

# Coordinating Heterogeneous Agents to Synthesize *Proactive Monitoring*

Amedeo Cesta, Gabriella Cortellessa, Federico Pecora, and Riccardo Rasconi

ISTC-CNR, National Research Council of Italy  
Institute for Cognitive Science and Technology  
Via S.Martino della Battaglia 44, I-00185 Rome, Italy  
<name.surname>@istc.cnr.it

## Abstract

This paper describes the results of the ROBOCARE, a project aimed at creating an integrated environment endowed with heterogeneous software and robotic agents for assisting an elderly person at home. Specifically, a proactive environment for continuous daily activity monitoring has been created in which an autonomous robot acts as the main interactor with the person. This paper describes how the synergy of different component technologies guarantees an overall intelligent behavior capable of personalized and contextualized interaction with the assisted person.

## Environment for continuous monitoring

In the ROBOCARE project we have been dealing with the problem of monitoring an older person during her daily activity at home. The use of intelligent technology for supporting older people at home has been addressed in various research projects in the last years, e.g., PEAT (Levinson 1997), PEARL (Pineau *et al.* 2003; Pollack 2005), I.L.S.A. (Haigh, Kiff, & Ho 2006).

In ROBOCARE we have followed the specific direction of creating a home environment dedicated to monitoring a person rather than concentrating all the functionalities on a single robot. The result is a prototypical intelligent environment that integrates robotic and software components to obtain a continuous behavior that we call here *Proactive Monitoring*. The goal underlying the intelligent environment concerns the ability to (a) maintain *continuity of behavior*, such as ensuring continuous monitoring of what happens in the environment (the state of the assisted elder and of his/her domestic context), (b) create a *context* at the knowledge level around the actions that the assisted elder performs (routinely, exceptionally, etc.), and (c) provide contextualized interaction services with the assisted elder aimed at *proactive assistance*.

In particular this paper describes how we have achieved continuous proactive monitoring in a domestic environment, that is, a specific implementation of the sense-plan-act cycle, by integrating separate intelligent components as a multi-agent system. A coordination algorithm guarantees a consistent collective behavior of the entire environment by continuously solving a Distributed Constraint Optimization Problem (DCOP). The coordination problem provides the “semantic glue” of the system, i.e., its resolution orients the en-

vironment toward guaranteeing safety of the observed person. In this paper we describe how we have obtained this comprehensive behavior by integrating: (1) the use of a distributed constraint optimization algorithm for coordination of the multiple agents involved in activity monitoring; (2) the use of a constraint based scheduling system, and in particular of the continuous schedule monitoring functionality employed to provide appropriate alerts and warnings at the occurrence of constraint violations; (3) the generation of relevant explanations from these constraint violations to be presented to the user and other observers of the process in the form of verbal interaction instances. This paper specifically focuses on the aspects related to the system’s context-awareness and interaction capabilities. In particular, we describe how the constraint-based scheduling technology is used to maintain a knowledge repository aimed at supporting on-demand specific interactions as well as enabling autonomous system initiative.

## Separate intelligent capabilities

The main “actor” in our smart home environment is a robotic agent with verbal interaction capabilities. The robot acts as a “mediator” through which the assisted person receives advice/warnings and can query the environment. As shown in Figure 1, the robot is composed of two distinct modules. The mobile robotic platform provides advanced mobility functionalities (referred to as “robot motion skills” in the figure<sup>1</sup>). A second module creates an additional level of competence for the robot, referred to as “interactive skills” in Figure 1(a). Indeed this capability groups and provides access to the functionalities of the overall intelligent system. Figure 1(b) shows how the interaction skills use (a) a simplified Interaction Manager, (b) a front-end for the interaction module consisting in a Talking Head and a Speech Recognition subsystem taken as external off-the-shelf modules, (c) a key component called Intelligent Activity Monitor whose role is very relevant for the situated interaction capability of the system.

<sup>1</sup>The robotic platform, developed by colleagues from University of Rome “La Sapienza”, consists of a Pioneer 2 integrated with a Sick laser scanner for localization. Additional work has been required to both integrate advanced SLAM algorithms and obtain robust navigation abilities which are suited for the domestic environment. Details are outside the scope of the paper.

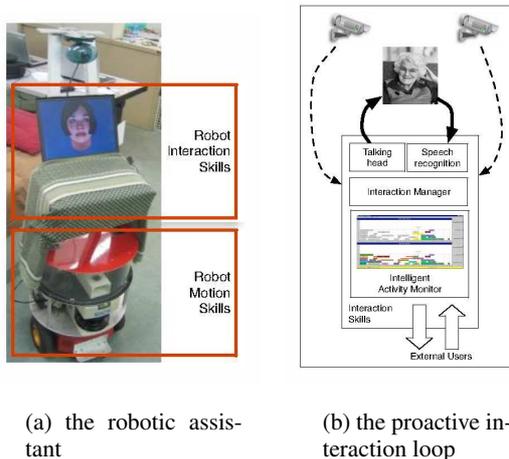


Figure 1: Components of the ROBOCARE demonstrator

The Activity Monitor is a separate module which integrates a constraint-based scheduler with additional features for knowledge engineering, in particular for problem modeling. Particular attention has been given to the definition of “user-oriented terminologies” in order to easily synthesize both the basic elements of a target domain as well as different problem instances in the particular domain (i.e., in the form of activities and constraints among activities). For example, it allows the definition of “home domain activities” like *breakfast*, *lunch*, *go-for-walk*, and also temporal/causal links among activities like *meal-bound-medication* to express rules like “aspirin cannot be taken before eating”. Through this high level language an external user (a doctor, a relative of the assisted person, etc.) may define a network of activities, a *schedule*, that the observed person is supposed to carry out in the home environment during the day. This schedule is dispatched for execution and monitored using the underlying schedule execution technology.

The key step for carrying out schedule execution is to integrate information coming from *sensors*. In particular, we used stereo cameras, which were distributed inside the prototype apartment and endowed with specific software for people localization and tracking developed by our colleagues from “La Sapienza”. Even if complete human activity recognition was outside the scope of the project, it is worth highlighting how the sequence of observations from the artificial vision sensors allows to follow the evolution of the activities of the observed person (e.g., if and when she took a pill, when she had lunch, etc.). The key focus of this paper is on the generation of a global cognitive support service through the combination of the “elementary” services provided by the separate intelligent components. This is made possible by three features, namely the original use of a multi-agent coordination algorithm based on distributed constraint reasoning (DCR), the specialized use of the activity monitor, and through the generation of speech acts for active interaction based on constraint violations.

### The coordination sub-problem

Coordination of multiple services is achieved by solving a Multi-Agent Coordination problem. This problem is cast

as a Distributed Constraint Optimization Problem (DCOP), and solved by ADOPT-N (Pecora, Modi, & Scerri 2006), an extension of the ADOPT (Asynchronous Distributed Optimization) algorithm (Modi *et al.* 2005) for dealing with  $n$ -ary constraints. Figure 2 gives an intuition of the chosen approach. We call  $Application_i$  the generic intelligent subsystem that is to be integrated in the overall multi-agent system, and  $Var_j$  one out of a set  $\mathcal{V}$  of variables in terms of which the coordination problem is defined. Each variable has an associated domain of Values  $D_j$ . Variables are bound by constraints like in regular Constraint Satisfaction Problems (CSP). Conversely, while constraints in CSP evaluate to *satisfied* or *unsatisfied*, in the optimization case constraints evaluate to costs, and can thus express what are often referred to as “soft constraints”. Such constraints are useful for modeling preferences, and in general requirements which have a “degree of satisfiability”. Constraints may involve an arbitrary subset of the variables ( $n$ -ary constraints): a constraint among the set  $C \subset \mathcal{V}$  of  $k$  variables is expressed as a function in the form  $f_C : D_1 \times \dots \times D_k \rightarrow \mathbb{N}$ . For instance, a constraint involving the three variables  $\{Var_1, Var_3, Var_7\}$  may prescribe that the cost of a particular assignment of values to these variables amounts to  $c$ , e.g.,  $f_{Var_1, Var_3, Var_7}(0, 3, 1) = c$ . The objective of a constraint optimization algorithm is to calculate an assignment  $\mathcal{A}$  of values to variables while minimizing the cost of the assignment  $\sum_{C \in \mathcal{C}} f_C(\mathcal{A})$ , where each  $f_C$  is of arity  $|C|$ .

In ROBOCARE, the valued constraints are decided in order to orient the solution of the DCOP toward preferred world situations (broadly speaking, those situations in which the person is maximally helped by the intelligent system). The system is composed of a number of heterogeneous applications: (a) an activity monitor, (b) the interaction manager plus the speech I/O modules, (c) the robot mobile platform, (d) one application for each of the cameras, each of them with the appropriate software for people localization and tracking.

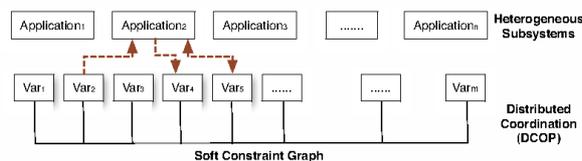


Figure 2: DCOP to maintain distributed coherence.

Each application manages one or more of the variables which are part of the DCOP. A variable may represent (a part of) the input to an application, its output, or both (see the dashed lines in Figure 2 as an example). When the DCOP resolution algorithm (ADOPT-N) is called into play, the values of the application-output variables are taken into account in the distributed resolution. When resolution reaches a fixed point, the input variables will reflect an updated input for the applications. The correspondence between the applications’ output at the  $i$ -th iteration and their input at iteration  $i + 1$  is a result of the propagation rules specified in the DCOP. Overall, the decisions of the applications constitute the input for the subsequent iteration of the cycle  $\langle DCOP-resolution; read-variable; application-$

*decision; write-variable*). The complete iterative procedure is depicted in Algorithm 1, which constitutes the core of the continuous operation requirement of the smart home. Specifically, let the situation, i.e., the state of the environ-

---

**Algorithm 1** Synchronization schema followed by each application *app*.

---

```

1.  iter ← 0; Siter ← getSensoryInput(Vinapp)
2.  while true do
3.    Siter-1 ← Siter
4.    while (Siter = Siter-1) ∧
      (iter ≥ app'.iter, ∀ app' ≠ app) do
5.      Siter ← getSensoryInput(Vinapp)
6.      iter ← iter + 1
7.      forall v ∈ Vinapp ∪ Voutapp do
8.        resetVarAssignment()
9.        A|Voutapp ← runAdopt() /** ADOPT-N termination **/
10.       triggerBehavior(A|Voutapp)

```

---

ment, of the assisted person, and of the services themselves, at time  $t$  be  $S_t$ . The DCOP formulation of the coordination problem represents the desired behavior of the system in function of the possible states of the environment and of the assisted person. Therefore, if  $S_t \neq S_{t-1}$ , the agents must trigger an “instance of coordination” so as to decide the assignment  $\mathcal{A}$  which represents the desired enactment of services.

The strong difference in nature between the various components of the multi-agent system reflects heavily on the coordination mechanism because of the uncertainty connected to the time employed by services to update the symbolic information which is passed on to the agents. For instance, the system contains devices which are driven by artificial stereo-vision algorithms, the convergence of which is strongly affected by a variety of factors. This problem also affects other components of the smart home, such as the activity monitor (described later in this article) which must propagate sensor-derived information on the temporal representation of the assisted person’s daily schedule. As a consequence, it is in general impossible to have strict guarantees on the responsiveness of the agents. For this reason the albeit asynchronous solving procedure needs to be iterated synchronously. More specifically, the agents continuously monitor the current situation and execute the ADOPT-N algorithm whenever a difference with the previous situation is found. The `getSensoryInput()` method in the pseudo-code samples the state of the environment which is represented by agent *app*’s input variables  $V_{in}^{app}$ . All agents concurrently initiate the ADOPT-N algorithm whenever the state changes or another agent has initiated the solving iteration. Thus, when an agent senses a difference in its input variables, its decision to run the coordination algorithm is cascaded to all other agents (see the condition in the internal while loop above).

Since ADOPT-N does not rely on synchronous communication between agents, it natively supports message transfer with random (but finite) delay. This made it possible to employ ADOPT-N within the smart home scenario without modifying the algorithm internally. Furthermore, while

most distributed reasoning algorithms are employed in practice as concurrent threads on a single machine (a situation in which network reliability is rather high), the asynchronous quality of ADOPT-N strongly facilitated the step towards “real” distribution, where delays in message passing increase in magnitude as well as randomness.

## Continuous monitoring of the assisted person

The ability to detect and follow the actions of the assisted person is one of the important features of the assistive environment. The main goal is in fact to guarantee cognitive and physical support while continuously maximizing the user’s independence, well-being and safety.

In this context we employed scheduling technology as a specific knowledge component. The Activity Monitor plays the role of activity *supervisor* and continuously tries to maintain situation awareness by updating the representation of the set of activities that the assisted person performs in the environment.

As mentioned above, the desired behavior the assisted person should adhere to is initially decided by a caregiver (a physician, or a family member) in terms of a set of activities to be monitored, i.e., the *schedule*. Activities in the schedule are bound to one another through potentially complex temporal relationships.

An important role of the intelligent assistant in this context is played by the management of all the temporal constraints present in the schedule. As the environment sensing cycle commences, the system periodically checks the state of the monitored area, trying to detect and recognize the execution state of all the activities. Regardless of the caregiver’s prescriptions, the assisted person is obviously free to act as she likes: this basically means that at each detection cycle, the system is called to precisely assess the possible differences between the actual and desired state. Assessing such differences does not necessarily entail the need for a system reaction. Conversely, when a true constraint violation occurs, reaction is triggered in order to issue suggestions and warnings.

The principal issue in employing schedule execution monitoring technology to the domestic assistance scenario is that the system has no control whatsoever on the actions the assisted person is going to perform, despite the caregiver’s prescriptions. Indeed, the system can only affect the person’s behavior indirectly, by maintaining an updated representation of the environment as it is perceived by the sensors, and possibly reacting to some significant events, if deemed necessary. The monitoring cycle therefore focuses on: (1) keeping the internal representation of the real world consistent with the behavioral decisions of the assisted person at all times, and (2) performing the necessary rescheduling actions so as to satisfy a maximum number of temporal constraints originally imposed by the caregiver.

**An example.** To be more concrete, let us consider a behavioral pattern described by a schedule composed of 6 different activities (*breakfast, lunch, dinner*, as well as taking one of three different medicines). Due to medical requirements, let

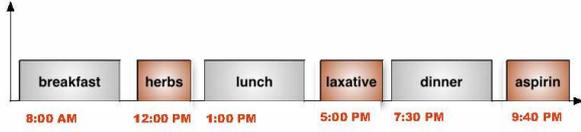


Figure 3: Example of desired behavior specified by the care giver for the assisted person in form of a *schedule*.

us also suppose that such activities must satisfy certain temporal requirements, such as “dinner should not begin before 7:30 PM, nor should it occur less than 5 hours after lunch” and “aspirin should only be taken after dinner, but no later than 20 minutes after”, and so on.

An “ideal schedule”, i.e., an enactment of these activities which does not violate any temporal constraint, is shown in Fig. 3. Broadly speaking, the objective of the Activity Monitor is to recognize deviations from this ideal situation. Specifically, the system should assess the extent to which the elder user’s behavior deviates from this situation. This equates to assessing which temporal constraints are progressively violated during the day. In a nutshell, system interventions are driven by constraint violations: warnings, alarms and suggestions result from violated constraints, which are processed by the interactive subsystem on board the robotic mediator.

**The execution monitoring algorithm.** Once the monitoring starts, the sensors are periodically queried and the nominal schedule is adjusted in accordance with the assisted person’s detected behavior. At each detection cycle, the execution status of each activity is checked: among the possible cases, some activities may be reported as under execution before their nominal start time, the execution of other activities may be reported as delayed, the duration of some activities may exceed the nominal value, and so on; each deviation from the nominal schedule may entail a conflict which has to be reacted upon.

As an example (see figure 3), let us suppose that the patient, after having dinner, sits on the sofa and starts watching TV: at each new sensor detection cycle, the system assesses the situation and delays the *aspirin-taking* activity. But since, according to the medical requirements, the aspirin should be taken no later than twenty minutes after dinner, delaying the aspirin activity beyond the prescribed time will eventually determine a time constraint insertion conflict in the temporal constraint database; upon the occurrence of this conflict, the intelligent assistant will respond by issuing a warning to the patient as a reminder for the forgotten action.

Algorithm 2 shows the execution monitoring algorithm we employ. As shown, an “environment sensing” action is periodically performed (line 2). This occurs by accessing the symbolic representation of the current situation ( $S_t$ ), that is obtained by reading the DCOP variable described previously. As a result, the set  $Events_t$  of the occurred events is periodically acquired. By *event* we mean any mismatch between the expected situation, according to the caregiver’s

prescriptions, and the actual situation (i.e., a planned action which fails to be executed, is considered as an event).

---

**Algorithm 2** The Execution Monitoring Algorithm.

---

```

1.  while true do
2.     $Events_t \leftarrow S_t$ 
3.    if  $Events_t \neq \emptyset$  then
4.       $C_{r,t} \leftarrow \text{removeConstraints}()$ 
5.       $\text{insertContingencies}(Events_t)$ 
6.       $K_t \leftarrow \emptyset$ 
7.      while  $C_{r,t} \neq \emptyset$  do
8.         $c_j \leftarrow \text{chooseConstraint}(C_{r,t})$ 
9.        if  $\neg \text{re-insertConstraint}(c_j)$  then
10.          $K_t \leftarrow K_t \cup c_j$ 

```

---

If events are detected, the first action is to remove all the active constraints present in the schedule (line 4). By *active* constraints, we mean those which do not completely belong to the past, with respect to the actual time of execution  $t_E$ <sup>2</sup>.

In the next step (line 5) all the detected contingencies, properly modeled as further constraints, are inserted in the plan. This is the step where the system updates the internal representation of the schedule in order to preserve consistency with the world’s true state.

Lines 7–10 implement the constraint re-insertion cycle, where the algorithm tries to restore as many caregiver requirements as possible given the current situation. Notice in fact that it is probable that not all the original constraints will be accepted at this point: the occurrence of the contingencies might in fact have changed the constrainedness of the temporal network, which in turn can make the complete re-insertion of the constraints removed at the previous step impossible. During the cycle, all the constraints which are rejected are stored in the set  $K_t$ . Constraints insertion (and rejection) is an extremely delicate issue, for many reasons:

- System reaction may consist in verbal suggestions or warning. The information conveyed by these messages strongly depends on the contents of the set  $K_t$ . As we will show, the analysis of all the rejected constraints quantitatively and qualitatively determines the system’s response. Given a temporal network  $TN$  underlying the current schedule, the set  $K_t = \{k_{t,1}, k_{t,2}, \dots, k_{t,r}\}$  must be such that: (1) the insertion of each  $k_{t,j}$  in  $TN$  causes a propagation failure; (2) the cardinality of  $K_t$  is maximum. Condition (1) ensures that every constraint in  $K_t$  plays a role in determining system’s reaction, ruling out false-positive situations; condition (2) ensures that no contingency escapes system’s attention.
- The acceptance of each constraint  $c_j$  (and complementarily, the contents of  $K_t$ ), is generally dependent on the particular order chosen for re-insertion. In general, a number of different choice heuristics ( $\text{chooseConstraint}()$  method) can be envisaged, leading to different

---

<sup>2</sup>More formally, given an execution instant  $t_E$  and a constraint  $c_k$  binding two time points  $t_a$  and  $t_b$ ,  $c_k$  is considered *idle* if and only if  $(t_a < t_E) \wedge (t_b < t_E)$ . All constraints that are not idle are active.

approaches for contingency management. To clarify this issue, let us consider a temporal network  $TN$  and two constraints  $c_1$  and  $c_2$  such that the attempt of posting both of them in  $TN$  would determine an inconsistency: in this case, if the insertion order is  $\{c_1, c_2\}$ , then  $c_2$  is going to be rejected; if the opposite order is used,  $c_1$  is rejected. Since in this context it is essential that the reaction be related to the closest contingency with respect to execution time  $t_E$ , the particular heuristic employed for re-insertion is backward-chronological. The result of this choice is that the rejected constraints will be the ones which are temporally closer to the actual instant of execution, therefore meeting the condition of reaction urgency. In other terms, the monitoring system is oriented towards synthesizing a suggestion regarding the primary cause of a violation, rather than forming one based on a distant effect of the assisted person's behavior. The constraints are chronologically ordered taking into account the values of the time point pairs they are connected to. More formally, given a set of constraints  $\{c_1(t_{1,s}, t_{1,e}), c_2(t_{2,s}, t_{2,e}), \dots, c_n(t_{n,s}, t_{n,e})\}$ , where each  $c_i(t_{i,s}, t_{i,e})$  connects the time points  $t_{i,s}$  and  $t_{i,e}$ , the constraint  $c_i(t_{i,s}, t_{i,e})$  chronologically precedes the constraint  $c_j(t_{j,s}, t_{j,e})$ , if  $\min(t_{i,s}, t_{i,e}) < \min(t_{j,s}, t_{j,e})$ .

### Managing assistant/assisted interaction

As already mentioned, interaction relies here on the embodied robotic assistant as the focal point between the user and the system. Communication between the user and the robotic mediator occurs verbally. We distinguish two forms of interaction based on *who takes the initiative* to start a dialogue:

**On-Demand interaction** in which the user takes the initiative first. The assisted person commences interaction, for instance, by querying the system's knowledge base: "have I taken my pills?", or "can I make an appointment for tomorrow at 5 PM?".

**Proactive interaction** in which the intelligent environment commences interaction guided by its internal reasoning. In ROBOCARE, constraint violations have been considered as a *trigger* for the system to take the initiative and perform some actions: issue an alarm in case of illness, or verbalize warnings and suggestions.

Our work explicitly focuses on the development of active and, at the same time, unobtrusive services to integrate within the artificial assistant. All interaction services rely on the Interaction Manager. This module essentially consists in a rule-based system that fires *situation-action* rules. In other words, it continuously assesses the situation and activates a particular sub-module as an action.

We categorize as *On-Demand* interaction the "Question-Answer" category of dialogues. This activity is triggered by a speech input from the assisted person. The generation of the answer is managed mostly internally to the manager that has information on the history of activities and/or on the current state of the environment, to answer questions like "Have I had lunch?" or "What time is it?", etc.

Instances of *Proactive* interaction are "Danger" and "Warning" scenarios. Undoubtedly, one of the important tasks for assistance is to recognize emergencies for the monitored person. The emergency trigger is fired by particular combinations of the input provided by the sensors that monitor the environment and the assisted person. As an example we can discriminate as a dangerous situation the case in which a person is "laying down in the kitchen floor" or "laying down in bed half an hour after usual wake up" as a danger, rather than "laying down in bed within a given period" which is recognized as a regular situation. The danger trigger is dealt with by a specific behavior of the multi-agent system that interrupts the usual flow of activities and undertakes an action: the robot is sent to the assisted person, a specific dialogue is attempted, and if no answer from the assisted person is obtained, an *Alarm* is immediately fired to the external world (call to a relative, to an emergency help desk, etc.).

A warning scenario is one in which constraint violations are detected by the Activity Monitor. Broadly speaking, the monitor decides the values for the variables that are used by the interaction manager to trigger a proactive dialogue with the assisted person. The content of the dialogue is synthesized on the basis of the monitor's internal knowledge.

Overall, the Interaction Manager is a simple planner that supervises the initiative of the "interactor" towards the assisted person. It is worth underscoring how the combination of this manager and the activity monitor endows the entire assistive environment with capabilities of proactive participation in a mixed-initiative interaction.

### From scheduler knowledge to interaction

The main issue here is how to translate constraint violation information into semantically meaningful speech acts that the user may immediately understand. First, we present the building blocks of this semantic analysis. At this level, all semantic inference will have to be derived exclusively from the analysis of information of temporal nature; the second step is to integrate temporal data with different types of environmental information.

Each element in the violated constraints set  $K_t$  is either a *minimum* or a *maximum* constraint. *Duration* constraints are defined through a minimum and a maximum constraint between the start and end time points of an activity, representing, respectively, the minimum and the maximum duration allowed. At a basic level, the violation of each constraint is immediately given a semantic interpretation: violation of the *minimum* constraint  $c_{min}^{ij}$  between activities  $A_i$  and  $A_j$  (where  $A_i$  is the *SOURCE* activity), directly involves the following interpretation: " $A_j$  is taking place too soon."; similarly, violation of the *maximum* constraint  $c_{max}^{ji}$  between activities  $A_j$  and  $A_i$  (where  $A_i$  is the *SOURCE* activity), enables the possible semantic interpretation: " $A_j$  is being delayed too much.". Duration constraints undergo a slightly different analysis: in fact, a violation of a *duration* constraint on activity  $A_i$  might either entail the violation of the minimum or of the maximum constraints involved. In the first case, we imply the semantics: " $A_i$  was too brief.";

in the second case, “ $A_i$  is lasting too long.”

Given these building blocks for higher level interpretation of the events related to constraint violation, we can employ other kinds of information to improve semantic precision. Again, the meaning of the violation of the constraint  $c_{min}^{ij}$  might take a different interpretation depending on the execution status of activity  $A_i$ . In fact, in case  $A_i$  has not yet been executed, it is easy to see that the violation of  $c_{min}^{ij}$  directly implies that the activities  $A_i$  and  $A_j$  have been temporally swapped, against the caregiver’s prescriptions. Therefore, a general verbalization consistent with the situation is: “*Shouldn’t  $A_i$  be performed first?*”. Obviously, in case  $A_i$  has been executed, a more realistic verbalization is: “*Shouldn’t you wait a little longer before performing  $A_j$ ?*”. Regarding the maximum constraints  $c_{max}^{ji}$ , a different interpretation might be given depending on whether or not activity  $A_i$  is the *SOURCE*. When  $A_i = SOURCE$ , we are describing an *absolute* time limit which involves the start time of  $A_i$ ; in the opposite case, the time constraint is *relative* to the end time of  $A_j$ . This difference affects the verbalization related to  $c_{max}^{ji}$  violation: in the first case, “*Expedite the execution of  $A_i$ .*” might suffice; in the second case, “*Too much time is passing between  $A_i$  and  $A_j$ .*” is more appropriate.

Another source of information that is used to enhance verbalization meaningfulness is related to the causal domain theory. In other words, information about casual links between direct neighbors of the activities involved in a constraint violation can be exploited to deliver explanations.

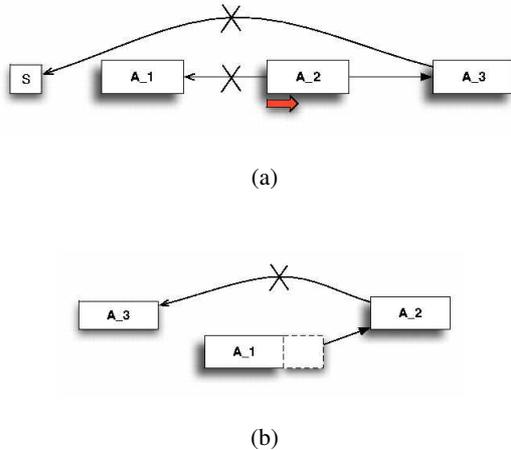


Figure 4: Exploiting causal information for meaningful semantic deduction.

An example is given in Figure 4(a): in this case, the delay on  $A_2$  involves also a delay on  $A_3$ , as both activities are causally linked; as shown, two maximum constraints might be simultaneously violated, and causal link analysis can interpret the situation according to the following semantics: “*Commence  $A_2$ , as it cannot be executed too late with respect to  $A_1$ , and  $A_3$  cannot begin later than a certain time.*”

Figure 4(b) shows another example:  $A_1$  is currently being executed and the causal link between  $A_1$  and  $A_2$  eventually gets the maximum constraint between  $A_2$  and  $A_3$  violated. Again, the deduced verbalization would be: “*Stop  $A_1$ , as  $A_2$  must be immediately executed because it cannot be performed too late with respect to  $A_3$ .*”

### Assessment of system performance

There are two meaningful measures with respect to the performance of the overall system: (1) the *application responsiveness*, i.e., the time  $t_{A_i}$  between a physical change in the environment and the recognition of the change in the input variable(s) of application  $A_i$ ; (2) the *coordination responsiveness*, i.e., the time  $t_{iter}$  it takes for a complete iteration of Algorithm 1 to terminate. The overall responsiveness of the system is therefore represented by  $t_{iter} + \max_{A_i} (t_{A_i})$ . The former measure is affected by the uncertainty connected to the time employed by services to update the symbolic information which they are responsible for deducing. For instance, the People Localization and Tracking service is realized through an artificial stereo-vision algorithm the convergence of which is strongly affected by a variety of factors (for instance, object tracking is difficult when many similar objects move in the same space, when the environment and the agents cannot be adequately structured, and when moving observers are used for monitoring large environments). A similar problem affects also the activity monitor, which must propagate sensor-derived information on the temporal representation of the behavioral constraints. This may require a combination of simple temporal propagation, re-scheduling, as well as other complex procedures (e.g., deciding which of the violated constraints are meaningful with respect to verbal warnings and suggestions). Dually, the responsiveness of coordination can be affected by two factors, namely network load and number of variables involved in the coordination (specifically, the complexity of the ADOPT family of DCOP algorithms is exponential in the number of variables). In practice, coordination responsiveness in the environment is mostly affected by the network load (which becomes high when services such as streaming video are activated). Qualitative observations have shown that application responsiveness can vary between about 1 and 10 seconds. In normal conditions (i.e., if no streaming video service is active) coordination responsiveness varies in the same order of magnitude of application responsiveness. The overall responsiveness of the system (e.g., the delay between an emergency situation occurring and the system activating the contingency plan) varies typically between about 5 and 30 seconds.

For a completely different evaluation that takes into account what real potential users think of the services generated within ROBOCARE, see (Cesta *et al.* 2007).

### Discussion and conclusions

The ROBOCARE project has addressed one of the open challenges in AI, namely that of integrating diversified intelligent capabilities to create a proactive assistant for everyday life in a domestic environment. Our effort to integrate

diverse intelligent components has allowed us to develop a system prototype which is capable not only of passively monitoring the execution of activities, but also of proactively initiating consistent and contextualized advice-giving dialogue.

The individual components of the system as they were available or developed specifically within ROBOCARE were not alone sufficient to obtain a system which could demonstrate intelligent and proactive behavior. The integration of the services provided by the various components through DCOP-based coordination provided the necessary “functional glue” to achieve this.

A key feature of our assistive environment is the ability to intervene proactively and contextually in the assisted person’s daily activity management. Specifically, this is achieved through the use of a continuous schedule monitoring functionality which provides appropriate alerts and warnings through the robotic mediator. The generation of relevant explanations stems from constraint violations from which verbal messages are synthesized and presented to the user. This particular use of the internal knowledge of a constraint-based scheduler is in line with current research in constraint-based explanation. Examples are (Bresina & Morris 2006), which focuses on special strategies to reason on temporal networks to generate explanations of “plan impossibilities”, and (Smith *et al.* 2005), which points out the need for a higher level terminological layer, a “user oriented ontology” connected to the temporal representation that better enables the user to understand conflicts.

While these related results refer to a problem solving task, in ROBOCARE we have used the information on constraint conflicts to tune interaction content for the relatively different task of producing “contextualized dialogues” on daily activities, determining both *when* the system has to interact and *how* to interact with the user (i.e., the content of the verbalization).

In conclusion, we should emphasize that the overall DCOP-based integration schema, thanks to its asynchronous quality, does not impose particular requirements on the types of components which are used to provide domestic services. This opens the possibility of integrating additional, different or enhanced state-of-the-art domotic components within the framework with minimal development overhead.

Being ROBOCARE a relatively small-scale project, our objective was to obtain a proof of concept. Authors are aware that further work is needed to generalize the obtained results. Nevertheless, the examples of behaviors that we are able to obtain with the ROBOCARE intelligent environment pave the way for further developments in the direction of autonomous, context-aware assistive environments.

**Acknowledgements.** This research has been partially supported by MIUR (Italian Ministry of Education, University and Research) under project ROBOCARE: “A MultiAgent System with Intelligent Fixed and Mobile Robotic Component”, L. 449/97 (<http://robocare.istc.cnr.it>). Special thanks to the colleagues of the Dept. of Computer and Systems Science (DIS) of the University of Rome “La Sapienza” for their work on the ROBOCARE intelligent environment.

## References

- Bresina, J., and Morris, P. 2006. Explanations and Recommendations for Temporal Inconsistencies. In *Proceedings of the 5<sup>th</sup> International Workshop on Planning and Scheduling for Space, IWSPSS’06*.
- Cesta, A.; Cortellessa, G.; Giuliani, M.; Pecora, F.; Scopelitti, M.; and Tiberio, L. 2007. Caring About the User’s View: The Joys and Sorrows of Experiments with People. In *ICAPS Workshop on Moving Planning and Scheduling Systems into the Real World, Providence, RI, USA*.
- Haigh, H. Z.; Kiff, L. M.; and Ho, G. 2006. The Independent Lifestyle Assistant (I.L.S.A.): Lessons Learned. *Assistive Technology* 18(1):87–106.
- Levinson, R. 1997. PEAT - The Planning and Execution Assistant and Trainer. *Journal of Head Trauma Rehabilitation*.
- Modi, P.; Shen, W.; Tambe, M.; and Yokoo, M. 2005. Adopt: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence* 161:149–180.
- Pecora, F.; Modi, P.; and Scerri, P. 2006. Reasoning About and Dynamically Posting n-ary Constraints in ADOPT. In *Proceedings of 7th Int. Workshop on Distributed Constraint Reasoning (DCR-06), at AAMAS’06*.
- Pineau, J.; Montemerlo, M.; Pollack, M.; Roy, N.; and Thrun, S. 2003. Towards Robotic Assistants in Nursing Homes: Challenges and Results. *Robotics and Autonomous Systems* 42(3–4):271–281.
- Pollack, M. 2005. Intelligent Technology for an Aging Population: The Use of AI to Assist Elders with Cognitive Impairment. *AI Magazine* 26(2):9–24.
- Smith, S.; Cortellessa, G.; Hildum, D.; and Ohler, C. 2005. Using a scheduling domain ontology to compute user-oriented explanations. In Castillo, L.; Borrajo, D.; Salido, M.; and Oddi, A., eds., *Planning, Scheduling, and Constraint Satisfaction: From Theory to Practice*. IOS Press.