The Use of Temporal Reasoning and Management of Complex Events in Smart Homes

Juan C. Augusto and **Chris D. Nugent** [1]

Abstract. Technological advancements have and will revolutionise the support offered to persons in their home environment. As the population continues to grow and in addition the percentage of elderly within the population increases we now face the challenge of improving individual autonomy and quality of life. Smart home technology offering intelligent appliances and remote alarm-based monitoring are moving close towards addressing these issues.

To date the research efforts on smart home technology have focused on communications and intelligent user interfaces. The trends in these areas must now, however, focus on the analysis on the data which is generated from the devices within the house as a means of producing 'profiles' of the users and providing intelligent interaction to support their daily activities. A key element in the implementation of these systems is the capability to handle time-related concepts. Here we report about one experience using Active Databases in connection with temporal reasoning in the form of complex event detection to accommodate prevention of hazardous situations.

1 INTRODUCTION

With changing population demographics and a shift towards a larger percentage of the population becoming elderly, we have witnessed a recent surge in the area of technology provision in the home environment to promote independent living. These developments are moving towards advanced technology solutions that can be 'embedded' into a person's normal living environment to support everyday activities. Such environments are referred to as 'smart homes' which are defined as living environments with the ability to proactively change the environment itself to provide services that promote an independent lifestyle for users [11]. Coupled with these general concepts is the notion of Ambient Intelligence which relates to a person's general surrounding environment. The vision of Ambient Intelligence is that before the end of the current decade people will be surrounded by intelligent and intuitive interfaces embedded in everyday objects and an environment recognizing and responding to the presence of individuals in a transparent manner [5].

To date many studies have addressed computing, communications and user interfaces in the Smart Home arena. Smart Homes can now provide the ability to monitor the whereabouts of the person [11], allow them to interact with various appliances in the home environment [12] and monitor health status [2].

Efforts must now, however, focus on the analysis on the data which is generated from the devices within the house as a means of producing 'profiles' of the users and providing intelligent interaction to change their environment to support independent living. A key element in the implementation of such systems is the capability to handle time-related concepts.

Here we bring together Active Databases (ADB) [14] and Temporal Reasoning (TR) [1] [6] applied to dynamic systems in order to merge advances in both areas, databases and Knowledge Representation (KR) and reasoning, that traditionally have worked separately despite the mutual benefits that collaboration may bring. Recently some work has been conducted on hybrid systems that obtain the best of both approaches [7] [8]. The research we are reporting here builds upon these experiences.

In the following section (section 2) we provide some details about our case study around which we have based our work. Then we explain the technology we use and how we apply it to solve the problem (section 3). Finally, we give some details about the future evolution of this system (section 4) and our conclusions are drawn in the final section (section 5).

2 A CASE STUDY

For the purposes of the current study we have modeled our Smart Home on a residential care institution in the United Kingdom. The environment is one of shared community care where approximately 30 individual apartments are contained within the same building each offering high technology solutions to promote independent living for the elderly. At the core of the environment is a central monitoring facility (CMF) which has the ability to detect all sensor and alarm events concurrently from each apartment.

Figure 1 depicts the typical layout of a person's apartment. The apartment has the ability to detect the movement of a person throughout the house and in addition provides a number of interfaces to monitor the person's interaction with various home appliances. Motion sensors have the ability to detect the presence of a person as they move around their living quarters. In the bathroom and bedroom, emergency switches provide the person with a direct means to raise an alarm to the CMF in instances of distress. A pressure pad sensor placed at the side of the bed detects a person getting out of bed. Taps in the bathroom and kitchen are fitted with sensors to detect when they have been turned on or off. Additionally, sensors detect if the fridge has been opened/closed or if the cooker has been turned on/off. Alarms may also be raised if the smoke or temperature sensors are activated.

In order to develop the necessary tools to process all of the information from these sensory elements and provide a means of support it is necessary to consider the possible activities that a person may undertake during normal and abnormal conditions. The key parameter is primarily to assess the current location of the person in the

[1] School of Computing and Mathematics, University of Ulster, UK. email: {jc.augusto, cd.nugent}@ulster.ac.uk

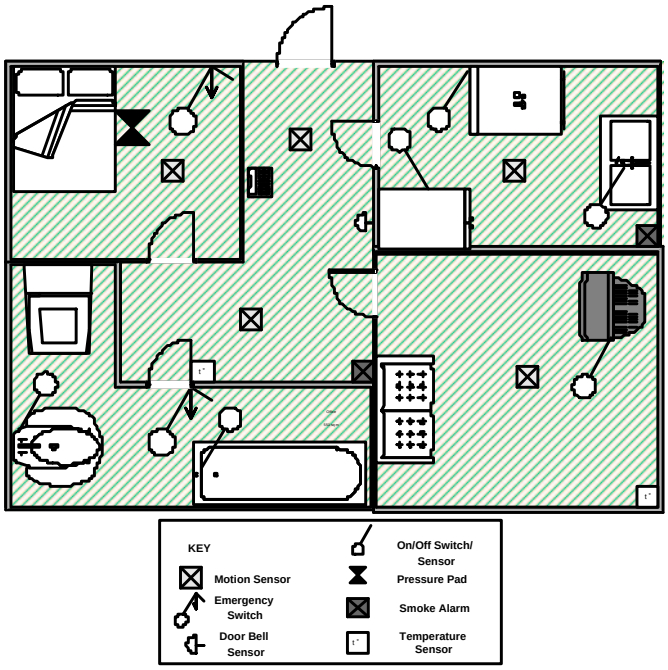

Figure 1. Layout of apartment indicating embedded technology to support independent living.

living environment. One this has been realized, it is then possible to determine if the sequence of actions the person becomes involved in are normal or abnormal.

In the first instance it is important to relate the whereabouts of the person in relation to the activities they have just performed. For example, if a person is in the bathroom and they wish to move to the kitchen, as an intermediate step they must pass through the reception area. This will result in the bathroom motion sensor initially being activated, then, the activation of the reception sensor and finally the kitchen sensor. Hence the sensor outputs provide detailed information on the movement and current position at any time of the person. This can be considered to be a normal condition. However, this can turn into a potentially hazardous situation if for example prior to leaving the bathroom the person turns on a tap and when entering into the kitchen turns on the cooker. A further third dimension to the problem can be introduced - the element of time. Re-considering the previous scenario, if the person returns to the bathroom and turns off the tap, yet leaves the cooker on for a 'significant' time period this can also be considered to be potentially hazardous.

The diversity of the types of information generated by the sensors provides a number of dimensions to the information which can be generated for a person. These can be considered to be (i) their whereabouts (ii) their interaction with appliances and (iii) the duration of these events. Hence, with such information, rules may be modeled and used to discriminate between normal conditions and potentially hazardous situations when an alarm condition should be raised.

3 ADB AND TEMPORAL REASONING

We first provide a general overview of the system which will be complemented with more detailed information in successive sections. At the core of the system there is an Active Database (ADB) manager (ADBM) with enhanced capabilities to detect complex events and contexts in order to distinguish between several situations of interest. Most importantly the system is used to anticipate potential or actual hazardous situations and intelligently discern how to best advise carers to increase safety and living standards for a person inside the monitored house. Although we consider a specific institution with specific sensory devices, we believe the general concept and the system itself to be adaptable to other similar contexts, especially in relation to healthcare, e.g. for monitoring in intensive care units or providing extra support at home to people with special needs.

ADB are strongly based on the concept of ECA (Event-Condition-Action) rules which will be the main focus of this paper. Basically an ECA rule has the form:

ON *event* IF *condition* THEN *action*

ECA rules in our system may cause the system to change its own perception about the state the house is in, for example from normal context to potentially hazardous context. In figure 2 we provide a general overview of the system where the ADBM will be given a set of ECA rules, a sequence of events coming one at a time, and the current state of the system. More than one event can occur simultaneously, for example smoke alarms, the doorbell and the phone can be activated at any time independently of other events. In our system they are all labeled with the same time of occurrence and then queued for batch processing.

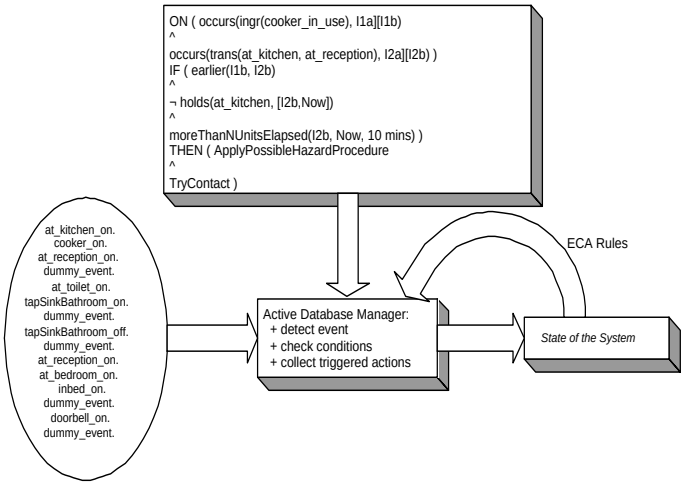

Figure 2. A general overview of the ADBM.

The current state of the system S is updated according to the incoming event and a possibly different state S' is obtained. When this incoming primitive event occurrs, new complex events may be detected. On the basis of new events detected and the conditions that are valid in S' one or more rules may be triggered. The action of the rule may contain instructions to carers or may produce a new change in the perception that the system has about the state of affairs in the house. This is represented in our system through the handling of contexts, e.g. it could be considered `normal_context`, `potentially_hazardous_context`, `danger_context`. These can be structured and inter-related as needed.

Assuming there is a non-empty and finite set of ECA rules: $R = \{R_1, R_2, \ldots, R_n\}$ with n a natural number, where each rule R_i for $1 <= i <= n$ is of the form specified in the Appendix A, the general algorithm of the process monitoring and triggering rules can be briefly described as follows. Each time an event

arrives the ON clauses are checked and from those rules in the subset of R: $R' = \{R_{i_1}, \ldots, R_{i_m}\}$ having a complex event definition detected the conditions stated in the IF clause are checked. For those rules in the subset of $R' : R'' = \{R_{j_1}, \ldots, R_{j_n}\}$ with their conditions satisfied the actions stated in the THEN clause are applied. Lets consider that the set of actions in the rules of R'' is $A'' = \{A_{j_1}, \ldots, A_{j_n}\}$, then in our system all the actions in A'' will be performed atomically in sequence.

The house being monitored is conceived as a dynamic system that goes through a sequence of states. Each state is defined by the place the person is in, which devices and alarms are activated, e.g. the temperature of the room, etc. Each of these parameters are discrete values, either boolean 'on'-'off' or within ranges like those used for temperatures. Each particular combination of possible values defines a potential state the system can be in. The initial state is that the person is in the reception area and all detectors are off. Other parameters depend on the environment and have to be set accordingly, e.g. temperature of the room. As explained above, from a perspective of knowledge representation, the evolution of the system will go through a cycle of sequence of states S_i provoked by the event occurrences E_i registered: $S_0, E_1, S_1, E_2, S_2, \ldots$. All the constituent parts of the theory (houses, sensors, ECA rules, possible events, possible states and time spanning since the initial state until the current time of use) can be defined as finite sets.

ECA rules are something all ADB systems accomodate for. An important difference between these systems is the definition of an *event*, a *condition* or an *action*. One important feature of our system is the language we use to specify the ON clauses and the IF clauses. Here we provide a general framework that is adaptable to expressiveness requirements demanded by different applications which is also more harmonic with previous research in knowledge representation for dynamic systems. Underlying our work is Galton's proposal to represent events and states inspired by natural language based research [6] (see also [10], [9]). We consider Galton's set of operators to offer a sound departing point which allows us to cleanly separate event detection from state holding checking. The set of operators provided in [6] are formally defined and provide concise ways to capture many situations of interest like instantaneous versus durative occurrences, repetitions, durations, sequences of events with different possible overlapping situations for the case of durative events, frequency of occurrence, past and progressive states, etc. One key predicate here is $Occurs(E, TR)$ stating that the event E occurs at/on the temporal reference TR depending on if TR is an instantaneous or a durative temporal reference. Instantaneous temporal references will be represented here as $i_1][i_2$ where $i_2 = i_1 + 1$. Durative temporal references will be represented as $[i_1, i_2]$ where $i_1 \leq i_2$. The predicate *Occurs* can be used to represent that a primitive event occurred, e.g. the location detector of the kitchen was activated, or that a non-primitive event occurs, e.g. detecting that somebody has been in bed at an unusual time for an unusually long period according to the person's normal habits. The terms event-types and event-tokens are used to distinguish between the general and abstract description of an event from its possible particular instances. To specify states we use the predicate $Holds(S, TR)$ stating that the state S holds at/on the temporal reference TR. This is one of the main components on the specification of IF clauses. Other predicates available allow us to obtain the current time of the system and to use a library of calendar-related predicates. Also we use some well known qualitative relationships providing temporal algebra operators considered in the literature of temporal reasoning for instant-instant, instant-interval, interval-instant [13] and interval-interval relationships [1].

Classic interesting AI-related problems arise due to the dynamics of the knowledge database. The triggering of one action can atempt to introduce in the knowledge base the negation of a previous stored fact, e.g. while making a transition from `normal_context` to `potentially_hazardous_context` and hence 'not normal'. To avoid ambiguities or inconsistencies we revise the system beliefs removing the contradictory concept before adding the incoming one. We assume persistency by default. For example, unless a sensor attached to a device is turned on the system will assume it is still off. Negation is handled as 'negation as failure'. Unless there is a record of it in the database, the system will assume that the event/state has not occurred/happened.

3.1 ECA Rules as a Bridge Between ADB and TR

Several proposals have been made in the last decade for ECA rule representation (see for example [3] and [4]). However it has been identified that further work is necessary in the area of complex event detection and ECA rules representation, see for example [7]. Here we provide a general language that is based on a formalization of temporal notions as referred in natural language expressions which allows us to distinguish two key notions events-forming and states-forming phenomena [6]. This will be used here to define the sublanguage to be used for complex event detection and for condition specifications. Previous proposals were focused on instantaneous events but the language we consider also accounts for durative events. This category of events possed interesting research challenges (see [7] and [8]).

To have a deeper understanding of the language defined below the reader is referred to one of the following documents: [6], [9] or [10]. Within the context of providing assistance for healthcare we can consider different categories of rules to improve security (e.g. detection of unexpected visitors and fire alarms), health (e.g. reminding that some medicine has to be taken), comfort (e.g. regulating temperature of the house), economy (e.g. turning off appliances that are not in use) and safe living (e.g. turning off dangerous devices that have been left unattended). We cannot include examples of all them here but we will illustrate the issues through the rules listed below (note that `earlier` and `moreThanNUnitsElapsed` are part of the calendar library):

If there is an ingression to a state where the cooker being in use followed by the person going out of the kitchen without returning to it for more than 10 minutes then apply the procedure to deal with a potential hazard and separately try contact.

```
ON (occurs(ingr(cooker_in_use), I1a][I1b) ∧
    occurs(trans(at_kitchen, at_reception),
                                 I2a][I2b))
IF (earlier(I1b, I2b) ∧
    ¬ holds(at_kitchen, [I2b,Now]) ∧
    moreThanNUnitsElapsed(I2b, Now, 10 mins))
THEN (ApplyPossibleHazardProcedure ∧
      TryContact )
```

If there is an ingression to a state where doorbell has been rung and is not followed by an ingression to a state were the person goes to the door in a reasonable time while it is known that the person is at home and is not hearing impaired then apply the procedure to deal with a potential emergency and separately try alternative ways of contact, e.g. visually or by phone.

```
ON (occurs(ingr(doorbell_rang), I1a][I1b) ∧
    ¬ occurs(trans(at_reception, at_outside),
                                  [I1b,Now]) )
IF (moreThanNUnitsElapsed(I1b, Now, 10 mins) ∧
    holds(at_home, [I1a,I1b]) ∧
    ¬ holds(hearingProblems, [I1a,I1b]))
THEN (ApplySecurityEmergencyProcedure ∧
      TryAlternativeWaysOfContact)
```

3.2 Non-Primitive Events Detection

A substantial concept in our system is about event handling. We consider primitive events, coming from the sensors within the house, and non-primitive/complex events, which are defined in terms of two or more primitive events and other complex events previously defined.

Examples of primitive events for our case study would be: `at_kitchen_on`, `cooker_on`, `cooker_off`, `alarm_kitchen_on`, `alarm_kitchen_off`, `at_living_on`, `tv_on`, `tv_off`, `doorbell_on`, `inbed_on`, `at_outside_on`. Part of our system is a program that will generate sequences of such events as if they are generated by the sensors within the house. The sequence of events can be used in two main ways within our system: a) batch mode, taken from a file, which could represent a recorded history or trail recorded from a prior interesting run. b) alternatively the system can benefit from a pseudo-random event generator. The algorithm will generate events in a random fashion but supplemented with domain knowledge which will help to provide a more realistic sequence of events.

From the initial state, a possible sequence of primitive events (preceded by their instant of occurrence) arriving at the system would be:

```
 0][1 at_kitchen_on     1][2 cooker_on
 2][3 at_reception_on   3][4 dummy_event
 4][5 at_toilet_on      5][6 tapSinkBathroom_on
 6][7 dummy_event       7][8 tapSinkBathroom_off
 8][9 dummy_event       9][10 at_reception_on
 10][11 at_bedroom_on   11][12 inbed_on
 12][13 dummy_event     13][14 dummy_event
 ...                    ...
```

Events with suffices '_on' and '_off' represent sensors changing values from 'off' to 'on' and viceversa, respectively. One special event we considered is the absence of an event which we denote as `dummy_event`. These represent that at a particular time no sensor changed its value.

Complex events can be detected using the language defined in [6], implemented and tested in [9] and [10], and included in the BNF in Appendix A of this article. It is not possible to give full coverage of all the operators with the space available so we just exemplify by providing two operators we included in the rules given above: `ingr` and `trans`. The following definitions are Galton's:

Ingr(S): occurs at $n][n+1$ iff $\neg S$ holds on $[n, n]$ and S holds on $[n+1, n+1]$.

Trans(S_1, S_2): If S_1 and S_2 are two mutually incompatible states, then the event $Trans(S_1, S_2)$ has:

1. an instantaneous occurrence at $n][n+1$ iff S_1 holds on $[n, n]$ and S_2 holds on $[n+1, n+1]$.
2. a durative occurrence in $[m, n]$ iff S_1 holds on $[m-1, m-1]$, S_2 holds on $[n+1, n+1]$ and both $\neg S_1$ and $\neg S_2$ hold on $[m, n]$.

Examples on how to use these operators were shown in the ECA rules given above. For example `occurs(ingr(cooker_in_use), 1][2)` denotes the instant in between those intervals at which the cooker was off and then on. `occurs(trans(at_kitchen, at_reception), 2][3)` denotes that there has been a sequence of sensor signals detecting the location of the person which implies the person has moved from the kitchen to the reception area. Assuming each primitive event takes one time unit to arrive then by the time the person is in bed at time 13 the condition that more than 10 units have elapsed since the person turned the cooker on without returning to the kitchen is satisfied. All the conditions will be fulfilled for the first of our two rules to be triggered:

```
          cooker_in_use
          |--------------------------->
            at_reception
              |-------|
                    not at_kitchen
              |-------------------------->
<---|---|---|---|---|--- ...---|---|--->
    1   2   3   4   5         13  14
              ......................
               (more than 10 units)
         I1a=1, I1b=I2a=2, I2b=3, now=14,
```

Note that these event operators can be composed of an arbitrary level of complexity and that here we are exemplifying simple situations compared with those which Galton's operators allow us to define and detect. As an example of a slightly more involved non-primitive event detection we can combine GSC (for General Sequential Composition of two different events) and Consec (for Consecutive occurrences of the same type of event) to define

```
occurs( po(gsc(change_of_medication,
               consec(high_blood_pressure,2)),
           [(2004, 8, 22, 13, 00, 00),
           (2004, 8, 23, 13, 00, 00)] )
```

to express that an occurrence of two consecutive high blood pressure records have been detected after a change in medication has been detected during the interval `[(2004, 8, 22, 13, 00, 00), (2004, 8, 23, 13, 00, 00)]`.

4 FURTHER WORK

The previous have detailed the rationale and developments of the integration of ADB and KR/TR for intelligent data analysis in smart home environments. Although the system at present physically supports these concepts we appreciate further work is still necessary.

Until now we have only considered the monitoring of one house. Once this process has been fully tested according to the verification and validation procedures the evolution of this system will be the generalization towards the concurrent monitoring of multiple houses. One immediate effect will be that more events will be produced at each time unit. From a logical point of view this will not add any major problem as they can be labeled by the house that is producing each event and kept separate. A multi-threaded process can apply the same algorithm to events coming from different houses and while this may bring performance concerns we believe the set of houses and the ratio of events produced by each for daily-life activities should not be a major challenge for today's processing capabilities. On the other hand there are some genuine logical extensions to be made to our

current framework as by considering more houses we bring the possibility that events in one house may have consequences on another house within the community. For example when a smoke detector is triggered in one house all the occupants of the other houses are also expected to be notified by an alarm. During the night, when fewer events have to be processed, the system can learn using various techniques, e.g. statistical analysis and case-based reasoning, to asses if there are any adjustments which need to be made to the ECA rules. At present our system cannot accommodate this feature but by addressing it will provide further flexibility, and challenges.

Some sensors, e.g. location detectors, may fail to detect movement. Until now we assume that extra hardware is used to ensure a person is detected in a room regardless brightness and other possible interferences. We are planning to replace such extra hardware with a more flexible diagnosis system.

5 CONCLUSION

As technology in the home environment to support independent living increases, research activities must now focus on analysis of all of the information gathered to provide intelligent means of changing a person's surrounding environment to assist with their daily activities. Research efforts to date have focused largely on the developments of communication systems and user interfaces, though little has been presented in the area of intelligent data analysis.

In the current study we have shown how through ADB and TR we can provide a framework to assess a person's whereabouts and activities in a home environment and facilitate support during potentially hazardous situations. Such developments address the growing problems we are currently witnessing in the increase in the size of the elderly population and the need to provide reliable and adaptable support systems.

From a computational perspective, an important feature of our approach is that we base our monitoring system on a language that addresses an area of computer science that still posses interesting technical challenges. The language we use to represent the ADB has clear syntax and semantics and the operators that it uses are grounded on the insights gained during the last decades on knowledge representation for dynamic systems and natural language analysis. We believe that this will allow for ease of future enhancement of the tool.

On the other hand we believe this work provides a natural blend of two areas, ADB and KR/TR that has been traditionally explored separately. This had undesirable consequences [7] that we are keen to avoid. From a more theoretical perspective this work is an attempt to reconciliate these two technical areas. From a more practical point of view this work brings the technology used in smart homes and in healthcare a step forward.

REFERENCES

[1] J. F. Allen, 'Maintaining Knowledge about Temporal Intervals', *Communications of the ACM*, **26**(11), 832 – 843, (1983).

[2] J. Barlow, S. Bayer, and R. Curry, 'Flexible homes, flexible care, inflexible attitudes? the role of telecare in supporting independence', in *Proceedings of the HAS Spring Conference 2003: Housing and Support*, pp. 1–21, (2003).

[3] E. Bertino, E. Ferrari, and G. Guerrini, 'An approach to model and query event-based temporal data', in *Proceeedings of TIME98*, pp. 122–131, (1998).

[4] S. Chakravarthy, V. Krishnaprasad, E. Anwar, and S. Kim, 'Composite Events for Active Databases: Semantics, Contexts and Detection', in *Proceedings of the International Conference on Very Large Data Bases (VLDB'94)*, pp. 606–617, Santiago de Chile, Chile, (1994).

[5] European Research Consortium for Informatics and Mathematics, 'Ambient intelligence', Technical report, (2001). 47.

[6] A. Galton, 'Eventualities', in *The Handbook of Time and Temporal Reasoning in Artificial Intelligence*, eds., Vila, van Beek, Boddy, Fisher, Gabbay, Galton, and Morris, MIT Press, (2004). (to be published).

[7] A. Galton and J. C. Augusto, 'Two approaches to event definition', in *Proceedings of 13th International Conference on Database and Expert Systems Applications (DEXA 2002), Aix-en-Provence, France, Springer-Verlag*, eds., R. Cicchetti A. Hameurlain and R. Traunmüller, pp. 547–556, (2002).

[8] R. Gómez and J.C. Augusto, 'Durative Event Composition in Active Databases', in *Proceedings of ICEIS'2004 (to appear)*, (2004).

[9] Rodolfo Gómez, Juan Carlos Augusto, and Antony Galton, 'Implementation and Testing for a Set of Event Detection Operators', Technical Report 398, School of Engineering and Computer Science, University of Exeter, United Kingdom, (2000). (`http://www.infj.ulst.ac.uk/~jcaug/rr398.pdf`).

[10] Rodolfo Gómez, Juan Carlos Augusto, and Antony Galton, 'Testing an Event Specification Language', in *Proceedings of the 13th. International Conference of Software Engineering and Knowledge Engineering (SEKE 2001)*, pp. 341–346, Buenos Aires, Argentina, (2001).

[11] S. Helal, B. Winkler, C. Lee, Y. Kaddourah, L. Ran, C. Giraldo, and W. Mann, 'Enabling location-aware pervasive computing applications for the elderly', in *Proceedings of the First Pervasive Computing Conference*, pp. 351–358, (2003).

[12] W. Mann and S. Helal, 'Smart phones for the elders: Boosting the intelligence of smart homes', in *Proceedings of the AAAI Workshop on Automation as Caregiver: The role of intelligent technology in Elder Care*, pp. 74–79, (2002).

[13] Itai Meiri, *Temporal Reasoning: A Constraint-Based Approach*, Ph.D. dissertation, University of California, 1992.

[14] N.W. Paton and O. Diaz, 'Active Database Systems', *ACM Computing Surveys*, **31**(1), 63–103, (1999).

A BNF OF THE LANGUAGE FOR ECA RULES

```
ECA_rule ::= ON (Event_Definition)
             IF (Condition_Definition)
             THEN (Action_Definition)
  Event_Definition ::=
      Primitive_Event | Composite_Event
    Primitive_Event ::= occurs(Event, TR)
    Composite_Event ::=
        Occurrence_Literal And_Or Occurrence_Literal
     Occurrence_Literal ::=
         Occurrence_atom | ¬ Occurrence_atom
     Occurrence_atom ::=
         occurs(Event, TR) |
         occurs(po(State), TR) |
         occurs(ingr(State), TR) |
         occurs(for(State, Number), TR) |
         occurs(consec(Event, Number), TR) |
         occurs(times(Event, Number), TR) |
         occurs(isc(Event, Event), TR) |
         occurs(gsc(Event, Event), TR) |
         occurs(rep(Event, Number), TR) |
         occurs(trans(State, State), TR) |
         ... (more can be defined)
  Condition_Definition ::= Condition_Literal |
      Condition_Literal And_Or Condition_Literal
    Condition_Literal ::=
        Condition_atom | ¬ Condition_atom
     Condition_atom ::=
         true | StateHolding |
         QualitativeRelations | CalendarFunctions
      StateHolding ::=
          holds(prog(Event), TR) |
          holds(perf(Event), TR) |
          holds(pros(Event), TR) |
          holds(freq(Event,(p,q)), TR) |
          holds(Context_ID, TR) |
          ... (more can be defined)
  Action_Definition ::= do(Action)
And_Or ::= ∧ | ∨
TR ::= Instantaneous | Durative
Instantaneous ::= a][b
Durative ::= [a, b] (with a, b natural numbers
                     and (a=b or a<b))
QualitativeRelations ::= (see section 3 above)
CalendarFunctions ::= e.g. nowIs(CalendarDate)
  where CalendarDate is given in the format
  (year, month, day, hours, minutes, seconds)
Primitive_Event ::= (events allowed in the application)
State ::= (states allowed in the application)
Context_ID::= (one of the contexts considered)
Action ::= (Actions allowed in the application)
Number ::= (a natural number)
```