunnhover Portugal, 2012) is an orientation system developed by Fraunhofer Portugal. Its functionalities include monitoring and alerts, navigation, call for help, and call for a taxi. The application runs on an Android device, and its interface is very simple. Its main focus is to to make the interpretation of the content seen on the screen easy. The direction a user should take is shown using an arrow that works as a compass, spinning around according to the direction the user sets on the mobile device. Along with the arrow, the user also sees the distance he must travel in that direction. The application also sends updates about the situation of the user to the caregiver through SMSs.

These are some of the most important works in the area of orientation systems for people with cognitive disabilities. At the level of the orientation method, i.e. the way in which orientation is presented to the user, the available systems either use static images or a compass-like display. Technologies such as augmented reality have not been explored enough in this domain. Augmented reality can increase the interactivity of applications and facilitate the understanding of instructions. The prompts of the applications assist the interpretation of the information provided by the orientation method and are normally audio and visual. Most of the existing systems work with a predetermined route, being useful only to take a user from a specific point A to a specific point B. There are only a few works that analyse the frequent behaviours of users to try to infer his route, recognize activities, or issue reminders if something is deviating from its normal course.

In previous work (Ramos et al., 2012; Ramos, Anacleto, Novais, Figueiredo, Almeida and Neves, 2013; Ramos, Costa, Novais and Neves, 2014; Ramos, Satoh, Novais and Neves, 2014), we have presented a functional prototype of an orientation and localization system. It is delivered through and Android application that makes use of augmented reality to guide the user along his route. It also enables the caregiver to know the location of the user in real time.

With the purpose of enhancing the guiding experience, a framework of Speculative Computation was devised. With the tentative computation based on default values, it is possible to generate the next step forward before it materializes. This feature can be used as the core of an alert system. The remaining content of this paper explains how this framework was integrated in the orientation system and describes the modifications and adjustments made to the system in order to accommodate it.

3 Reasoning Under Incomplete Information

The issue of incomplete information in orientation systems has not been widely explored. In such systems, it would be advantageous to explore patterns of user behaviour in order to know frequent habits of an individual so as to provide him a better guidance. This has a particular application in people with cognitive disabilities. Given their diminished cognitive capacity, they are prone to errors when following instructions, which makes it easy for them to get lost. As such, identifying the critical points of a route, where an individual is most likely to choose the wrong path, would allow a system to issue warnings in order to avoid the situation. There are, however, some difficulties with this. Given the complexity that travelling routes may have, it is significantly difficult to analyse them and extract patterns. Moreover, there is a need for a reasoning model that accommodates these patterns and structures reasoning with them.

One of such reasoning models is the Markov Decision Process (MDP) (Littman, Dean and Kaelbling, 1995), a discrete time stochastic control process with four components: a set of states, which represent the current status of the world; a set of actions, which represent possible alternatives to act; a set of state transitions, representing the effects of the actions; and a set of immediate rewards, specifying the reward after taking an action. The objective of MDPs is to map the elements of a state to the possible actions one may take so as to maximize the immediate reward after taking the action. For cases in which the state of the world cannot be fully grasped, *i.e.*, when there is incomplete information, there is a special type of MDP, called the Partially Observable Markov Decision Process (POMDP) (Chong, Kreucher and Hero, 2009). In it, an agent uses information from previous iterations in order to estimate the current process state, based on a probability distribution over the possible states. POMDPs were initially developed in the 1980s by the artificial intelligence community has a way of searching for near optimal solutions for complex planning problems under uncertainty. POMDPs have been used in artificial intelligence for robot navigation (Kaelbling, Littman and Cassandra, 1998), healthcare (Bennett and Hauser, 2013), and assistive technology (*e.g.*, assisting people with dementia in hand washing (Hoey, Bertoldi, Poupart and Mihailidis, 2007)). The drawbacks associated with both MDPs and POMDPs is that they require data-intensive estimation steps to get accurate estimation models and their is quite complex.

Another reasoning model is Case-based Reasoning (CBR) (Kolodner, 1992). CBR models are based on recalling past experiences which are similar to the current one and using the solution of those past experiences in the new situation. Another form of putting the problem would be to recall things that went wrong in past experiences and avoid repeating the same mistakes. The advantage of this approach is that it resembles human reasoning in problem solving.

The effectiveness of a CBR system depends highly on the experiences of the reasoner, its ability to understand the new experiences in terms of the old ones, its capacity to adapt an old case to the new one, and its capacity to evaluate a solution. Depending on the particular objective of the CBR implementation, *i.e.*, whether the new situation should be considered identical to an old one based on its similarities and differences or the goal is to get a solution for the new case based on the adaptation of a previous one, the type of CBR used can be classified as interpretative CBR or problem solving CBR (Lopez and Plaza, 1997). It is a cyclic process consisting of four phases: retrieval of past cases; reusing a set of best past cases; revising the solutions in order to avoid poor ones; and retaining new solutions. CBR has been widely applied in clinical decision making in problems of planning and knowledge management for patients with Alzheimer's disease (Corchado, Bajo and Abraham, 2008), in diagnosis and classification of stress-related disorders (Begum, Ahmed, Funk, Xiong and Von Schéele, 2009), and other applications in the field of assistive technologies. As far as this research goes there are no significant examples of CBR applied to orientation systems. The drawbacks of CBR are related with the great number of cases and the large processing time it takes to find similar cases in the case-base.

The final model exposed in this section, and the object of study of this work, is Speculative Computation. Speculative Computation was derived from Default Reasoning and Abductive Logic Programming. It was first presented in Satoh et al. (Satoh, Inoue, Iwanuma and Sakama, 2000) as a method for problem solving in multi-agent systems when the communication between agents is not ensured. Speculative Computation deals with incomplete information by resorting to a set of default ground literals which represent the information being exchanged. Based on these defaults, tentative computations are made. When the information arrives from the system components, the computations are revised and their consistency is checked against it. Based on this, some processes are started and others are removed. The initial setting for which Speculative Computation was presented was that of a master-slave multi-agent system where only the master performs Speculative Computation and the abducibles assume the form of ground literals. Speculative Computation will be explained in more detail in the subsequent sections of this paper.

The search for reasoning models was conducted by taking into consideration the domain of application of this work. As such, only models applied in situations that resemble the context presented herein were considered.

4 System Description

The aim of the CogHelper system is to help people with cognitive disabilities as well as their caregivers, through a lightweight application . Its main purpose is to assist the user in moving between two locations without getting lost. In order to help the caregiver in his tasks, the system also has tracking functionalities which allow the caregiver to locate the user at any time.

The system was devised for outdoor use and its architecture can be seen in Figure 1. It is composed of server-side components, such as the Database and the Agency for the Integration

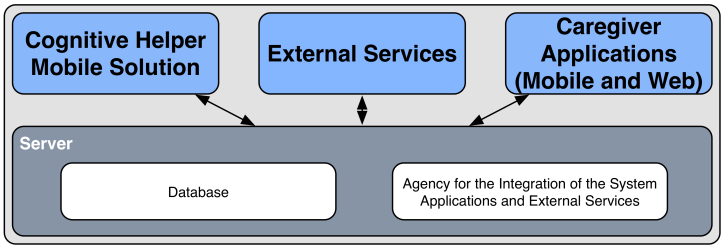

Figure 1: Architecture of the CogHelper System.

of System Applications and External Services Modules, and client side components, such as the Cognitive Helper Mobile Solution, the External Services, and the Caregiver Applications. All the services are connected to the Server, making it the core of the system. The Database Module contains all the information necessary for the system to accomplish its tasks. This information includes usernames, locations, and points of interest. As for the Agency for Integration of System Applications and External Services Module, it provides the facilities to communicate with both internal and external services.

The component of the system aimed specifically for people with cognitive disabilities to use is the Cognitive Helper Mobile Solution which encompasses three layers, displayed in Figure 2. It was developed for Android OS and its main functionality is orientation, enabling the user to travel independently without getting lost. The first layer is the Localization Layer which has methods for getting the current location of the user. This is done using either the GPS of the mobile device or the network through a coarse location. With the information of user location, it is possible to determine the frequent paths of the user. As for the Decision Layer, it provides warnings in order to ensure that the user travels in the right direction. It does so through a Decision Algorithm which crosses data from the camera, accelerometer, and the magnetic sensor with information about the destination and the correct path.

The directions are given through an augmented reality Interface such as the one depicted in Figure 3. The system keeps the caregiver informed of the steps of the user through e-mails and short messages. If the system detects that the user is disorientated or confused through his shaking the device too much or by going forwards and backwards in the same restricted area, the system automatically makes a phone call to the caregiver. The third and last layer

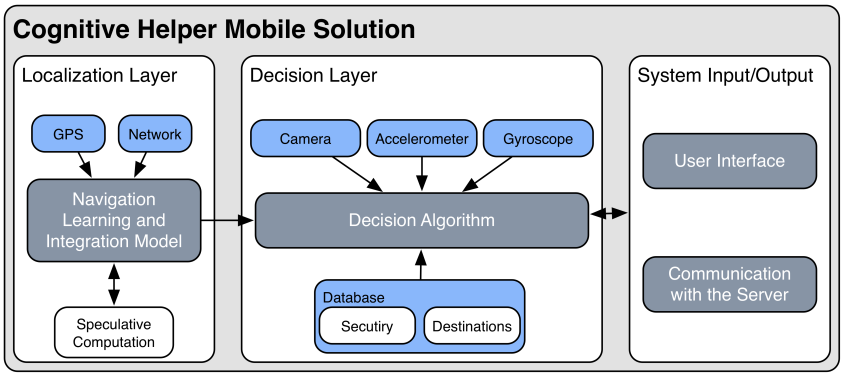

Figure 2: Information layers of the Cognitive Helper Mobile Solution.

Figure 3: Augmented reality interface of the CogHelper Mobile Application for users with cognitive disabilities.

is the System Input/Output. It features a User Interface and a communication module whose function is to pass on the information between the server and the user. Such information may include the update of user destinations, the update of the current location, and so forth.
There are two applications for caregivers, a mobile application for Android OS and a Web application. Their main goal is to inform the caregiver about the current location and state of the user. Despite this, there are some differences between the two. While the Web application is more complex, enabling the caregiver to add, edit or remove user preferences,(*e.g.* destination points), the mobile application is simpler in terms of the functionalities it provides, focusing more on monitoring tasks.
To improve the features of the system, Speculative Computation was included in the mobile application for people with cognitive disabilities under the Localization Layer, since it needs information about the current location of the user, provided by the GPS device or by the network.

5 A Speculative System for Users with Cognitive Disabilities

The procedure initially presented by (Kakas, Kowalski and Toni, 1998) on abduction in logic programming was a starting point for the theory of Speculative Computation and Abduction. This theory was an extension of (Kakas et al., 1998) procedure made by Satoh (Satoh et al., 2000). Under this theory the computation of a given action/task continues when facing an incomplete information scenario. Applying this theory means that the computation does not stop (or does not suspend) when there is a lack of information. Using a default value the computation continues and tries to generate a tentative solution for the problem. Whenever the missing information is received, the current computation is re-examined in order to verify if the default value is consistent with the real one.
During the execution of the speculative framework the computation may change between two phases, Process Reduction Phase and Fact Arrival Phase. During the normal execution the process is in the former phase changing temporally to the latter one when a value is returned. Thus, the Fact Arrival Phase is considered an interruption phase.

In this section we define the framework of Speculative Computation that is used and present a case to show the execution of the framework.

5.1 Framework of Speculative Computation

To develop an orientation method for people with cognitive disabilities using Speculative Computation is necessary to define a framework (designed by SF_{OM}) according to the tuple $\langle \Sigma, \mathcal{E}, \Delta, \mathcal{A}, \mathcal{P}, \mathcal{I} \rangle$, which consists of five sets. The first element of the tuple (Σ) represents a finite set of constants, which contains each system module. The set represented by $\mathcal{E}$ includes the predicates called external predicates. When Q is a literal belonging to an external predicate and S is the identifier of the information source, $Q@S$ is called an askable literal. We define $\sim(Q@S)$ as $(\sim Q)@S$.
Default values are assumed whenever the information is incomplete. They are included in the set represented by Δ (default answer set). This is a set of ground askable literals which satisfies the following condition: Δ does not contain at the same time $p(t_1, \ldots, t_n)@S$ and $\sim p(t_1, \ldots, t_n)@S$. $\mathcal{A}$ is the set of abducible predicates. Q is called abducible when it is a literal with an abducible predicate.
The set of rules ($\mathcal{P}$) that the program is going to execute are in the form:

- $H \leftarrow B_1, B_2, \ldots, B_n$ where H is a positive ordinary literal and each of $B_1, \ldots, B_n$ is an ordinary literal, an askable literal or an abducible literal; and
- H is the non-empty head of the rule (named $head(R)$) in which R is the rule of the form $H \leftarrow B_1, \ldots, B_n$. The body of the rule is represented by $B_1, \ldots, B_n$ (named $body(R)$) and may be replaced by the boolean value true.

The last set, denoted by $\mathcal{I}$, contains the integrity constraints, which do not allow contradictions during the execution of the speculative framework. The integrity constraints are in the form $\bot \leftarrow B_1, \ldots, B_n$ where $\bot$ is the symbol for the contradiction. $B_1, \ldots, B_n$ are ordinary literals, askable literals or abducibles. At least one of $B_1, \ldots, B_n$ must be an askable literal or an abducible.
An askable literal may have two different meanings, namely:

- An askable literal $Q@S$ in a rule $R \in P$ represents a question that is asked to a system module S;
- An askable literal in Δ denotes a default true value ($true$ or $false$), *i.e.*, $p(t_1, \ldots, t_n)@S \in \Delta$, $p(t1, \ldots, tn)@S$ is usually true for a question to a system module S, and $\sim p(t_1, \ldots, t_n)@S \in \Delta$, $p(t_1, \ldots, t_n)@S$ is generally false for a question to a system module S.

In the logic program given below the literal $path(a, b)$ denotes that there is a physical connection between the locations a and b, thus the user may travel between them. The literal $show\_next\_point$ is used to indicate that the system must show the next location to which the user should travel. This location may be an intermediate point or the final destination. Whenever the user travels in the wrong direction the literal $show\_user\_warning$ is activated indicating

to the system that it must alert the user. In the set of the external predicates there are the predicates $user\_travel(a, b)$ (which says that the user will travel from location a to location b) and $included(a)$ (to indicate if a location a is part of the route). These external predicates ask information from sources gps_sensor and $recognizer$, respectively. The former verifies if the user is travelling from point A to B. The latter checks if point B is included in the set of valid locations. The framework of Speculative Computation that is given below ensures that the user is travelling in the correct path and assesses the need for issuing alerts when he misses a turning point. This framework is given in terms of the following logic programming suite.

- $\Sigma = \{gps\_sensor, recognizer\}$
- $\mathcal{E} = \{user\_travel, included\}$
- $\Delta = \{user\_travel(1,3)@gps\_sensor, \sim user\_travel(1,2)@gps\_sensor,$
  $user\_travel(3,4)@gps\_sensor, user\_travel(4,3)@gps\_sensor,$
  $user\_travel(3,2)@gps\_sensor,$
  $included(1)@recognizer, included(2)@recognizer,$
  $included(3)@recognizer, \sim included(4)@recognizer$
- $\mathcal{A} = \{show\_next\_point, show\_user\_warning\}$
- $\mathcal{P}$ is a mark of the following set of rules:

$$
\begin{aligned}
&guide(A, A) \leftarrow .\\
&guide(A, B) \leftarrow\\
&\qquad path(A, F),\\
&\qquad show\_next\_point(F),\\
&\qquad user\_travel(A, F)@gps\_sensor,\\
&\qquad guide(F, B).\\
&guide(A, B) \leftarrow\\
&\qquad path(A, F),\\
&\qquad user\_travel(A, F)@gps\_sensor,\\
&\qquad show\_user\_warning(F),\\
&\qquad guide(F, B).\\
&path(1, 2) \leftarrow .\\
&path(1, 3) \leftarrow .\\
&path(3, 4) \leftarrow .\\
&path(3, 2) \leftarrow .\\
&path(4, 3) \leftarrow .
\end{aligned}
$$

- $\mathcal{I}$ denotes the following set of integrity constraint or invariants:

$$
\begin{aligned}
&\bot \leftarrow\\
&\qquad show\_next\_point(F),\\
&\qquad \sim included(F)@recognizer.
\end{aligned}
$$

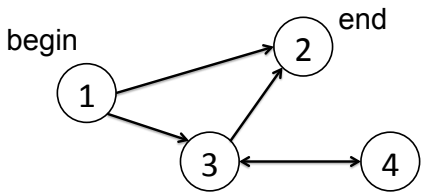

Figure 4: Possible ways to travel between locations 1 and 2.

$$\begin{aligned}\bot \leftarrow\\ &show\_user\_warning(F),\\ &included(F)@recognizer.\end{aligned}$$

The two invariants included in set $\mathcal{I}$ ensure that the system does not show a warning when the user travels in the correct path and that it does not show a location that is not valid (is not part of the route) as the next travelling point.
In the setting depicted above the system assumes as default that the user will travel between locations 1 and 2 through an intermediate location 3. Figure 4 presents the possible paths that may be done by the user.

5.2 Preliminary Definitions

To be able to create a proof procedure for the previously described frame-work there is the need of defining some concepts.

Definition 5.1. The tuple $\langle GS, OD, IA, ANS\rangle$ represents a process in which GS (Goal Set) is a set of extended literals to prove. These literals express the current status of an alternative computation; Outside Defaults (OD) is the set of askable literals and represents the assumed information about the outside world during the process; the set of negative literals or abducibles is named Inside Assumptions (IA) and contains the values that are assumed during the process; finally, the set ANS is the set of instantiations of variables in the initial query.

Definition 5.2. PS (Process Set) is a set of processes. AAQ is a set of askable literals that have already been asked. The Current Belief State (CBS) is the set of askable literals.

The set of processes PS expresses all the alternative computations considered. AAQ ensures that redundant questions are not made to the sensors. The current status of the outside world is expressed through the CBS. During the execution of the framework there are different types of process, which is important to define: active processes and suspended processes.

Definition 5.3. Let $\langle GS, OD, IA, ANS\rangle$ be a process and CBS be a current belief state. A process is active with respect to CBS if $OD \subseteq CBS$. A process is suspended with respect to CBS otherwise.

The previous definition states that an active process is a process in which the outside defaults are consistent with the current belief state.

5.3 Process Reduction Phase

During this phase a process may suffer some changes. In the next description a changed PS, AAQ and CBS are defined as $NewPS$, $NewAAQ$ and $NewCBS$.

Initial Step: Let GS be an initial goal set. The tuple $\langle GS, \emptyset, \emptyset, ANS\rangle$ is given to the proof procedure where ANS is a set of variables in GS. That is, $PS = \{\langle GS, \emptyset, \emptyset, ANS\rangle\}$. Let $AAQ = \emptyset$ and $CBS = \Delta$.

Iteration Step: Do the following:

- ▷ **Case 1:** If there is an active process $\langle \emptyset, OD, IA, ANS\rangle$ with respect to CBS in PS, terminate the process by returning outside defaults OD, inside assumptions IA, and instantiation for variables ANS.
- ▷ **Case 2:** If there is no active process, terminate the process by reporting a failure of the goal;
- ▷ **Case 3:** Select an active process $\langle GS, OD, IA, ANS\rangle$ with respect to CBS from PS and select an extended literal L in GS. Let $PS' = PS - \{\langle GS, OD, IA, ANS\rangle\}$ and $GS' = GS - \{L\}$. For the selected extended literal L, do the following:
  - **Case 3.1:** If L is a positive ordinary literal, $NewPS = PS' \bigcup \{\langle (\{body(R)\} \bigcup GS')\theta, OD, IA, ANS\theta\rangle | \exists R \in \mathcal{P}$ and ∃most general unifier (mgu) θ so that $head(R)\theta = L\theta$.
  - **Case 3.2:** If L is a ground negative ordinary literal or a ground abducible then:
    - **Case 3.2.1:** If $L \in IA$ then $NewPS = PS' \bigcup \{\langle GS', OD, IA, ANS\rangle\}$.
    - **Case 3.2.2:** If $\overline{L} \in IA$ then $NewPS = PS'$.
    - **Case 3.2.3:** If $L \notin IA$ then $NewPS = PS' \bigcup \{\langle NewGS, OD, IA \bigcup \{L\}, ANS\rangle\}$ where $NewGS = \{fail(BS) | BS \in resolvent(L, \mathcal{P} \bigcup \mathcal{I})\} \bigcup GS'$ and $resolvent(L, T)$ is defined as follows:
      - If L is a groung negative ordinary literal, $resolvent(L, T) = \{\{L_1\theta, \ldots, L_k\theta\} | H \leftarrow L_1, \ldots, L_k \in T$ so that $\overline{L} = H\theta$ by a ground substitution $\theta\}$
      - If L is a ground abducible, $resolvent(L, T) = \{\{L_1\theta, \ldots, L_{i-1}\theta, L_{i+1}\theta, \ldots, L_k\theta\} | \bot \leftarrow L_1, \ldots, L_k \in T$ so that $L = L_i\theta$ by a ground substitution $\theta\}$.
  - **Case 3.3:** If L is $fail(BS)$, then
    - If $BS = \emptyset$, $NewPS = PS'$;
    - IF $BS \neq \emptyset$, then do the following:
      (1) Select B from BS and let $BS' = BS - \{B\}$.
      (2) **Case 3.3.1:** If B is a positive ordinary literal, $NewPS = PS' \bigcup \{\langle NewGS \bigcup GS', OD, IA, ANS\rangle\}$ where $NewGS = \{fail((\{body(R)\} \bigcup BS')\theta | \exists R \in \mathcal{P}$ and ∃mgu θ so that $head(R)\theta = B\theta\}$
      Case 3.3.2: If B is a ground negative ordinary literal or a ground askable literal or an abducible, $NewPS = PS' \bigcup \{\langle \{fail(BS')\} \bigcup GS', OD, IA, ANS\rangle\} \bigcup \{\langle \{\overline{B}\} \bigcup GS', OD, IA, ANS\rangle\}$.

– **Case 3.4:** If L is a ground askable literal, $Q@S$, then do the following:

(1) If $L \notin AAQ$ and $\overline{L} \notin AAQ$, then send the question Q to the slave agent S and $NewAAQ = AAQ \bigcup \{L\}$.

(2) If $\overline{L} \in OD$ then $NewPS = PS'$ else $NewPS = PS' \bigcup \{\langle GS', OD \bigcup \{L\}, IA, ANS\rangle\}$.

5.4 Fact Arrival Phase

During the process reduction phase it is asked information to the information sources. Whenever this information is returned from the source the current belief state is revised according to it. Supposing that an answer Q is returned from a sensor S. Let $L = Q@S$. After finishing a step of the process reduction phase, do the following:

- If $\overline{L} \in CBS$, then $NewCBS = CBS - \{\overline{L}\} \bigcup \{L\}$
- Else if $L \notin CBS$, then $NewCBS = CBS \bigcup \{L\}$.

Some askable literals might not be included in the initial belief set. Thus, if there is a process that is using such askable literal or their complements, they are suspended until the answer arrives.

5.5 Execution Example

An execution example of the program introduced in Section 5.1 is presented bellow. For the process reduction, the following strategy is used:

▷ When a positive literal is reduced, new processes are created along according to the rule order in the program, which are unifiable with the positive literal;

▷ A newly created or newly resumed process and the most left literal is always selected.

In the next execution trace for $guide(1,2)$ the selected literal is underlined in the selected active process. AAQ and CBS are only shown when a change occurs.
During the next execution trace the user travels between locations 1 and 2 through intermediate location 3 and takes the wrong direction towards location 4. At this stage the system alerts the user and guides him to the correct path.

①

$$\begin{aligned} PS &= \{\langle \{\underline{guide(1,2)}\},\ \emptyset,\ \emptyset\rangle\} \\ AAQ &= \emptyset \\ CBS &= \{user\_travel(1,3)@gps\_sensor, \sim user\_travel(1,2)@gps\_sensor, \\ &\quad user\_travel(3,4)@gps\_sensor, user\_travel(4,3)@gps\_sensor, \\ &\quad user\_travel(3,2)@gps\_sensor, included(1)@recognizer, \\ &\quad included(2)@recognizer, included(3)@recognizer, \\ &\quad \sim included(4)@recognizer \end{aligned}$$

② By case 3.1

$$\begin{aligned} PS = \{&\langle \{\underline{path(1,F)},\ show\_next\_point(F),\ user\_travel(1,F)@gps\_sensor,\ guide(F,2)\},\ \emptyset,\ \emptyset\rangle, \\ &\langle \{path(1,F),\ user\_travel(1,F)@gps\_sensor,\ show\_user\_warning(F),\ guide(F,2)\},\ \emptyset,\ \emptyset\rangle\} \end{aligned}$$

③ By case 3.1

$PS = \{\langle\{\underline{show\_next\_point(2)},\ user\_travel(1,2)@gps\_sensor,\ guide(2,2)\},\ \emptyset,\ \emptyset\rangle,$
$\langle\{show\_next\_point(3),\ user\_travel(1,3)@gps\_sensor,\ guide(3,2)\},\ \emptyset,\ \emptyset\rangle,\ P_1$ [1] $\}$

At step 3 the system starts to check if it is possible to use the path between locations 1 and 2 to guide the user.

④ By case 3.2.3

$PS = \{\langle\{\underline{fail(\sim included(2)@recognizer)},\ user\_travel(1,2)@gps\_sensor,\ guide(2,2)\},\ \emptyset,$
$\{show\_next\_point(2)\}\rangle,$
P_2 [2] $,\ P_1\}$

⑤ By case 3.3.2

$PS = \{\langle\{\underline{fail(\emptyset)},\ user\_travel(1,2)@gps\_sensor,\ guide(2,2)\},\ \emptyset,\ \{show\_next\_point(2)\}\rangle,$
$\langle\{included(2)@recognizer,\ user\_travel(1,2)@gps\_sensor,\ guide(2,2)\},\ \emptyset,$
$\{show\_next\_point(2)\}\rangle,$
$P_2,\ P_1\}$

⑥ By case 3.3

$PS = \{\langle\{\underline{included(2)@recognizer},\ user\_travel(1,2)@gps\_sensor,\ guide(2,2)\},\ \emptyset,$
$\{show\_next\_point(2)\}\rangle,$
$P_2,\ P_1\}$

⑦ By case 3.4

$included(2)$ is asked to the $recognizer$

$PS = \{\langle\{\underline{user\_travel(1,2)@gps\_sensor},\ guide(2,2)\},\ \{included(2)@recognizer\},$
$\{show\_next\_point(2)\}\rangle,$
$P_2,\ P_1\}$
$AAQ = \{included(2)@recognizer\}$

⑧ By case 3.4

$user\_travel(1,2)$ is asked to the gps_sensor

$PS = \{\langle\{guide(2,2)\},\ \{included(2)@recognizer,\ user\_travel(1,2)@gps\_sensor\},$
$\{show\_next\_point(2)\}\rangle,$
$P_2,\ P_1\}$
$AAQ = \{included(2)@recognizer,\ user\_travel(1,2)@gps\_sensor\}$

At this stage, since the system does not have an answer for the question $user\_travel(1,2)@gps\_sensor$, it uses the default $false$ value and suspends the computation of this branch. The system begins the same process for a new branch that uses a path between locations 1 and 3.

⑨

$PS = \{\langle\{\underline{show\_next\_point(3)},\ user\_travel(1,3)@gps\_sensor,\ guide(3,2)\},\ \emptyset,\ \emptyset\rangle$
$P_1,\ P_3$ [3] $\}$

⑩ By case 3.2.3

$PS = \{\langle\{\underline{fail(\sim included(3)@recognizer)},\ user\_travel(1,3)@gps\_sensor,\ guide(3,2)\},\ \emptyset,$
$\{show\_next\_point(3)\}\rangle$
$P_1,\ P_3\}$

---

[1] $P_1 = \langle\{path(1,F),\ user\_travel(1,F)@gps\_sensor,\ show\_user\_warning(F),\ guide(F,2)\},\ \emptyset,\ \emptyset\rangle$
[2] $P_2 = \langle\{show\_next\_point(3),\ user\_travel(1,3)@gps\_sensor,\ guide(3,2)\},\ \emptyset,\ \emptyset\rangle$
[3] $P_3 = \langle\{guide(2,2)\},\ \{included(2)@recognizer,\ user\_travel(1,2)@gps\_sensor\},\ \{show\_next\_point(2)\}\rangle$

⑪ By case 3.3.2

$$PS = \{\langle\{\underline{fail(\emptyset)},\ user\_travel(1,3)@gps\_sensor,\ guide(3,2)\},\ \emptyset,\ \{show\_next\_point(3)\}\rangle,$$
$$\langle\{included(3)@recognizer,\ user\_travel(1,3)@gps\_sensor,\ guide(3,2)\},\ \emptyset,$$
$$\{show\_next\_point(3)\}\rangle,$$
$$P_1,\ P_3\}$$

⑫ By case 3.3

$$PS = \{\langle\{\underline{included(3)@recognizer},\ user\_travel(1,3)@gps\_sensor,\ guide(3,2)\},\ \emptyset,$$
$$\{show\_next\_point(3)\}\rangle,$$
$$P_1,\ P_3\}$$

⑬ By case 3.4

$included(3)$ is asked to the $recognizer$

$$PS = \{\langle\{\underline{user\_travel(1,3)@gps\_sensor},\ guide(3,2)\},\ \{included(3)@recognizer\},$$
$$\{show\_next\_point(3)\}\rangle,$$
$$P_1,\ P_3\}$$
$$AAQ = \{included(2)@recognizer,\ user\_travel(1,2)@gps\_sensor,\ included(3)@recognizer\}$$

⑭ By case 3.4

$user\_travel(1,3)$ is asked to the gps_sensor

$$PS = \{\langle\{guide(3,2)\},\ \{included(3)@recognizer,\ user\_travel(1,3)@gps\_sensor\},$$
$$\{show\_next\_point(3)\}\rangle,$$
$$P_1,\ P_3\}$$
$$AAQ = \{included(2)@recognizer,\ user\_travel(1,2)@gps\_sensor,\ included(3)@recognizer,$$
$$user\_travel(1,3)@gps\_sensor\}$$

The execution of the program assumed, at this stage, that the user is travelling from 1 to 3. Then, it suspends this branch and resumes the branche that is used to alert the user whenever needed.

⑮

$$PS = \{\langle\{path(1,F),\ user\_travel(1,F)@gps\_sensor,\ show\_user\_warning(F),\ guide(F,2)\},\ \emptyset,\ \emptyset\rangle,$$
$$P_3,\ P_4\ [4]\ \}$$

⑯ By case 3.1

$$PS = \{\langle\{\underline{user\_travel(1,2)@gps\_sensor},\ show\_user\_warning(2),\ guide(2,2)\},\ \emptyset,\ \emptyset\rangle,$$
$$\langle\{user\_travel(1,3)@gps\_sensor,\ show\_user\_warning(3),\ guide(3,2)\},\ \emptyset,\ \emptyset\rangle$$
$$P_3,\ P_4\}$$

⑰ By case 3.4

$$PS = \{\langle\{\underline{show\_user\_warning(2)},\ guide(2,2)\},\ \{user\_travel(1,2)@gps\_sensor\},\ \emptyset\rangle,$$
$$P_5\ [5],\ P_3,\ P_4\}$$

⑱ By case 3.2.3

$$PS = \{\langle\{\underline{fail(included(2)@recognizer)},\ guide(2,2)\},\ \{user\_travel(1,2)@gps\_sensor\},$$
$$\{show\_user\_warning(2)\}\rangle,$$
$$P_5,\ P_3,\ P_4\}$$

⑲ By case 3.3.2

$$PS = \{\langle\{\underline{fail(\emptyset)},\ guide(2,2)\},\ \{user\_travel(1,2)@gps\_sensor\},\ \{show\_user\_warning(2)\}\rangle,$$
$$\langle\{\sim included(2)@recognizer,\ guide(2,2)\},\ \{user\_travel(1,2)@gps\_sensor\},$$
$$\{show\_user\_warning(2)\}\rangle$$
$$P_5,\ P_3,\ P_4\}$$

[4] $P_4 = \langle\{guide(3,2)\},\ \{included(3)@recognizer,\ user\_travel(1,3)@gps\_sensor\},\ \{show\_next\_point(3)\}\rangle$

[5] $P_5 = \langle\{user\_travel(1,3)@gps\_sensor,\ show\_user\_warning(3),\ guide(3,2)\},\ \emptyset,\ \emptyset\rangle$

20 By case 3.3

$PS = \{\langle\{\underline{\sim included(2)@recognizer},\ guide(2,2)\},\ \{user\_travel(1,2)@gps\_sensor\},$
$\{show\_user\_warning(2)\}\rangle$
$P_5,\ P_3,\ P_4\}$

21 By case 3.4

$PS = \{\langle\{guide(2,2)\},\ \{user\_travel(1,2)@gps\_sensor,\ \sim included(2)@recognizer\},$
$\{show\_user\_warning(2)\}\rangle$
$\langle\{\underline{user\_travel(1,3)@gps\_sensor},\ show\_user\_warning(3),\ guide(3,2)\},\ \emptyset,\ \emptyset\rangle$
$P_3,\ P_4\}$

22 By case 3.4

$PS = \{\langle\{\underline{show\_user\_warning(3)},\ guide(3,2)\},\ \{user\_travel(1,3)@gps\_sensor\},\ \emptyset\rangle$
$P_3,\ P_4,\ P_6$ [6] $\}$

23 By case 3.2.3

$PS = \{\langle\{\underline{fail(included(3)@recognizer)},\ guide(3,2)\},\ \{user\_travel(1,3)@gps\_sensor\},$
$\{show\_user\_warning(3)\}\rangle$
$P_3,\ P_4,\ P_6\}$

24 By case 3.3.2

$PS = \{\langle\{\underline{fail(\emptyset)},\ guide(3,2)\},\ \{user\_travel(1,3)@gps\_sensor\},\ \{show\_user\_warning(3)\}\rangle$
$\langle\{\sim included(3)@recognizer,\ guide(3,2)\},\ \{user\_travel(1,3)@gps\_sensor\},$
$\{show\_user\_warning(3)\}\rangle$
$P_3,\ P_4,\ P_6\}$

25 By case 3.3

$PS = \{\langle\{\underline{\sim included(3)@recognizer},\ guide(3,2)\},\ \{user\_travel(1,3)@gps\_sensor\},$
$\{show\_user\_warning(3)\}\rangle$
$P_3,\ P_4,\ P_6\}$

26 By case 3.4

$PS = \{\langle\{guide(3,2)\},\ \{user\_travel(1,3)@gps\_sensor,\ \sim included(3)@recognizer\},$
$\{show\_user\_warning(3)\}\rangle$
$P_3,\ P_4,\ P_6\}$

Since locations 2 and 3 are valid, the two branches that are used to alert the user are suspended. The integrity contraints show that the system does not alert the user when he is travelling in the correct path and this is shown in these previous steps.

27
$user\_travel(1,3)$ is returned from gps_sensor
nothing changes

From this point on the execution will do the same as in step 1: guide the user from the intermediate location 3 to the final location 2. It starts by finding a possible path to the destination and tries to show if the location is valid or alerts the user if not. For each possible location two new branches are derived from the current one.

28

$PS = \{\langle\{\underline{guide(3,2)}\},\ \{included(3)@recognizer,\ user\_travel(1,3)@gps\_sensor\},\ \{show\_next\_point(3)\}\rangle$
$P_3,\ P_6,\ P_7$ [7] $\}$

[6] $P_6 = \langle\{guide(2,2)\},\ \{user\_travel(1,2)@gps\_sensor,\ \sim included(2)@recognizer\},\ \{show\_user\_warning(2)\}\rangle$
[7] $P_7 = \langle\{guide(3,2)\},\ \{user\_travel(1,3)@gps\_sensor,\ \sim included(3)@recognizer\},\ \{show\_user\_warning(3)\}\rangle$

29 By case 3.1

$PS = \{\langle\{\underline{path(3,F)},\ show\_next\_point(F),\ user\_travel(3,F)@gps\_sensor,\ guide(F,2)\},$
$\{included(3)@recognizer,\ user\_travel(1,3)@gps\_sensor\},\ \{show\_next\_point(3)\}\rangle$
$\langle\{path(3,F),\ user\_travel(3,F)@gps\_sensor,\ show\_user\_warning(F),\ guide(F,2)\},$
$\{included(3)@recognizer,\ user\_travel(1,3)@gps\_sensor\},\ \{show\_next\_point(3)\}\rangle$
$P_3,\ P_6,\ P_7\}$

30 By case 3.1

$PS = \{\langle\{\underline{show\_next\_point(2)},\ user\_travel(3,2)@gps\_sensor,\ guide(2,2)\},$
$\{included(3)@recognizer,\ user\_travel(1,3)@gps\_sensor\},\ \{show\_next\_point(3)\}\rangle$
$\langle\{show\_next\_point(4),\ user\_travel(3,4)@gps\_sensor,\ guide(4,2)\},$
$\{included(3)@recognizer,\ user\_travel(1,3)@gps\_sensor\},\ \{show\_next\_point(3)\}\rangle$
P_8[8], $P_3,\ P_6,\ P_7\}$

31 By case 3.2.3

$PS = \{\langle\{\underline{fail(\sim included(2)@recognizer)},\ user\_travel(3,2)@gps\_sensor,\ guide(2,2)\},$
$\{included(3)@recognizer,\ user\_travel(1,3)@gps\_sensor\},\ \{show\_next\_point(3),$
$show\_next\_point(2)\}\rangle$
P_9[9], $P_8,\ P_3,\ P_6,\ P_7\}$

32 By case 3.3.2

$PS = \{\langle\{\underline{fail(\emptyset)},\ user\_travel(3,2)@gps\_sensor,\ guide(2,2)\},\ \{included(3)@recognizer,$
$user\_travel(1,3)@gps\_sensor\},\ \{show\_next\_point(3),\ show\_next\_point(2)\}\rangle$
$\langle\{included(2)@recognizer,\ user\_travel(3,2)@gps\_sensor,\ guide(2,2)\},$
$\{included(3)@recognizer,\ user\_travel(1,3)@gps\_sensor\},\ \{show\_next\_point(3),$
$show\_next\_point(2)\}\rangle$
$P_9,\ P_8,\ P_3,\ P_6,\ P_7\}$

33 By case 3.3

$PS = \{\langle\{\underline{included(2)@recognizer},\ user\_travel(3,2)@gps\_sensor,\ guide(2,2)\},$
$\{included(3)@recognizer,\ user\_travel(1,3)@gps\_sensor\},\ \{show\_next\_point(3),$
$show\_next\_point(2)\}\rangle$
$P_9,\ P_8,\ P_3,\ P_6,\ P_7\}$

34 By case 3.4

$PS = \{\langle\{\underline{user\_travel(3,2)@gps\_sensor},\ guide(2,2)\},\ \{included(3)@recognizer,$
$user\_travel(1,3)@gps\_sensor,\ included(2)@recognizer\},\ \{show\_next\_point(3),$
$show\_next\_point(2)\}\rangle$
$P_9,\ P_8,\ P_3,\ P_6,\ P_7\}$

35 By case 3.4

$user\_travel(3,2)$ is asked to the gps_sensor

$PS = \{\langle\{\underline{guide(2,2)}\},\ \{included(3)@recognizer,\ user\_travel(1,3)@gps\_sensor,$
$included(2)@recognizer,\ user\_travel(3,2)@gps\_sensor\},\ \{show\_next\_point(3),$
$show\_next\_point(2)\}\rangle$
$P_9,\ P_8,\ P_3,\ P_6,\ P_7\}$

$AAQ = \{included(2)@recognizer,\ user\_travel(1,2)@gps\_sensor,\ included(3)@recognizer,$
$user\_travel(1,3)@gps\_sensor,\ user\_travel(3,2)@gps\_sensor\}$

---

[8] $P_8 = \langle\{path(3,F),\ user\_travel(3,F)@gps\_sensor,\ show\_user\_warning(F),\ guide(F,2)\},$
$\{included(3)@recognizer,\ user\_travel(1,3)@gps\_sensor\},\ \{show\_next\_point(3)\}\rangle$

[9] $P_9 = \langle\{show\_next\_point(4),\ user\_travel(3,4)@gps\_sensor,\ guide(4,2)\},\ \{included(3)@recognizer,$
$user\_travel(1,3)@gps\_sensor\},\ \{show\_next\_point(3)\}\rangle$

㊱ By case 3.1

$PS = \{\langle\emptyset,\ \{included(3)@recognizer,\ user\_travel(1,3)@gps\_sensor,\ included(2)@recognizer,\ user\_travel(3,2)@gps\_sensor\},\ \{show\_next\_point(3),\ show\_next\_point(2)\}\rangle$
$\langle\{\underline{show\_next\_point(4)},\ user\_travel(3,4)@gps\_sensor,\ guide(4,2)\},\ \{included(3)@recognizer,\ user\_travel(1,3)@gps\_sensor\},\ \{show\_next\_point(3)\}\rangle$
$P_8,\ P_3,\ P_6,\ P_7\}$

After step 36 the execution trace finds the total path to travel, however there are other branches that may be computed. Thus, the system verifies if the unfinished branch is viable to be computed. In this case it computes over the path that connects location 3 and location 4.

㊲ By case 3.2.3

$PS = \{\langle\{\underline{fail(\sim included(4)@recognizer)},\ user\_travel(3,4)@gps\_sensor,\ guide(4,2)\},\ \{included(3)@recognizer,\ user\_travel(1,3)@gps\_sensor\},\ \{show\_next\_point(3),\ show\_next\_point(4)\}\rangle$
$P_8,\ P_3,\ P_6,\ P_7,\ P_{10}$[10]$\}$

㊳ By case 3.3.2

$PS = \{\langle\{\underline{fail(\emptyset)},\ user\_travel(3,4)@gps\_sensor,\ guide(4,2)\},\ \{included(3)@recognizer,\ user\_travel(1,3)@gps\_sensor\},\ \{show\_next\_point(3),\ show\_next\_point(4)\}\rangle$
$\langle\{included(4)@recognizer,\ user\_travel(3,4)@gps\_sensor,\ guide(4,2)\},\ \{included(3)@recognizer,\ user\_travel(1,3)@gps\_sensor\},\ \{show\_next\_point(3),\ show\_next\_point(4)\}\rangle$
$P_8,\ P_3,\ P_6,\ P_7,\ P_{10}\}$

㊴ By case 3.3

$PS = \{\langle\{\underline{included(4)@recognizer},\ user\_travel(3,4)@gps\_sensor,\ guide(4,2)\},\ \{included(3)@recognizer,\ user\_travel(1,3)@gps\_sensor\},\ \{show\_next\_point(3),\ show\_next\_point(4)\}\rangle$
$P_8,\ P_3,\ P_6,\ P_7,\ P_{10}\}$

The current computed branch is suspended since the default value for $included(4)$ is $false$, *i.e.*, location 4 is not a valid location and the program must not show this location as a location to travel to. The current branch is suspended and the branch suspended on step 29 is resumed.

㊵ By case 3.4

$included(4)$ is asked to the $recognizer$

$PS = \{\langle\{user\_travel(3,4)@gps\_sensor,\ guide(4,2)\},\ \{included(3)@recognizer,\ user\_travel(1,3)@gps\_sensor,\ included(4)@recognizer\},\ \{show\_next\_point(3),\ show\_next\_point(4)\}\rangle$
$\langle\{\underline{path(3,F)},\ user\_travel(3,F)@gps\_sensor,\ show\_user\_warning(F),\ guide(F,2)\},\ \{included(3)@recognizer,\ user\_travel(1,3)@gps\_sensor\},\ \{show\_next\_point(3)\}\rangle$
$P_3,\ P_6,\ P_7,\ P_{10}\}$

$AAQ = \{included(2)@recognizer,\ user\_travel(1,2)@gps\_sensor,\ included(3)@recognizer,\ user\_travel(1,3)@gps\_sensor,\ user\_travel(3,2)@gps\_sensor,\ included(4)@recognizer\}$

㊶ By case 3.1

$PS = \{\langle\{\underline{user\_travel(3,4)@gps\_sensor},\ show\_user\_warning(4),\ guide(4,2)\},\ \{included(3)@recognizer,\ user\_travel(1,3)@gps\_sensor\},\ \{show\_next\_point(3)\}\rangle$
$\langle\{user\_travel(3,2)@gps\_sensor,\ show\_user\_warning(2),\ guide(3,2)\},\ \{included(3)@recognizer,\ user\_travel(1,3)@gps\_sensor\},\ \{show\_next\_point(3)\}\rangle$
$P_3,\ P_6,\ P_7,\ P_{10},\ P_{11}$[11]$\}$

[10] $P_{10} = \langle\emptyset,\ \{included(3)@recognizer,\ user\_travel(1,3)@gps\_sensor,\ included(2)@recognizer,\ user\_travel(3,2)@gps\_sensor\},\ \{show\_next\_point(3),\ show\_next\_point(2)\}\rangle$

$user\_travel(3,4)$ is asked to the gps_sensor

$PS = \{\langle\{\underline{show\_user\_warning(4)},\ guide(4,2)\},\ \{included(3)@recognizer,$
$user\_travel(1,3)@gps\_sensor,\ user\_travel(3,4)@gps\_sensor\},\ \{show\_next\_point(3)\}\rangle$
P_{12}[12], $P_3,\ P_6,\ P_7,\ P_{10},\ P_{11}\}$

43 By case 3.2.3

$PS = \{\langle\{\underline{fail(included(4)@recognizer)},\ guide(4,2)\},\ \{included(3)@recognizer,$
$user\_travel(1,3)@gps\_sensor,\ user\_travel(3,4)@gps\_sensor\},\ \{show\_next\_point(3),$
$show\_user\_warning(4)\}\rangle$
$P_{12},\ P_3,\ P_6,\ P_7,\ P_{10},\ P_{11}\}$

44 By case 3.3.2

$PS = \{\langle\{\underline{fail(\emptyset)},\ guide(4,2)\},\ \{included(3)@recognizer,\ user\_travel(1,3)@gps\_sensor,$
$user\_travel(3,4)@gps\_sensor\},\ \{show\_next\_point(3),\ show\_user\_warning(4)\}\rangle$
$\langle\{\sim included(4)@recognizer,\ guide(4,2)\},\ \{included(3)@recognizer,$
$user\_travel(1,3)@gps\_sensor,\ user\_travel(3,4)@gps\_sensor\},\ \{show\_next\_point(3),$
$show\_user\_warning(4)\}\rangle$
$P_{12},\ P_3,\ P_6,\ P_7,\ P_{10},\ P_{11}\}$

45 By case 3.3

$PS = \{\langle\{\underline{\sim included(4)@recognizer},\ guide(4,2)\},\ \{included(3)@recognizer,$
$user\_travel(1,3)@gps\_sensor,\ user\_travel(3,4)@gps\_sensor\},\ \{show\_next\_point(3),$
$show\_user\_warning(4)\}\rangle$
$P_{12},\ P_3,\ P_6,\ P_7,\ P_{10},\ P_{11}\}$

Using default values the system detects that the user may travel towards location 4. This location is not a valid one so the system alerts the user and guides him to the correct path.

46 By case 3.4

$\sim included(4)$ is asked to the $recognizer$

$PS = \{\langle\{guide(4,2)\},\ \{included(3)@recognizer,\ user\_travel(1,3)@gps\_sensor,$
$user\_travel(3,4)@gps\_sensor,\ \sim included(4)@recognizer\},\ \{show\_next\_point(3),$
$show\_user\_warning(4)\}\rangle$
$\langle\{\underline{user\_travel(3,2)@gps\_sensor},\ show\_user\_warning(2),\ guide(3,2)\},$
$\{included(3)@recognizer,\ user\_travel(1,3)@gps\_sensor\},\ \{show\_next\_point(3)\}\rangle$
$P_3,\ P_6,\ P_7,\ P_{10},\ P_{11}\}$

Before deriving new branches from $guide(4,2)$ the system resumes another branch that verifies if an alert show be sent to the user if he travels towards location 2.

47 By case 3.4

$PS = \{\langle\{\underline{show\_user\_warning(2)},\ guide(3,2)\},\ \{included(3)@recognizer,$
$user\_travel(1,3)@gps\_sensor,\ user\_travel(3,4)@gps\_sensor\},\ \{show\_next\_point(3)\}\rangle$
P_{13}[13], $P_3,\ P_6,\ P_7,\ P_{10},\ P_{11}\}$

---

[11] $P_{11} = \langle\{user\_travel(3,4)@gps\_sensor,\ guide(4,2)\},\ \{included(3)@recognizer,$
$user\_travel(1,3)@gps\_sensor,\ included(4)@recognizer\},\ \{show\_next\_point(3),\ show\_next\_point(4)\}\rangle$

[12] $P_{12} = \langle\{user\_travel(3,2)@gps\_sensor,\ show\_user\_warning(2),\ guide(3,2)\},\ \{included(3)@recognizer,$
$user\_travel(1,3)@gps\_sensor\},\ \{show\_next\_point(3)\}\rangle$

[13] $P_{13} = \langle\{guide(4,2)\},\ \{included(3)@recognizer,\ user\_travel(1,3)@gps\_sensor,$
$user\_travel(3,4)@gps\_sensor,\ \sim included(4)@recognizer\},\ \{show\_next\_point(3),\ show\_user\_warning(4)\}\rangle$

48 By case 3.2.3

$PS = \{\langle\{\underline{fail(included(2)@recognizer)},\ guide(3,2)\},\ \{included(3)@recognizer,\ user\_travel(1,3)@gps\_sensor,\ user\_travel(3,4)@gps\_sensor\},\ \{show\_next\_point(3),\ show\_user\_warning(2)\}\rangle$
$P_{13},\ P_3,\ P_6,\ P_7,\ P_{10},\ P_{11}\}$

49 By case 3.3.2

$PS = \{\langle\{\underline{fail(\emptyset)},\ guide(3,2)\},\ \{included(3)@recognizer,\ user\_travel(1,3)@gps\_sensor,\ user\_travel(3,4)@gps\_sensor\},\ \{show\_next\_point(3),\ show\_user\_warning(2)\}\rangle$
$\langle\{\sim included(2)@recognizer,\ guide(3,2)\},\ \{included(3)@recognizer,\ user\_travel(1,3)@gps\_sensor,\ user\_travel(3,4)@gps\_sensor\},\ \{show\_next\_point(3),\ show\_user\_warning(2)\}\rangle$
$P_{13},\ P_3,\ P_6,\ P_7,\ P_{10},\ P_{11}\}$

50 By case 3.3

$PS = \{\langle\{\underline{\sim included(2)@recognizer},\ guide(3,2)\},\ \{included(3)@recognizer,\ user\_travel(1,3)@gps\_sensor,\ user\_travel(3,4)@gps\_sensor\},\ \{show\_next\_point(3),\ show\_user\_warning(2)\}\rangle$
$P_{13},\ P_3,\ P_6,\ P_7,\ P_{10},\ P_{11}\}$

Since location 2 is a valid location the current branch is suspended and the one resumed in step 46 is resumed, deriving two more branches.

51 By case 3.4

$PS = \{\langle\{guide(3,2)\},\ \{included(3)@recognizer,\ user\_travel(1,3)@gps\_sensor,\ user\_travel(3,4)@gps\_sensor,\ \sim included(2)@recognizer\},\ \{show\_next\_point(3),\ show\_user\_warning(2)\}\rangle$
$\langle\{\underline{guide(4,2)}\},\ \{included(3)@recognizer,\ user\_travel(1,3)@gps\_sensor,\ user\_travel(3,4)@gps\_sensor,\ \sim included(4)@recognizer\},\ \{show\_next\_point(3),\ show\_user\_warning(4)\}\rangle$
$P_3,\ P_6,\ P_7,\ P_{10},\ P_{11}\}$

52

$\sim user\_travel(3,4)$ is returned from $recognizer$

$CBS = \{user\_travel(1,3)@gps\_sensor, \sim user\_travel(1,2)@gps\_sensor,\ \sim user\_travel(3,4)@gps\_sensor, user\_travel(4,3)@gps\_sensor,\ user\_travel(3,2)@gps\_sensor, included(1)@recognizer,\ included(2)@recognizer, included(3)@recognizer,\ \sim included(4)@recognizer$

53 By case 3.1

$PS = \{\langle\{\underline{show\_next\_point(3)},\ user\_travel(4,3)@gps\_sensor,\ guide(3,2)\},\ \{included(3)@recognizer,\ user\_travel(1,3)@gps\_sensor,\ user\_travel(3,4)@gps\_sensor,\ \sim included(4)@recognizer\},\ \{show\_next\_point(3),\ show\_user\_warning(4)\}\rangle$
$P_3,\ P_6,\ P_7,\ P_{10},\ P_{11},\ P_{14}$[14]$\}$

54 By case 3.2.3

$PS = \{\langle\{\underline{fail(\sim included(3)@recognizer)},\ user\_travel(4,3)@gps\_sensor,\ guide(3,2)\},\ \{included(3)@recognizer,\ user\_travel(1,3)@gps\_sensor,\ user\_travel(3,4)@gps\_sensor,\ \sim included(4)@recognizer\},\ \{show\_next\_point(3),\ show\_user\_warning(4)\}\rangle$
$P_3,\ P_6,\ P_7,\ P_{10},\ P_{11},\ P_{14}\}$

[14] $P_{14} = \langle\{guide(3,2)\},\ \{included(3)@recognizer,\ user\_travel(1,3)@gps\_sensor,\ user\_travel(3,4)@gps\_sensor,\ \sim included(2)@recognizer\},\ \{show\_next\_point(3),\ show\_user\_warning(2)\}\rangle$

55 By case 3.3.2

$PS = \{\langle\{\underline{fail(\emptyset)},\ user\_travel(4,3)@gps\_sensor,\ guide(3,2)\},\ \{included(3)@recognizer,\ user\_travel(1,3)@gps\_sensor,\ user\_travel(3,4)@gps\_sensor,\ \sim included(4)@recognizer\},\ \{show\_next\_point(3),\ show\_user\_warning(4)\}\rangle$
$\langle\{included(3)@recognizer,\ user\_travel(4,3)@gps\_sensor,\ guide(3,2)\},\ \{included(3)@recognizer,\ user\_travel(1,3)@gps\_sensor,\ user\_travel(3,4)@gps\_sensor,\ \sim included(4)@recognizer\},\ \{show\_next\_point(3),\ show\_user\_warning(4)\}\rangle$
$P_3,\ P_6,\ P_7,\ P_{10},\ P_{11},\ P_{14}\}$

56 By case 3.3

$PS = \{\langle\{\underline{included(3)@recognizer},\ user\_travel(4,3)@gps\_sensor,\ guide(3,2)\},\ \{included(3)@recognizer,\ user\_travel(1,3)@gps\_sensor,\ user\_travel(3,4)@gps\_sensor,\ \sim included(4)@recognizer\},\ \{show\_next\_point(3),\ show\_user\_warning(4)\}\rangle$
$P_3,\ P_6,\ P_7,\ P_{10},\ P_{11},\ P_{14}\}$

57 By case 3.4

$PS = \{\langle\{\underline{user\_travel(4,3)@gps\_sensor},\ guide(3,2)\},\ \{included(3)@recognizer,\ user\_travel(1,3)@gps\_sensor,\ user\_travel(3,4)@gps\_sensor,\ \sim included(4)@recognizer\},\ \{show\_next\_point(3),\ show\_user\_warning(4)\}\rangle$
$P_3,\ P_6,\ P_7,\ P_{10},\ P_{11},\ P_{14}\}$

58 By case 3.4

$user\_travel(4,3)$ is asked to the gps_sensor

$PS = \{\langle\{\underline{guide(3,2)}\},\ \{included(3)@recognizer,\ user\_travel(1,3)@gps\_sensor,\ user\_travel(3,4)@gps\_sensor,\ \sim included(4)@recognizer\},\ \{show\_next\_point(3),\ show\_user\_warning(4)\}\rangle$
$P_3,\ P_6,\ P_7,\ P_{10},\ P_{11},\ P_{14}\}$

$AAQ = \{included(2)@recognizer,\ user\_travel(1,2)@gps\_sensor,\ included(3)@recognizer,\ user\_travel(1,3)@gps\_sensor,\ user\_travel(3,2)@gps\_sensor,\ included(4)@recognizer,\ user\_travel(4,3)@gps\_sensor\}$

At this stage the system will derive four new branches: two for location 2 and two for 4. As the system knows that the user is not going in the direction of location 4, both branches will be suspended. The only branch that will continue the computation is the one that points to location 2 as the next point to travel. These processes are not shown in detailed steps due to the extension of the proof procedure and they will be similar to the steps from step 28 onwards. For the previous execution the framework starts with the objective of guiding the user between the starting location 1 and the ending 2. At an initial step it tries to find a path between location 1 and the destination. According to Figure 4 there are two possibilities, which are locations 2 (the goal) and 3. The system starts processing 2 branches for each next possible location: one branch of the pair lets the execution continue if the next location is valid (*i.e.*, is included in the valid locations set) and the user is travelling to it; the other branch represents the case in which the location is not valid and the user is travelling towards it. In the former the system keeps guiding the user to the next point (since he is travelling in the correct path) whereas in the latter the system alerts the user, informing him about his mistake and guides him to the correct path. Any branch that tries to execute an alternative combination of the previously described situations is suspended since it is not valid (*e.g.*, alert the user when the next location is valid). During the execution of a branch the program may query the information source. While this information is not returned the program continues its execution, so branches which queried for

$user\_travel(1,2)@gps\_sensor$ are paused (since the default value is negative). The only branch that continues the computation is the one that assumes the default value $user\_travel(1,3)@gps\_sensor$. Whenever an answer is returned from the information source the computation is revised. The current branch may be paused and a previously paused one may be resumed. The system continues to start, pause, and resume branches during the entire processing until the destination point is reached.
Besides not knowing an answer, the system continues the computation using Speculative Computation. This situation occurs at different steps in which a default value is assumed.
At Step 9, $show\_next\_point(3)$ is assumed and the integrity constraints are checked. Thus, it is ensured that there is no contradiction by checking if $\sim included(3)@recognizer$ is not derived. In this step an ordinary abduction is performed. When an answer is returned and confirms the default value, nothing changes.
In situations like the previous one the Speculative Computation is in an advanced stage of the computation and it does not have to be revised.
The previous execution trace represents a computation example and it has not been implemented. Its purpose is to check if it is possible to formalize the orientation problem through Speculative Computation. According to the proof, applying Speculative Computation is expressive and adequate.

6 Conclusions

Orientation may be a serious problem for people with cognitive disabilities. Some may even be prevented by caregivers to go out their homes by themselves due to the risk of getting lost. In order to minimize this risk and increase the autonomy of people researchers have been developing orientation methods for people with cognitive disabilities, so that they may travel alone and be remotely monitored.
Our orientation system used augmented reality, surpassing limitations of different systems that use static pictures with overlayered arrows or other symbols, or of systems that resemble a compass in which the user needs to interpret and understand the direction. On the other hand, our system enables caregivers to know, in real time, the current location of the person with cognitive disabilities on a map. In order to increase the effectiveness of the system, it is possible to connect it to external services expanding its features.
To increase the system responsiveness and enable it to be ready for a possible error situation before it happens (predictive feature) it was envisioned a suite for speculative computation having in mind the public that will use our system.
Applying the framework of Speculative Computation enables the system to deal with incomplete information. Instead of pausing the computation and wait for a value to calculate the correct path to follow, the framework uses a default value and keeps the computation running. When the missing data is returned the computation is revised. If the returned value is coherent with the default one then the computation is in an advanced stage of execution. However, if the returned value is contradictory with the default then the computed branch is suspended and another may be started or resumed.

The Speculative Computation framework lets the system be in a constant execution stage. Instead of pausing the computation and be in an idle state. Thus, the system may have a predictive feature since it may be ready to alert the user before he turns in the wrong direction. Lastly, the Speculative Computation framework does not compute the set of default values, *i.e.*, it just uses the values that compose the set. The method for the detection of user behaviour patterns is independent of the framework, but Speculative Computation can be used in combination with it, providing a structured reasoning framework capable of coping with missing information. Future work includes analysing behaviour extraction methods, specific for the problem of orientation, which can be useful for the extraction of default values.
The detection of frequent trajectory patterns had been studied in (Giannotti, Nanni, Pinelli and Pedreschi, 2007; Monreale, Pinelli and Trasarti, 2009; Lee, Chen and Ip, 2009). In these studies the authors use different location acquisition methods (*e.g.*, GPS and GSM networks) to gather the user location. The detection of patterns is the result of a data mining process in which different algorithms are applied. Knowing the user preferences (frequent trajectories) our system may use this information to create a route that better adapts to the user characterists. Thus, the path used to guide the user may not be the shortest one, but the one that the user knows better.

Acknowledgment

This work is part-funded by ERDF - European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within project FCOMP-01-0124-FEDER-028980 (PTDC/EEI-SII/1386/2012). The work of João Ramos is supported by a doctoral grant by FCT - Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) SFRH/BD/89530/2012. The work of Tiago Oliveira is also supported by the FCT grant with the reference SFRH/BD/85291/-2012.

References

Alper, S. and Raharinirina, S.: 2006, Assisitive Technology for Individuals with Disabilities: A Review and Synthesis of the Literature, *Journal of Special Education Technology* **21**(2), 47–64.

Begum, S., Ahmed, M. U., Funk, P., Xiong, N. and Von Schéele, B.: 2009, A Case-based decision support system for individual stress diagnosis using fuzzy similarity matching, *Computational Intelligence* **25**, 180–195.

Bennett, C. C. and Hauser, K.: 2013, Artificial intelligence framework for simulating clinical decision-making: A Markov decision process approach, *Artificial Intelligence in Medicine* **57**(1), 9–19.

Carmien, S., Dawe, M., Fischer, G., Gorman, A., Kintsch, A. and Sullivan, J. F.: 2005, Socio-technical environments supporting people with cognitive disabilities using public transportation, *ACM Transactions on Computer-Human Interaction* **12**(2), 233–262.

Carneiro, D., Novais, P., Costa, R., Gomes, P., Neves, J., Tscheligi, M., De Ruyter, B., Markopoulus, P., Wichert, R., Mirlacher, T., Meschterjakov, A. and Reitberger, W.: 2009, EMon: Embodied Monitorization, *Proceedings of the European Conference on Ambient Intelligence* **5859**, 133–142.

Chong, E. K. P., Kreucher, C. M. and Hero, A. O.: 2009, Partially Observable Markov Decision Process Approximations for Adaptive Sensing.

Corchado, J. M., Bajo, J. and Abraham, A.: 2008, GerAmi: Improving healthcare delivery in geriatric residences, *IEEE Intelligent Systems* **23**, 19–25.

Dawe, M.: 2006, Desperately seeking simplicity: how young adults with cognitive disabilities and their families adopt assistive technologies, *Proceedings of the SIGCHI conference on Human Factors in computing systems*, CHI '06, ACM, pp. 1143–1152.

Dawe, M.: 2007, "Let Me Show You What I Want": Engaging Individuals with Cognitive Disabilities and their Families in Design, *Technology* pp. 2177–2182.

Fraunnhover Portugal: 2012, AlzNav.
URL: *http://www.fraunhofer.pt/en/fraunhofer_aicos/projects/internal_research/alznav.html*

Friedman, M. G. and Bryen, D. N.: 2007, Web accessibility design recommendations for people with cognitive disabilities, *Technology and Disability* **19**(4), 205–212.

Giannotti, F., Nanni, M., Pinelli, F. and Pedreschi, D.: 2007, Trajectory Pattern Mining, *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, ACM, pp. 330–339.

Hoey, J., Bertoldi, A. V., Poupart, P. and Mihailidis, A.: 2007, Assisting Persons with Dementia during Handwashing Using a Partially Observable Markov Decision Process, *Proceedings of the 5th International Conference on Vision Systems*.

Kaelbling, L. P., Littman, M. L. and Cassandra, A. R.: 1998, Planning and acting in partially observable stochastic domains.

Kakas, A. C., Kowalski, R. A. and Toni, F.: 1998, The Role of Abduction in Logic Programming, *Handbook of Logic in Artificial Intelligence and Logic Programming* **5**, 235–324.

Kolodner, J. L.: 1992, An introduction to case-based reasoning, *Artificial Intelligence Review* **6**(1), 3–34.

Lee, A. J. T., Chen, Y. A. and Ip, W. C.: 2009, Mining frequent trajectory patterns in spatial-temporal databases, *Information Sciences* **179**(13), 2218–2231.

Littman, M., Dean, T. and Kaelbling, L.: 1995, On the complexity of solving Markov decision problems, *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pp. 394–402.

Liu, A. L., Hile, H., Borriello, G., Kautz, H., Brown, P. A., Harniss, M. and Johnson, K.: 2009, Informing the Design of an Automated Wayfinding System for Individuals with Cognitive Impairments, *Proceedings of Pervasive Health '09*, Vol. 9, London UK, p. 8.

Lopez, R. and Plaza, E.: 1997, Case-Based Reasoning : An Overview, *AI Communications* **10**, 21–29.

Monreale, A., Pinelli, F. and Trasarti, R.: 2009, WhereNext : a Location Predictor on Trajectory Pattern Mining, *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '09*, pp. 637–645.

Ramos, J., Anacleto, R., Costa, A., Novais, P., Figueiredo, L. and Almeida, A.: 2012, Orientation System for People with Cognitive Disabilities, *in* P. Novais, K. Hallenborg, D. I. Tapia and J. M. C. Rodríguez (eds), *Ambient Intelligence - Software and Applications*, Vol. 153 of *Advances in Intelligent and Soft Computing*, Springer Berlin Heidelberg, pp. 43–50.

Ramos, J., Anacleto, R., Novais, P., Figueiredo, L., Almeida, A. and Neves, J.: 2013, Geo-localization System for People with Cognitive Disabilities, *in* J. B. Pérez, J. M. C. Rodríguez, J. Fähndrich, P. Mathieu, A. Campbell, M. C. Suarez-Figueroa, A. Ortega, E. Adam, E. Navarro, R. Hermoso and M. N. Moreno (eds), *Trends in Practical Applications of Agents and Multiagent Systems*, Vol. 221 of *Advances in Intelligent Systems and Computing*, Springer International Publishing, pp. 59–66.

Ramos, J., Costa, A., Novais, P. and Neves, J.: 2014, Interactive Guiding and Localization Platform, *International Journal of Artificial Intelligence (IJAI)* **12**(1), 63–78.

Ramos, J., Satoh, K., Novais, P. and Neves, J.: 2014, Modelling an Orientation System based on Speculative Computation, *in* S. Omatu, H. Bersini, J. M. Corchado, S. Rodríguez, P. Pawlewski and E. Bucciarelli (eds), *Distributed Computing and Artificial Intelligence, 11th International Conference*, Vol. 290 of *Advances in Intelligent Systems and Computing*, Springer International Publishing, pp. 319–326.

Sadri, F.: 2007, Multi-Agent Ambient Intelligence for Elderly Care and Assistance, *Aip Conference Proceedings*, Vol. 2007, Aip, pp. 117–120.

Satoh, K., Inoue, K., Iwanuma, K. and Sakama, C.: 2000, Speculative Computation by Abduction under Incomplete Communication Environments, *MultiAgent Systems, 2000. Proceedings. Fourth International Conference on Speculative computation by abduction under incomplete communication environments*, Vol. 12, pp. 263–270.

Scherer, M. M. J., Hart, T., Kirsch, N. and Schulthesis, M.: 2005, Assistive Technologies for Cognitive Disabilities, *Critical Reviews in Physical and Rehabilitation Medicine* **17**(3), 195–215.

Spitzer, R. L., Gibbon, M., Skodol, A. E. and First, M. B.: 1994, *DSM-IV casebook: A learning companion to the Diagnostic and Statistical Manual of Mental Disorders*, 4 edn, American Psychiatric Association.

Stefanov, D. H., Bien, Z. and Bang, W.-C.: 2004, The smart house for older persons and persons with physical disabilities: structure, technology arrangements, and perspectives., *IEEE Transactions on Neural and Rehabilitation Systems Engineering* **12**(2), 228–250.