An Alert Mechanism for Orientation Systems based on Speculative Computation

João Ramos, Tiago Oliveira, Paulo Novais, José Neves
Algoritmi Centre, Department of Informatics
University of Minho
Braga, Portugal
Email:{jramos,pjon,toliveira,jneves}@di.uminho.pt

Ken Satoh
National Institute of Informatics
Hitotsubashi, Chyoda-ku
Tokyo, Japan
Email: ksatoh@nii.ac.jp

***Abstract*—The role of assistive technologies is to help users with diminished capabilities in the fulfillment of their everyday tasks. One of such tasks is orientation. It is crucial for the autonomy of an individual and, at the same time, it is one of the most challenging tasks for an individual with cognitive disabilities. Existing solutions that tackle this problem are mostly concerned with guidance, tracking and the display of information. However, there is a dimension that has not been the object of concern in existing projects, the prediction of user actions. This work presents a Speculative Module for an orientation system that is used to alert the user for potential mistakes in his path, anticipating possible shifts in the wrong direction in critical points of the route. With this module, it becomes possible to issue warnings to the user and increase his attention so as to avoid a deviation from the correct path.**

I. Introduction

The term cognitive disability is used to represent the condition of an individual who has difficulties in one or more types of mental tasks, when compared to ordinary people, according to [1]. Causes may include traumatic brain injury, stroke, Alzheimer's and developmental disability, among others. One can distinguish four phases of cognitive disabilities: mild, moderate, severe and extreme. While individuals in the severe or extreme phases need continuous assistance with every aspect of their everyday lives, people in the mild or moderate phases are capable of leading an independent life, requiring assistance only in certain activities. One of the mental tasks that is greatly affected is orientation. It is, at the same time, something that is vital for the autonomy of a person. As such, there is a clear necessity for technologies that increase the independence of individuals with cognitive disabilities outdoors. These technologies can be materialized in orientation systems that assist the user during his travels and compensate for his diminished orientation capabilities. Using these orientation/way finding systems the user may be guided from his current location to a predefined destination. The current approaches to this include systems that are focused exclusively on guidance, display of information and communication with a caregiver or a support community [2], [3], [4]. However, it would also be useful to anticipate the actions of the user and provide alerts when he is expected to make a wrong turn in his path. Through different alerts it would be possible to indicate to the user that he is approaching a critical point in his route, in which experience shows that he normally makes a mistake.

This work proposes an orientation system endowed with such a feature. Besides guidance, an intuitive display of information, and a tracking suite for caregivers (previously disclosed in [5]), the system has a mechanism based on Speculative Computation that issues alerts to the user if he reaches a point in his path where it is likely for him to make a mistake. As such, the main contributions of this work are: i) a method for the generation of default values regarding the habits of the user when following a route; ii) the integration of these defaults in a framework for Speculative Computation with constraint processing; and iii) a module for an orientation system that analyses a route and issues alerts to the user, increasing his level of attention.

This paper is organized as follows. Section II presents related work on orientation systems for people with cognitive disabilities. Section III provides a brief description of the orientation system that hosts the Speculative Module. The Speculative Module and the framework for Speculative Computation are described and formulated in Section IV. Finally, conclusions are drawn and future work considerations are made in Section V.

II. Related Work

There are a few research projects that focus on the development of orientation systems for people with cognitive disabilities. However, they lack the kind of predictive capability that would allow them to anticipate the actions of users and employ preemptive measures to avoid them, in the event that they are undesirable.

With the recognition that smartphones could play a pivotal role in the development of orientation systems for individuals with cognitive disabilities [6], researchers started to focus on these handheld devices as the ideal vehicles for conveying guidance. An example of this is the work in [2] which provides an application that enables a user to travel between locations using a public transportation system such as a bus. This is possible through a precompiled list of instructions, created by a caregiver, and delivered through a smartphone. As the user moves around, a personal travel assistant uses the GPS module to deliver the next set of instructions. As such, this system is mostly focused on providing an appropriate support for both the user and the caregiver.

Worries with the way in which information is displayed to users were behind the work in [6]. In their system, static images with overlaid arrows, audio messages, and text messages are used to guide a user. The objective is to find a tailored way of providing directions to individuals with cognitive disabilities. To achieve this, the preferences of users are modeled in a Markov Decision Process (MDP) [7]. Another feature is the use of recognizable landmarks near the user to facilitate guidance.

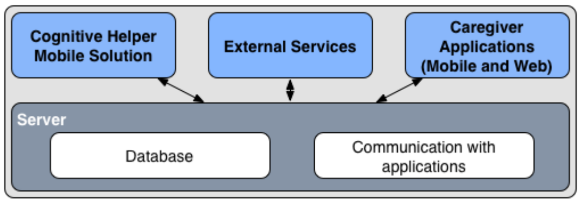
Fraunhofer Portugal also develops work in this area with their AlzNav orientation system [4]. The objective of the system is to make the interpretation of on-screen instructions as easy as possible. The direction that a user should take is shown by an arrow that works as a compass, spinning around as the user spins his phone. Along with the compass information, the user also sees on the screen the distance he should travel in a certain direction. Moreover, the system provides updates about the situation of the user to the caregiver through SMSs.

Although the existing approaches tackle some of the most important aspects of orientation for individuals with cognitive disabilities, it is still possible to find aspects in which they are lacking. Predicting the steps of users would provide an advantage. By making use of usage patterns, it becomes possible to identify critical points in the routes that are usually taken by the user. A critical point may be defined as a point where something usually goes wrong, or, in this specific case, where a user starts going in the wrong direction. Then, if the system is capable of predicting where this will happen, it can issue alerts to the user and reinforce the right path. This is the kind of feature that is proposed in the present work. The objective is to develop an orientation system that adapts to the user and tries to maximize user autonomy by minimizing the risk of getting lost.

III. System Description

The CogHelper System has the aim of guiding the user (person with cognitive disabilities) by adapting not only the level of prompting (indicating wether the user is traveling in the correct or wrong path) but also adjusting the guiding path to user preferences. This goal is complemented by a secondary objective of providing a tracking system for caregivers, which allows the caregiver to locate the user at any time.

The orientation method provided is devised for outdoor and the system architecture is shown in Figure 1. The architecture may be divided in four main modules: the server-side components (such as the database and the agency for the integration of the system applications and external services); the client-side applications (such as Cognitive Helper Mobile Solution and Caregiver Applications); and the external services, which allow the integration of CogHelper with other systems or other applications. The core of the system is the server since all the services are connected to it. The database module stores all the information necessary for the correct execution of the system (like usernames, locations and points of interest) and the remaining module provides the facilities for the communication between the server and the services (both internal and external).


Cognitive Helper Mobile Solution
External Services
Caregiver Applications (Mobile and Web)
Server
Database
Communication with applications
Fig. 1. Architecture of CogHelper System

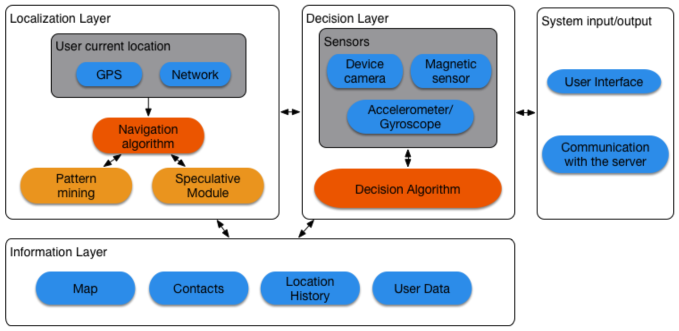


Fig. 2. Information layers of the CogHelper Module for People with Cognitive Disabilities

The component of the system that is specifically aimed at people with cognitive disabilities is the Cognitive Helper Mobile Solution (depicted in Figure 2), which consists of four layers. This component was developed for Android OS and its main functionality is to guide the user, enabling him to travel outside alone without getting lost. The Localization Layer has the methods that enable the system to get current location of the user, which may be done through the GPS of the mobile device or through the network (giving a coarse location). With this information the Navigation Algorithm is able to use the pattern mining module (responsible for getting the path that best fits the needs of the user) and the Speculative Module (which ensures that the user is traveling in the right path and alerts him before he takes a wrong turn). The Decision Algorithm (Decision Layer) uses information from the device camera, magnetic sensor and accelerometer to get the device orientation and ensure that the user is correctly oriented within the path.

The directions and alerts are given through an augmented reality interface. This interface module is included in the System Input/Output Layer, which is also includes a communication module. The latter is responsible for passing the information between the server and the user application. User destination updates, user current location updates, and so forth are some of the messages exchanged by these services. The last Layer depicted in Figure 2 (Information Layer) is used as a local database for storing information about the map, user contacts, his location history and other data associated with the user.

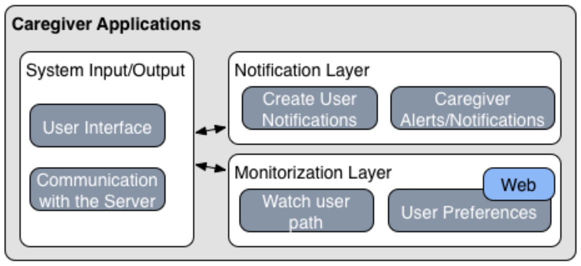


Fig. 3. Information layers of the CogHelper Module for Caregivers

Caregivers may use two different applications, a mobile application for Android OS and a Web application. Despite being developed for different platforms, these two applications have the same goal, provide a monitoring platform for the caregiver to check the current position of the user without having to be physically present. However there are some differences between the two. As depicted in Figure 3 the difference resides in the User Preferences module that is only present in the Web application. This module is responsible for adding, editing and removing user preferences such as destination points. This feature makes the Web application more complex. The Watch User Path module allows the caregiver to know the current location of the user. Inside the Notification Layer there are two modules that generate and receive alerts from and to the user application. The third Layer (System Input/Output) has the User Interface, which enables the user to interact with the application and shows the necessary information to the user. The other module ensures the communication with the server.

IV. A Speculative Module for Users with Cognitive Disabilities

The objective of the Speculative Module is to predict the next step of the user of the orientation system and manipulate that prediction in order to determine if an alert/warning should be issued. The whole procedure is controlled by Speculative Computation, acting as an interface between the instructions for the correct path (encoded as rules in the framework), the predictions about user transitions from one point to another in the path (encoded as default values), and the real information about these transitions that arrive from the other modules of the system. The Speculative Module has two main components, the framework of Speculative Computation and the method for the Generation of Default Values.

A. *Framework of Speculative Computation*

Speculative Computation as a reasoning framework was initially presented in [8] for problem solving in multi-agent systems when the communication between agents is not ensured. The Speculative Computation part of the Speculative Module is based on a logic programming framework that uses abductive reasoning [9]. As such, it includes a dynamic belief mechanism about the outside world. A Framework of Speculative Computation for an Orientation Method (SF_{OM}) is a tuple $\langle \Sigma, \mathcal{E}, \Delta, \mathcal{A}, \mathcal{P}, \mathcal{I} \rangle$. The meaning of each element is the following. Σ is a finite set of constants. An element in Σ is a module of the orientation system. $\mathcal{E}$ is a set of predicates called external predicates, representing the decision criteria. When Q is a literal with an external predicate and S is the identifier of a system module, $Q@S$ is called an askable literal. $\sim(Q@S)$ is defined 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 $\perp \leftarrow B_1, \ldots, B_n$ where $\perp$ is the symbol for 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: an askable literal $Q@S$ in a rule $R \in \mathcal{P}$ represents a question that is asked to a system module S; or an askable literal in Δ denotes a default truth 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. The literals in Δ represent the defaults about travel habits of the user in a specific route and the inclusion of locations in that route.

In the logic program given below, which provides a formalization of the example in Figure 4, 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 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 traveling 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 traveling in the correct path and assesses the need for issuing alerts when he may miss a turning

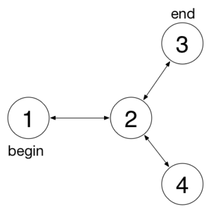


Fig. 4. Possible ways to travel between locations 1 and 3.

point. This framework is given in terms of the following logic program:

- $\Sigma = \{gps\_sensor, recognizer\}$
- $\mathcal{E} = \{user\_travel, included\}$
- $\Delta = \{user\_travel(1,2)@gps\_sensor,$
  $user\_travel(2,4)@gps\_sensor,$
  $\sim user\_travel(4,2)@gps\_sensor,$
  $user\_travel(2,3)@gps\_sensor,$
  $included(1)@recognizer, included(2)@recognizer,$
  $included(3)@recognizer, \sim included(4)@recognizer$
- $\mathcal{A} = \{show\_next\_point, show\_user\_warning\}$
- $\mathcal{P}$ is the following set of rules:
  $guide(A,A) \leftarrow .$
  $guide(A,B) \leftarrow$
  $path(A,F),$
  $show\_next\_point(F),$
  $user\_travel(A,F)@gps\_sensor,$
  $guide(F,B).$
  $guide(A,B) \leftarrow$
  $path(A,F),$
  $user\_travel(A,F)@gps\_sensor,$
  $show\_user\_warning(F),$
  $guide(F,B).$
  $path(1,2) \leftarrow .$
  $path(2,4) \leftarrow .$
  $path(2,3) \leftarrow .$
  $path(4,2) \leftarrow .$
- $\mathcal{I}$ denotes the following set of integrity constraint or invariants:
  $\perp \leftarrow$
  $show\_next\_point(F),$
  $\sim included(F)@recognizer.$
  $\perp \leftarrow$
  $show\_user\_warning(F),$
  $included(F)@recognizer.$

The two invariants included in set $\mathcal{I}$ ensure that the system does not show an alert 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 traveling point. In the setting depicted above the framework assumes as default that the user normally travels from 1 to 2, from 2 to 4, and from 2 to 3, but not from 4 to 2. The framework also assumes that 1, 2 and 3 are included in the path, but 4 is not. Figure 4 presents the setting for the decision of whether to show or not a warning. It is a graph representing the path that a user should take. At location 2, there is an alternative path that may take the user to location 4. This is a point where the user may follow the wrong path.

B. Preliminary Definitions

The normal execution of the framework comprises the reduction of a non askable literal into subgoals under the rules in $\mathcal{P}$. Upon the appearance of an askable literal a query is sent to the system modules . When the answers are returned from Σ, they are added to the execution. To be able to create a proof procedure for the previously described framework there is the need for a few definitions:

Definition 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 2. *APS is a set of active processes and SPS is a set of suspended processes.*

Definition 3. *AAQ is a set of askable literals that have already been asked. The Current Belief State (CBS) is the set of askable literals.*

The APS expresses the set of processes that are consistent with the CBS and the SPS represents those which are not. 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, namely active processes and suspended processes.

Definition 4. *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 definitions state that an active process is a process in which the outside defaults are consistent with the current belief state. As such, if a process that states that an alert to the user should be issued is active, it means that the system will perform that action.

C. Process Reduction Phase

During this phase a process may suffer some changes. It represents the normal reduction of literals according to the rules in $\mathcal{P}$, the integrity constraints $\mathcal{I}$, and the CBS. In the following description 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 $\exists$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, $resolvent(L,T) = \{\{L_1\theta, \ldots, L_{i-1}\theta, L_{i+1}\theta, \ldots, L_k\theta\} | \perp \leftarrow L_1, \ldots, L_k \in T$ 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 $\exists$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 $NewPS = PS' \bigcup \{\langle GS', OD \bigcup \{L\}, IA, ANS\rangle\}$.

D. Fact Arrival Phase

During the process reduction phase information is asked from the system modules. 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 system module 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.

E. Generation of Default Values

The Framework of Speculative Computation uses the values from the *default answer set* (defined by Δ) whenever it is under a scenario of incomplete information. During the execution of the framework it asks the information source and keeps the execution using the value from this set. The *default answer set* needs to be generated each time the user wants to travel from the location he currently is to the destination selected inside the mobile application. This generation process is very important because its goal is to find a traveling route that is adjusted to the preferences of the user. Instead of guiding the user though the shortest path, it might guide him through a longer, but preferred, path. This set also includes values of possible mistakes that the user may make, reflecting his habits when following that route. They are used in order to alert and prevent him from taking the wrong path.

The generation of the *default values* is a complex process and is dependent on the number of times the person has traveled with the aid of the CogHelper System. If it is the first time the person with cognitive disabilities is using the application, then there is no historic data about him and the generated route will use a shortest path method (*e.g.*, Dijkstra algorithm). If there is historic data available, then all similar routes are retrieved from the database. This set of routes includes those with the same (or approximately) starting point or routes that go through the current starting point to the intended destination, *i.e.*, routes that started elsewhere and, from a given point, become similar to the other routes retrieved from the database.

After this first retrieval step, a pattern mining method is applied [10], [11], [12] to get the best route (the one that best fits user preferences). Figure 5 represents a calculated path using the raw data retrieved from the database in which a pattern mining method has been applied. This example route will be used to guide the user between the points.

To be able to use the resulting route to guide the user it is necessary to remove the error of the raw data. For this normalizing step there are multiple online services like *RoadMatcher*[13], *TrackMatching*[14] and *GraphHopper*[15], to name a few. To use these services, the system sends the route raw points and receives the points without the inherent raw error. This second phase ends with a route like the one presented in Figure 6.

Removing the error from the raw GPS data enables the system to guide the user in a more precise way, however if we use this calculated route for the guiding process it would generate too many assertive messages informing the user to travel small distances, given the density of points. This situation would be distracting to the user since the number of prompts would be exacerbated. Thus, after this process we need to reduce the number of routing points to the minimum. Using the map information from OpenStreetMap, it is possible to get the starting and ending points of a street and get

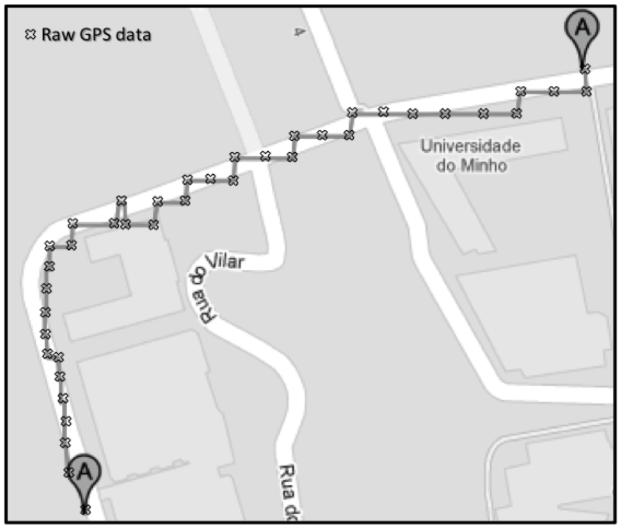


Fig. 5. Route obtained from database raw data and after applying a pattern mining method

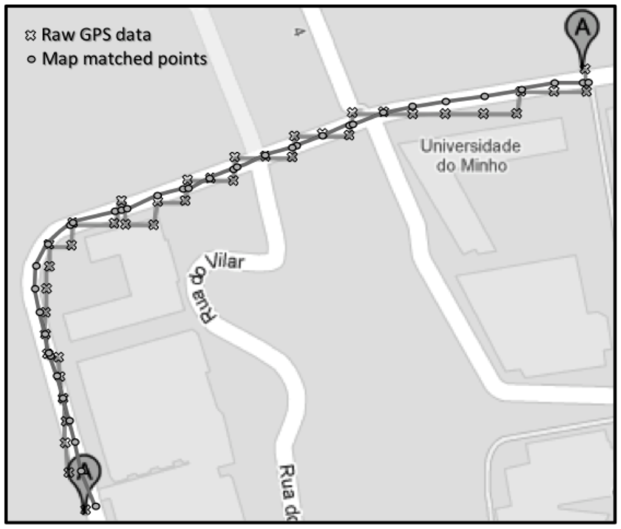


Fig. 6. Route obtained after applying a pattern mining method

intersections with other streets. With these points, it becomes possible to guide the user and predict if he is going to take a wrong turn at an intermediate intersection (using historic data from the database). This reduction process is illustrated in Figure 7.

With the reduced route the system may guide the user to the intended destination. However, there is the need to convert the GPS points (latitude and longitude coordinates) so that the framework of Speculative Computation may use this information and create the alerts whenever they are necessary. This data conversion transforms single location points into arcs that connect these points. Thus, if the generated route has 30 valid points (including start, end and intermediate locations) 29 arcs are created. These are used to indicate the next location point that the user should travel to. In order to alert the user we need to create another arc between the error point X (see

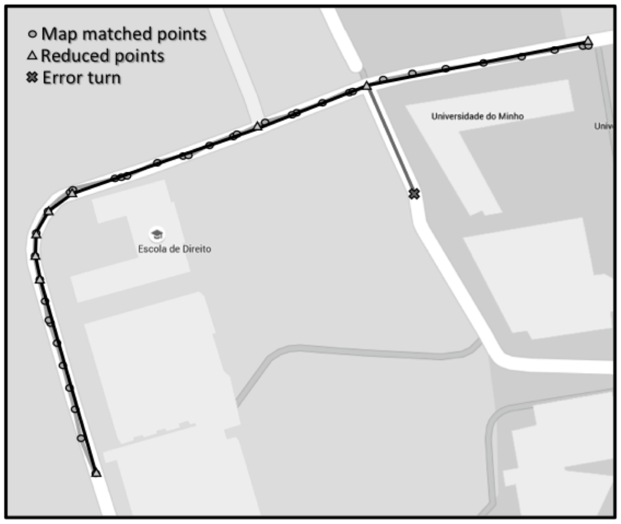


Fig. 7. Route with reduced number of points

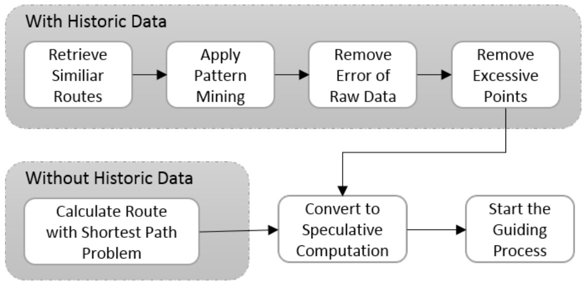


Fig. 8. Generation of default values schema

error turn in Figure 7) and an intersection with a valid point. This arc together with the literal $\sim included(X)$ indicates that the user is traveling to a wrong direction and an alert must be shown. The entire process of generating the default values depends on historic data of each user. Note that a user may have historic information about previous interactions with the application, but it may not suit the current guidance because the user may be traveling to a different destination or from a different starting location. Figure 8 shows a schematic of the tasks necessary to get the *default answer set*.

F. Execution Example

An execution example of the Program introduced in Section IV-A is presented below. For the process reduction, when a positive literal is reduced, new processes are created according to the rule order in the program, if the rules are unifiable with the positive literal.

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

①
$APS = \{\langle\{\underline{guide(1,3)}\},\ \emptyset,\ \emptyset\rangle\}$
$AAQ = \emptyset$
$CBS = \{user\_travel(1,2)@gps\_sensor,$
$user\_travel(2,3)@gps\_sensor, user\_travel(2,4)@gps\_sensor,$
$included(1)@recognizer, included(2)@recognizer,$
$included(3)@recognizer, \sim included(4)@recognizer\}$

② By case 3.1
$APS = \{\langle\{\underline{path(1,F)},\ show\_next\_point(F),$
$user\_travel(1,F)@gps\_sensor,\ guide(F,3)\},\ \emptyset,\ \emptyset\rangle,$
$\langle\{\underline{path(1,F)},\ user\_travel(1,F)@gps\_sensor,$
$show\_user\_warning(F),\ guide(F,3)\},\ \emptyset,\ \emptyset\rangle\}$

③ By case 3.1
$APS = \{\langle\{\underline{show\_next\_point(2)},$
$user\_travel(1,2)@gps\_sensor, guide(2,3)\},\ \emptyset,\ \emptyset\rangle,$
$\langle\{user\_travel(1,2)@gps\_sensor,$
$show\_user\_warning(2), guide(2,3)\},\ \emptyset,\ \emptyset\rangle\}$

④ By case 3.2.3, 3.3.2 and 3.3
$APS = \{\langle\{\underline{included(2)@recognizer},$
$user\_travel(1,2)@gps\_sensor, guide(2,3)\}, \emptyset,$
$\{show\_next\_point(2)\}\rangle,$
$\langle\{user\_travel(1,2)@gps\_sensor,$
$show\_user\_warning(2), guide(2,3)\},\ \emptyset,\ \emptyset\rangle\}$

⑤ By case 3.4
$included(2)$ is asked to the $recognizer$
$APS = \{\langle\{\underline{user\_travel(1,2)@gps\_sensor},\ guide(2,3)\},$
$\{included(2)@recognizer\}, \{show\_next\_point(2)\}\rangle,$
$\langle\{\underline{user\_travel(1,2)@gps\_sensor},$
$show\_user\_warning(2), guide(2,3)\},\ \emptyset,\ \emptyset\rangle\}$
$AAQ = \{included(2)@recognizer\}$

To an easier comprehension of this execution example the branch that would check if an alert should be issued when the user travels towards location 2 is omitted since it would become a suspended process.

⑥ By case 3.4
$APS = \{\langle\{\underline{guide(2,3)}\},\ \{included(2)@recognizer,$
$user\_travel(1,2)@gps\_sensor\}, \{show\_next\_point(2)\}\rangle\}$

⑦ By case 3.1
$APS = \{\langle\{\underline{path(2,F)},\ show\_next\_point(F),$
$user\_travel(2,F)@gps\_sensor,\ guide(F,3)\},$
$\{included(2)@recognizer, user\_travel(1,2)@gps\_sensor\},$
$\{show\_next\_point(2)\}\rangle,$
$\langle\{\underline{path(2,F)}, user\_travel(2,F)@gps\_sensor,$
$show\_user\_warning(F),\ guide(F,3)\}, \{included(2)@recognizer,$
$user\_travel(1,2)@gps\_sensor\}, \{show\_next\_point(2)\}\rangle\}$

⑧ By case 3.1
$APS = \{\langle\{\underline{show\_next\_point(3)},$
$user\_travel(2,3)@gps\_sensor,\ guide(3,3)\},$
$\{included(2)@recognizer, user\_travel(1,2)@gps\_sensor\},$
$\{show\_next\_point(2)\}\rangle,$
$\langle\{show\_next\_point(4), user\_travel(2,4)@gps\_sensor,$
$guide(4,3)\}, \{included(2)@recognizer,$
$user\_travel(1,2)@gps\_sensor\}, \{show\_next\_point(2)\}\rangle^{P_1},$
$\langle\{user\_travel(2,3)@gps\_sensor, show\_user\_warning(3),$
$guide(3,3)\}, \{included(2)@recognizer,$
$user\_travel(1,2)@gps\_sensor\}, \{show\_next\_point(2)\}\rangle^{P_2}$
$\langle\{user\_travel(2,4)@gps\_sensor, show\_user\_warning(4),$
$guide(4,3)\}, \{included(2)@recognizer,$
$user\_travel(1,2)@gps\_sensor\}, \{show\_next\_point(2)\}\rangle^{P_3}\}$

⑨ By case 3.2.3, 3.3.2 and 3.3
$APS = \{\langle\{\underline{included(3)@recognizer},$
$user\_travel(2,3)@gps\_sensor,\ guide(3,3)\},$
$\{included(2)@recognizer, user\_travel(1,2)@gps\_sensor\},$
$\{show\_next\_point(2), show\_next\_point(3)\}\rangle,$
$P_1, P_2, P_3\}$

⑩ By case 3.4
$included(3)$ is asked to the $recognizer$
$APS = \{\langle\{\underline{user\_travel(2,3)@gps\_sensor},\ guide(3,3)\},$
$\{included(2)@recognizer, user\_travel(1,2)@gps\_sensor,$
$included(3)@recognizer\},$
$\{show\_next\_point(2), show\_next\_point(3)\}\rangle,$
$P_1,$
$\langle\{\underline{user\_travel(2,3)@gps\_sensor}, show\_user\_warning(3),$
$guide(3,3)\}, \{included(2)@recognizer,$
$user\_travel(1,2)@gps\_sensor\}, \{show\_next\_point(2)\}\rangle,$
$P_3\}$
$AAQ = \{included(2)@recognizer,$
$user\_travel(1,2)@gps\_sensor, included(3)@recognizer\}$

⑪ By case 3.4
$user\_travel(2,3)$ is asked to the gps_sensor
$APS = \{\langle\{\underline{guide(3,3)}\}, \{included(2)@recognizer,$
$user\_travel(1,2)@gps\_sensor, included(3)@recognizer,$
$user\_travel(2,3)@gps\_sensor\},$
$\{show\_next\_point(2), show\_next\_point(3)\}\rangle^{P_4},$
$P_1,$
$\langle\{\underline{show\_user\_warning(3)},\ guide(3,3)\},$
$\{included(2)@recognizer, user\_travel(1,2)@gps\_sensor,$
$user\_travel(2,3)@gps\_sensor\}, \{show\_next\_point(2)\}\rangle,$
$P_3\}$
$AAQ = \{included(2)@recognizer,$
$user\_travel(1,2)@gps\_sensor, included(3)@recognizer,$
$user\_travel(2,3)@gps\_sensor\}$

⑫ By case 3.2.3, 3.3.2 and 3.3
$APS = \{P_4, P_1,$
$\langle\{\underline{\sim included(3)@recognizer},\ guide(3,3)\},$
$\{included(2)@recognizer, user\_travel(1,2)@gps\_sensor,$
$user\_travel(2,3)@gps\_sensor\},$
$\{show\_next\_point(2), show\_user\_warning(3)\}\rangle,$
$P_3\}$

According to what was previously described, the branch represented by P_3 is not shown since the system must not indicate the user to travel towards location 4. Thus, after executing this branch, it will become a suspended process.

⑬ By case 3.4
$included(4)$ is asked to the $recognizer$
$APS = \{P_4,$
$\langle\{\underline{user\_travel(2,4)@gps\_sensor}, show\_user\_warning(4),$
$guide(4,3)\}, \{included(2)@recognizer,$
$user\_travel(1,2)@gps\_sensor\}, \{show\_next\_point(2)\}\rangle\}$
$AAQ = \{included(2)@recognizer,$
$user\_travel(1,2)@gps\_sensor, included(3)@recognizer,$
$user\_travel(2,3)@gps\_sensor, included(4)@recognizer\}$

⑭ By case 3.4
$user\_travel(2,4)$ is asked to the gps_sensor
$APS = \{P_4,$
$\langle\{\underline{show\_user\_warning(4)}, guide(4,3)\}, \{included(2)@recognizer,$

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

⑮ By case 3.2.3, 3.3.2 and 3.3
$APS = \{P_4,$
$\langle\{\sim included(4)@recognizer,\ guide(4,3)\},$
$\{included(2)@recognizer, user\_travel(1,2)@gps\_sensor,$
$user\_travel(2,4)@gps\_sensor\},$
$\{show\_next\_point(2), show\_user\_warning(4)\}\rangle\}$

⑯ By case 3.4
$APS = \{\langle\{guide(3,3)\}, \{included(2)@recognizer,$
$user\_travel(1,2)@gps\_sensor, included(3)@recognizer,$
$user\_travel(2,3)@gps\_sensor\},$
$\{show\_next\_point(2), show\_next\_point(3)\}\rangle,$
$\langle\{guide(4,3)\}, \{included(2)@recognizer,$
$user\_travel(1,2)@gps\_sensor, user\_travel(2,4)@gps\_sensor,$
$\sim included(4)@recognizer\},$
$\{show\_next\_point(2), show\_user\_warning(4)\}\rangle^{P_5}\}$

⑰ By case 3.1
$APS = \{\langle\emptyset, \{included(2)@recognizer,$
$user\_travel(1,2)@gps\_sensor, included(3)@recognizer,$
$user\_travel(2,3)@gps\_sensor\},$
$\{show\_next\_point(2), show\_next\_point(3)\}\rangle,$
$P_5\}$

At this stage the system has an active process that will also become suspended since the user is alerted before making the mistake of turning to location 4. When this occurs the execution trace ends since there is no more active processes.

V. Conclusion and Future Work

This work proposes a default generation method which produces predictions for the direction that a user may follow at particular points in his route. The process is based on data from previous runs of the system which are used to identify critical points, such as intersections, where he may make a mistake in his path. The habits of the user at those locations are also assessed in order to determine the direction he usually follows when he is there. These habits will become the default values. Here, Speculative Computation is used as a control mechanism that helps the system determine whether it is necessary to issue an alert or not. The integration of both the Generation of Defaults and Speculative Computation to create a predictive feature for orientation systems is the main contribution of this work. The method for the detection of user orientation patterns is independent of the framework, but Speculative Computation can be used in combination with it, providing a structured reasoning framework. With the general definition of the Generation of Defaults, it becomes necessary to explore different pattern mining techniques in order to determine which is the most appropriate for the problem at hand. Therefore, this will be the object of study in future works.

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) and within Project Scope UID/CEC/00319/2013. The work of João Ramos is supported by a doctoral the FCT grant SFRH/BD/89530/2012. The work of Tiago Oliveira is also supported by the FCT grant with the reference SFRH/BD/85291/2012.

References

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

[2] S. Carmien, M. Dawe, G. Fischer, A. Gorman, A. Kintsch, and J. F. Sullivan, "Socio-technical environments supporting people with cognitive disabilities using public transportation," *ACM Transactions on Computer-Human Interaction*, vol. 12, no. 2, pp. 233–262, 2005.

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

[4] Fraunnhover Portugal, "AlzNav," 2012. [Online]. Available: http://www.fraunhofer.pt/en/fraunhofer_aicos/projects/internal_research/alznav.html

[5] J. Ramos, A. Costa, P. Novais, and J. Neves, "Interactive Guiding and Localization Platform," *International Journal of Artificial Intelligence (IJAI)*, vol. 12, no. 1, pp. 63–78, 2014.

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

[7] C. C. White and D. J. White, "Markov decision processes," *European Journal of Operational Research*, vol. 39, no. 1, pp. 1–16, 1989.

[8] K. Satoh, K. Inoue, K. Iwanuma, and C. Sakama, "Specultative Computation by Abduction under Incomplete Communication Environments," in *Proceedings of the Fourth International Conference on MultiAgent Systems*, vol. 12. IEEE, 2000, pp. 263–270.

[9] K. Satoh, "Speculative Computation and Abduction for an Autonomous Agent," *IEICE - Transactions on Information and Systems*, vol. E88-D, no. 9, pp. 2031–2038, 2005.

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

[11] E. Masciari, G. Shi, and C. Zaniolo, "Sequential pattern mining from trajectory data," *Proceedings of the 17th International Database Engineering & Applications Symposium on - IDEAS '13*, pp. 162–167, 2013. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2513591.2513653

[12] L. Chen, M. Lv, Q. Ye, G. Chen, and J. Woodward, "A personal route prediction system based on trajectory data mining," *Information Sciences*, vol. 181, no. 7, pp. 1264–1284, 2011.

[13] Vivid Solutions, "RoadMatcher." [Online]. Available: http://wiki.openstreetmap.org/wiki/RoadMatcher

[14] F. Marchal, "TrackMatching." [Online]. Available: https://mapmatching.3scale.net

[15] P. Karich, "GraphHopper," 2015. [Online]. Available: http://wiki.openstreetmap.org/wiki/GraphHopper