A Logical Framework for Behaviour Reasoning and Assistance in a Smart Home

Liming Chen, Chris Nugent, Maurice Mulvenna, Dewar Finlay, Xin Hong, Michael Poland

School of Computing and Mathematics and Computer Science Research Institute,
University of Ulster, Northern Ireland
Tel: 0044-28-90368837, Fax: 0044-28-90366216
{l.chen, cd.nugent, md.mulvenna, d.finlay, x.hong, m.poland}@ulster.ac.uk

Abstract- Smart Homes (SH) have emerged as a realistic intelligent assistive environment capable of providing assistive living for the elderly and the disabled. Nevertheless, it still remains a challenge to assist the inhabitants of a SH in performing the "right" action(s) at the "right" time in the "right" place. To address this challenge, this paper introduces a novel logical framework for cognitive behavioural modelling, reasoning and assistance based on a highly developed logical theory of actions - the Event Calculus. Cognitive models go beyond data-centric behavioural models in that they govern an inhabitant's behaviour by reasoning about its knowledge, actions and environmental events. In our work we outline the theoretical foundation of such an approach and describe cognitive modelling of SH. We discuss the reasoning capabilities and algorithms of the cognitive SH model and present the details of the various tasks it can support. A system architecture is proposed to illustrate the use of the framework in facilitating assistive living. We demonstrate the perceived effectiveness of the approach through presentation of its operation in the context of a real world daily activity scenario.

Index Terms – Event calculus, cognitive modelling, behaviour reasoning, smart homes, assistive living.

1. Introduction

Smart Homes (SH) have emerged as one of the mainstream approaches to support technology-driven independent living for elderly and disabled persons [1] [2] [3]. Extensive research has been undertaken on the underpinning technologies of SH such as sensor networks [4], data communication and fusion, standards, middleware, intelligent processing [5] and proof-of-concept prototypes [6] [7]. Whilst significant progress has been made in the aforementioned areas, it remains difficult, however, to assist a SH inhabitant to perform the "right" action(s) at the "right" time in the "right" place. As such, there is an increasing demand for novel approaches and associated methods to enable complex, formal action modelling and reasoning for the assistance in the completion of Activities of Daily Living (ADL). This is becoming particularly true for those suffering from cognitive deficiencies such as Alzheimer's disease [8] [9].

We contend that with the underpinning technologies in place it is time to direct research emphasis more towards the approaches and mechanisms for high-level behavioural assistance. In this paper we propose a novel framework for cognitive modelling, event reasoning and behaviour assistance within a SH. Cognitive models can be considered to go beyond current implementations of data-centric and reactive models currently in place in SH in that they govern what an inhabitant knows, how that knowledge is acquired, and how it can be used to predict and/or infer the ensuing actions. The theoretical foundation upon which our work has been based is the highly developed logical theory of actions, i.e., the Event Calculus (EC) [10] [11] [12]. EC describes complex, dynamically changing worlds in sorted first-order logic and has the ability of also incorporating a temporal dimension in the description process. For example with the proposed logical framework hazard prevention can be mapped to a deductive task such as temporal projection and providing assistance with ADL completion can be mapped to temporal explanation and/or planning. In the following Sections we describe cognitive modelling and reasoning mechanisms in further detail and based on these concepts we conceive a system architecture to support the implementation of the proposed framework. We consider the application of this approach by applying the concepts within the scenario of the ADL of "making a cup of tea" to demonstrate its usage.

The paper is organised as follows: Section 2 discusses related work in the area of SH. Section 3 introduces the theoretical foundations of the proposed framework. Section 4 describes logical formalization, cognitive modelling and reasoning for a SH. We elaborate upon logic-based methods for behaviour assistance and present a system architecture in Section 5. We present an example use scenario in Section 6 and Section 7 concludes the paper and outlines our visions for future work.

2. Related Work

The provision of general assistance and support in the completion of ADLs within SH has undergone many studies. The fundamental differences between these approaches relate to the way in which the activity and an inhabitant's profiles are modelled and represented. One approach is centred on data-intensive probabilistic

methods, which use probability theories such as Markovian models [13] [14], Bayesian networks [15] and the Dempster-Shafer theory [16], for ADL modelling and incorporate the inhabitant's preferences by tuning the initial values of the parameters of the probabilistic models. With these approaches the basic idea is to specify a set of possible activity patterns (plans) and define a stochastic model that can update the likelihood of occurrence of each pattern according to dynamic observations and to the known state of the system. A major strength with probabilistic behaviour recognition methods is that they can capture the fact that certain plans are, a priori, more likely than others. Hence, the probabilities are used to model the uncertainty related to the recognition task directly in the plan likelihood. Nevertheless, the major disadvantage with these approaches is that the model is largely static and subjective in probabilistic variable configuration.

An alternative approach in the management and processing of the data within SH concentrates on those techniques which adopt a machine learning paradigm to identify and extract ADL patterns from observed activities, and at some later point in time use the patterns as predictive models [17] [18]. Some of these learning techniques are also based on probabilistic approaches [17]. Instead of using a pre-established stochastic model to update the likelihood of the activity pattern, learning probabilistic methods keep a trace of their previous observed experiences and use them as the basis to dynamically learn the parameters of the stochastic model, in order to create a predictive model based on the observed agent's habits. While this approach increases system adaptation, it suffers from data sparsity and diminished inter-object applicability

A third alternative strand of research focuses on logical approaches that view behaviour recognition and reasoning as plan recognition [19] [20] [21]. The basic idea is to use logical formalisms to formalise and create domain theories including axioms and a plan (i.e. activity pattern) library. Observed actions can then be interpreted against the domain theory through various reasoning tasks that provide explanations and/or predictions for the observed agent's behaviour. Central to these techniques is the way that plans are interpreted in order to derive desirable conclusions based on observed actions. Some approaches, as in [19], consider all derived plans inferred as equi-probable. Others as in [20] attempt to define a partial relation organizing the plan's elements in terms of level of plausibility but are too complex and computationally expensive to make operational in a real context. These previously explored approaches also fail to address uncertainty and nondeterministic choice of actions – a situation that can often occur in the situation of a person within a SH.

In terms of the three aforementioned approaches our work belongs to the latter, i.e., a logical approach that uses event theory for representing ADL models, and exploits deduction or abduction for explanation and predication. The event theory is based on sorted first order logic with the desirable feature of temporal handling that is not possible in other approaches. The compelling feature of our approach can be viewed as the concept of compound actions. It provides a flexible way of specifying a behaviour pattern or "sketch plan". The novelty is that "sketch plans" do not need to specify the full details of a plan but are only required to provide an outline of the pattern. Behaviour reasoning here amounts to recursive hierarchical compound event decomposition in which event details of a "sketch plan" can be automatically generated in a progressive way. This empowers our approach to deal with uncertainty and nondeterministic choice of actions. Component events also provide an effective way of incorporating domain heuristics and details of inhabitant profiles. As such, event patterns (plans) can be constructed that significantly reduce the event search space, and can be interpreted in a way that best match an inhabitant's behaviour. Our approach is not dependent on the assumption of an inhabitants' rationality, but based on real-world use cases which offer pragmatic and practical solutions in a more intuitive and natural way.

3. Theoretical Foundation

The proposed logical framework is based on event calculus - a formalism stemming from the domain of Artificial Intelligence (AI) used for representing and reasoning with events and their effects. It was initially introduced by Kowalski [10], and later extended by a number of different researchers [11] [12]. Our work has been based on the concepts and proposals as presented in [12] and is expressed in sorted first-order predicate calculus with circumscription.

3.1 The Ontology and Predicates

The core ingredients of a logical formalism are ontologies, predicates and axioms. The ontology is used to specify the types of things in a problem domain over which quantification is permitted. The basic ontology of the EC comprises events (or actions), *fluents* and time points. Events are the fundamental instrument of change in our ontology. All changes to a world are the result of named events. A possible world history is simply a sequence of events that happen at different points in time.

Any property of a world that can change over time is known as a *fluent*. A *fluent* is a function of the time point. If a fluent represents a relation and it takes on values from a boolean number system, i.e. true or false, then it is called a relational *fluent*. Otherwise if a *fluent's* denotations vary from one time point to another time point, then it is called a functional *fluent*.

Predicates define relations between entities that specify what happens when, what *fluents* hold at what times, and describe the initial situation and the effects of events. Table 1 displays the EC predicates together with their explanations. For example, **Initiates**(α, β, τ) specifies the effects of an event, i.e., the *fluent* β starts to hold after action α at time τ. **Happens**(α, $\tau1$, $\tau2$) specifies the occurrence of an event, i.e., the action α starts at time $\tau1$ and ends at time $\tau2$. The EC can

handle time-related concepts in these predicates, thus making it particularly suitable for SH where events are often time sensitive..

Based on the causal relations of the predicates in the EC we can derive a set of axioms. Axioms state how and when the truth holds based on causal relations of predicates. Table 2 lists some EC axioms. For instance, the first axiom asserts that a *fluent* f holds at time *t* if it held at time 0, and has not been terminated between time 0 and time *t*. Axioms serve as inference rules in reasoning.

Table 1 Examples of EC Predicates

Initiates(α, β, τ) - Fluent β starts to hold after action α at time τ
Terminates(α, β, τ) - Fluent β ceases to hold after action α at time τ
Releases(α, β, τ) - Fluent β is not subject to inertia after α at time τ]
InitiallyP(β) - Fluent β hold from time 0
InitiallyN (β) - Fluent β does not hold from time 0
Happens(α, τ1, τ2) - Action α starts at time τ1 and ends at time τ2]
τ1 < τ2 - Time point τ1 is before time point τ2]
HoldAt(β, τ) - Fluent β holds at time τ]
Clipped(τ1, β, τ2) - Fluent β is terminated between times τ1 and τ2
Declipped(τ1, β, τ2) - Fluent β is initiated between times τ1 and τ2
Cancels(α1, α2, β) - The occurrence of α1 cancels the effect of a simultaneous occurrence of α2 on fluent β
Cancelled(α, β, τ1, τ2) - Other concurrent event occurs from time τ1 to time τ2 that cancels the effect of action α on fluent β
Trajectory(β1, τ, β2, δ) - If fluent β1 is initiated at time τ then fluent β2 becomes true at time τ + δ

Note:
α, β and τ denote an action, a fluent and time point respectively.
The terms action and event are used interchangeably in this paper.

3.2 The Frame and Ramification Problems

The frame problem is that of trying to infer what remains unchanged by an action. It is overcome through circumscription. That is, all events that may happen are modelled. There are no unexpected event occurrences and actions have no unexpected effects. This is similar to the 'closed world' assumption. In most cases the formula describing event occurrences and action effects will be conjunctions of Horn clauses, and the circumscription will reduce to predicate completions.

The ramification problem is the frame problem for actions with indirect effects, i.e., actions with effects beyond those described explicitly by their associated effect axioms. This can be overcome through state constraints – refer to [11] for details.

3.3 Compound Actions

The previous sections outlined the basics of the EC for representing and reasoning about simple actions. Nevertheless, it failed to address the problem of expressing and reasoning about compound actions such as "**If** action α is successful **then** take action β **else** take action δ", "**While** there is a fire alarm **do** run outside and call the fire services**"**. The EC introduces the concept of compound actions to denote actions that are composed of other actions. To facilitate the construction of compound actions, some additional extra-logical symbols, such as **||, ;, ?, while** and **if** that act as abbreviations of control structures for logical expressions, have been developed. Table 3 gives the definition and semantics of these extra-logical control structures.

Table 2 Examples of EC Axioms

$\text{HoldAt}(\beta, \tau) \leftarrow \text{InitiallyP}(\beta) \wedge \neg \text{Clipped}(0, \beta, \tau)$ (EC1)

$\text{HoldAt}(\beta, \tau3) \leftarrow \text{Happens}(\alpha, \tau1, \tau2) \wedge \text{Initiates}(\alpha, \beta, \tau1) \wedge \neg \text{Cancelled}(\alpha, \beta, \tau1, \tau2) \wedge \tau2 < \tau3 \wedge \neg \text{Clipped}(\tau1, \beta, \tau3)$ (EC2)

$\text{Clipped}(\tau1, \beta, \tau4) \leftrightarrow \exists \alpha, \tau2, \tau3[\text{Happens}(\alpha, \tau2, \tau3) \wedge \tau1 < \tau3 \wedge \tau2 < \tau4 \wedge [\text{Terminates}(\alpha, \beta, \tau2) \vee \text{Releases}(\alpha, \beta, \tau2)] \wedge \neg \text{Cancelled}(\alpha, \beta, \tau2, \tau3)]$ (EC3)

$\neg \text{HoldAt}(\beta, \tau) \leftarrow \text{InitiallyN}(\beta) \wedge \neg \text{Declipped}(0, \beta, \tau)$ (EC4)

$\neg \text{HoldAt}(\beta, \tau3) \leftarrow \text{Happens}(\alpha, \tau1, \tau2) \wedge \text{Terminates}(\alpha, \beta, \tau1) \wedge \neg \text{Cancelled}(\alpha, \beta, \tau1, \tau2) \wedge \tau2 < \tau3 \wedge \neg \text{Declipped}(\tau1, \beta, \tau3)$ (EC5)

$\text{Declipped}(\tau1, \beta, \tau4) \leftrightarrow \exists \alpha, \tau2, \tau3[\text{Happens}(\alpha, \tau2, \tau3) \wedge \tau1 < \tau3 \wedge \tau2 < \tau4 \wedge [\text{Initiates}(\alpha, \beta, \tau2) \vee \text{Releases}(\alpha, \beta, \tau2)] \wedge \neg \text{Cancelled}(\alpha, \beta, \tau2, \tau3)]$ (EC6)

$\text{Happens}(\alpha, \tau1, \tau2) \rightarrow \tau1 \leq \tau2$ (EC7)

$\text{Cancelled}(\alpha1, \beta, \tau1, \tau2) \leftrightarrow \text{Happens}(\alpha2, \tau1, \tau2) \wedge \text{Cancels}(\alpha2, \alpha1, \beta)$ (EC8)

$\text{HoldAt}(\beta2, \tau3) \leftarrow \text{Happens}(\alpha, \tau1, \tau2) \wedge \text{Initiates}(\alpha, \beta1, \tau1) \wedge \neg \text{Cancelled}(\alpha, \beta, \tau1, \tau2) \wedge \tau2 < \tau3 \wedge \tau3 = \tau2+d \wedge \text{Trajectory}(\beta1, \tau1, \beta2, d) \wedge \neg \text{Clipped}(\tau1, \beta1, \tau3)$ (EC9)

Note:
α, β and τ denote an action, a fluent and time point respectively.

Consider that time points are regarded as real values and the corresponding comparative predicates and arithmetic functions are taken for granted. Compound actions can then be represented as a combination of primitive actions and other compound actions in terms of these extra-logical symbols. For convenience compound actions are usually denoted as procedures, which are actually macros of primitive actions and other compound actions. As such, a program can be viewed as a top-level procedure that consists of combinations of sequences and/or parallel declarations of procedures and primitive actions. During execution, the program expands recursively into genuine formulae of the EC that can be instantiated and reasoned over.

4. Behaviour Modelling and Reasoning

The central idea of the proposed framework in the current work is to (1) use the EC as the representation medium for the domain knowledge specification of SH, and (2) exploit the reasoning capabilities of EC such as deduction, abduction and induction as a means of

computation for hazard prevention, prediction and behaviour assistance. This is discussed in details in the following Sections.

Table 3 Extra-logical Control Structures

Sequence α;β - do action α, followed by action β.
Nondeterministic choice of actions α | β - do action α, or action β.
Concurrency α || β - actions α and β occur concurrently.
Iteration while *p* do α end - do α while *p* is true.
Conditionals if *p* then α else β - do α if *p* is true, otherwise do β.
Test *p*? - true if formula *p* holds, otherwise false.
Nondeterministic choice of arguments - (πx) α(x) - pick some argument *x* and perform the action α(*x*).
Non-deterministic iteration α^* - do α zero or more times.
Iteration - while *p* do α end - do α while *p* is true.
Concurrency with different priorities α >> β - α has higher priority than β, and β may only be executed when α is done or blocked.
Interrupt $<\vec{X} : \phi \rightarrow \alpha>$ - if the interrupt gets control and the condition formula ϕ is true for some binding of the variable vectors $\vec{X}$, the interrupt triggers and the action body is executed with the variables taking these values. Once the actions finish, the interrupt may trigger again.
Procedures **Proc** $P(X_1, X_2, \ldots X_n)$ α **endproc** declares a compound action that uses $P(X_1, X_2, \ldots X_n)$ as procedure calls. In the macro expansion, the actual parameters (x_i) are first evaluated with respect to the current time point τ ($x_i[\tau]$) before passing them to the procedure *P*, so the procedure mechanism we are defining is call-by-value.
Note: α, β and τ denote an action, a fluent and time point respectively. *p*, ϕ denote a logical formula. *x* is a variable denoting an event.

4.1 SH Conceptualisation and Formalisation

A SH is a dynamic environment in which an inhabitant performs ADL by interacting with different types of domestic devices and appliances. Sensors monitor the environment and collect corresponding data reflecting the events which the inhabitant causes. The temporal unfolding of ADL in a SH can be conceptually projected as a situation tree, refer to Figure 1, where *e* and *s* refer to an event and a situation, respectively. The node of the tree denotes a situation or goal that can be viewed as a "snapshot" of states of the SH at any specific point in time *t*. The links between nodes are events that evolves a SH from one situation to another. At the root of the tree is the initial situation S_{t_0}, and each path through the tree represents a possible sequence of events leading to a specific situation (goal).

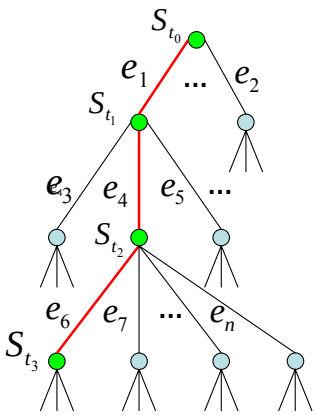

Fig. 1 A Fragment of a Situation Tree

An assistive system such as a software agent requires an explicit representation of a SH so that it acquires the states of the SH, know an inhabitant's whereabouts or behaviour, and further uses domain knowledge for behaviour prediction, inference and assistance. In the EC, we represent any property related to the state of a SH by a *fluent* and specify its value through a state axiom. For example, at_location(*x*) is a *fluent* representing the location *x* of an inhabitant. The formulae Hold(at_location(*kichen*), t_0) denotes that an inhabitant is in the place *kitchen* at time t_0. The EC assumes that any change to a property state is brought about by an event(s), initiated by either an inhabitant or the SH environment. A change is modelled by an effect formula and an event occurrence formula. For instance, goto(*x*) is an action of going to location *x* with its corresponding occurrence formula Happen(goto(*x*), t_2, t_3) and effect formulae Initiate(goto(*x*), at_location(*x*), t_3). These formula state that an inhabitant starts to go to location *x* at time t_2 and finishes the action at time t_3, and by then the inhabitant is in the location *kitchen*.

A key step in cognitive modelling is to identify and specify the domain ontologies and to further formalise their relationships using the EC predicates. This will lead to a complete **domain theory**, i.e. cognitive model, which typically comprises the following aspects.

a) A set of declarations, specifying uniqueness-of-names for events and *fluents*, state constraints, and effect and causal constraints.

b) A set of Initiates, Terminates and Releases formulae describing the effects of the primitive, low-level events.

The occurrences and effects of compound actions can be formalised in the same way as with primitive actions. Therefore, the **domain theory** of the cognitive model will contain all compound actions and a set of Initiates, Terminates and Releases formulae describing the effects of all compound events.

c) A set of Happens formulae defining high-level compound actions in terms of more primitive ones. These definitions are constructed using extra-

logical control structures such as sequence, choice and recursion. A compound action is actually an activity pattern that incorporates domain heuristics and commonsense knowledge.

d) A set of Initiates, Terminates and Releases formulae describing the effects of the compound, high-level events.

4.2 Situation Sensing

Events in a SH are dynamic and unpredictable. To accommodate this and provide timely responses, an assistive living system needs to perform time-critical assessment of the environment in order to capture the up-to-date status of a situation. Through the use of EC we model the sensing of an environment as a knowledge-production action which does not have any effect on the SH, however, produces the desired effect of changing the knowledge base of the SH's assistive system. We introduce a generic epistemic fluent *Knows* to represent acquired knowledge [22]. The *Know fluent* specifies the state of an assistive system's knowledge and has exactly the same status as other *fluents*. The formula HoldAt(Know(ϕ), τ) represents that the assistive system has the knowledge expressed in formula ϕ. Using this epistemic *fluent*, we can formalize the knowledge producing effects of any actions just like formalizing any other *fluents*. For example, the effect formula **Initiate**(lookAt(ovenTimer), Know(timeForCooking), *t*) represents that the cooking time in the oven can be acquired by a knowledge producing action "looking at the oven timer".

4.3 Behaviour Reasoning

The EC can support different types of reasoning tasks as shown in Figure 2. The "What happens when" part refers to a narrative of sequential and/or concurrent events. The "What actions do" part describes the effects of events and the "what's true when" denotes the current status of a situation. The formal logical definitions are given below:

"What happens when": a finite conjunction Δ of Happens formulae and temporal ordering among them such as Happen(*event1*, t_0, t_1), Happen(*event2*, t_1, t_2) … Happen(*eventn*, t_{n-1}, t_n) where $t_0 < t_1 < \ldots < t_{n-1} < t_n$.

"What actions do": a conjunction $\sum$ of Initiates, Terminates and Releases formulae describing the effects of events.

"What's true when": a finite conjunction Γ of formulae HoldAt(β_1, t), $\neg$HoldAt(β_2, t) … ($\neg$)HoldAt(β_n, t) where β_x is a ground *fluent* term and *t* is a ground time point. The Γ is actually a specific situation S at time point *t*.

Domain theory: a conjunction Γ_0 of InitiallyP and InitiallyN describing the initial situation S_{t_0}, a finite conjunction ψ of state constraints describing the indirect effect of actions, a conjunction Ω of uniqueness-of-names axioms for actions and *fluents*, and the event calculus axioms EC.

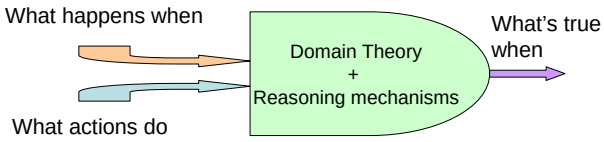

Fig. 2 The Reasoning Mechanisms of the EC

As can be seen from Figure 2, given any two parts the third part can be inferred using the underlying EC logical machinery. This corresponds to three broadly categorised reasoning tasks, namely i) deductive task, ii) abductive task and iii) inductive task. The deductive task infers "what's true when" based on "what happens when" and "what actions do". In other words, a deductive task seeks the consequence of a known sequence of events. It provides a mechanism for temporal prediction, and thus can be used for hazard prevention within the SH. The abductive task infers "what happens when" in terms of "what actions do" and "what is true and when". It aims to obtain a sequence of actions that leads to a given outcome. An abductive task provides an approach for temporal explanation and planning, thus it is considered useful for advising an inhabitant's ensuing action within the SH context.

The inductive task infers "what actions do" given "what happens when" and "what is true and when". It seeks a set of general rules and a theory of the effects of actions that account for the observed situation. An inductive task is usually used for learning and domain theory formation. In the following Section we describe the mechanisms for using the cognitive modelling and reasoning machinery for assistive living along with providing details of specific algorithms for each reasoning task.

5. Cognitive Assistance and Architecture

Existing reminder systems usually provide suggestions or reminding advice at pre-specified times or locations. Cognitive assistance is different from previous work in that it is intended to help an inhabitant take the "right" actions based on the real-time context and dynamic situations by observing and reasoning the unfolding of events in a SH. As such, cognitive behaviour assistance can be directly mapped to different reasoning tasks of the EC.

5.1 Predicting Consequences

To capture the consequence of a narrative of events is a typical deductive task. Given an initial situation S_{t_0}, each time an event happens a new situation will be arrived at. For example, the occurrence Happen(*event*, t_0, t_1) at S_{t_0} will lead to S_{t_1}. Therefore, for a sequence of events Δ, the temporal projection of a SH corresponds to a sequence of situations S_{t_0}, S_{t_1}, S_{t_2} … S_{t_n}. The situation S_{t_x} at t_x is actually the accumulation of all effects of these events that happen before t_x. Therefore, the consequences Γ for a Δ can be defined logically and formally as follows.

CIRC[Σ; Initiates, Terminates, Releases] $\wedge$ CIRC[$\Delta_0 \wedge \Delta$; Happens] $\wedge \psi \wedge$ EC $\wedge$ Ω $\models$ Γ

Where, CIRC stands for circumscription. CIRC[Σ; Initiates, Terminates, Releases] and CIRC[Δ; Happens] guarantee the non-effects of events and the non-occurrence of events by supplying explicitly the completion of predicates, thus solving the frame problem.

By sensing and capturing the up-to-date state changes in a SH, an assistive system can predict undesirable consequences and take appropriate actions to prevent them from happening. For example, suppose that an inhabitant takes the following actions, getUp, goTo(*kichen*), putFoodIntoOven, switch_on(*oven*), goTo(*sittingroom*), switch_on(*TV*) and watchTV. The assistive system will acquire the knowledge that the oven is on at a time point t_{ovenon}. If the inhabitant does not take any other actions that involve switching off the oven, and the time elapsed from t_{ovenon} to the current time $t_{current}$ goes over a pre-specified threshold, the assistive system will then take appropriate actions either by raising an alarm to attract the attention of the inhabitant or by shutting off the oven's power through an actuator.

5.2 Suggesting Actions

If a situation, i.e. the states of some *fluents*, is the goal of an inhabitant's ADL e.g. make a drink, use the telephone, reasoning and deriving a sequence of actions leading to that particular situation can be directly mapped to a typical AI planning problem. This can be realised through an abductive reasoning process in the EC (see Figure 2).

Previous research [20] [23] has viewed the prediction of ensuing actions as a 'plan recognition' problem. In this approach, a plan base enumerating all possible action paths is assumed *a priori,* the system takes as input a sequence of actions which have happened in an environment and infers the possible goals pursued by the inhabitant and subsequently predicts the next action in terms of an action path. The weakness of the above approach is that it is based on a number of assumptions, notably the availability of all possible event paths, the rationality of the observed agent (inhabitants) and the decidability of possible goals. For example, if there are *n* primitive actions, theoretically there will be n^2 event paths. Given any observed action, it is necessary to try every path, e.g. through depth-first or breadth-first searching, to identify the actual event route an inhabitant takes. This is not only computationally expensive and time-consuming, but also it may not be practically feasible to specify all event paths. Furthermore, the assumption of rationality and decidability is also weak because the inhabitants of SH are usually the elderly who may suffer from cognitive impairments and who do not have a consistent intention or persistent goal.

Rather than trying to identify all theoretically possible and complete ensuing actions in terms of some hypotheses, our approach intends to develop realistically efficient and pragmatic solutions for the genuine needs of SH inhabitants who are suffering from cognitive impairments. The rationale is to make use of commonsense knowledge of the ADL along with the ADL profiles of a specific inhabitant to create a set of practically sensible event paths. For example, an individual normally gets up in the morning, visits the bathroom and then may perform some other basic household tasks. Alternatively, a particular inhabitant may have the typical profile of going for a walk, reading the morning newspaper, then having breakfast, and finally drinking tea whilst watching television. In this case, the event path "getting up, going for a walk, reading paper, have breakfast, watch television" is more sensible for this particular person. There could be some more alternative event paths to incorporate potential deviations in behaviour. Nevertheless, the total number of event paths will significantly decrease and time will be saved, hence the possibility of finding the "right" event route will substantially increase. Theoretically the use of the inhabitant's profile and commonsense domain knowledge is equivalent to reducing the search space of an event path for a given goal by pruning the exhaustive event paths (see Figure 1), which is exponential to the number of the events in an event path, to those that are conformed with an inhabitant's ADL profile.

Our approach exploits the compound action concept (see section 3.3) to provide a natural way to incorporate heuristic knowledge about an inhabitant's ADL. For example, the commonsense knowledge that people usually go toileting after getting up in the morning can be encoded and represented by a sequence compound event (getUp **:** goToileting). Similarly, consider that an inhabitant has two living habits: either making tea or making breakfast after washing, and eating breakfast while watching TV. Then we can construct the non-deterministic and concurrent compound events, i.e., (washFinished **:** (makeTea **|** MakeBreakfast)) and (HaveBreakfast || WatchTV), to encode and model these patterns. A sequence compound action (α : β) can be regarded as reducing a search space after action α to only one choice β, and a two variable nondeterministic

choice of actions as only two choices α and β no matter how many actions are available.

Based on heuristic knowledge of an inhabitant's profile and commonsense creation of specific ADL, we can build a set of compound events for a particular inhabitant within a SH. These compound actions, together with primitive actions, can then be used to build procedures. Each of them can lead to a specific goal (situation), i.e., taking a SH from one situation to another. Therefore, suggesting the ensuing actions can be defined as: given an inferred goal, find out a procedure that leads to that situation based on observed actions, and decompose the procedure into sub-compound actions and/or primitive actions depending on what conditions hold. The primitive action immediately following the observed actions in an event path is the next action that the system would advise the inhabitant to take. The distinguishing feature of our approach is that by recursive decomposition it can decide the next action when the first compound actions consisting of observed actions is identified without the need to know in detail the following actions. For example, when a getUp event is observed in the morning, the goToileting action can be advised no mater what events follow (except emergency events); when a haveBreakfast action is observed, the watchTV action could be suggested in terms of the inhabitant's profile. As both primitive actions and compound actions incur state changes that can be viewed as a goal or situation, a procedure for a desired goal can be viewed as the sequential pursuit of a number of sub-goals. In particular, a procedure, i.e. its contained sub-goals, could be nondeterministic. This reflects the exact nature of inconsistency of an inhabitant's intention and the flexibility of the ADL. For example, an inhabitant gets up in the morning, goes to their lounge area within their house, and then has the ptential to engage in a number of actions e.g. to read newspapers, open windows, make a phone call, turning on TV, watching TV all before going to make breakfast. By taking into consideration the inhabitant's ADL profile we can formally represent an event path using a sequence and nondeterministic construct in the following procedure {(getUp **:** goToileting) **:** gotoLounge **:** ((collectNewspaper : readNewspaper) | openWindow | makePhoneCall | (turnOnTV : watchTV)) **:** prepareBreakfast, haveBreakfast || watchTV}. Based on the observed action, the next action could be inferred from the corresponding compound action. For instance, if collectNewspaper is perceived, then readNewspaper should be recommended should the inhabitant seemingly forget what to do next. Other logical constructs provide rich capabilities for modelling diversity of the ADL, e.g. an interrupt construct can facilitate the handling of emergency events.

Given the formal domain descriptions (see Section 4.1), the algorithm for reasoning and suggesting the ensuing actions is as follows: suppose that an assistive system derives an inhabitant's desired goal based on its ADL profile and commonsense knowledge. The goal should be represented as a situation Γ in the form of the HoldAt formulae. Using resolution against formulae in ***d***, the underlying reasoner of the assistive system will identify all high-level procedures *ps* that will achieve Γ. The reasoner then decomposes *ps* using resolution against formulae in ***c*.** This decomposition may yield any combination of the following.

- Executable primitive actions, i.e. Happens formulae.
- Sub compound actions within the procedure, i.e. Happens formulae in ***c***.
- Conditions that are equivalent to sub-goals, i.e. HoldAt formulae.

For primitive actions, the system will compare them with observed actions to decide the ensuing action in the procedure which will be recommended. If an observed action appears in multiple procedures with the same goal, theoretically any action in these procedures can be recommended. A procedure may have conditional tests in order to determine an event flow. A condition about a state *x* is represented as a HoldAt(*x*) formula. Therefore, if a condition is not met, the system needs to seek a sequence of actions to make HoldAt(*x*) true. This process should be exactly the same as discussed above.

In general, reasoning and suggesting an inhabitant's action is a cycle of (sub)goal inference, compound action search and decomposition, primitive action generation and comparison with observed actions. Nevertheless, the system can start suggesting actions when the first executable action is available while the future activities are still unknown or nondeterministic. If there is no corresponding high-level compound action available for a given goal in formulae *d*, the system will use the formulae in *b* to derive a plan from scratch as a typical planning problem. This is beyond the scope of this current work.

The overall process of suggesting the next actions interweaves sensing, action recognition, event reasoning and advising whenever necessary. Sensor events or states (represented as *fluents*) serve as pre-conditions for subsequent actions or conditional test criteria for alternative event path routes. They are also used as conditions for terminating current actions. Event reasoning is a resolution-based abductive theorem proving process working on a collection of event calculus formulae. The most salient feature is the exploitation of incremental prediction via compound actions decomposition. This facilitates advising next actions in progressive order, which help address the inconsistency and flexibility of ADL intentions and goals. If a sensing produces an unexpected event or state that fails to satisfy the required conditions for subsequent actions, then the system will have the ability to trace back to try other event paths.

5.3 The Architecture

Fig. 3 shows a conceptual system architecture for the proposed approach of behaviour modelling and reasoning. Central to the architecture is the Reasoning Engine, which provides the inference capabilities for manipulating the logical formulae describing events and their effects in a SH. The engine uses recognised

observed events as inputs, and infers event consequences or ensuing actions against its *a priori* knowledge. It provides instructions to assistive services to take concrete assistive actions. The *a priori* knowledge is provided through the Inhabitant Profile, the Domain Theory and the Event and Pattern Repository components. The Event and Pattern Repository component contains event and pattern (compound actions) formulae defining the preconditions and effects of events and compound actions. The Domain Theory consists of all events and *fluents* that are identified as essential events and necessary attributes for the description of a SH. It will also define state constraints, effect constraints and causal constraints, which express logical relationships that have to hold between *fluents* at all times.

The Inhabitant Profile component contains details of the inhabitant's profile and habits in addition to their daily life preferences. These can be formalized using the EC formalism and incorporated into the Domain Theory and Pattern Repository components. Domain experts, formal and informal carers and the inhabitant will collaborate to capture and model various types of domain knowledge and encode them into EC formulae into the aforementioned components for reasoning.

The Event Recognition component analyses collected sensor data from various sensors, it recognise the occurrence of events, and passes on the observed events to the reasoning engine for inference. The event recognition algorithm for each individual device should be different dependent on their nature and configuration, which we shall not discuss here in detail. The recognised events will be modelled as a narrative of Happen(anObseredEvent, t_i, t_{i+1}) formula and used as an input for the reasoning tasks, i.e., consequence prediction and/or action advising.

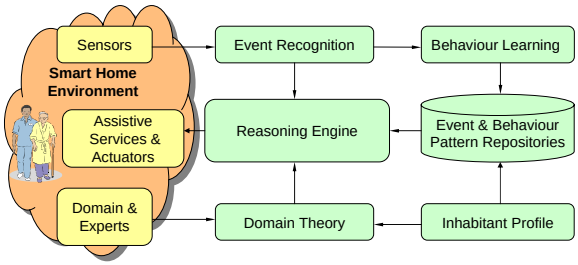

Fig. 3 The Proposed System Architecture based on the Concepts of EC

The Behaviour Learning component will use machine learning algorithms against observed occurrences of events to derive ADL patterns from daily observations and populate the pattern repositories automatically through dynamic learning capabilities. This enables assistive systems to capture dynamic and unexpected events and further adapt to their occurrence by using previously observed patterns. The Assistive Services and Actuators component is responsible for the provision of assistance to the inhabitant within the SH. Examples of assistance provision may be audio or video reminding messages, emergency calls being placed to carers or direct activation of actuators for example door and window openers for direct control of the SH environment.

The three most compelling features of the system can be considered as the following. Firstly, as compound actions are built using heuristics and inhabitant's profiles, they reduce the search space of potential event paths to a limited set of paths that are most plausible, thus making the solution more pragmatic and scalable. Secondly, by recursive decomposition an assistive system can decide the next action when the first compound action consisting of observed actions is identified without the need to know in detail the following actions. This subsequently enables advising of ensuing actions in progressive order. Thirdly, through rich logical structures an assistive system can specify and further infer complex event patterns such as nondeterministic or concurrent events. It is important to give ADL advice in such situations because the behaviour of an inhabitant with cognitive deficiency is usually inconsistent and nondeterministic.

6. Case Study – Drink Preparation

We shall use the activity of preparing a cup of tea, denoted as makingACupOfTea, as an example to show how the proposed framework works. In the following we illustrate the construction of a SH domain theory in the context of "makingACupOfTea" in bullet (i) to (vi). The implementation of an assistive system is presented in bullet (vii) and the reasoning mechanism is discussed in bullet (viii). Due to limited space, details of all formulae will not be included.

(i) Primitive actions

takeTo(*thing*, *location*) - to take the *thing* to the *location*.
add(*thing*, *container*) and remove(*thing*, *container*) - to add/remove the *thing* into/out of the *container*.
turnOn(*device*) and turnoff(*device*) – to turn on/off the *device*.
boilWater and stirAround are two other action involved in the makingACupOfTea activity.

Here, a *thing* refers to any item or substance ranging from an inhabitant, raw food, water, a teabag, boiled water, sugar, milk, to name but a few. A *location* refers to any position such as a room, a table, a sofa, a window, a bed, a fridge or a position represented in 2-D coordinates. A *container* refers to any thing that can hold a smaller item such as a house, a room, bathroom, a kitchen, a washing machine, a freezer, a kettle and a tea mug.. A *device* refers to any thing that can be turned on and off such as a tap, a light, a TV, a cooker, a microwave, a radio, a fire alarm and a smoke alarm.

As can be seen actions can take one or two argument variables, this allows them to perform the same actions against different objects. For the makingACupOfTea scenario, instantiated actions include takeTo(kettle, basin), turnOn(tap), add(cold water, kettle), turnOff(kettlePower), boilWater, add(teabag, mug), add(boiled water, mug), remove(teabag, mug),

add(sugar, mug), add(milk, mug), stirAround, takeTo(mug, kitchen table), takeTo(milk, kitchen table), takeTo(sugar, kitchen table), takeTo(milk, fridge) takeTo(milk, cupboard).

(ii) Fluents, i.e. state variables

at_pos(*thing*, *location*) --- The *thing* is at the *location*.
inside(*thing*, *container*) --- The *thing* is inside the *container*.
available(*thing*) --- The *thing* is available for use.
on(*device*) --- The *device* is on.

Here, *thing*, *container*, *location* and *device* are the same as previously defined. We use the negation of these variables to represent their false status. For example, ¬ at_pos(*thing*, *location*) means the *thing* is not in the *location*; ¬ on(*device*) means the *device* is off.

(iii) Effect formulae (some examples)

Initiates(takeTo(*thing*, *location*), at_pos(*thing*, *location*), *t*)
Initiates(add(*thing*, *container*), inside(*thing*, *container*), *t*)
Initiates(turnOn(*device*), on(*device*), *t*)
Initiates(boilWater, available(boiledWater), *t*) ← HoldAt(inside(cold water, kettle), *t*)

Some instantiated effect formulae are listed below.

Initiates(takeTo(kettle, basin), at_pos(kettle, basin), *t*) ← HoldAt(at_pos(inhabitant, kitchen), *t*)
Initiates(turnOn(tap), on(tap), *t*) ← HoldAt(at_pos(inhabitant, basin), *t*)
Initiates(add(cold water, kettle), inside(cold water, kettle), *t*) ← HoldAt(at_pos(inhabitant, basin), *t*)
Initiates(boilWater, available(boiledWater), *t*) ← HoldAt(), inside(cold water, kettle), *t*)
Initiates(takeTo(mug, kitchen table), at_pos(mug, kitchen table), *t*) ← HoldAt(at_pos(inhabitant, kitchen), *t*)
Initiates(add(teabag, mug), inside(teabag, mug), *t*) ← HoldAt(at_pos(teabag, kitchen table), *t*) ∧ HoldAt(at_pos(mug, kitchen table), *t*)
Initiates(add(boiledWater, mug), inside(boiledWater, mug), *t*) ← HoldAt(inside(teabag, mug), *t*) ∧ HoldAt(available(waterBoiled), *t*)
Initiates(stirAround, available(readyTea), *t*) ← HoldAt(¬ inside(teabag, mug), *t*) ∧ HoldAt(inside(boiledWater, mug), *t*) ∧ HoldAt(inside(sugar, mug), *t*) ∧ HoldAt(inside(milk, mug), *t*)

(iv) Heuristics and inhabitant profile

Suppose that an inhabitant has the habit of adding sugar to tea and of having tea around 10:30am and 4:30pm respectively each day, and normally spends half an hour to make a cup of tea. Using this heuristic and the background knowledge about making a cup of tea, we can define a compound action makeACupOfTea as a narrative of actions – see the definition of the makeACupOfTea compound action below. The ADL profile can be represented either as states HoldAt(available (teaReady), 10:30am) and HoldAt(available(teaReady), 4:30pm) or as compound action Happen (makeACupOfTea, 10am, 10:30am) and Happen(makeACupOfTea, 4pm, 4:30pm).

(v) Compound action and its effect

The makingACupOfTea compound action can be defined as follows:
Proc. makingACupOfTea
{Happen(takeTo(kettle, basin), t_0) ; Happen(turnOn(tap), t_1) ; Happen(add(cold water, kettle), t_2, t_3) ; Happen(turnOff(tap), t_4) ; (Happen(boilWater, t_5, t_6)) || (Happen(takeTo(mug, kitchen table), t_5) ; Happen(takeTo(teabag, kitchen table), t_{51}) ; Happen(takeTo(milk, kitchen table), t_{52}) ; Happen(takeTo(sugar, kitchen table), t_{53}))
waterBoiled ?
Happen(add(teabag, mug), t_7) ; (Happen(add(boiledWater, mug), t_8) ; Happen(add(sugar, mug), t_9) ; Happen(add(milk, mug), t_{10})) ;
Happen(stirAround, t_{11}, t_{12}) ;
Happen(takeTo(milk, fridge), t_{13}) ; Happen(takeTo(sugar, cupboard), t_{14}))
Here $t_0 < t_1 < t_2 < t_3 < t_4 < t_5 < t_6$, $t_5 < t_{51} < t_{52} < t_{53} < t_6$, and $t_7 < t_8 < t_9 < t_{10} < t_{11} < t_{12} < t_{13} < t_{14}$.

Note that the ;, ||, ? are extra-logical constructs denoting the way actions are taken. For example, the || operator indicates that mug, teabag, milk and sugar can be prepared while water is boiled, i.e. taking actions in parallel. The ? operator is used to test if boiled water is available before adding hot water, milk, and sugar into the mug.
The effect formula of the compound action is Initiates(makingACupOfTea, available(teaReady), t_{14}).

(vi) Initial situation

InitiallyP(at_pos(inhabitant, kitchen))
InitiallyP(at_pos(inhabitant, basin))
InitiallyP(at_pos(milk, fridge))
InitiallyP(at_pos(sugar, cupboard))
and more …..

(vii) Assistive system design

The Domain Theory component will contain two Uniqueness-of-Names Axioms (UNA) for all actions in (**i**) and (**v**), and all *fluents* in (**ii**) respectively:

UNA[takeTo, add, remove, turnOn, turnOff, boilWater, stirAround, makeACupOfTea]
UNA[at_pos, inside, available, on]

It can also contain state constraints, effect and causal constraints which we do not discuss in this example.
The Event and Behaviour Pattern component will consist of all effect axioms in (**iii**) and (**v**). The Inhabitant Profile component will contain information about the inhabitant's living habits, routine activities and special ADLs as defined in (**iv**). Such heuristics, in combination with common sense knowledge, will be used for the construction of compound actions and decision-making during the reasoning process.
We assume that the Event Recognition component can observe an inhabitant's activities through the Sensor component and further recognize and represent them as individual primitive actions. The detail of raw data collection and ADL recognition algorithms is beyond the scope of this paper.

(viii) Assistance provisioning

Based on the formal SH modelling described above the assistive system will have the makeACupOfTea compound action and two desired goals HoldAt(available(teaReady), 10:30am) and HoldAt(available(teaReady), 4:30pm). To achieve the two goals, the system will infer that two compound actions, i.e., Happen(makeACupOfTea, 10am, 10:30am) and Happen(makeACupOfTea, 4pm, 4:30pm), should take place at the specified time points in terms of the effect axioms and the 30 minute requirements from the inhabitant's profile. In this case, if the assistive system does not detect expected actions at the corresponding time, for example, the takeTo(kettle, basin) action at or shortly after 10.00am and 4.00pm, then it can activate reminding services to remind the inhabitant of the activity. Similar assistance can be provided during the process of making a cup of tea if the expected action in the compound action is not recognised in a considerable time. For example, if boiled water is ready for quite a long time and the add(boiledWater, mug) action is not observed, then a reminder can be issued.

By the compound action mechanism, we can build high-level ADL patterns recursively based on inhabitants' profiles. For example, the following procedure {getup(at_morning_time) ; goToileting; (collectNewspapers || makeACupOfTea || makeBreakfast); (haveBreakfast || drinkTea || readNewspaper || watchTV)} specifies an inhabitant's morning activity based on her/his profile in which makeACupOfTea becomes a building block. While we omit details, the logical operators and sequences incorporate inhabitants' ADL heuristics and needs, which can lead to personalized knowledge-based living assistance.

7. Conclusions

This paper has proposed an EC-based logical framework for behaviour reasoning leading to personalised just-in-time behaviour assistance within a SH. The concepts of a compound action and its hierarchical construction mechanism enable assistive systems to incorporate ADL heuristics and inhabitant profiles and hence achieve a degree of personalised assistance. The approach avoids the assumptions of inhabitant's rationality and the time-consuming planning processes of traditional approaches. Therefore the proposed solution can be considered to be more pragmatic and with greater potential for wide application. We have developed a conceptual system architecture and demonstrated its use through the scenario of "making a cup of tea".

There are many topics which we have not considered within this work such as how to decide the level of granularity at which cognitive modelling is conducted, how to make the approach scalable to large-scale complex system, and how to automatically or semi-automatically construct compound actions. Our future work will first focus on the development of a prototype assistive system to carry out full experimenting and evaluation of the approach based on the case scenario presented in this paper. We shall investigate the use of learning techniques to construct compound actions by mining activity patterns. In addition, we shall look into the ontology technologies from the semantic Web domain to facilitate domain formalisation and knowledge acquisition such as ADL heuristics and user preferences.

References

[1] Ambient Assisted Living (AAL) joint programme, http://www.aal-europe.eu/.

[2] S. Giroux and H. Pigot, From Smart Home to Smart Care, Assistive Technology Research Series, in Proc. of the 3rd International Conference On Smart homes and health Telematics, ICOST05, Canada, 2005.

[3] M. Chan, D. Estève, C. Escriba and E. Campo, A review of smart homes—Present state and future challenges, Computer Methods and Programs in Biomedicine, vol.91, no.1, pp.55-81, 2008.

[4] F.L. Lewis, Wireless Sensor Network, in Cook D.J. and Das S.K., editors, Smart Environment: Technology, Protocols and Applications, Wiley, 2004.

[5] M. Pollack, Intelligent technology for an aging population: the use of AI to assist elder with cognitive impairment, AI Magazine, pp1-27, 2005.

[6] Smart Medical Home Researh Laboratory, Center for Future Health, University of Rochester, http://www.futurehealth.rochester.edu/smart_home

[7] The PlaceLab, House_n research Group, MIT Department of Architecture, http://architecture.mit.edu/house_n/placelab.html.

[8] M. Knapp and M. Prince, Dementia UK, the Alzheimer's Society, 2007. http://www.psige.org /psige-pdfs/Dementia_UK_Summary.pdf.

[9] C. Baum and D. Edwards, Cognitive performance in senile dementia of the alzheimer's type: The kitchen task assessment. The American Journal of Occupational Therapy, Vol. 47 (5), 431436, 1993.

[10] R.A. Kowalski and M.J. Sergot, A Logic-based Calculus of Events, New Generation Computing, Vol.4 pp.67-95, 1986.

[11] M.P. Shanahan, Solving the Frame Problem: A Mathematical Investigation of the Common Sense Law of Inertia, MIT Press, 1997.

[12] L. Chen, An Architecture for Animated Human-like Interface Agent, PhD thesis, 2002.

[13] M. Philipose, K.P. Fishkin, D.J. Patterson, D. Fox, H. Kautz, D. Hahnel, Inferring activities from interactions with objects. IEEE Pervasive Computing, pp.50-57, 2004.

[14] J. Boger, P. Poupart, J. Hoey and C. Boutilier, A. Mihailidis, A Decision-Theoretic Approach to Task Assistance for Persons with Dementia. In Proc. of the International Joint Conference on Artificial Intelligence (IJCAI'05), pp.1293-1299, 2005.

[15] D.W. Albrecht and I. Zukerman, Bayesian Models for Keyhole Plan Recognition in an Adventure Game, User Modelling and User-Adapted Interaction, (8) 5-47, 1998.

[16] X. Hong, C. Nugent, M. Mulvenna, S. McClean, B. Scotney, S. Devlin, "Evidential fusion of sensor data for activity recognition in smart homes," Pervasive and Mobile Computing, 2008 (in press).

[17] J. Bauchet and A. Mayers, Modelisation of ADL in its Environment for Cognitive Assistance, In Proc. of the 3rd International Conference on Smart homes and health Telematics (ICOST'05), Sherbrooke, Canada, pp.3-10, 2005.

[18] J. Hoey and P. Poupart, Solving POMDPs with continuous or large discrete observation spaces. In Proc. Int'l. Joint Conf. on Artificial Intelligence, pp.1332–1338, 2005.

[19] H. Kautz, A Formal Theory of Plan Recognition and its Implementation, Reasoning About Plans, Allen J., Pelavin R. and Tenenberg J. eds., Morgan Kaufmann, San Mateo, C.A., (1991), 69-125.

[20] W. Wobke, Two Logical Theories of Plan Recognition, Journal of Logic Computation, Vol.12(3), pp.371-412, 2002.

[21] B. Bouchard and S. Giroux, A Smart Home Agent for Plan Recognition of Cognitively-impaired Patients, Journal of Computers, Vol.1, No.5, pp.53-62, (2006)

[22] H. Levesque, What Is Planning in the Presence of Sensing? Proceedings AAAI96, pp.1139-1146, 1996.

[23] S. Carberry, Techniques for Plan Recognition, User Modeling and User Adapted-Interaction, (11) 31-48, 2001.