Journal of Ambient Intelligence and Smart Environments 4 (2012) 347–367
DOI 10.3233/AIS-2012-0157
IOS Press

A constraint-based approach for proactive, context-aware human support

Federico Pecora [*,**], Marcello Cirillo, Francesca Dell'Osa, Jonas Ullberg and Alessandro Saffiotti
Center for Applied Autonomous Sensor Systems, Örebro University, 70182 Örebro, Sweden
E-mail: <name>.<surname>@aass.oru.se

Abstract. In this article we address the problem of realizing a service-providing reasoning infrastructure for pro-active human assistance in intelligent environments. We propose SAM, an architecture which leverages temporal knowledge represented as relations in Allen's interval algebra and constraint-based temporal planning techniques. SAM provides two key capabilities for contextualized service provision: human activity recognition and planning for controlling pervasive actuation devices. While drawing inspiration from several state-of-the-art approaches, SAM provides a unique feature which has thus far not been addressed in the literature, namely the seamless integration of these two key capabilities. It does so by leveraging a constraint-based reasoning paradigm whereby both requirements for recognition and for planning/execution are represented as constraints and reasoned upon continuously.

Keywords: Context awareness, activity recognition, continuous planning, temporal reasoning, constraint reasoning

1. Introduction

> *Malin lives alone in her small apartment. With the help of her grandchildren, she has equipped the apartment with a series of service robots, sensors and actuators which help her manage some of the physical and cognitive difficulties she has due to her age. Her home alerts her if she appears to be over-cooking her meals, and autonomously organizes when and where to dispatch her robotic vacuum cleaner so as to minimize its intrusiveness in her daily activities. The home recognizes when Malin is sleeping, eating and performing other usual activities at home, and can be easily set up to monitor and respond to the occurrence of specific patterns of behavior, like getting her a drink from the fridge when she watches TV.*

State of the art robotic and sensor systems can be leveraged to achieve intelligent functionalities that are useful in a number of domains, such as assistive workplaces, or domestic care of the elderly. As suggested by Malin's futuristic home, two important issues underlying the realization of this vision are *context awareness* and *proactiveness*. The former can be achieved today through the use of sensor systems coupled with scene understanding and activity recognition techniques (for instance Hidden Markov Models for activity recognition [42]). Examples of the latter capability have been demonstrated by integrating robotic systems with intelligent control, planning and/or multi-agent coordination techniques (such as Hierarchical Task Network planning for service synthesis [24]).

However, it is increasingly evident that providing services that are effective in supporting human users in real-world situations require both cognitive capabilities concurrently. In order to be effective, these two cognitive processes must operate in unison, informing each other in order to synthesize appropriate, timely and relevant support services. An approach that seamlessly integrates these key capabilities is missing from the current spectrum of techniques.

We propose to use constraint reasoning as the basis for achieving an integration of activity recognition, task planning and execution. Our solution employs a

[*]Corresponding author. E-mail: federico.pecora@oru.se.
[**]This work is partially supported by NovaMedTech project "Ängen" (EU regional funds) and by the Swedish Knowledge Foundation (KK Stiftelsen).

uniform formalism based on Allen's interval algebra to represent both the criteria for context recognition and a planning domain for proactive service enactment. Constraint-based temporal reasoning techniques are used in a closed loop with deployed sensors and actuators to seamlessly interleave context deduction and plan generation/execution.

The architecture, called SAM[1], is conceived to satisfy a number of important requirements stemming from realistic application settings:

- *Modularity:* it should be possible to add new sensors and actuators with minimal reconfiguration effort, and the specific technique employed for context recognition and planning should not be specific to the type of devices employed;
- *Long temporal horizons:* the system must be capable of recognizing patterns of human behavior that depend on events that are separated by long temporal intervals, e.g., recognizing activities that occur every Monday;
- *On-line recognition and execution:* we require the system to be capable of recognizing activities as soon as they occur, a necessity which arises in contexts where the inferred activities should lead to the timely enactment of appropriate procedures;
- *Multiple hypothesis tracking:* finally, the system must be capable of modeling and tracking multiple hypotheses of human behavior, in order to support alternative and/or multiple valid explanations of sensor readings.

The paper is organized as follows. After giving a brief overview of related work, we describe the constraint-based domain representation language that is employed to represent context recognition and actuation requirements in SAM. We then describe how SAM achieves integrated context recognition and proactive support service planning through temporal reasoning. We conclude by presenting several experimental runs of SAM aimed at assessing its adherence to the requirements above, and which include several test runs carried out in a prototypical intelligent environment with human users.

2. Related work

The development of a service-providing reasoning framework relates to a wide range of state of the art-techniques in AIand robotics, which include planning and scheduling, automated reasoning and execution monitoring. Our work explicitly focuses on building a service providing loop around a human user. In this context, specific automated reasoning techniques for human activity recognition on one hand, and "human-aware" planning methods on the other are particularly relevant to our work.

Current approaches to the problem of recognizing human activities can be roughly categorized as *data-driven* or *model-driven*. In data-driven approaches, models of human behavior are acquired from large volumes of data over time. Notable examples of this approach employ Hidden Markov Models (HMMs) in conjunction with learning techniques for inferring transition probabilities [31,42]. Extensions of these approaches have been proposed for dealing with realistic features of the domain, such as interleaved activities [12,28] and multiple persons [35].

Although highly effective in specific domains, such systems are typically brittle to changes in the nature and quantity of sensors, requiring significant retraining when the application context changes. Liao et al. [23] have described a data-driven approach which partially overcomes these limitations using conditional random fields, showing that learned behavior models can be generalized to different users. However, this has been empirically proved only for the specific context of activity recognition using GPS traces and location information, and does not address the problem of contextual recognition and planning/execution. A complementary approach is followed by Helaoui et al. [18] to overcome some of the limitations of purely data-driven techniques. Specifically, the authors incorporate modeling capabilities to capture features such as qualitative temporal relations which describe how events relate to each other. One of the key features of the approach is its capability to recognize interleaved activities. However, it is limited to detecting sensor context, and does not address the issue of actuation.

The lack of integration with actuation is a common problem for data-driven approaches. A notable exception is reported by Boger et al. [3], in which a real-time system recognizes sequences of handwashing-related tasks through vision-based sensors, and assists cognitively impaired users through visual prompts. However, the system relies heavily on a single, ad-hoc (albeit highly optimized) vision-based sensor. Also actuation involves one device (the prompt-generating system), the behavior of which is inferred through a hand-coded Partially Observable Markov Decision Process

[1] SAM stands for "SAM the Activity Manager".

(POMDP) model. Building support for multiple, heterogeneous sensors and actuators into such a system would severely affect the computational complexity of policy generation.

Model-driven approaches to activity recognition follow a complementary strategy in which patterns of observations are modeled from first principles rather than learned or inferred from large quantities of data. Such approaches typically employ an abductive process, whereby sensor data is explained by hypothesizing the occurrence of specific human activities[2]. Examples include work by Goultiaeva and Lespérance [16], where the Situation Calculus is used to specify very rich plans, as well as the work of Pinhanez and Bobick [32], Augusto and Nugent [2], and Jakkula et al. [19], all of which employ rich temporal representations to model the conditions under which patterns of human activities occur. Other techniques used to perform context recognition include ontological reasoning. For instance, Springer and Turhan [39] employ OWL-DL to specify models of complex situations, the argument being that the more complex the situation to recognize, the more sophisticated the behavior of the smart environment. However, the loop with actuation is not closed, and time is considered only implicitly. Riboni and Bettini [34] combine ontological and statistical reasoning to reduce errors in context inference, albeit without addressing temporal relationships between activities.

Data- and model-driven approaches have complementary strengths and weaknesses: the former provide an effective way to recognize elementary activities from large amounts of continuous data – relying, however, on the availability of accurately annotated datasets for training; conversely, model-driven approaches provide a means to easily customize the system to different operational conditions and users through expressive modeling languages – which, though, is based on the ability of a domain modeler to identify criteria for recognition appropriately from first principles. More importantly, the same inference mechanism that is used for model-based activity recognition can be leveraged for plan synthesis. This is particularly so in cases where the expressiveness of the modeling language affords plan-generation and execution monitoring capabilities. Examples of such knowledge representation formalisms are constraint languages such as the restricted Allen's Interval Algebra [1,40], which is extensively used in planning and plan execution monitoring. Examples include several continuous planning systems such as IxTeT [15], OMPS [14], the NASA planning infrastructures [20,22,30] and the T-REX model-based executive [26]. SAM employs similar constraint-based reasoning techniques[3], but towards the aim of combining context inference, plan synthesis and execution monitoring capabilities.

Constraint-based modeling and inference have also been employed to perform schedule execution monitoring for domestic activities. Two notable representatives of this direction are described by Pollack et al. [33] and Cesta et al. [4]. These systems differ from our work in that they employ pre-compiled (albeit highly flexible) schedules as models for human behavior. In the present work, we employ a planning process to actually instantiate such candidate schedules on-line.

SAM is related to the chronicle recognition approach described by Dousson and Le Maigat [11], which also employs temporal reasoning techniques to perform on-line recognition of temporal patterns of sensory events. An approach based on evidence theory augmented with temporal features presented by Mckeever et al. [27] underscores the advantage of explicitly accounting for activity durations. In SAM, however, both recognition and actuation are integrated at the reasoning level as well as being modeled in the same formalism. While both former approaches provide in principle a means to "trigger" events as a result of recognized situations, SAM generates contingent plans whose elements are flexibly constrained to sensory events or recognized activities as they evolve in time.

Several techniques have been proposed for realizing planning systems which exhibit some level of adaptability to human behavior, such as learning techniques for adjusting plan execution parameters in the light of learned models of human behavior and reactions [21]. Other approaches focus on a specific aspect of human-aware planning, e.g., robot motion planning [38] and/or safety [17]. The focus in SAM lies at a higher level of abstraction and involves the contextual and proactive generation of assistive plans. The level of abstraction in SAM is comparable to that of

[2] An approach similar to Shanahan's work [37] on deducing context in a robot's environment.

[3] SAM builds on the OMPS framework, leveraging its constraint representation language and temporal propagation algorithms.

systems such as SHARY [8], in which, however, the emphasis is not on human activity/plan recognition rather on the cooperative achievement of joint goals. Finally, the type of human behavior employed in the work of Cirillo et al. [6,7] to generate human-aware robot plans is structurally very similar to that considered in the present work. However, the above work does not address the issue of inferring human plans.

3. Domain representation

At the center of SAM lies a knowledge representation formalism which is employed to model how human behavior should be inferred as well as how the evolution of human behavior should entail services executed by assistive technology components. Both aspects are modeled through temporal constraints in a domain description language similar to DDL.1 [14,5] and Europa's NDDL [13]. SAM leverages this constraint-based formalism to model the dependencies that exist between sensor readings, the state of the human user, and tasks to be performed in the environment. Domains expressed in this formalism are used to represent both requirements on sensor readings and on devices for actuation, thus allowing the system to infer the state of the user and to contextually synthesize action plans for actuators in the intelligent environment.

The domain description language is grounded on the notion of *state variable*, which models elements of the domain whose state in time is represented by a symbol. State variables are used to represent the parts of the real world that are relevant for SAM's decision process. These include the actuation and sensing capabilities of the physical system as well as the various aspects of human behavior that are meaningful in a specific domain. For instance, a state variable can be used to represent actuators, such as a robot which can navigate the environment and grasp objects, or an automated refrigerator which can open and close its door; similarly, a state variable can represent the interesting states of the human being, e.g., being asleep, cooking, eating, and so on; sensors are represented as state variables whose possible values correspond to the possible sensor readings, e.g., a stove that can be on or off, or room temperature with values in $\{\mathbf{cold}, \mathbf{cool}, \mathbf{warm}, \mathbf{hot}\}$.

The activity recognition and plan synthesis capabilities developed in SAM essentially consist in a procedure for taking *decisions* on state variables.

Definition 1. A decision $d^{\mathrm{x}}_{\mathbf{v}}$ is a triple $\langle \mathrm{x}, \mathbf{v}, [I_s, I_e] \rangle$, where x is a state variable, $\mathbf{v}$ is a possible value of the state variable x, and I_s, I_e represent, respectively, an interval of admissibility of the start and end times of the decision.

A decision therefore describes an assertion on the possible evolutions in time of a state variable. For instance, a decision on the actuated fridge described above could be to open its door no earlier than time instant 30 and no later than time instant 40. In this case, assuming the door takes five seconds to open, the flexible interval is $[I_s = [30, 40], I_e = [34, 44]]$.

SAM supports disjunctive values for state variables, e.g., a decision on a state variable that models a mobile robot could be $\langle \mathrm{Robot}, \mathbf{navigate} \vee \mathbf{grasp}, [I_s, I_e] \rangle$, representing that the robot should be in the process of either navigating or grasping an object during the flexible interval $[I_s, I_e]$.

For the purpose of building SAM, state variables are partitioned into three sets: those that represent observations from *sensors*, those that model the capabilities of *actuators*, and those that represent the various aspects of *human behavior*. This distinction is due to the way in which decisions are imposed on these state variables: decisions on sensors are imposed by *continuous sensing processes* to maintain an updated representation of the evolution of sensor readings as they are acquired through physically instantiated sensors; decisions on state variables modeling human behavior are imposed by SAM's *continuous inference process*, and model the recognized human activities; finally, decisions on actuators are also imposed by the inference process, and represent the tasks to be executed by the physical actuators in response to the inferred behavior of the human. In Section 4 we illustrate how these processes interact towards the aim of achieving a seamless integration of activity recognition and actuation.

3.1. Modeling knowledge for plan synthesis

The core intuition behind SAM's domain theory is the fact that decisions on certain state variables may entail the need to assert decisions on other state variables. For instance, the decision to dock the robot to the fridge may *require* that the fridge door has already been opened. Such dependencies among decisions are captured in a domain theory through what are called *synchronizations*.

Definition 2. A *synchronization* is a tuple $\langle \langle \mathbf{v_{ref}}, \mathrm{x} \rangle, \mathcal{R} \rangle$, where

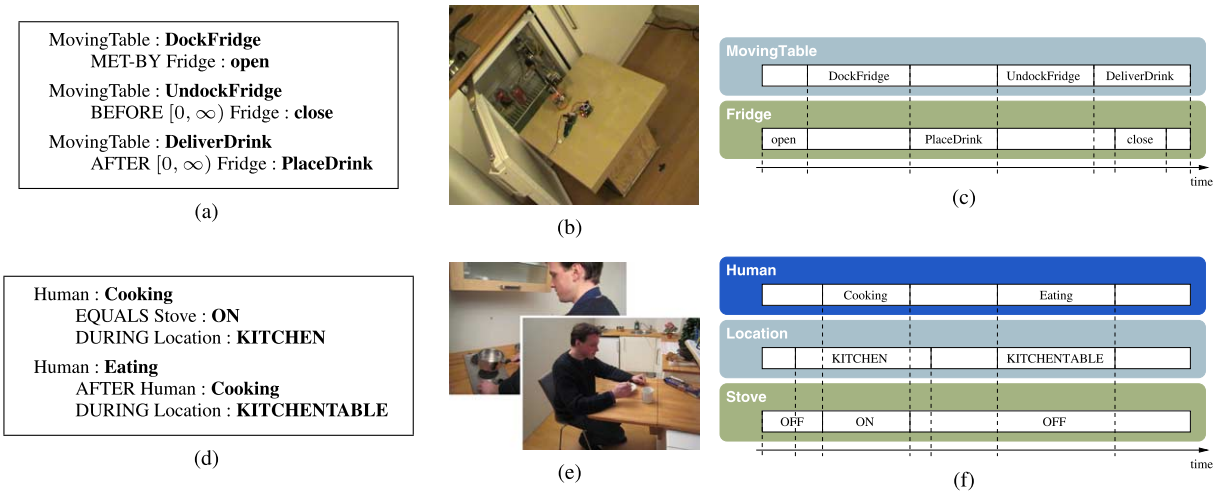

Fig. 1. *Top row:* three synchronizations in a possible domestic robot planning domain (a), the corresponding real actuators available in our intelligent environment (b), and a possible timeline for the corresponding two state variables (c). *Bottom row:* two synchronizations in a possible domestic activity recognition domain (d), the corresponding situations as enacted by a test subject in a test environment (e), and a possible timeline for the three state variables (f).

- $\mathbf{v_{ref}}$ is a *reference value* of a *reference state variable* x;
- $\mathcal{R}$ is a set of *requirements*, each in the form $\langle\langle \mathbf{v_i}, \mathrm{j}\rangle, R_i\rangle$ where
  - $\mathbf{v_i}$ is a value of a state variable j (called a *target value*);
  - R_i is a bounded temporal constraint between the reference value $\mathbf{v_{ref}}$ of state variable x and the target value $\mathbf{v_i}$ of state variable j.

A synchronization describes a set of requirements expressed in the form of temporal constraints. Such constraints are bounded variants of the relations in the restricted Allen's Interval Algebra [1,40]. Specifically, temporal constraints enrich Allen's relations with bounds through which it is possible to fine-tune the relative temporal placement of constrained decisions. For instance, the constraint **A** DURING $[3, 5][0, \infty)$ **B** states that **A** should be temporally contained in **B**, that the start time of **A** must occur between 3 and 5 units of time after the beginning of **B**, and that the end time of **A** should occur some time before the end of **B**.

Figure 1(a) shows an example of how temporal constraints can be used to model requirements among actuators in Malin's home (for syntactic convenience, we write each requirement $\langle\langle \mathbf{v_i}, \mathrm{j}\rangle, R_i\rangle$ using the format R_i j : $\mathbf{v_i}$). The three synchronizations involve two actuators: a robotic table and an intelligent fridge (represented, respectively, by state variables MovingTable and Fridge). The moving table can dock and undock the fridge, and navigate to the human user to deliver a drink. The fridge can open and close its door, as well as grasp a drink inside it and place it on a docked table. The above three synchronizations model three simple requirements of this domain, namely: (1) since the fridge's door cannot open if it is obstructed by the moving table (see Fig. 1(b)), and we would like the door to be kept open only when necessary, docking the fridge must occur directly after the fridge door is opened (MET-BY constraint); (2) for the same reasons, the fridge door should close only after the moving table has completed the undocking procedure (BEFORE constraint); and (3) delivering a drink to the human is possible only after the drink has been placed on the table (AFTER constraint).

Decisions and temporal constraints asserted on state variables are maintained in a *decision network*, that is at all times kept consistent through *temporal propagation*. This ensures that the temporal intervals underlying the decisions are kept consistent with respect to the temporal constraints, while decisions are anchored flexibly in time. In other words, adding a temporal constraint to the decision network will either result in the calculation of updated *bounds* for the intervals I_s, I_e for all decisions, or in a propagation failure, indicating that the added constraint or decision is not admissible. Temporal constraint propagation is based on a Simple Temporal Network [9], and is therefore a polynomial time operation.

SAM provides built-in methods to extract the *timeline* of each state variable in the domain. A timeline represents the behavior of the state variable in time as it is determined by the imposed decisions and constraints in the decision network. Figure 1(c) shows a possible timeline for the two actuators Fridge and MovingTable of the previous example.

3.2. Modeling knowledge for activity recognition

In the previous example, temporal constraints are used to model the requirements that exist between two actuators in carrying out the task of retrieving a drink from the fridge. Conversely, the values of state variables modeling sensors represent *sensor readings* rather than commands to be executed. Consequently, whereas temporal constraints among the values of actuator state variables represent temporal dependencies among commands to be executed that should be upheld in proactive service enactment, temporal constraints among sensors represent temporal dependencies among sensor readings that are the result of specific human activities. For instance, the synchronizations in Fig. 1(d) describe possible conditions under which the human activities of **Cooking** and **Eating** can be inferred (where omitted, temporal bounds are assumed to be $[0, \infty)$). The synchronizations involve three state variables, namely one representing the human inhabitant of the intelligent environment, a state variable representing a stove state sensor (values **ON**, **OFF**), and another state variable representing the location of the human as it is determined by a person localization sensor in the environment [29]. The synchronizations model how the relative occurrence of specific values of these state variables in time can be used as evidence of the human cooking or eating: the former is deduced as a result of the user being located in the **KITCHEN** (DURING constraint) and is temporally equal to the sensed activity of the Stove sensor; similarly, the requirement for asserting the **Eating** activity consists in the human being having already performed the **Cooking** activity (AFTER constraint) and his being seated at the **KITCHENTABLE**.

3.3. Combining recognition and plan synthesis knowledge

A unique feature of SAM is that the same formalism can be employed to express requirements both for enactment and for activity recognition. Adding a contextual plan to a synchronization that models an activity to be recognized equates to adding one or more requirements to this synchronization whose target state variable represents an actuator. For instance, suppose we want to trigger the fan above the stove when cooking has started and make it switch off some amount of time Δ after cooking has finished. This can be accounted for by adding to the **Cooking** synchronization shown above the requirement $\langle\langle \mathbf{ON}, \mathrm{Fan}\rangle, \mathrm{CONTAINS}\,[0,0]\,[\Delta,\Delta]\rangle$.

One of the benefits of this integrated representation is that a single inference algorithm based on temporal constraint reasoning provides a means to concurrently deduce context from sensors and to plan for actuators. As we show in the next section, inference, sensing and actuation operate continuously, each contributing to a common decision network which acts as a dynamically evolving repository of knowledge regarding current sensor readings, inferred activities and actuator status.

4. Recognizing activities and executing proactive services in SAM

SAM employs state variables to model the aspects of the user's context that are of interest. For instance, in the examples that follow we will use a state variable whose values are $\{\mathbf{Cooking}, \mathbf{Eating}, \mathbf{InBed}, \mathbf{WatchingTV}, \mathbf{Out}\}$ to model the human user's domestic activities. For each sensor and actuator state variable, an interface between the real-world sensing and actuation modules and the decision network is provided. For sensors, the interface interprets data obtained from the physical sensors deployed in the intelligent environment and represents this information as decisions and constraints in the decision network. For actuators, the interface triggers the execution on a real actuator of a planned decision.

The decision network acts as a "blackboard" where decisions and constraints re-construct the reality observed by sensor interfaces as well as the current hypotheses on what the human being is doing. These hypotheses are deduced by a continuous re-planning process which attempts to infer new possible states of the human being and any necessary actuator plans.

SAM is implemented as a collection of concurrent processes (described in detail in the following sections), each operating continuously on the decision network:

Sensing processes: each sensor interface is a process that adds decisions and constraints to represent the real-world observations provided by the intelligent environment. There is one such process for each sensor.

Inference process: the current decision network is manipulated by the continuous inference process, which adds decisions and constraints that model the hypotheses on the context of the human and any proactive support operations to be executed by the actuators.

Actuator processes: actuator interfaces ensure that decisions in the decision network that represent operations to be executed are dispatched as commands to the real actuators and that termination of actuation operations are reflected in the decision network as they are observed in reality. There is one such process for each actuator.

Sensing, inference and actuation processes implement a *sense-plan-act* loop which adds decisions and constraints to the decision network in real-time. Access to the decision network is scheduled by an overall process scheduler, and constraint propagation is triggered each time a constraint or decision is added to the decision network.

For simplicity, we assume in the following that the sensor readings never entail the possibility to infer more than one hypothesis on the user's current activity. We will relax this assumption in Section 6, where we tackle the problem of multiple hypothesis tracking.

4.1. Sensing processes

Every sensor in the environment is represented by a state variable whose values model its possible sensor readings. Each sensor interface implements a sensing process for its state variable. If DN_t represents the decision network at time t, running the sensing process at time $t' > t$ for the state variable x will yield a new decision network $DN_{t'} = Sense_{\mathrm{x}}(DN_t, t')$. This function, which is run iteratively at a given rate f, reads the interface of the sensor and models the current sensor reading in the decision network, yielding a network in which the state variable is constrained to take on the sensed values in the appropriate time intervals. Specifically, if a new reading $\mathbf{v}$ is sensed at time t_0 by sensor x, a decision $\langle \mathrm{x}, \mathbf{v}, [I_s, I_e] \rangle$ is added to the decision network with a fixed interval of admissibility $I_s = [t_0, t_0]$ for its start time, and a flexible interval $I_e = [t_0, \infty)$ for its end time which reflects the uncertainty about the reading's temporal evolution. The lower bound on the end time interval is then periodically updated to reflect the incoming sensor readings: if at time $t_i > t_0$ the sensor provides the same reading, then the $Sense_{\mathrm{x}}$ procedure will constrain I_e to $[t_i, \infty)$, reflecting the fact that the sensor value persists at least until the current time t_i; conversely, if the sensor reading changes at time t_i, the interval of admissibility for the end time is fixed to $[t_i, t_i]$, reflecting the knowledge that the sensed value $\mathbf{v}$ persisted in the interval $[[t_0, t_0], [t_i, t_i]]$.

4.2. Continuous inference process

Along with a sensing procedure for each sensor, SAM also runs an iterative inference process, that operates at the same frequency f as the sensing procedures, realizing an on-line sensing-inference loop. The inference procedure employs a domain theory which describes the conditions under which patterns of sensor readings indicate particular values of one or more state variables representing the user. These prerequisites are specified as synchronizations as explained earlier.

The inference procedure continuously attempts to assess the applicability of a set of synchronizations in the decision network. This is done in three steps: (1) adding a decision that represents a hypothesis on the current state of the state variable representing the user; (2) selecting a synchronization whose reference value $\mathbf{v_{ref}}$ matches the current hypothesis; and (3) *expanding* the synchronization's requirements. The first step is performed once for every distinct value of the state variable representing the human. For instance, if the domain contains only the two synchronizations shown in Fig. 1(d), inference would be performed once with the hypothesis $d^{\mathrm{Human}}_{\mathbf{Eating}}$ and once with the hypothesis $d^{\mathrm{Human}}_{\mathbf{Cooking}}$.

The second step identifies one or more synchronizations whose reference value matches the current hypothesis. In the example, assuming the current hypothesis is $d^{\mathrm{Human}}_{\mathbf{Eating}}$, this process would select the second synchronization in Fig. 1(d) as it is the only one whose reference value is **Eating**.

The third step consists in expanding the set of requirements $\mathcal{R} = \{\langle \langle \mathbf{v_i}, \mathrm{j} \rangle, R_i \rangle\}$ of the selected synchronization. This equates to finding a suitable set of target decisions α in the decision network whose values are equal to the values $\mathbf{v_i}$ and imposing the con-

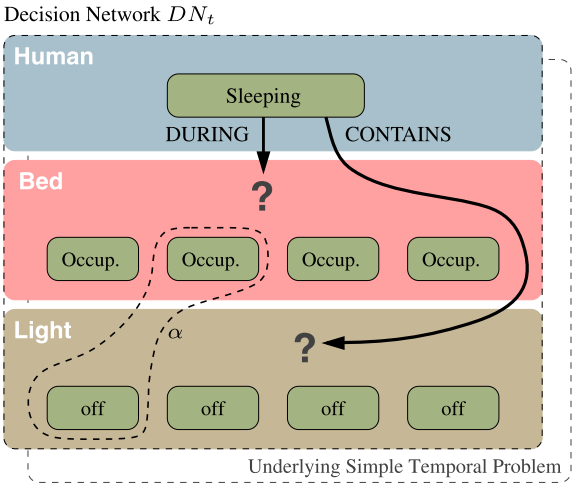

Fig. 2. Support for a hypothesis asserting that the human being is sleeping must be evaluated by applying the required constraints to all possible support sets α, i.e., all possible combinations of decisions on sensory state variables Bed and Light that unify with the requirements.

straints R_i between the reference decision $d^{\text{Human}}_{\mathbf{v_{ref}}}$ and these decisions. In the example, expansion of the **Eating** synchronization will identify all the pairs of decisions $\{d^{\text{Human}}_{\mathbf{Cooking}}, d^{\text{Location}}_{\mathbf{KITCHENTABLE}}\}$ and attempt to constrain them with the current hypothesis $d^{\text{Human}}_{\mathbf{Eating}}$ with the constraints prescribed in the synchronization (i.e., AFTER and DURING, respectively).

If the imposition of the constraints modeled in a selected synchronization does not lead to a propagation failure, then we say that the hypothesis has been *supported*. The procedure can be seen as an operator $Support(DN_t, d^{\text{x}}_{\mathbf{v_{ref}}}, \alpha)$ that returns true if the decision $d^{\text{x}}_{\mathbf{v_{ref}}}$ on state variable x (in our examples, the Human state variable) can be supported in a decision network DN_t using a set of target decisions α, and false otherwise.

In the remainder of this paper, we indicate that there is a one-to-one matching between the values of decisions in a set α and the values of a set of requirements $\mathcal{R}$ with the notation $Unifies(\alpha, \mathcal{R})$.

For each synchronization, the inference procedure must attempt to impose the required constraints between the hypothesis and a number of possible decisions whose values unify with the target values. An example of this is shown in Fig. 2, where support is sought for a candidate decision asserting that the human being is sleeping through one of the sixteen possible combinations of decisions modeling the state of the bed and light sensors in the environment.

4.3. Actuation processes

As explained earlier, synchronizations can model how actions carried out by actuators should be temporally related to recognized activities. For instance, in addition to requiring that **Cooking** should be supported by requirements such as "being in the kitchen" and "using the stove", a requirement involving an actuator can be added, such as "turn on the ventilation over the stove". More in general, for each actuation-capable device in the intelligent environment, an actuator state variable is modeled in the domain. This state variable's values represent the possible actions that can be performed by the device. As sensor interfaces represent real world sensor readings as decisions and constraints in the decision network, actuator interfaces trigger commands to real actuators when decisions involving them appear in the decision network. By stating requirements on actuator state variables in the synchronizations of the domain, entire plans can be generated as a direct result of context inference.

It should be noted that we cannot assume to have strict guarantees on when and for how long given commands will be executed. For this reason, actuator interfaces also possess a sensory capability that is employed to feed information on the status of command execution back into the decision network. As in the case of sensors, actuator interfaces write this information directly into the decision network, thus allowing the re-planning process to take into account the current state of execution of the actions. This executive layer built into the actuator interfaces operates similarly to other well known continuous planning and execution infrastructures (e.g., T-REX [26]). Specifically, the actuator interfaces determine whether or not to issue a command to execute based on the current earliest start times of the decisions in the plan. Similarly to sensor interfaces, actuator interfaces reflect command execution, termination and delays in execution as constraints in the decision network.

In summary, sensing, inference and actuation processes work together to maintain a dynamic constraint network (the decision network) which at all times maintains a consistent representation of (a) observations from the environment, (b) current hypotheses of human behavior deduced from the domain theory, (c) contextually appropriate plans for the service-providing components in the environment, and (d) the current state of execution of these plans. The uniform representation of requirements for recognition and plan synthesis, along with the inference process

as described so far, specifically address one of the four key requirements of realistic application settings, namely modularity. A series of test cases exemplifying modular domain development and concrete usage examples of SAM is provided in Section 8.2. We now turn our attention to addressing the remaining three requirements, namely long temporal horizons and online recognition/execution (Section 5), and multiple hypothesis tracking (Sections 6 and 7).

5. Pruning the search space

As described above, the inference process attempts to support all synchronizations in the domain theory to the current decision network. In the worst case, all possible selections of target decisions need to be explored by the inference process: given a synchronization with requirements $\mathcal{R} = \{R_1 \dots R_{|\mathcal{R}|}\}$ and assuming there are n_i decisions in the decision network which unify with each requirement R_i, then it is necessary to perform $\prod_{i=1}^{|\mathcal{R}|} n_i$ tests. Under the simplified assumption that there are an equal number m of applicable decisions for each requirement, the number of tests to be performed is $O(m^{|\mathcal{R}|})$. Notice also that every attempt to employ a combination of sensed decisions to provide support for a hypothesis will require temporal constraint propagation, which is polynomial in the number of decisions in the decision network. Overall, it is clear that this will not scale well during long-term monitoring since the number of decisions in the decision network grows as time progresses.

In order to reduce the cost of supporting decisions, we introduce the concepts of *flexible* and *fixed* decision.

Definition 3. A decision $d^{\mathrm{x}}_{\mathbf{v}} = \langle \mathrm{x}, \mathbf{v}, [[l_s, u_s], [l_e, u_e]] \rangle$ is *flexible* when $l_s \neq u_s$ or $l_e \neq u_e$. Conversely, if $l_s = u_s$ and $l_e = u_e$ we say that d is *fixed*.

A decision on a state variable modeling a sensor becomes fixed when the state variable's sensing procedure has bounded its end time (i.e., when the sensor reading is no longer being sensed). An important property of such decisions in our approach is that they can no longer provide new information to the inference process. In fact, since constraints on sensor decisions are only tightened, is easy to see that if $Support(DN_t, d^{\mathrm{x}}_{\mathbf{v_{ref}}}, \alpha)$ is false at time t, where α is a set of fixed decisions, then $Support(DN_{t'}, d^{\mathrm{x}}_{\mathbf{v_{ref}}}, \alpha)$ will be false for all $t' > t$. The opposite also holds, i.e., if $Support(DN_t, d^{\mathrm{x}}_{\mathbf{v_{ref}}}, \alpha)$ is true, then $Support$ holds true for the same set of target decisions for all $t' > t$. It should also be noted that if a new decision appears in the decision network (such as a new percept) and it is taken into account for inference, i.e., one of the decisions in α is flexible, then the decision network may indeed be found to support the hypothesis $d^{\mathrm{x}}_{\mathbf{v_{ref}}}$. Overall, sets of fixed decisions represent support that will never affect the inference process in a new way. All decisions on sensory state variables are bound to become fixed, as the $Sense_{\mathrm{y}}(DN_t, t')$ operator for sensor y will eventually set the upper bound of any sensed decision to a fixed time.

Procedure `ActivityRecognition(x,`DN_t`)`

```
1  foreach v ∈ PossibleValues(x) do
2      d^x_v ← ⟨x, v, [[0, ∞), [0, ∞)]⟩
3      foreach Synch. S = ⟨⟨v_ref, x⟩, R⟩ : v_ref = v do
4          Ω ← ∅
5          foreach α ⊆ DN_t : Unifies(α, R) do
6              if ∃ d ∈ α : d is flexible then
7                  Ω ← Ω ∪ α
8          foreach α ∈ Ω do
9              if Support(DN_t, d^x_v, α) then
10                 DN_t ← DN_t ∪ d^x_v
11                 return success
12     return failure
```

As shown in procedure `ActivityRecognition`, fixed decisions can be leveraged for pruning. Specifically, the procedure is applied to a monitored state variable x given a decision network DN_t, and attempts to find support for one of its possible values. Support for a possible value is attempted through all applicable synchronizations in the domain theory (line 3), and only sets of supporting decisions in the decision network that contain at least one flexible decision are considered (lines 5–7). The procedure terminates either when support is found or all synchronizations have been attempted. Note that the `ActivityRecognition` procedure is greedy, in that the first acceptable hypothesis is selected in support of current sensor readings. Also note that in line 10 the decision network is incrementally updated when a hypothesis is confirmed.

As shown in Theorem 1 (see Appendix), the procedure is complete in the sense that if support for a hypothesis exists, the procedure will return success. The added requirement that states that at least one of the target decisions in α should be flexible increases the performance during long term monitoring since

the bulk of the decisions will be fixed (i.e., the number of flexible decisions is bounded by the number of sensors). For example, consider a synchronization that requires two decisions $\mathbf{A}$ and $\mathbf{B}$, and assume that there are seven fixed and three flexible decisions with the value $\mathbf{A}$, and five fixed and two flexible decisions with the value $\mathbf{B}$. The number of support sets α that contain at least one flexible decision are therefore $(3\times2)+(3\times5)+(7\times2) = 35$, while a naïve approach would in this case have to attempt $(7+3)*(5+2) = 70$ combinations.

Without loss of generality, assume that there are an equal number m of decisions that unify with each target value in $\mathcal{R}$. Under this assumption, the decision network contains $|\mathcal{R}| \cdot m$ decisions. As stated in Theorem 2 (see Appendix), requiring that at least one decision in α is flexible reduces the complexity of the activity recognition procedure from $O(m^{|\mathcal{R}|})$ to $O(m^{|\mathcal{R}|-1})$. This entails a significant gain in practical terms, as hypotheses are typically expressed as requirements among few "sensed" values (i.e., $|\mathcal{R}|$ is in the order of two to five), therefore making the gain in computational cost rather noticeable.

5.1. Temporal propagation

While pruning reduces the cost of the `Activity-Recognition` procedure by one order of magnitude, the existence of fixed decisions in the decision network brings with it another advantage which can be leveraged to further increase performance. Specifically, temporal propagation occurs every time $Support(DN_t, d^{\mathrm{x}}_{\mathbf{v_{ref}}}, \alpha)$ is evaluated. This incurs a cost which is polynomial in the number n of decisions in DN_t. Our specific implementation of the underlying temporal propagation is based on the Floyd-Warshall algorithm, whose computational cost is $O(n^3)$ [9]. Notice, however, that since fixed decisions are not flexible in time, they do not need to partake in temporal constraint propagation. This is achieved in our implementation by periodically removing pairs of timepoints corresponding to fixed decisions from the underlying temporal network. Any effect these fixed timepoints have on other (flexible) timepoints is modeled as constraints on these timepoints, thus effectively reducing the number of timepoints over which the Floyd-Warshall propagation procedure iterates (two for every flexible decision).

The exclusion of fixed decisions is the primary method for affecting the complexity of temporal constraint propagation (as Floyd-Warshall is also $\Theta(n^3)$), and when this is done in conjunction with pruning, a significant increase in overall performance can be achieved. In fact, notice that in the worst case, one synchronization requires $m^{|\mathcal{R}|-1}$ propagations, each with a cost of n^3 (due to Floyd-Warshall's temporal propagation). Again under the assumption that there are an equal number m of decisions that unify with each target value in $\mathcal{R}$, the decision network contains $|\mathcal{R}| \cdot m$ decisions. As a consequence, the cost per synchronization would be $O(m^{|\mathcal{R}|+2})$ if inflexible timepoints were never excluded from propagation. Conversely, by excluding fixed decisions from temporal propagation our system periodically curtails this computational load, therefore guaranteeing a cost of $O(m^{|\mathcal{R}|-1})$.

6. Tracking multiple hypotheses

As stated previously, we wish to endow SAM with the ability to track multiple hypotheses of the user's behavior. For instance, it may be possible to discern between having a snack and having a full meal only if certain temporal requirements are met (e.g., the duration should be longer than a certain threshold). This would entail the need to model synchronizations with the same reference value and different requirements. It may also be the case that sensors simply do not provide sufficient information. This would require us to model synchronizations which represent complementary explanations of the sensor readings. Both these cases contrast with the assumption put forth in Section 4.1, i.e., that sensor readings never entail the possibility to infer more than one hypothesis. Overall, we want to avoid having to impose any sort of structural requirement on the domain theory: it should be possible to include multiple synchronizations modeling the same hypothesis with different requirements, as well as synchronizations modeling different hypotheses grounded on the same sensor patterns. Such domains afford greater flexibility and realism, as it is not always possible or desirable to provide a domain theory which maps sensor readings to hypotheses unambiguously.

In the following, we relax both assumptions to allow concurrent or alternative hypotheses on human behavior[4]. Specifically, given a domain theory

$$\mathcal{D} = \{S_1 = \langle\langle \mathbf{v^1_{ref}}, \mathrm{x}_1\rangle, \mathcal{R}_1\rangle, \ldots, S_m = \langle\langle \mathbf{v^1_{ref}}, \mathrm{x}_m\rangle, \mathcal{R}_m\rangle\}$$

[4]Note that we do not support interleaved activities.

we are interested in modeling

- *Multiple explanations:* the same behavior can be supported by different patterns of sensor readings, i.e., $\exists\, S_i, S_j \in \mathcal{D} \mid \mathrm{x}_i = \mathrm{x}_j \wedge \mathbf{v^i_{ref}} = \mathbf{v^j_{ref}} \wedge \mathcal{R}_i \neq \mathcal{R}_j$;
- *Multiple hypotheses:* different behaviors can be supported by non-independent patterns of sensor readings, i.e., $\exists\, S_i, S_j \in \mathcal{D} \mid \mathrm{x}_i = \mathrm{x}_j \wedge \mathbf{v^i_{ref}} \neq \mathbf{v^j_{ref}} \wedge \mathcal{R}_i \cap \mathcal{R}_j \neq \emptyset$,

where $\mathcal{R}_i \cap \mathcal{R}_j \neq \emptyset$ *iff* there exists at least one pattern of sensor readings that supports both $\mathcal{R}_i$ and $\mathcal{R}_j$.

Introducing these possibilities entails two problems. The first problem arises due to the fact that multiple explanations of the same hypothesis may not necessarily co-exist in the decision network. As a simple example, suppose that two synchronizations S_1 and S_2 have the same reference value $\mathbf{v_{ref}}$ on state variable x and different requirements $\mathcal{R}_1$ and $\mathcal{R}_2$ such that $d^{\mathrm{x}}_{\mathbf{v_{ref}}}$ AFTER $d^{\mathrm{sensor}}_{\mathbf{v}} \in \mathcal{R}_1$ and $d^{\mathrm{x}}_{\mathbf{v_{ref}}}$ BEFORE $d^{\mathrm{sensor}}_{\mathbf{v}} \in \mathcal{R}_2$. Clearly no matter what the other constraints in $\mathcal{R}_1$ and $\mathcal{R}_2$ state, these two requirements cannot be imposed at the same time in the decision network. Similarly, it may be impossible to consistently maintain alternative hypotheses in the decision network. This is the case when $\mathcal{R}_i \cap \mathcal{R}_j \neq \emptyset$ but the sensor readings support $\mathcal{R}_j$ in a way that is inconsistent with $\mathcal{R}_i$ (i.e., the sensor readings are such that there exists no solution for either $\mathcal{R}_i$ or $\mathcal{R}_j$ that falls within $\mathcal{R}_i \cap \mathcal{R}_j$).

Accordingly, we modify SAM's abductive procedure to maintain multiple decision networks. Specifically (see Fig. 4), we maintain a decision network DN^{sensors}_t containing only the decisions and constraints that model the history of sensor readings and we update it with current sensor readings as described above. Let $dom(DN)$ be the set of decision networks obtainable as a result of separately applying each synchronization in the domain to the decision network DN. Starting from a decision network representing the current sensor readings at time t_0 ($DN^{\mathrm{sensors}}_{t_0}$), applying the domain theory yields a (possibly empty) set of decision networks. In the figure, three hypotheses are supported by $DN^{\mathrm{sensors}}_{t_0}$, namely $\mathbf{v^1_{ref}}$, $\mathbf{v^2_{ref}}$ and $\mathbf{v^3_{ref}}$. These hypotheses are maintained in decision networks containing all the decisions and constraints that are unique to the application of the hypothesis $d^{\mathrm{x}}_{\mathbf{v^i_{ref}}}$. These networks are obtained as follows:

$$DN^{\mathbf{v^i_{ref}}}_t \leftarrow \left(DN^{\mathrm{sensors}}_t \oplus \left\{d^{\mathrm{x}}_{\mathbf{v^i_{ref}}} \cup \mathcal{R}_i\right\}\right) \setminus DN^{\mathrm{sensors}}_t,$$

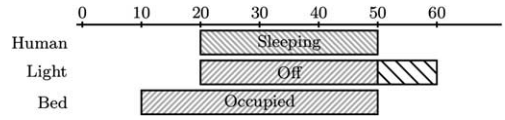

Fig. 3. Timelines relevant to the hypothesis that the human user is sleeping (deduced at time 40 and no longer consistent at time 50).

where the operator $\oplus$ represents the act of adding a set of decisions and propagating a set of constraints. Overall, our model of how human behavior correlates to patterns of sensor readings (expressed in the form of synchronizations) will lead at each time t to a collection of decision networks that contain only the decisions and constraints necessary to describe a hypothesis. Each of these networks is indexed with the time t at which it was calculated, and is used as the basis for computing further hypotheses when sensor readings are updated.

As the computed hypotheses are constrained in time with their support sets, subsequent evolutions of sensor decisions may have invalidated one or more hypotheses. For instance, suppose that the domain contains the synchronization $\langle\langle \mathbf{Sleeping}, \mathrm{Human}\rangle, \mathcal{R}\rangle$ where

$$\mathcal{R} = \{\langle\langle \mathbf{Occupied}, \mathrm{Bed}\rangle, \mathrm{DURING}\rangle, \\ \langle\langle \mathbf{Off}, \mathrm{Light}\rangle, \mathrm{CONTAINS}\rangle\}$$

modeling that a human being is sleeping during an interval of time when the bed sensor is activated and temporally contains a sensor reading indicating that the light is off. Let this hypothesis be inferred at time $t = 40$. Suppose now that at time $t' = 50$ the night light is still off, while the pressure sensor under the bed is deactivated (see Fig. 3). The requirements of the synchronization that supported this hypothesis at time t are no longer consistent at time t' as they induce the constraint $\langle \mathbf{Occupied}, \mathrm{Bed}\rangle$ CONTAINS $\langle \mathbf{Off}, \mathrm{Light}\rangle$, a situation which will not be true henceforth.

Testing whether a hypothesis $DN^{\mathbf{v^i_{ref}}}_t$ computed at time t is still consistent at time $t' > t$ occurs by ascertaining whether the decision network

$$DN^{\mathrm{sensors}}_{t'} \oplus DN^{\mathbf{v^i_{ref}}}_t$$

is consistent. As mentioned, this can be done in cubic time as the underlying temporal constraint network is a STP. Figure 4 shows an example where one of the hypotheses calculated at time t_1 ceases to be consistent when sensors are updated at time t_2.

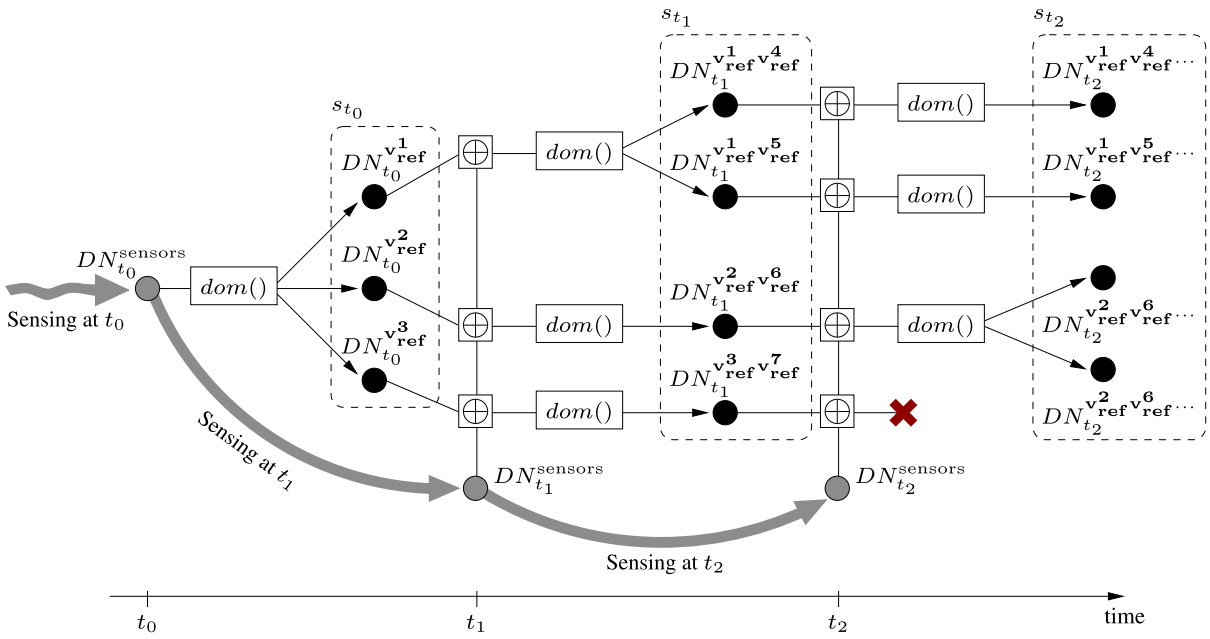

Fig. 4. Evolution in time of multiple hypotheses. $dom()$ represents domain theory application (i.e., inference) as described in procedure `ActivityRecognition`. At every time instant, sensing occurs through sensing processes yielding $DN^{\text{sensors}}_{t_i}$. These decision networks are used to ascertain the continued consistency of previously computed hypotheses through temporal propagation (denoted by $\oplus$).

6.1. Formal properties and the Markov assumption

In SAM, we are interested in tracking all possible evolutions in time of monitored state variables. This entails that dropping a hypothesis when it is found to be invalid cannot be done unless complete information on the cause of the failure is computed. This is because the inconsistency of a constraint network representing, say, hypotheses $\mathbf{v^3_{ref}}$ and $\mathbf{v^7_{ref}}$, does not alone tell us whether the sensors have invalidated the support for $\mathbf{v^3_{ref}}$, for $\mathbf{v^7_{ref}}$, or for both. Reconstructing the cause of the inconsistency can be done by maintaining all decision networks computed at previous time steps, thus ensuring that all inferred hypotheses, including the partial hypotheses, are checked for consistency. In other words, at every node $DN^{\mathbf{V}}_{t_i}$, where $\mathbf{V}$ is a set of hypotheses $\{\mathbf{v^1_{ref}}, \ldots, \mathbf{v^k_{ref}}\}$, it is necessary to check the consistency of all partial hypotheses over which $DN^{\mathbf{V}}_{t_i}$ is built, i.e., all decision networks $DN^{\mathbf{V}'}_{t_i}, \mathbf{V}' \in 2^{\mathbf{V}}$.

Maintaining the power set of hypotheses at each t_i ensures completeness in the sense that if a particular combination of hypotheses is supported at any given time, then the `ActivityRecognition` algorithm will find it. In SAM, this is achieved by considering a "null synchronization" when applying $dom(DN^{\mathbf{v^i_{ref}}}_t)$ which yields $DN^{\mathbf{v^i_{ref}}}_t$ itself. In the example shown in Fig. 4, this would result in the presence of $DN^{\mathbf{v^3_{ref}}}_{t_1}$ (i.e., $DN^{\mathbf{v^3_{ref}}}_{t_0}$ at time t_1), which would thus maintain the possibility of inferring further hypotheses based on $DN^{\mathbf{v^3_{ref}}}_{t_1}$ in the event that $DN^{\mathbf{v^3_{ref}v^7_{ref}}}_{t_1}$ fails at time t_2. Applying the null synchronization ensures that the set of hypotheses s_{t_i} will also contain the power set of all hypotheses inferred thus far. Notice that it is also necessary to discard duplicate hypotheses in order to ensure that s_{t_i} is exactly the power set of all hypotheses computable at time t_i. The number of duplicate hypotheses to discard is linear in the size $|\mathcal{R}|$ of the domain, since in the worst case all synchronizations will be applicable at any point in time.

For the sake of mathematical and computational tractability, many approaches to observation decoding (e.g., the Viterbi Algorithm [41]) make the assumption that future states only depend on the current states. SAM's activity recognition capability differs from most other approaches to activity recognition: these approaches are typically data-driven, and are therefore by nature approximate; conversely, SAM leverages temporal inference in a given model of how sensors relate to activities, and is therefore geared towards complete inference. In addition, most algorithms for ac-

tivity recognition retain the Markov property, namely that state(s) at time t_i result from inference computed only on state(s) computed at t_{i-1}. Also SAM's activity recognition capability can be understood as a Markovian process: the states are the sets of decision networks at time t_i, and computation of the set of consistent hypotheses at the following sensing/inference time point t_{i+1} occurs only based on the state at time t_i. Completeness, however, comes at the cost of an exponentially large state (while in other Markovian activity recognition algorithms the state increases linearly in the number of recognized activities). In the following section, we illustrate how a heuristic based on temporal flexibility can be leveraged to prune the set of hypotheses to maintain while guaranteeing completeness.

6.2. Heuristics and practical applicability

Due to the exponential number of hypotheses to maintain, the practical tractability of multiple hypothesis tracking depends largely on the amount of ambiguity in the domain. The exponential worst case cost rests on the assumption that a significant number of synchronizations becomes applicable at *every* sensing/inference interval, and that *none* of these synchronizations yield a hypothesis that has already been inferred previously. Clearly, this is seldom the case in real scenarios, as it would entail that the observed human is constantly engaging in a new activity every time inference occurs. Thus the computational overhead of maintaining completeness is significantly lower in real-world cases, where humans engage in new activities at a frequency that is much lower than the frequency of sensing/inference.

More importantly though, another observation of practical nature suggests a heuristic for pruning a large number of the partial hypotheses that are maintained over time. Specifically, a recognized partial hypothesis ceases to be a hypothetical conjecture on the state of the monitored human when it is no longer subject to temporal flexibility. In fact, notice that it is necessary to maintain a hypothesis $DN_{t_i}^{\mathbf{v'_{ref}}}$ which has been proved consistent only until another hypothesis $DN_{t_{j>i}}^{\mathbf{v'_{ref}}\mathbf{v''_{ref}}}$ that builds on $DN_{t_i}^{\mathbf{v'_{ref}}}$ becomes fixed. When this occurs, we can safely discard $DN_{t_i}^{\mathbf{v'_{ref}}}$ from the set of hypotheses to maintain at future iterations. This will not hinder completeness, as fixed hypotheses are still maintained so long as no other hypothesis that builds on them becomes fixed. This heuristic makes the system usable in realistic scenarios, as illustrated by the evaluation in Section 8.

Finally, regarding practical applicability, we should note that any sensor noise (e.g., frequent but sparse false-positive readings of a pressure sensor) can potentially affect the performance of the abductive reasoning. It is easy to heed against noise-generated hypotheses by imposing a minimum duration to recognized activities – e.g., imposing a minimum duration of a few minutes for the **Sleeping** activity avoids recognizing it when the light is **Off** and the pressure sensor under the bed erroneously signals **Occupied**. Note, however, that each and every reading would still result in a new pair of timepoints, thus uselessly over-burdening the underlying temporal propagation. For this reason, we avoid hooking noisy sensors directly into the system, rather we use simple techniques to filter sensor readings before they are fed into the constraint network for processing.

7. Multiple hypotheses and actuation

Since context recognition and planning occur concurrently, hypotheses will also contain decisions representing actuator commands. While it is always possible to drop a hypothesis if it ceases to be consistent (as described above), we must ensure that this hypothesis has not lead to irreversible enactment in the real world. One way to do this is to assume that the domain specification also specifies which elements of an actuator's plan can be retracted once started. For instance, it may be admissible to abort the operation of dispatching the moving table to the fridge. However, once the can is on the table it would be necessary to enact a contingent plan to place the can back into the fridge once it has been recognized that delivering it to the user is no longer needed. This option requires adding an extra layer of semantics to the domain definition, and extending the domain with knowledge about how to react to failed plans.

Another way of avoiding irreversible change in the real world is to adopt a conservative approach, i.e., to start execution of actuator plans only in response to confirmed hypotheses. In this case, SAM's inference process must be provided the means to prove when there exists no further possibility of failure of a hypothesis. In other words, we must devise a strategy to ensure the future consistency of hypotheses.

In SAM, we adopt the latter approach. Specifically, we employ a sufficient condition to guarantee

that hypotheses cannot be disproved in the future. As seen earlier, such a sufficient condition should indicate when $Support(DN_t, d^{\mathrm{x}}_{\mathbf{v_{ref}}}, \alpha)$ will hold for all possible $DN^{\mathrm{sensors}}_{t'>t}$. For each hypothesis containing actuation plans, we procrastinate the plan's decisions until the sufficient condition holds.

An obvious sufficient condition for guaranteeing hypothesis consistency is the total absence of temporal flexibility in the sensor readings supporting the hypothesis. In fact, if all sensor decisions underlying a hypothesis are fixed (recall Definition 3), then it is clearly not possible to invalidate the hypothesis, as the temporal dependencies underlying it will never change. However, this entails that actuation will occur only once hypotheses cease to evolve in time, e.g., delivering a drink to the user only once he/she has stopped watching TV. In order to circumvent this shortcoming, we state a less restrictive sufficient condition which provides criteria for ruling out the possibility of hazardous induced constraints in the network (see Theorem 3 in the Appendix). Operationally, assessing whether the condition holds equates to performing a simple analysis of the worst case constraining power of future sensor readings on the support set of a hypothesis. The condition can be verified before the sensor readings underlying a hypothesis cease to evolve, therefore making it possible to determine whether support for a hypothesis will persist in the future. As it is a sufficient condition, it guarantees the correctness of our approach to multiple hypothesis tracking with actuation, in the sense that any deduced contextual plan is valid with respect to the domain theory. In SAM, each successful application of the $Support$ procedure (line 10 in `ActivityRecognition`) is followed by a test ascertaining whether the above sufficient condition holds. If the added hypothesis together with its supporting set fails the sufficient condition, any actuation decisions in the hypothesis are delayed, thus guaranteeing that states that may not be supported in the future are never committed to.

The sufficient condition stated in Theorem 3 can be checked in polynomial time with respect to the number of decisions in a hypothesis' support set α. Enforcing this test will effectively delay actuation until no doubt exists as to the future consistency of the hypothesis.

Finally, although the sufficient condition allows to prove future consistency earlier than a naïve approach, it is not always the case that such temporal inconsistencies can be proved early enough for execution. This is the case in the sleeping example shown earlier, where the hypothesis **Sleeping** can be proved to be consistent only when the light is turned back on (signaling that the decision $\langle \mathrm{Light}, \mathbf{Off}, [I_s, I_e] \rangle$ has ceased to evolve in time). In such cases, there is no way to avoid modeling appropriate reaction strategies into the domain theory.

8. Evaluation

As mentioned, the realization of SAM is motivated by the need for modularity, long temporal horizons, on-line recognition, and multiple hypothesis tracking. This section seeks to evaluate how well these requirements are met by the current implementation of SAM. We start by addressing the issue of performance, which directly affects the second and third requirement. We then present a series of tests in a physical smart home testbed environment aimed at assessing the ability to deploy SAM in incrementally rich environments and its capability to deal with a realistic scenario in which multiple hypotheses are tracked.

8.1. Performance

The `ActivityRecognition` procedure provides a means to achieve the required performance as it reduces the cost of supporting a hypothesis with a synchronization from $O(m^{|\mathcal{R}|})$ to $O(m^{|\mathcal{R}|-1})$. This effectively means that finding support for a decision through a synchronization with two requirements can be done in linear time with respect to the number of applicable target decisions in the decision network.

In order to assess whether the performance increase obtained as a result of the admissible pruning can support long-term monitoring scenarios, we compare the performance of two implementations of our system, one employing no optimization, and one which prunes the search space as shown earlier. All tests described in this section were carried out on an Intel Core2 Duo processor @ 2.33 GHz.

First, we experimentally verify the bounds on complexity shown earlier by performing two tests with a domain theory containing only one synchronization. In the first test, the domain contains one synchronization stating that a value $\mathbf{v_{ref}}$ should be recognized if it occurs DURING value $\mathbf{A}$ (on one sensory state variable) and should CONTAIN value $\mathbf{B}$ (on another sensory state variable). The sensory input shown in Fig. 5 (top) was continuously fed to the systems (the first 10 seconds of sensory input

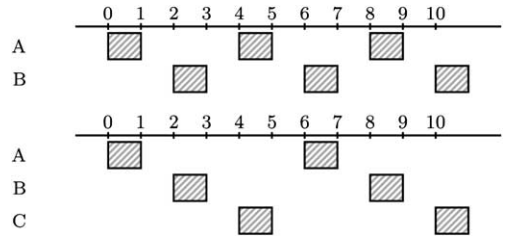

Fig. 5. Recurring patterns used in the performance tests in Fig. 6 (top and bottom, respectively; first 10 time instants shown).

are shown in the figure). Notice that the number of combinations of support decisions that need to be explored – i.e., the number of sets α used to attempt $Support(DN_t, \langle x, \mathbf{v_{ref}}, [0,\infty)[0,\infty)\rangle, \alpha)$ – constantly increases over time. Also, notice that the sensor readings occur in a repeating pattern such that no combination of targets can act as support for $\mathbf{v_{ref}}$. This situation represents the worst case, as all combinations of supporting decisions must be attempted in order to conclude that $\mathbf{v_{ref}}$ cannot be supported. Figure 6 (top) compares the CPU time required by the `ActivityRecognition` procedure in the two systems. As shown, when pruning is employed, the performance of the system grows linearly with the number of sensory events, while in the absence of pruning we obtain a quadratic increase in CPU time.

Figure 6 (bottom) shows the second test, where a similar synchronization that has three requirements was used. As for the first test, the input for the second test (Fig. 5, bottom) feeds sensor readings that never support the reference value $\mathbf{v_{ref}}$, thus yielding a quadratic increase in CPU time with pruning, as opposed to cubic complexity without pruning.

Our last experiment aims to assess how the system performs in a more realistic scenario, where a domain theory containing ten synchronizations models meaningful activities of daily living. The human activities, each depending on at most $|\mathcal{R}| = 3$ sensor readings, include the previously described **Sleeping** activity, as well as several other activities such as **Cooking**, **WatchingTV** and **HavingLunch**. Six sensor state variables were employed, and the sensory input was simulated in such a way that it would constitute feasible input from a real world scenario (e.g., the location state variable providing the position of the human being was fed realistic movements of a human being in a topologically correct model of a small apartment).

Figure 7 shows the performance of the system obtained over a monitoring horizon of one week. The test serves as an experimental proof of the fact that it is

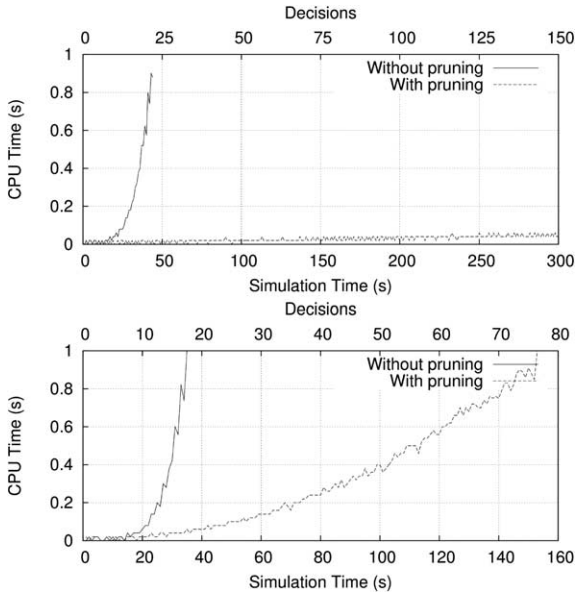

Fig. 6. CPU time required to prove lack of support for a hypothesis using a synchronization with two (top), and three (bottom) requirements.

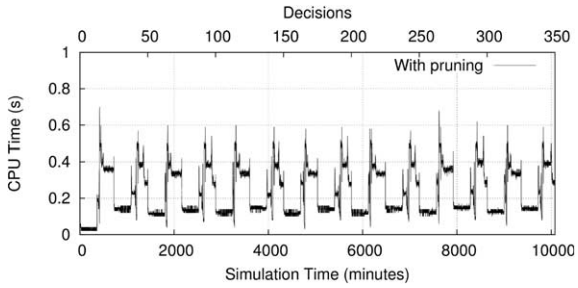

Fig. 7. CPU time required to recognize human activities once each minute during a one-week long scenario.

possible to use the current system to recognize activities over realistically-scaled periods of time. Activity recognition and sensing occurs once per minute. The criteria for choosing this rate is related to the resolution that is necessary in the specific monitoring scenario. Clearly, this rate has to be low enough to allow the `ActivityRecognition` process to terminate, but also high enough to guarantee that meaningful variations of sensor readings are detected. In the present test, variations of sensor readings occurring over intervals of one minute or longer are sufficient for detecting all meaningful activities. Notice that it is possible to continuously recognize activities at a frequency of approximately 1.4 Hz throughout the entire scenario, as the maximum CPU time of the `ActivityRecognition` procedure is about 0.7

seconds. This would enable the system to recognize activities whose temporal granularity is finer than one minute. Notice also that the computational load of inference increases once per day, as shown by the corresponding fourteen peaks in Fig. 7. This is due to the increased number of possible support sets during periods of high activity (e.g., activities carried out during the day as opposed to the relative inactivity during the night).

At the end of the week, the decision network contained close to 350 decisions (a combination of sensor readings and the recognized activities), with an average production of approximately 50 decisions per day. As can be seen, the system scales well with the growing number of decisions, which indicates that recognizing activities at a greater level of detail (e.g., activities with a shorter time-span that occur more often, such as relocating etc.) is not believed to cause any performance problems as long as the number of decisions are reasonable. For instance, we can assume that the current system can detect activities with a similar level of detail during ten weeks while maintaining an adequate performance for most domestic applications, but not necessarily for an entire year.

8.2. *Incremental case studies in the* PEIS-HOME

We illustrate the use of real-world sensors in SAM with four runs performed in the PEIS-HOME, a prototypical intelligent environment deployed at Örebro University[5]. The environment provides ubiquitous sensing and actuation devices, including the robotic table [10] and intelligent fridge described in earlier examples.

In the first run our aim is to assess the sleep quality of a person by employing three physical sensors: a pressure sensor, placed beneath the bed, a luminosity sensor placed close to the night light, and a person tracker based on stereo vision. We then define a domain with three sensors and the two synchronizations shown in Fig. 8. Note that the human user is modeled by means of two distinct state variables, Human and HumanAbstract. This allows us to reason at different levels of abstraction on the user: while the decisions taken on state variable Human are always a direct consequence of sensor readings, synchronizations on values of HumanAbstract describe knowledge that can be inferred from sensor data as well as previously

[5]See http://aass.oru.se/~peis and [36].

1) Human : **InBed**
DURING Location : **NOPOS**
EQUALS Bed : **ON**
2) HumanAbstract : **Awake**
DURING Human : **InBed**
EQUALS NightLight : **ON**

Fig. 8. Synchronizations defined in our domain for the Human and HumanAbstract state variables to assess quality of sleep.

recognized Human and HumanAbstract activities. The first synchronization models two requirements for recognizing that the user has gone to bed: first, the user should not be observable by the tracking system, since the bedroom is a private area of the apartment and, therefore, outside the field of view of the cameras; second, the pressure sensor beneath the bed should be activated. The resulting **InBed** decision has a duration EQUAL to the one of the positive reading of the bed sensor. The second synchronization grasps the situation in which, although lying in bed, the user is not sleeping. The decision **Awake** on the state variable HumanAbstract depends therefore on the decision **InBed** of the Human and on the sensor readings of NightLight.

This simple domain was employed to test SAM in our intelligent home environment with a human subject. The overall duration of the experiment was 500 seconds, with the concurrent inference and sensing processes operating at a rate of about 1 Hz. Figure 11(a) is a snapshot of the five state variables' timelines at the end of the run (from top to bottom, the three sensors and the two state variables modeling the human).

The outcome of a more complex example is shown in Fig. 11(b). In this case the scenario contains four instantiated sensors. Our goal is to determine the afternoon activities of the user living in the apartment, detecting activities like **Cooking**, **Eating** and the more abstract **Lunch**. To realize this example, we define five new synchronizations (Fig. 9), three for the Human state variable and two for the HumanAbstract state variable. Synchronization (3) identifies the human activity **Cooking**: the user should be in the kitchen and its duration is EQUAL to the activation of the Stove. Synchronization (5) models the **Eating** activity, using both the Location sensor and an RFID reader placed beneath the kitchen table (state variable KTRfid). A number of objects have been tagged to be recognized by the reader, among which dishes whose presence on the table is required to assert the decision **Eating**. Syn-

1) HumanAbstract : **Lunch**
STARTED-BY Human : **Cooking**
FINISHED-BY Human : **Eating**
DURING Time : **afternoon**
2) HumanAbstract : **Nap**
AFTER HumanAbstract : **Lunch**
EQUALS Human : **WatchingTV**
3) Human : **Cooking**
DURING Location : **KITCHEN**
EQUALS Stove : **ON**
4) Human : **WatchingTV**
EQUALS Location : **COUCH**
5) Human : **Eating**
DURING Location : **KITCHENTABLE**
EQUALS KTRfid : **DISH**

Fig. 9. Synchronizations modeling afternoon activities of the human user.

1) Human : **WatchingTV**
EQUALS Location : **COUCH**
START MovingTable : **DeliverDrink**
2) MovingTable : **DockFridge**
MET-BY Fridge : **OpenDoor**
3) MovingTable : **DeliverDrink**
AFTER Fridge : **PlaceDrink**
4) MovingTable : **UndockFridge**
BEFORE Fridge : **CloseDoor**
5) Fridge : **PlaceDrink**
MET-BY MovingTable : **DockFridge**
MEETS MovingTable : **UndockFridge**
6) OpenDoor : **OpenDoor**
MET-BY Fridge : **GraspDrink**

Fig. 10. Synchronizations defining temporal relations between human activities and proactive services.

chronization (4) correlates the presence of the user on the couch with the activity of **WatchingTV**.

Synchronizations (1) and (2) work at a higher level of abstraction. The decisions asserted on HumanAbstract are inferred from sensor readings (Time), from the Human state variable and from the HumanAbstract state variable itself. This way we can identify complex activities such as **Lunch**, which encompasses both **Cooking** and the subsequent **Eating**, and we can capture the fact that after lunch the user, sitting in front of the TV, will most probably fall asleep.

Also this example was executed in the PEIS-HOME. It is worth mentioning that the decision corresponding to the **Lunch** activity on the HumanAbstract state variable was identified only when both **Cooking** and **Eating** were asserted on the Human state variable. Also it can be noted that **Nap** is identified as the current HumanAbstract activity only after the lunch is over and that on the first occurrence of **WatchingTV**, **Nap** was not asserted because it lacked support from the **Lunch** activity.

As an example of how the domain can include actuation as synchronization requirements on monitored state variables, let us consider the following run of SAM in a setup which includes the robotic table and autonomous fridge devices described earlier.

As shown in Fig. 10, we use abductive reasoning to infer when the user is watching TV. In this case, however, we modify the synchronization (4) presented in Fig. 9 to include the actuators in the loop. The new synchronization (Fig. 10, (1)), not only recognizes the **WatchingTV** activity, but also asserts the decision **DeliverDrink** on the MovingTable state variable. This decision can be supported only if it comes AFTER another decision, namely **PlaceDrink** on state variable Fridge (synchronization (3)). When SAM's replanning procedure attempts to support **WatchingTV**, synchronization (5) is called into play, stating that **PlaceDrink** should occur right after (MET-BY) the MovingTable has docked the Fridge and right before the undocking maneuver (MEETS). The remaining three synchronizations – (2), (4) and (6) – are attempted to complete the chain of support, that is, the Fridge should first grasp the drink with its robotic arm, then open the door before the MovingTable is allowed to dock to it, and finally it should close the door right after the MovingTable has left the docking position.

This chain of synchronizations leads to the presence in the decision network of a plan to retrieve a drink from the fridge and deliver it to the human who is watching TV. Notice that when the planned decisions on the actuator state variables are first added to the decision network, their duration is minimal. The actuator processes update these durations at every re-planning period until the devices that are executing the tasks signal that execution has completed. Also, thanks to the continuous propagation of the constraints underlying the plan, decisions are appropriately delayed until their earliest start time coincides with the current time (see Section 4.3). A complete run of this scenario was performed in our intelligent environment and a snapshot of the final timelines is shown in Fig. 11(c).

8.3. A test run for multiple hypothesis tracking

In order to gauge the performance of SAM under realistic conditions involving multiple hypothesis tracking, we performed a prolonged experimental run in the PEIS-HOME with a human user and an ambiguous domain description. The run involved eight of the sensors

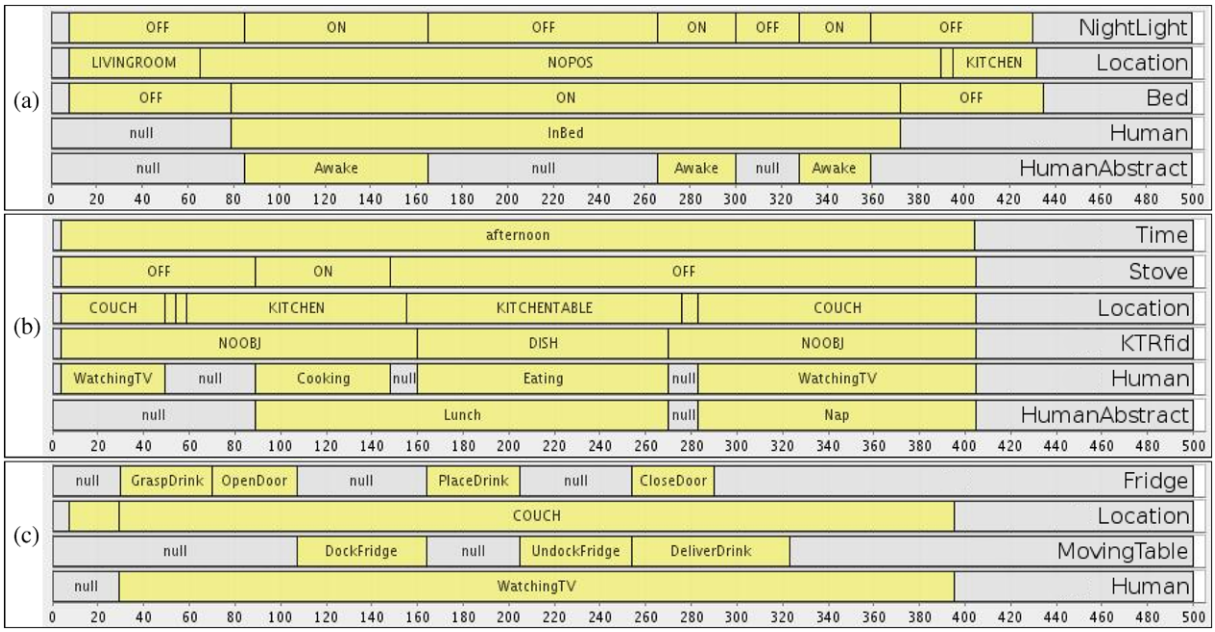

Fig. 11. Timelines resulting from the runs performed in our intelligent home using the sleep monitoring (a), afternoon activities (b) and proactive service (c) domains.

Fig. 12. The system supports two non-mutually exclusive hypotheses: watching TV and having a snack.

available in the home, and monitoring occurred over a time-span of four hours. The domain theory used by the inference procedure, which describes the requirements for recognizing twelve activities in total, contains ambiguous synchronizations: the same activity can be explained in alternative ways and the same sensor readings can support multiple hypotheses. The human subject performed a series of actions that can be associated with a normal working day.

The system kept track of each hypothesis that could be inferred from the domain theory and the sensor readings – the inferred states in one particular moment of the run are shown in the inset of Fig. 12. All the activities were recognized correctly, and the number of possible hypotheses during the experiment varied between 0 and 4. The four hour run operated at an average of 0.45 Hz (performing more than 6400 inference iterations), with an average computation time of about 2 seconds.

The good performance of the system obtained in the experimental run (in the light of the exponential complexity of maintaining multiple hypotheses) is due to two factors: (1) the fact that the worst case scenario implied by the assumptions made in the complexity analysis are extremely unrealistic, and (2) the high number of temporally inflexible hypotheses that are pruned by the heuristic. In order to assess the effect of the heuristic on the performance of the system, a further run subject to realistic human behavior was performed in which fixed hypotheses were maintained. The cost of inference became unmanageable after approximately 100 iterations (less than four minutes), thus indicating that temporal flexibility analysis provides a good criteria for proving the relevance of previously computed hypotheses.

9. Conclusions

In this article we have presented SAM, an architecture for concurrent activity recognition, planning and execution. The architecture leverages a constraint-based approach and a modular domain description language to realize a proactive activity monitor which operates in a closed loop with physical sensing and actuation components in an intelligent environment.

SAM employs concepts drawn from constraint-based planning and execution frameworks in conjunction with efficient temporal reasoning techniques for activity recognition. By blending these techniques SAM introduces a key novelty, namely a single architecture that integrates recognition and planning/execution abilities. These two aspects of activity management are uniformly represented in a single constraint-based formalism, reasoned upon by the same inference mechanism, and anchored to the real world through specialized interfaces with physical sensors and actuators.

SAM is built to satisfy four key requirements stemming from realistic application settings, namely modularity, the ability to operate over long temporal horizons, on-line recognition and execution and multiple hypothesis tracking. In this article, we have demonstrated the feasibility of the approach with a number of artificial worst case scenarios and experimental runs in a real environment with a human user.

As the performance of the system ultimately depends on STP propagation cost as well as the number of support sets to be attempted, future work will consider different algorithms to solve the STP in order to fully evaluate the performance gain of the current approach (e.g., [43]).

Another important direction we will investigate is the integration of our approach with data-driven methods in order deal with sensor noise and possibly provide on-line model training and adaptation. Our system currently relies on hand-coded domain knowledge – we will investigate how to relax this assumption through data-driven model learning, as done in existing approaches [18,34].

Dealing with uncertainty is also an important avenue of investigation. We have begun to explore the issue through fuzzy temporal constraint propagation [25].

Appendix: Formal Properties

Theorem 1 (Completeness)**.** *If recognition is carried out every time new sensory information is obtained, i.e.,* `ActivityRecognition`$(\mathrm{x}, DN_{t'})$ always follows $Sense_Y(DN_t, t')$, where Y is a set of sensors that are updated at time t', then the `ActivityRecognition` procedure is complete.

Proof. We are guaranteed that if a set α of decisions cannot support a hypothesis $d^{\mathrm{x}}_{\mathbf{v_{ref}}}$ at time t, this set need only be attempted in subsequent calls to `ActivityRecognition` as a subset of a support set $\alpha \cup \alpha'$ where α' contains at least one flexible decision. A decision d^{y} on a sensor state variable $\mathrm{y} \in Y$ can become fixed only as a consequence of the sensing procedure $Sense_{\mathrm{y}}(DN_t, t')$. Also, when a sensor signals a new sensed value, this is modeled as a decision with unbounded end-time. If the `ActivityRecognition` procedure is always applied after a sensor update, we are thus guaranteed that no sensory decision will become fixed without having previously been employed in an attempt to support a hypothesis. In other words, every set of decisions containing at least one flexible decision will be considered for support, thus proving completeness. □

Theorem 2. *Given a synchronization* $\langle\langle \mathbf{v_{ref}}, \mathrm{x}\rangle, \mathcal{R}\rangle$*, let* DN_t *be a decision network containing* m *decisions that unify with each target value* $\mathbf{v_i}$ *in* $\mathcal{R}$*. The cost of searching for a set* α *such that* $Unifies(\alpha, \mathcal{R})$ *and* $Support(DN_t, d^x_{hyp}, \alpha)$ *holds (lines 4–11 in the* `ActivityRecognition` *procedure) is* $O(m^{|\mathcal{R}|-1})$*.*

Proof. If we do not prune sets of decisions that are all fixed, a synchronization will require

$$\prod_{i=1}^{|\mathcal{R}|}(Fx_i + Fl_i)$$

combinations to be tried, where $|\mathcal{R}|$ is the number of requirements for the synchronization, and Fx_i and Fl_i are, respectively, the number of fixed and flexible decisions that the ith target value in $\mathcal{R}$ can unify against. Conversely, if sets of all fixed decisions are discarded, it is only necessary to attempt

$$\prod_{i=1}^{|\mathcal{R}|}(Fx_i + Fl_i) - \prod_{i=1}^{|\mathcal{R}|}(Fx_i)$$

different sets of decisions. Due to the binomial theorem, and assuming without loss of generality that each sensory state variable has an equal number of flexible

and fixed decisions that unify with each target value, i.e., $Fx_i = Fx$ and $Fl_i = Fl$, we obtain:

$$\begin{aligned}&(Fx + Fl)^{|\mathcal{R}|} - Fx^{|\mathcal{R}|}\\ &= \sum_{i=0}^{|\mathcal{R}|} \binom{|\mathcal{R}|}{i} Fx^i Fl^{|\mathcal{R}|-i} - Fx^{|\mathcal{R}|}\\ &= \sum_{i=0}^{|\mathcal{R}|-1} \binom{|\mathcal{R}|}{i} Fx^i Fl^{|\mathcal{R}|-i},\end{aligned}$$

where the maximum power of Fx is $|\mathcal{R}| - 1$, thus decreasing the cost per synchronization by one order of magnitude. □

Theorem 3 (Correctness)**.** *Let $d^{\mathrm{x}}_{\mathbf{v_{ref}}}$ be a hypothesis such that $Support(DN_t, d^{\mathrm{x}}_{\mathbf{v_{ref}}}, \alpha)$ holds. The following conditions are sufficient for guaranteeing that $Support(DN_{t'}, d^x_{hyp}, \alpha)$ will hold for every $t' > t$:*

- *for every pair $d_i, d_j \in \alpha$, imposing the constraint d_i DEADLINE $[l_i, l_i]$ does not change the bounds of the end time of d_j, where l_i is the lower bound of the end time of d_i;*
- *for every pair $d_i, d_j \in \alpha$, imposing the constraint d_i DEADLINE $[u_i, u_i]$ does not change the bounds of the end time of d_j, where u_i is the upper bound of the end time of d_i.*

Proof. Given that the underlying temporal problem is a Simple Temporal Problem (as all constraints represent simple intervals defining the distance between decisions' start and end times), the temporal relations induced by the network on any two time points can be modeled as a simple distance constraint between the two time points [9]. Assume that at time t we consider two decisions d_i and d_j of the support set α, and that the constraint induced by the temporal network on the end times t^i_e, t^j_e of these decisions imposes $t^j_e - t^i_e \geqslant l$ and $t^j_e - t^i_e \leqslant u$. We at this point attempt two tests. First, we constrain the interval of admissibility of t^i_e to its upper bound. The imposition of this further constraint may or may not affect the lower bound of t^j_e. Whether it does so depends on the positive slack l allowed by the induced constraint. Second, we constrain the interval of admissibility of t^i_e to its lower bound, in which case the slack allowance u will determine whether this further constraint affects the upper bound of t^j_e. It is easy to see that the forward and backward slack allowed by network, i.e., l and u, is maximally exhausted by imposing one of the two constraints above on the interval of admissibility of t^i_e, and that no other constraint on t^i_e has more power to reduce the interval of admissibility of t^j_e.

The above observation, together with the fact that the $Sense$ function will only constrain the end times of the decisions in α more as time goes by, proves that the above tests are sufficient for guaranteeing that sensor decisions in a support set α can end at any time in the future without introducing inconsistencies. In essence, testing the impact of constraining end times of decisions in α on the end times of each other decision in α as described above will expose any indirect dependencies introduced by activity recognition. □

References

[1] J.F. Allen, Towards a general theory of action and time, *Artificial Intelligence* **23**(2) (1984), 123–154.

[2] J.C. Augusto and C.D. Nugent, The use of temporal reasoning and management of complex events in smart homes, in: *Proc. of the 16th Eureopean Conference on Artificial Intelligence (ECAI)*, 2004.

[3] J. Boger, P. Poupart, J. Hoey, C. Boutilier, G. Fernie, and A. Mihailidis, A decision-theoretic approach to task assistance for persons with dementia, in: *Proc. of the 19th International Joint Conference on Artificial Intelligence (IJCAI)*, 2005.

[4] A. Cesta, G. Cortellessa, R. Rasconi, F. Pecora, M. Scopelliti, and L. Tiberio, Monitoring elderly people with the robocare domestic environment: Interaction synthesis and user evaluation, *Computational Intelligence,* Special Issue on Scheduling and Planning Applications **27**(1) (2011), 60–82.

[5] A. Cesta and A. Oddi, DDL.1: A formal description of a constraint representation language for physical domains, in: *New Directions in AI Planning*, 1996, pp. 341–352.

[6] M. Cirillo, L. Karlsson, and A. Saffiotti, A human-aware robot task planner, in: *Proc. of the 19th International Conference on Automated Planning and Scheduling (ICAPS)*, 2009.

[7] M. Cirillo, L. Karlsson, and A. Saffiotti, Human-aware task planning: An application to mobile robots, *ACM Transactions on Intelligent Systems and Technology*, Special Issue on Applications of Automated Planning **15** (2010), 1–26.

[8] A. Clodic, H. Cao, S. Alili, V. Montreuil, R. Alami, and R. Chatila, SHARY: A supervision system adapted to human-robot interaction, in: *Experimental Robotics*, O. Khatib, V. Kumar and G. Pappas, eds, Tracts in Advanced Robotics, Vol. 54, Springer, 2009, pp. 229–238.

[9] R. Dechter, I. Meiri, and J. Pearl, Temporal constraint networks, *Artificial Intelligence* **49**(1-3) (1991), 61–95.

[10] E. Di Lello, A. Loutfi, F. Pecora, and A. Saffiotti, Robotic furniture in a smart environment: The peis table, in: *Proc. of the 4th International Workshop on Artificial Intelligence Techniques for Ambient Intelligence (AITAmI)*, 2009.

[11] C. Dousson and P. Le Maigat, Chronicle recognition improvement using temporal focusing and hierarchization, in: *Proc. of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, 2007.

[12] T.V. Duong, H.H. Bui, D.Q. Phung, and S. Venkatesh, Activity recognition and abnormality detection with the switching hidden semi-Markov model, in: *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[13] EUROPA, Europa software distribution web site, https://babelfish.arc.nasa.gov/trac/europa, 2008.

[14] S. Fratini, F. Pecora, and A. Cesta, Unifying planning and scheduling as timelines in a component-based perspective, *Archives of Control Sciences* **18**(2) (2008), 231–271.

[15] M. Ghallab and H. Laruelle, Representation and Control in IxTeT, a Temporal Planner, in: *Proc. of the 2nd International Conference on Artificial Intelligence Planning and Scheduling (AIPS)*, 1994.

[16] A. Goultiaeva and Y. Lespérance, Incremental plan recognition in an agent programming framework, in: *Working Notes of the AAAI Workshop on Plan, Activity, and Intention Recognition (PAIR)*, 2007.

[17] B. Graf, M. Hans, and R. Schraft, Mobile robotassistants: Issues for dependable operation in direct cooperation with humans, *IEEE Robotics and Automation Magazine* **2**(11) (2004), 67–77.

[18] R. Helaoui, M. Niepert, and H. Stuckenschmidt, Recognizing interleaved and concurrent activities: A statistical-relational approach, in: *Proc. of the IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 2011.

[19] V. Jakkula, D.J. Cook, and A.S. Crandall, Temporal pattern discovery for anomaly detection in a smart home, in: *Proc. of the 3rd IET Conference on Intelligent Environments (IE)*, 2007.

[20] A.K. Jónsson, P.H. Morris, N. Muscettola, K. Rajan, and B. Smith, Planning in interplanetary space: Theory and practice, in: *Proc. of the International Conference on Artificial Intelligence Planning and Scheduling (AIPS)*, 2000.

[21] A. Kirsch, T. Kruse, and L. Mösenlechner, An integrated planning and learning framework for human-robot interaction, in: *Proc. of the 4th Workshop on Planning and Plan Execution for Real-World Systems (at ICAPS)*, 2009.

[22] R. Knight, G. Rabideau, S. Chien, B. Engelhardt, and R. Sherwood, Casper: Space exploration through continuous planning, *IEEE Intelligent Systems* **16**(5) (2001), 70–75.

[23] L. Liao, D. Fox, and H. Kautz, Extracting places and activities from gps traces using hierarchical conditional random fields, *Robotics Research* **26**(1) (2007), 119–134.

[24] R. Lundh, L. Karlsson, and A. Saffiotti, Autonomous functional configuration of a network robot system, *Robotics and Autonomous Systems* **56**(10) (October 2008), 819–830.

[25] M. Mansouri, Constraint-based activity recognition with uncertainty, Master's thesis, Örebro University, School of Science and Technology, 2011.

[26] C. McGann, F. Py, K. Rajan, J. Ryan, and R. Henthorn, Adaptive control for autonomous underwater vehicles, in: *Proc. of the 23rd National Conference on Artificial Intelligence (AAAI)*, 2008.

[27] S. Mckeever, J. Ye, L. Coyle, C. Bleakley, and S. Dobson, Activity recognition using temporal evidence theory, *Ambient Intelligence and Smart Environments* **2**(3) (2010), 253–269.

[28] J. Modayil, T. Bai, and H. Kautz, Improving the recognition of interleaved activities, in: *Proc. of the 10th International Conference on Ubiquitous Computing (UbiComp)*, 2008.

[29] R. Muñoz Salinas, E. Aguirre, and M. García-Silvente, People detection and tracking using stereo vision and color, *Image Vision Computing* **25**(6) (2007), 995–1007.

[30] N. Muscettola, G.A. Dorais, C. Fry, R. Levinson, and C. Plaunt, IDEA: Planning at the core of autonomous reactive agents, in: *Proc. of the 3rd International NASA Workshop on Planning and Scheduling for Space*, 2002.

[31] D.J. Patterson, D. Fox, H. Kautz, and M. Philipose, Fine-grained activity recognition by aggregating abstract object usage, in: *Proc. of the 9th IEEE International Symposium on Wearable Computers,* 2005.

[32] C. Pinhanez and A. Bobick, Fast constraint propagation on specialized allen networks and its application to action recognition and control, Technical Report 456, M.I.T. Media Lab, Perceptual Computing Section, 1998.

[33] M.E. Pollack, L. Brown, D. Colbry, C.E. McCarthy, C. Orosz, B. Peintner, S. Ramakrishnan, and I. Tsamardinos, Autominder: An intelligent cognitive orthotic system for people with memory impairment, *Robotics and Autonomous Systems* **44**(3-4) (2003), 273–282.

[34] D. Riboni and C. Bettini, Context-aware activity recognition through a combination of ontological and statistical reasoning, in: *Ubiquitous Intelligence and Computing*, D. Zhang, M. Portmann, A.-H. Tan and J. Indulska, eds, Lecture Notes in Computer Science, Vol. 5585, 2009, pp. 39–53.

[35] S. Geetika, D.J. Cook, and M. Schmitter-Edgecombe, Recognizing independent and joint activities among multiple residents in smart environments, *Ambient Intelligence and Humanized Computing* **1**(1) (2010), 57–63.

[36] A. Saffiotti, M. Broxvall, M. Gritti, K. LeBlanc, R. Lundh, J. Rashid, B.S. Seo, and Y.J. Cho, The PEIS-ecology project: Vision and results, in: *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2008.

[37] M.P. Shanahan, Robotics and the common sense informatic situation, in: *Proc. of the 12th European Conference on Artificial Intelligence (ECAI)*, 1996.

[38] E.A. Sisbot, L.F. Marin-Urias, R. Alami, and T. Siméon, A human aware mobile robot motion planner, *IEEE Transactions on Robotics* **23**(5) (2007), 874–883.

[39] T. Springer and A.-Y. Turhan, Employing description logics in ambient intelligence for modeling and reasoning about complex situations, *Ambient Intelligence and Smart Environments* **1**(3) (2009), 235–259.

[40] M. Vilain, H. Kautz, and P. van Beek, Constraint propagation algorithms for temporal reasoning: A revised report, in: *Readings in Qualitative Reasoning about Physical Systems,* D.S. Weld and J. de Kleer, eds, 1989, pp. 373–381.

[41] A. Viterbi, Error bounds for convolutional codes and an asymptotically optimum decoding algorithm, *IEEE Transactions on Information Theory* **13**(2) (April 1967), 260–269.

[42] J. Wu, A. Osuntogun, T. Choudhury, M. Philipose, and J. Rehg, A scalable approach to activity recognition based on object use, in: *Proc. of the 11th IEEE International Conference on Computer Vision (ICCV)*, 2007.

[43] L. Xu and B.Y. Choueiry, A new efficient algorithm for solving the simple temporal problem, in: *Proc. of the International Symposium on Temporal Representation and Reasoning*, 2003.