

# A Visual Editor to Support the Use of Temporal Logic for ADL Monitoring

A. Rugnone<sup>1</sup>, F. Poli<sup>1</sup>, E. Vicario<sup>1</sup>, C. D. Nugent<sup>2</sup>, E. Tamburini<sup>3</sup>, and C. Paggetti<sup>3</sup>

<sup>1</sup> Department of Systems and Computer Science, University of Florence, Italy

<sup>2</sup> Faculty of Engineering, University of Ulster at Jorsanstown, Northern Ireland

<sup>3</sup> I+ srl, Florence, Italy

**Abstract.** The use of technology within the home environment has been established as an acceptable means to support independent living for elderly and disabled people. An area of particular interest within this domain relates to monitoring of Activities of Daily Living for those persons with a form of cognitive decline. In this area, specific tasks undertaken by the persons in the context of their normal day-to-day lives reveal a wealth of information to be used to customize their environment to improve their living experience. In our current work we investigate the development of models which can be used to represent, classify and monitor basic human behaviors and support observation and control of activities of daily living. In particular, in this paper we focus on the problem of automated recognition of sequences of events that may indicate critical conditions and unexpected behaviors requiring intervention and attention from caregivers. Our work is based on a formal framework developed with temporal logic used for the specification of critical sequences of patterns and a behavior checking engine for automated recognition. In addition we have also developed an approach to provide a means of interaction with user. A visual formalism for the specification of Linear Temporal Logic expressions reduces the barrier of technical complexity enabling the involvement of experts in the domain of healthcare services to interact with the system.

**Keywords:** Model Checking, Temporal Logic, Activity Daily Living, Patient Behavior Models.

## Introduction

As our elderly population continues to grow the need to develop intelligent environments to support independent living is ever increasing. Recent results have shown that the introduction of technological solutions within the home environment can have significant impact on the quality of life of the person. Examples of these solutions include basic motion sensors to activate lights, to safety devices, to prevent flooding in bathrooms, to elaborate wearable systems and to monitor and assess vital signs [1]. The common goal of all these solutions is to improve the living experience and quality of life for the person, customize their living environment and extend the period of time they can remain at home without the requirement for institutionalization. Although, on the whole, the introduction of technology can have positive effects, there are a number of potential concerns. Viable solutions for wide-scale uptake should not be expensive,

should be compatible or interoperable with other devices/solutions and should integrate with existing means for healthcare delivery. In addition, a realistic reimbursement scheme requires establishment for service realization.

In our current work, we have focused our efforts towards the monitoring and analysis of human behavior to support the control of Activities of Daily Living (*ADL*). Many efforts have recently been focused on ADL recognition: Augusto et al. [2] proposed a smart home framework centered on Event-Condition-Action (*ECA*) rules to capture events accurately, applied to monitoring of activities of elderly people with cognitive impairment; Bauchet and Mayers [3] proposed a hierarchical model for the description of ADL; Bouchard et al. [4] proposed a non-quantitative logic approach to ADL recognition for Alzheimer's patients, based on lattice theory and action description logic.

In this paper, we propose a formal approach to critical behavior recognition in ADL analysis based on the use of temporal logic [5] and algorithms, usually applied in the context of model checking [6] [7], of Run Time Execution Monitoring (REM) [8] [9] and also in on-line systems intrusion detection [10].

Based on these developments, we describe a service model that may be deployed to exploit this solution. Then we describe, with a partial degree of completeness and formalism, the characteristics of the logic deployed and the model checking algorithm. With the formalisms in place, we exemplify the kind of behavior that the system can describe and devise the type of service organization which it can effectively support. We also focus our attention on the visual formalisms to support the specification of the logic formulae, to support the involvement of users in the development of such systems, especially those with expertise in the domain of home based healthcare provision.

## 1 Service Model

The service model is a means of defining the stakeholders, their responsibility, the links and the interaction between them, the devices and services and how the care will actually be delivered within the realms of home healthcare delivery. Within our current study we are initially focusing on the requirements of healthcare operators and experts within such a system. Healthcare operators and experts both have responsibility to define the necessary requirements to be used to monitor the patient. The expert user has the responsibility to define the expected behavior of the patient and the bounds of their deviation from normal conditions, while the operator has the responsibility to perform the necessary intervention when the patient is reporting a deviation from normal.

To provide an automated approach for service delivery the required system would require three main modules. These modules would involve:

1. A repository to store the representative behaviors (system rules) of the patient;
2. A monitoring platform to monitor multiple patients within their home environment [11];
3. A Behavior Checking Engine (BC Engine) acting as a model checker, which compares the specification of critical behavior against the observed sequence events.

Our work is based on the assumption that the subjects who are being monitored and their living environment is equipped with a suite of pervasive middleware sensors. The

information provided by the sensors records low level events such as “*door is open*” or “*cooker is on*”. Using such information it is then possible to model the *ADL* as a set of observable variables that provide a sequence of conditions started with a initial state, formed by their initial values. The representation of what happens in the environment can be modeled by a sequence of states, differentiated by changes in measured values with events such as { “*Water added to the kettle*”, “*The kettle is full*”, “*Cooker has been turned on*”,... }. The operator monitors the single state of the end user via the monitoring platform. When a critical behavior occurs the Behavior Checking Engine automatically notifies the event via the monitoring platform. The format of the event notification issued by the Behavior Checking engine is dictated by the Behaviors Model Repository (*BMR*).

The expert user has the responsibility to define the specification of the rules dictating the expected behavior of the patient. If, however, we wish to deploy complex formalisms to represent the underlying rules and model behavior it is necessary to provide a form of visual based rule editor which requires limited technical knowledge of the underlying notation. Based on the rules such an editor would produce, the model checking engine would have the ability to process them according to predefined formalisms on the fly. This approach makes possible the definition of a set of rules that may be deemed exportable and reusable all under the control of expert healthcare user.

The remainder of this paper focuses on the development of such an interface.

## 2 Behavior Checking

We propose the use of temporal logic to specify critical behaviors of persons within their home environment and a behavior checking algorithm to process the behaviors and hence identify their deviation from normal which implicitly represent situations of concern. In particular we propose the use of an adaptation of Past Linear Temporal Logic (*PLTL*) [12] to satisfy the assumptions made in Sect. 1 and also to deal with the constraints imposed by the application domain. The logic used is linear as we must accommodate for a sequence of states. In addition, we must accommodate for previous observations in addition to those at the present moment and the time elapsed between two events.

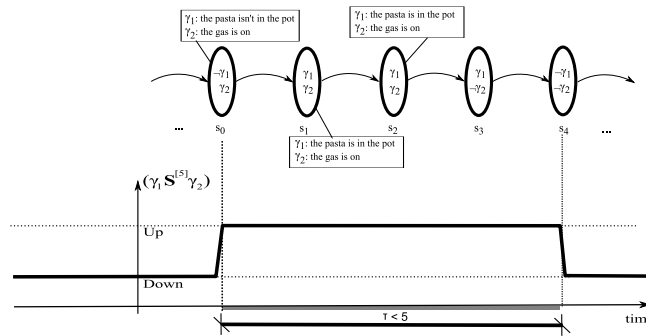
### 2.1 Temporal Logic : Syntax and Semantic

The language of temporal logic relies on describing conditions over paths ( $\rho$ ), i.e. sequences of states. A formula defining rules on a path is called a *path formula*. Path formulae are built as a combination of atomic propositions on states, by means of Boolean connectors and temporal operators:

$$\gamma ::= p_s \mid \neg\gamma_1 \mid \gamma_1 \wedge \gamma_2 \mid \gamma_1 \mathbf{S}^\tau \gamma_2 \mid \mathbf{H}^\tau \gamma_1 \mid \mathbf{P}^\tau \gamma_2 \mid \mathbf{V} \gamma_2 \mid \mathbf{C}_{[min,max]}^\tau(\gamma_1)$$

where  $p_s$  is an atomic proposition on state. The temporal operators are **S** (Since), **H** (Historically), **P** (Past), **V** (Previous) and **C** (Count). By using an index  $\tau$  we can specify that formulae built by these operators are constrained within a temporal interval  $[-\tau, 0]$ . It is also possible to omit  $\tau$  from the writing of a temporal operator, thus implicitly setting it to  $\infty$ , and unbounding the operator time interest interval.

Meanings and uses of these operators are best described through the use of examples. Consider, in the first instance, a person within their home environment called Marco. A proposition which is not considered to be elementary could be represented by “Marco doesn’t add the water to the kettle if the kettle is full”. We can decompose the phrase into two elementary propositions “Marco doesn’t add the water to the kettle”, labelled with  $\gamma_1$ , and “The Kettle is full”, labelled with  $\gamma_2$ . The initial statement can now be represented with the simple formula  $\gamma_1 \wedge \gamma_2$ . Another example is the following: “Marco has been sleeping for the past 8 hours” could be modelled as  $H^{[8]}\gamma_1$ , where  $\gamma_1$  represents “Marco sleeps”. The logic can also be used to model more complex representations, for example consider the following; “Marco is adding some flavour to his coffee and in the past he has used sugar”. Based on the previous approaches we decompose the proposition into the following elementary components, “Marco is adding some flavour to his coffee”,  $\gamma_1$ , and “Marco has used sugar”,  $\gamma_2$ , and hence the scenario can be redefined with the following formula  $\gamma_1 \wedge P(\gamma_2)$ . Consider, as a final example: “The pasta has been in the pot since the gas was turned on 5 minutes ago”. By adopting the same decomposition method, we can model the phrase with  $\gamma_1 S \gamma_2$ , where  $\gamma_1$  is “The pasta is in the pot”, and  $\gamma_2$  is “the gas is on”; taking  $\rho_4$  a path  $s_0 \dots s_4$  as in 1. This means  $\rho \models \gamma_1 S \gamma_2$ , read  $\rho$  models  $\gamma_1 S \gamma_2$ , if it existed a sub-sequence of  $\rho_3 = s_0 \dots s_3$  that satisfies in a time minor of 5 minutes and each sub-sequence of  $\rho_3 = s_0 \dots s_3$  satisfies  $\gamma_1$ . With this scenario we have a trend on the time of formula as represented in the Fig. 1.



**Fig. 1.** The pasta is in the pot since the gas is on. In figure is represented the trend of formula on the sequence of states.

## 2.2 Behavior Checking Algorithms

The complexity of TL formulae tends to rapidly grow when applied to realistic behavioral pattern. ‘Divide et Impera’ is good practice to manage this complexity. To this scenario, we decompose instances of complex behavior into a set of simpler and complementary representations; however, doing so contributes to the increasing number of formulae required. In addition to this level of complexity and in accordance with the model to be created, the analysis of the sequence and duration of states require processing. This places the requirement for the model checking algorithm to satisfy rigid constraints relating to both time and space complexity and to provide a real time response.

To address these requirements, we propose an algorithm based on the recursive evolution of two variables, called  $\Sigma$  and  $T$ , whose calculation is done for each state notification.  $\Sigma$  means the validity of the formula to be checked, while  $T$  is a temporal evaluation on sequence of states. If  $\Sigma$  is true the formula satisfies the path which is currently being evaluated. Below, as an example we define the single step calculation, for the *Since* temporal operator previously introduced. Similar algorithms can be defined for the other operators.

$$\begin{cases} T_n = (s_n \models \gamma_2) ? \tau_{s_n} : T_{n-1} \\ \Sigma_n = ((s_n \models \gamma_2) \vee (\Sigma_{n-1} \wedge s_0 \models \gamma_1)) \wedge (\tau_{s_n} - T_n) \leq \tau \end{cases}$$

where  $s_n$  is the state presently observed,  $s_0$  is the starting state of observation.  $\tau_{s_n}$  is present time and  $T_0 = 0$  and  $\Sigma_0 = (\gamma_2 \models s_0)$  are the booting values of the sequence. It means, in the case of *Since* operator, that  $T_n$  stores the last time in which the condition  $\gamma_2$  is satisfied on a state, while  $\Sigma_n$  involves checking conditions both on state properties (the left hand of  $\wedge$  operator), and on time (the right hand).

### 2.3 Visual Formalism

The Visual Editor is a tool based on simple visual formalisms that aims to make the process of specification of temporal logic based formula more *intuitively based* as opposed to *mathematical focused*. This approach facilitates the design of temporal logic expressions by Healthcare professionals, without them requiring an in depth knowledge of the underlying computational principles. The visual formalisms have been designed following a previously conducted user-centered usability engineering process [13]. An heuristic design process was applied to maximize consistency, i.e. to minimize the complexity of the visual metaphor mapping textual sentences into their respective visual representation.

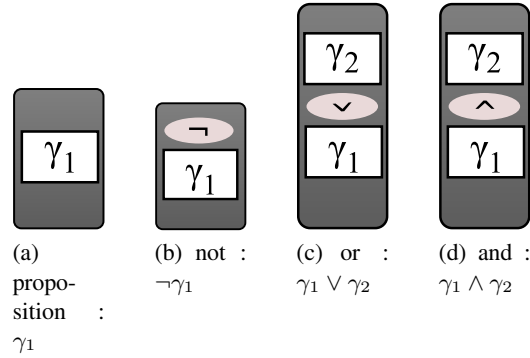
The Basic visualization primitives associated with the semantic constructs of temporal progression, timing constraints, and propositional logic, can be combined into a set of visual re-writing rules, which associate each terminal token of PLTL with a concrete drawing. Matching the recursive organization of the PLTL syntax and semantics, these rules reduce the visualization of a formula to the recursive visualization of its sub-formulae and to the concrete drawing of graphic icons representing terminal symbols. While the structure of rules matches the PLTL syntax, the concrete drawings that these rules produce reflects the semantics of terminal symbols.

The PLTL expressivity combines three distinct fragments:

1. Basic Proposition;
2. Sequencing conditions;
3. Instantaneous conditions.

Following a commonly accepted standard, basic propositions are represented in textual form inside the frame of a test sheet icon (see Fig. 2(a)). Sequencing conditions expressed by temporal operators, underlie the basic concept of a sequence of temporal contexts through which the system progresses following the metaphor of timeline graphs. An example of such a sequence is visualized as a horizontal path line in Fig. 3.

For representation of instantaneous conditions captured through Boolean outputs, the process adopted suggests that operands should be arranged in a vertical direction (see Fig. 2). As an example, Figure 4 reports the visual representation for the textual formula of  $C_{[3,inf]}(P^{[10min]}(A))$ .



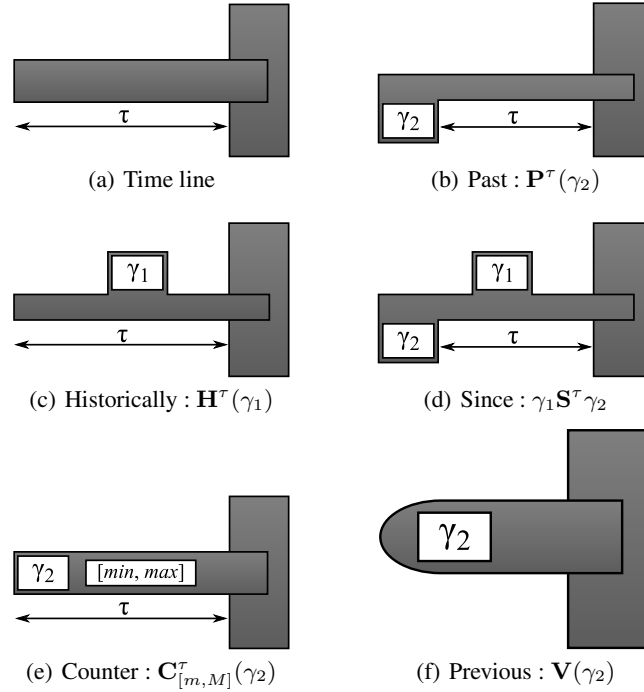
**Fig. 2.** Graphical Logical operators. Figure 2(a) represents a proposition or an alias for a more complex formula (e.g. a predefined monitoring *macro*); figure 2(b) represents the negation of a subformula  $\gamma_1$ . 2(b), 2(c), 2(d) represent instantaneous conditions based on boolean operators.

### 3 An Example of Use

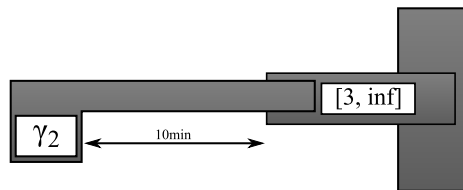
As a simple example of usage of the proposed approach, consider the scenario of a patient exhibiting a form of Confusional State Behavior. In this case, we consider two topics:

1. Repetitive Confusion : repetition of the same action in a relatively short time interval whilst not being considered as continuous;
2. Random Confusion : repetition of different actions in a relatively short time interval.

For instance, confusional behavior is a sequence like “A1, A2, A3, A1, A2, A3...” or a repetitive sequence like “A1, A1, A1...”. Generally we can consider confusion as any cycle in the ADL model which is repeated too much in a short space of time. We can further progress this concept by defining “*too much*” repetitions as a quantity higher than 3 and “*a short space*” of time as a quantity less than 5 minutes. Although these are sharp decision boundaries in terms of temporal values and counter values, they are based only on experience in the domain, but are equally questionable. It is a generally accepted fact that this feature of temporal logic is not a good asset and can be considered as a major limitation of the approach. Nevertheless, these sharp decision boundaries are offset by good expressiveness and excellent flexibility to define system specifications, as we will examine. A cycle is a path that begins with a state, for example A, and also finishes with this state. We can model the confusion as  $C_{[3,inf]}(P^{[10min]}(A))$  where A is any action or state, e.g. “Add water to kettle”,  $P_{[10min]}(A)$  means at least 2 minutes ago action A is performed and, the last, that in the past  $C_{[3,inf]}(P^{[10min]}(A))$  means



**Fig. 3.** Visual formalisms on time constraints and temporal operators. Figure 3(a) shows the basic concept of arranging graphical elements in the formalism: the horizontal dimension represents the temporal line, the horizontal rectangle represents a sequence of states evolving within temporal interval long  $\tau$ , while the vertical rect represent the present. Figure 3(b) shows the formalism associated to the Past temporal operator, describing that at least once in the past, within a  $\tau$  time from now, a condition  $\gamma_2$  has been satisfied. Similarly, the other formalisms represent the remaining temporal operators: Historically (Fig. 3(c), requiring that a condition  $\gamma_1$  holds continuously during the last  $\tau$  time), Since (Fig. 3(d), requiring that since a condition  $\gamma_2$  has been verified in the past, within a time  $\tau$  from now,  $\gamma_1$  holds continuously), Counter (Fig. 3(e) requiring that a condition  $\gamma_2$  has been satisfied a number of times between  $min$  and  $max$  during the last  $\tau$  time) and Previous (Fig. 3(f) requiring  $\gamma_2$  is satisfied that in the previous state).



**Fig. 4.** Model of Confusion behavior. The formula  $C_{[3,inf]}(P^{[10min]}(A))$  is expressed with use of visual formalism defined in Sect. 2.3.

that  $P_{[10min]}(A)$  became true at least 3 times in the past. In similar way, an expert user can provide a set of template formula to describe a specific disease. This form of representation is modelled in Fig. 4 based on the visual formalism introduced in Fig. 4.

## 4 Conclusion

We have presented a formal model of a monitoring service based on temporal logic. We have deployed a behavior checking approach, relying on description of critical behaviors based on a temporal logic formalism, where behavior checking is considered as a meta-algorithm which can be programmed through the specification of a temporal logic formula rather than through direct programming. Through the addition of a visual layer on top of the temporal logic based formalisms, the use of such an approach by users with lower technical skills (relating to temporal logic) and higher domain knowledge, e.g. Healthcare professional, is made easier. This results in the only requirement from the user who is using the tool being one relating to specifying the needs of the task need and user requirements. This implicitly avoids the user requiring to know or indeed specify all possible permutations that patient can perform, but instead only to provide details on sequences of conditions that detect situations of concern, thus introducing a paradigm shift in the approach to comparison user's behavior against an expected task model. In fact, it offers the possibility to specify a set of "abnormal" behaviors without requiring the specification of the full set of "normal" ones, as required using operational user task models such as Automata, Petri Nets and Activity Diagrams.

## References

1. Lymberis, A., de Rossi, D.: Wearable eHealth Systems for Personalised Health Management. In: State of the Art and Future Challenges, vol. 108, IOS Press, Amsterdam (2004)
2. Augusto, J.C., Nugent, C.D.: Designing Smart Homes, The Role of Artificial Intelligence. In: Augusto, J.C., Nugent, C.D. (eds.) Designing Smart Homes. LNCS, vol. 4008, Springer, Heidelberg (2006)
3. Bauchet, J., Mayers, A.: Modelisation of adls in its environment for cognitive assistance. In: Proc. of the 3rd International Conference on Smart Homes and Health Telematic (ICOST'05), Sherbrooke, Canada, pp. 3–10. IOS Press, Amsterdam (2005)
4. Boucharde, B., Giroux, S., Bouzouane, A.: A logical approach to adl recognition for alzheimer's patients. In: Proc. of the 4th International Conference on Smart Homes and Health Telematic (ICOST'06), pp. 1–8 (2006)
5. Manna, Z., Pnueli, A.: The Temporal Logic of Reactive and Concurrent Systems. Springer-Verlag, New York (1992)
6. Clarke, E.M., Emerson, E.A., Sistla, A.P.: Automatic verification of finite-state concurrent systems using temporal logic specifications. ACM Trans. Program. Lang. Syst. 8(2), 244–263 (1986)
7. Clarke, E.M., Grumberg, O., Hamaguchi, K.: Another look at ltl model checking. Formal Methods in System Design 10(1), 47–71 (1997)
8. Drusinsky, D., Shing, M.T.: Monitoring temporal logic specifications combined with time series constraints. J. UCS 9(11), 1261–1276 (2003)
9. Kim, M., Viswanathan, M., Kannan, S., Lee, I., Sokolsky, O.: Java-mac: A run-time assurance approach for java programs. Formal Methods in System Design 24(2), 129–155 (2004)

10. Naldurg, P., Sen, K., Thati, P.: A temporal logic based framework for intrusion detection. In: de Frutos-Escrig, D., Núñez, M. (eds.) FORTE. LNCS, vol. 3235, pp. 359–376. Springer, Heidelberg (2004)
11. Paggetti, C., Tamburini, E.: Remote management of integrated home care services: the dghome platform. In: Press, I. (ed.) From Smart Homes to Smart Care: Icost (2005), pp. 298–302 (2005)
12. Benedetti, M., Cimatti, A.: Bounded model checking for past ltl. In: Garavel, H., Hatcliff, J. (eds.) TACAS 2003. LNCS, vol. 2619, pp. 18–33. Springer, Heidelberg (2003)
13. Lusini, M., Vicario, E.: Engineering the usability of visual formalisms: a case study in real time logics. In: Catarci, T., Costabile, M.F., Santucci, G., Tarantino, L. (eds.) AVI, pp. 114–123. ACM Press, New York (1998)