HAL
open science

Automated Generation of Models of Activities of Daily Living

Jérémie Saives, Gregory Faraut

To cite this version:

Jérémie Saives, Gregory Faraut. Automated Generation of Models of Activities of Daily Living. 12th International Workshop on Discrete Event Systems-WODES 2014, May 2014, Cachan, France. pp.13-20. hal-00999505

HAL Id: hal-00999505

https://hal.science/hal-00999505v1

Submitted on 3 Jun 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Automated Generation of Models of Activities of Daily Living

Jeremie Saives * **Gregory Faraut** *

* *Automated Production Research Laboratory(LURPA), ENS Cachan, France (e-mail: firstname.lastname@lurpa.ens-cachan.fr).*

Abstract: In order to increase the safety of autonomous elderly people in their home, Ambient Assisted Living technologies are currently emerging. Namely, the recognition of their activities might be a way to detect eventual health problems, and can be performed in a Smarthome equipped with binary sensors. Hence, this communication aims at providing means to automatically generate a formal model of the Activities of Daily Living. A data mining approach in order to discover frequent habits of the observed inhabitant from a database of sequences of sensor events is proposed. Those frequent habits are then formally modelled using finite automata, leading to the construction of a map of habits mirroring the behaviour of the inhabitant. Such a model could then be used for online identification of habits, and even predictions of the upcoming behaviour. Results obtained on a case study are also presented.

Keywords: Activities of Daily Living, Discrete Event Systems, Data Mining, Automated Modelling

1. INTRODUCTION

The increase in life expectancy leads to an ageing population, whose needs in terms of health care are also increasing. In order to help the elderly to keep their autonomy at their home, emerging technologies for Ambient Assisted Living are being developed (Nehmer et al., 2006). The services provided by those technologies can be divided into three categories, defined in (Kleinberger et al., 2007): Emergency Assistance, Autonomy Enhancement and Comfort. Emergency Assistance is of crucial interest, and sustains the task of preventing health problems, or providing assistance should one occur. Monitoring the behaviour of the inhabitant allows for detection or even prediction of eventual irregularities. For that purpose, some tools are developed to identify the status of the inhabitant. The status could be split into the current position of the inhabitant, which can be evaluated by Location Tracking (Danancher et al., 2013), and the ongoing activity he is performing. A lot of works are hence dedicated to the recognition of Activities of Daily Living (*e.g.* waking up, showering, flushing the toilet, cooking . . .).

Those works require equipping the house with different kinds of sensors. In (Yu et al., 2009) cameras are used to detect whether the monitored inhabitant has fallen. In (Chernbumroong et al., 2013), wearable sensors are used to determine the posture and movement of the equipped inhabitant using accelerometers. In (Patterson et al., 2005), RFID sensors are used in order to detect which objects are being used by the inhabitant wearing the RFID reader. However, all those approaches are either intrusive or require the inhabitant to wear a special device. Another solution would be to use a Wireless Sensor Network, using a collection of various sensors with binary output, such as the universal switch sensors defined in (Intille et al., 2003). The network provides a flow of sensor events, each event containing very little information, but activities can be identified from a sequence issued of the flow.

In order to recognize activities out of a sequence of sensor events, which is assumed as a classification task (or supervised classification), all models share the same strategy : a learning phase on a set that has been studied by an expert, then the recognition phase, during which new set of data are classified, and parameters of the model are adapted in order to improve the recognition. The most classical classification model would be the Hidden Markov Model (HMM) (Cheng et al., 2010). Other models can be found in the literature. For instance, expert-designed boolean functions on the statuses of the sensors are used in (Botia et al., 2012), with an emphasis on the adaptation part. Expert models of activities are also designed in (Ros et al., 2013), each activity being described by an ordered set of action sequences, and the adaptation being achieved by learning automata. Data mining techniques are exploited in (Chikhaoui et al., 2011) to find frequent patterns in a sequence of events and compare them to expert models. Nevertheless, all those works focus only on classification, based on an expert knowledge.

The approach proposed in this paper is oriented towards providing a formal model of habits and activities discovered in a dataset of sequences, without *a priori* knowledge, hence in an unsupervised way (Dimitrov et al., 2010), and provides an automated method of building a map of the habits and activities of a monitored inhabitant. Such a model could be devoted to online real-time recognition of activities, and prediction when possible. Section II summarizes the approach. Section III deals with the data mining phase that discovers patterns in the dataset. Section IV presents the method and the formalism used to build the model of the patterns discovered. Section V provides an example of application to a case study.

2. OVERVIEW OF THE PROPOSED METHOD

The following method provides a way to get a formal model (a *map*) of all the habits of a monitored inhabitant, with possible applications such as real-time recognition of whether an activity is being executed or not, and prediction of what might be the next activity pursued. Such a model could therefore be used for the application of formal techniques such as diagnosis (in order to detect a faulty behaviour), and could detect either small incidents (such as an uncompleted activity) or slow behaviour alterations (such as an habit that disappears).

A smart home equipped with a binary sensor network will generate *events*, that correspond to a change of state of the sensors (which are assumed to be fault-free). Events are assumed to be not simultaneous, and an activity can thus be associated to a sequence of events. The behaviour of a human being is however arbitrary, hence multiple sequences of events can be images of the same activity, hardening the difficulty of building an expert model. A possibility is to use a training set (a few sequences that have already been observed), within which the sequences are still very different. Numerous observations would be required to depict all the possible sequences that depict the activity. Nevertheless, the sequences share sub-sequences, which are the fundamental basis of the activities, because they are very often played. Those frequent sequences would thus represent fundamental habits of the inhabitant.

The first contribution revolves around the discovery of these habits, which can be achieved by data mining methods (**Mining Step** of Figure 1). The dataset can stand in two forms. On one hand, it might be a unique sequence of events, within which frequent episodes can be found (Mannila et al., 1995),(Magnusson, 2000). A few days worth of observation could lead to a single sequence of events. On the other hand, one could use distinct sequences of events, as long as they represent the same temporal window, and compare each other in order to extract the habits. This last solution has been chosen. For the remainder of this work, it is also supposed that a single inhabitant is being monitored. Once the patterns have been found, habits can be identified within the set of patterns (**Identification Step** of Figure 1).

Then, the second contribution consists in the automated modelling of the discovered habits, which will lead to the building of a *map* of the habits of the inhabitant (**Automated Building Step** of Figure 1). This map allows for online recognition : when an event e occurs, the active states of the map change, leading to a set of habits that might be currently ongoing. The accuracy of the recognition remains out of the scope of this work.

It is worth noting that until the map is built, no expert knowledge has been injected in the model. The goal is to recognize frequent habits (that have already been observed and not expertly designed), instead of trying to split and classify every sequence observed into activities. Nevertheless, once the map is obtained, an expert can study it and determine which parts and which habits correspond to which activities. That part could also be helped by coupling the habit model with a location tracking model,

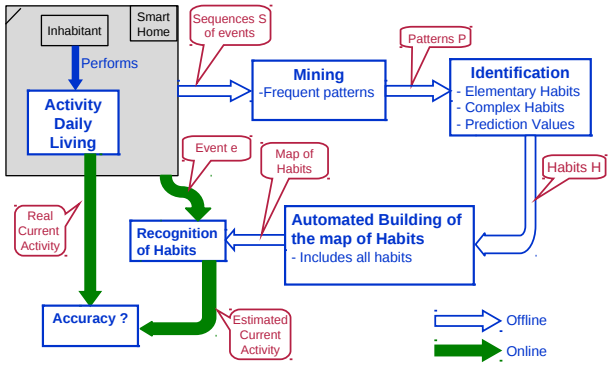

Fig. 1. Building a model from sequences of events issued of a Smart-Home

thus getting information on both the location and the activity of the monitored inhabitant.

3. DISCOVERING PATTERNS

Sequence mining is a specific field of data mining that deals with the search for relevant patterns in sequences and strings. Most of the methods used in sequence mining look for frequent itemsets in databases, in order to discover association rules between items frequently found together. Such techniques would be the *Apriori* algorithm found in (Agrawal and Srikant, 1994), or the use of FPTrees found in (Han et al., 2000). Since the ordering of the items is irrelevant, those methods consider only databases that have been previously lexicographically ordered. However, in the case of a sequence of events, since we want to discover succession of events which would be images of habits, the order is of great importance and an adaptation is required. Based on the Apriori algorithm, the proposed algorithm is thus designed to find continuous ordered patterns.

3.1 Definitions

Definition 1. Sequence and events

Let $D = \{S_i\}_{i=1..n}$ be a database containing n sequences. A *sequence* S_i is an ordered list of events such as $S_i=[e^i_1, e^i_2, e^i_3 \ldots, e^i_{l(S_i)}]$, where $l(S_i) \geq 2$ is the *length* of the sequence, and e^i_k is the k-th event of the sequence.

If Σ is the set of all the events that can be generated by the sensor network, then $\forall i \in [\![1, n]\!], \forall k \in [\![1, l(S_i)]\!], e^i_k \in \Sigma$.

Definition 2. Pattern

A *pattern* P is a sequence of events $P = [e^P_1, e^P_2, \ldots, e^P_{l(P)}]$, where $l(P) \geq 2$ is the length of the pattern. An event is not enough to define a pattern. Hence, a pattern is said *of elementary length* when $l(P) = 2$, which is the minimum length.

A pattern P is contained in a sequence S_i if, and only if $\exists s \in [\![1, l(S_i) - l(P) + 1]\!]$, $[e^i_s, e^i_{s+1}, \ldots, e^i_{s+l(P)-1}] = [e^P_1, e^P_2, \ldots, e^P_{l(P)}]$, *i.e.* P is a

continuous subsequence of S_i . It will be written $P \subset S_i$. A pattern can also be contained in another one.

Definition 3. Support

The *support* $Supp(P)$ of a pattern P is the number of sequences $S_i \in D$ in which P is contained, *i.e.* it mirrors the frequency of the pattern in the database. Thus, $\forall P, Supp(P) \in [\![0, |D|]\!]$, with $Supp(P) = 0$ meaning that the pattern is not present in the database, and $Supp(P) = |D|$ that the pattern is present in each sequence.

Let $Supp_{min} \in [\![0, |D|]\!]$ be a *minimum support*. Then, a pattern P is said to *satisfy* the minimum support iff $Supp(P) \geq Supp_{min}$. The minimum support defines the minimum presence for a pattern to be considered relevant.

3.2 Algorithm of sequence mining

The global goal of the algorithm of sequence mining is to discover every pattern contained in the database, and evaluate their support. Given a minimum support, it will return every pattern whose support is equal or higher. Running it multiple times for various values of minimum support is required in order to obtain a complete overview of the dataset.

Algorithm 1 consists of three steps, the last two being repeated until no new pattern is discovered :

Init An initialization step, to discover the patterns of elementary length, described in algorithm 2
- All patterns of elementary length that satisfy the minimum support are stocked in a list of elementary patterns

CandGen A candidate generation step, described in algorithm 3. The objective of this step is to make the patterns grow in length, in order to find the patterns of maximum length.
- If two already discovered patterns partially recover themselves, then a bigger pattern including both patterns is a potential candidate for being a pattern satisfying the minimum support itself.

Count A counting step, to evaluate the support of the candidates (function *EvalSupp* in the algorithms).
- Only the patterns satisfying the minimum support are kept and added to the list of discovered patterns.
- The shorter patterns that helped generate the newly discovered pattern are deleted, because no longer relevant.

Let $|\Sigma|$ be the size of the set of events, $|D|$ the size of the database, and $|S|_{max}$ the length of the longest sequence in the database. *Evalsupp* handles the problem of finding subsequences in the whole database, thus is of complexity $O(|D||S|^2_{max})$.
There are at most $|List_{disc}| = |\Sigma|(|\Sigma| - 1)/2$ patterns of length 2, so Algorithm 2 is of complexity $O(|\Sigma|^2|D||S|^2_{max})$. Then, Algorithm 3 being of complexity $O(|List_{disc}|^2|S|_{max})$, and $List_{cand}$ being necessary empty after $|S|_{max}$ candidate generations, Algorithm 1 is of complexity $O(|\Sigma|^4|D||S|^3_{max})$. Since it is of polynomial complexity, it can be used to handle large databases, the main limiting parameter being the maximum length of the sequences $|S|_{max}$.

Algorithm 1 Continuous Pattern Discovery

Require: Database D of n sequences, minimum support Supp_{min}, set of events Σ
  List_{disc}=Init(D, Supp_{min}, Σ)
  List_{cand}=CandGen(List_{disc})
  while $\text{List}_{cand} \neq \emptyset$ **do**
    for pattern_{cand} in List_{cand} **do**
      EvalSupp(pattern_{cand})
      if Supp(pattern_{cand})$\geq$ Supp_{min} **then**
        Add pattern_{cand} to List_{disc}
        for pattern_{disc} in List_{disc} **do**
          if $\text{pattern}_{disc} \subset \text{pattern}_{cand}$ **then**
            Delete pattern_{disc}
          end if
        end for
      end if
    end for
    List_{cand}=CandGen(List_{disc})
  end while
  return List_{disc}, List of discovered patterns satisfying minimum support

Algorithm 2 *Init*-Initialization function

Require: Database D of n sequences, minimum support Supp_{min}, set of events Σ
  List_{elem}=[]
  for Event1 in Σ **do**
    for Event2 in Σ **do**
      pattern_{test} = [Event1,Event2]
      EvalSupp(pattern_{test})
      if Supp(pattern_{test})$\geq$ Supp_{min} **then**
        Add pattern_{test} to List_{elem}
      end if
    end for
  end for
  return List_{elem}, List of patterns of elementary length

Algorithm 3 *CandGen*-Candidate Generation Function

Require: List of previously discovered patterns List_{disc}
  List_{cand}=[]
  for Pattern_1=$[e^1_1, e^1_2, \ldots, e^1_n]$ in List_{disc} **do**
    for Pattern_2=$[e^2_1, e^2_2 \ldots, e^2_m]$ in List_{disc} **do**
      if $[e^1_2, \ldots, e^1_n]$=$[e^2_1, \ldots, e^2_{m-1}]$ **then**
        Add $[e^1_1, e^1_2$=$e^2_1, \ldots, e^1_n$=$e^2_{m-1}, e^2_m]$ to List_{cand}
      end if
    end for
  end for
  return List_{cand}, list of possible patterns

Example 1 Let D be the following dataset, over the set of events Σ ={1,2,3,4,5,6,7,8} :

- S_1=[1,2,3,4,5,6]
- S_2=[1,2,3,4,7,8]
- S_3=[1,2,4,7,8,5]
- S_4=[1,2,3,7,8,4]

Let the minimum support be 2. Then :

Init List_{elem}=[1,2 ; 2,3 ; 3,4 ; 4,7 ; 7,8]

CandGen [1,2] and [2,3] overlaps, thus leading to [1,2,3] being a potential candidate, and so on.
List$_{cand}$=[1,2,3 ; 2,3,4 ; 3,4,7 ; 4,7,8]
Count Supp([3,4,7])=1, hence pattern [3,4,7] does not satisfy the minimum support, and so on.
List$_{disc}$=[1,2 ; 2,3 ; 3,4 ; 4,7 ; 7,8 ; 1,2,3 ; 2,3,4 ; 4,7,8].
[1,2] is contained in [1,2,3], can thus be cleaned out, and so on. Hence : List$_{disc}$=[1,2,3 ; 2,3,4 ; 4,7,8]
CandGen List$_{cand}$=[1,2,3,4]
Count List$_{disc}$=[1,2,3,4 ; 4,7,8]
CandGen List$_{cand}$=$\emptyset$

Finally, List$_{disc}$=[1,2,3,4 ; 4,7,8] contains the largest patterns whose minimum support is 2.

By repeating for supports 3 and 4 :

- List$_{Support4}$ = [1,2]
- List$_{Support3}$ = [1,2,3 ; 7,8]
- List$_{Support2}$ = [1,2,3,4 ; 4,7,8]

Now that all patterns and their supports have been found, each pattern could be treated individually, and independent models for recognition could be built. However, they are strongly dependent, since the shortest patterns are often included in longer pattern of lowest support. In order to build model of activities, or study the links between the habits, those inclusions have to be considered.

3.3 Structuring discovered patterns and building habits

Generally speaking, the discovered patterns follow a tendency : the longer the pattern, the lower the support. Hence, short patterns with high support depict a fundamental *habit* of the observed inhabitant. Those habits are often contained in longer patterns of lower support. Hence, these latter patterns are an image of *activities*, built out of habits. Multiple fundamental habits form an activity, and there are multiple long patterns that can represent the same activity.

As a pattern can be contained in a sequence, it can also be contained in another discovered pattern. Should a pattern not contain any other discovered pattern, it will be called *elementary*. Patterns of elementary length can not contain another one, and are thus natively elementary. A pattern can only be contained in a pattern of lower support. A pattern containing at least an other one will be called *complex*. A complex pattern might contain more than one other pattern, and thus the decomposition into smaller patterns is not unique. Nevertheless, Algorithm 4 can be used in order to find one of those possible decompositions for each complex pattern.

Algorithm 4 handles every pattern one after another, following a decreasing order of support. The first patterns handled are those of highest support. For the following patterns, the algorithm checks whether one of the already structured patterns, hence of a higher support, is contained within. When no more already structured patterns can be found in the handled pattern, it is added in its structured form to the list of structured patterns.

Let n be the number of discovered patterns, and L the length of the longest discovered pattern. Algorithm 4 is of complexity $O(n^2L^2)$, hence polynomial as well as Algorithm 1.

Algorithm 4 Building the hierarchy of patterns

Require: List of patterns and their supports List$_{patterns}$
  List$_{str}$=[ ]
  for support=Support$_{max}$ **to** 2 (decreasing) **do**
    for pattern $\in$ {List$_{patterns}$ } such that Supp(pattern)=support **do**
      while $\exists$ pattern$_{str}$ $\in$ List$_{str}$ such that pattern$_{str}$ $\subset$ pattern **do**
        Substitute pattern$_{str}$ in pattern
      end while
      Add pattern to List$_{str}$
    end for
  end for
  return List$_{str}$, List of structured patterns

Example 2 Let List$_{patterns}$ be the following :

- P$_1$=[1,2], Support : 10
- P$_2$=[4,5], Support : 10
- P$_3$=[1,2,3], Support : 8
- P$_4$=[6,7,8], Support : 5
- P$_5$=[1,2,3,4,5], Support : 4

Then, using algorithm 4 :

P$_1$ (natively) Elementary pattern.
P$_2$ (natively) Elementary pattern.
P$_3$ P$_1$ is included in P$_3$ $\rightarrow$ P$_3$=[P$_1$,3]
P$_3$ No more pattern is included in P$_3$
P$_4$ Elementary pattern.
P$_5$ P$_1$ is included in P$_5$ $\rightarrow$ P$_5$=[P$_1$,3,4,5]
P$_5$ P$_2$ is included in P$_5$ $\rightarrow$ P$_5$=[P$_1$,3,P$_2$]
P$_5$ P$_3$ is included in P$_5$ $\rightarrow$ P$_5$=[P$_3$,P$_2$]
P$_5$ No more pattern are included in P$_5$

Finally, and after the conversion of patterns into habits, List$_{str}$ is :

- H$_1$=[1,2], Support : 10
- H$_2$=[4,5], Support : 10
- H$_3$=[H$_1$,3], Support : 8
- H$_4$=[6,7,8], Support : 5
- H$_5$=[H$_3$,H$_2$], Support : 4

On one hand, habits H$_1$ and H$_2$ can be considered as fundamental, with high support. On the other hand, H$_5$, which consists of the succession of two habits, can be considered as a way to perform an activity. H$_5$ also provides information on the observed behaviour : it states that sometimes, H$_3$ is followed by H$_2$. Hence, should H$_3$ be observed, there is a chance for H$_2$ to start next. The structure underlying the complex patterns can thus efficiently be exploited for a predictive purpose.

3.4 Predicting the next habit

Suppose that, during an on-line recognition phase, the generation of an event has led to a habit being recognized. That habit might be included in another one of lower support. Using this available knowledge of the hierarchy of the habits of the inhabitant, an operator to estimate what event and/or habit comes up next is defined.

Definition 4. Let $\{H_i,H_j\}$ be two habits and e an event such that $[H_i, e] \subset H_j$. Let D_i be the restriction of the database to the sequences that contain the habit H_i, and $Occ(H_i \in S_k)$ be the number of occurences of the habit H_i in any sequence S_k. The operator *pred* is defined as following :

$$pred(H_i \rightarrow e) = \frac{1}{|D_i|} \sum_{S_k \in D_i} \frac{Occ(H_j \in S_k)}{Occ(H_i \in S_k)}$$

Notice that $|D_i|$=Support(H_i). Hence, if every habit has been observed only once in every sequence, then $pred(H_i \rightarrow e) = \frac{Support(H_j)}{Support(H_i)}$. The definition can be extended to the succession of two habits, if e is the first event of a second habit also contained in H_j.

Example 3 Looking at *Example 2*, habits H_3 and H_5 are complex and thus give possibility for a prediction value. It is assumed that these habits have been observed at most once in every sequence. Hence :

- $pred(H_1 \rightarrow 3)$= 8/10 = 80%
- $pred(H_3 \rightarrow H_2\ (4)\ )$= 4/8 = 50%

At the end of this section, the raw data has been explored, patterns have been extracted and structured, and the structure provides prediction information.

4. AUTOMATED BUILDING OF A MODEL

The objective of this section is to propose a model that will be able to identify when a pattern has been reproduced in an online real-time flow of sensors events. Hence, when a new event is generated, the model must react accordingly, by identifying which patterns could currently be at stake, which ones might have just begun, and which ones have just ended. For that purpose, each pattern could, in its elementary form, be modelled by a Finite-State Automaton.

For further applications, the model should also provide a prediction of the upcoming events it expects, when possible. The structure discovered between the patterns needs therefore to be exploited, leading to the choice of an extended class of automata in order to keep that structure, which is presented in the following.

4.1 Extended Finite Automata

The formalism of the Extended Finite-state Automata (EFA), as defined in (Sköldstam et al., 2007) is recalled here :

Definition 5. A (Deterministic) Extended Finite-state Automaton is a 7-tuple : $(Q \times V, \Sigma, G, A, \delta, (q_0 \times v_0), (Q_m \times V_m))$, with :

- Q x V is an extended set of states, where Q is a finite set of locations and V the finite domain of definition of the variables
- Σ an alphabet (non empty set of events)
- G a set of guard predicates over 2^V
- A a set of actions over V
- $\delta : Q \times \Sigma \times G \times A \rightarrow Q$ a transition function
- (q_0, v_0) the initial state
- $(Q_m, V_m) \subset (Q \times V)$ a set of marked states

Σ is the alphabet that contains every event that can be generated by the binary sensor network. Let $q \in Q$ be the current state, $q' \in Q$ be another state such that $\exists(e, g, a) \in \Sigma \times G \times A, \delta(q, e, g, a) = q'$. Then, if event e occurs, the transition will be fired if and only if the guard g is satisfied. The current state of the automaton will then be q', and the action a will be executed, changing the values of the variables.

4.2 Modelling elementary patterns

Let P=$[e_1, e_2, , \ldots, e_n]$ be an elementary pattern of length n. In order for the automaton to identify in real-time if that pattern has been played, it should satisfy a few properties:

- As long as the pattern has not started (e_1 has not occurred yet), the automaton should stay in its initial state
- If the pattern has been completed, the current state of the automaton should be marked
- If the pattern is interrupted by an unexpected event, the automaton should react accordingly : if e_1 occurs, the pattern might have started again before having been completed ; if another unexpected event occurred, the pattern has stopped, and the automaton should be in its initial state again.

Figure 2 proposes such an automaton for $n = 4$. Let us say that the current state is $P_1.2$, *i.e.* the last events that occurred have been [1,2]. The transition function states : $\delta(P_1.2, 3, ., .) = P_1.3$, $\delta(P_1.2, 1, ., .) = P_1.1$ and, $\forall e \in \Sigma - \{1, 3\}, \delta(P_1.2, e, ., .) = P_1.0$. If 3 occurs, the current state of the automaton becomes $P_1.3$ and the pattern keeps being identified. If 1 occurs, the pattern starts again, and if any other event happens, the pattern halts, and the automaton reaches its initial state.

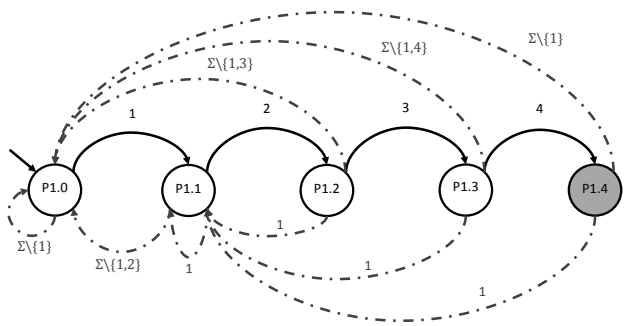

Fig. 2. EFA modelling the elementary pattern [1,2,3,4]

Elementary patterns can be modelled by simple Finite-State Automata, for no variables are required. However, when complex patterns are concerned, the need for variables will arise, as will be shown in the next subsection.

4.3 Modelling complex patterns

The construction process of an automaton in order to model a complex pattern is also more complex. The patterns must be distinguishable, and there must be no false detection. Three cases have to be considered :

Case 1 The contained pattern shares the first event with the complex pattern. It is impossible to distinguish the two patterns at the beginning, hence they share the first states and transitions. The complex pattern becomes a prolongation of the contained one. No variables are required in this specific case. See Figure 3 for an example with P_1=[1,2] being an elementary pattern contained in P_2=[P_1,3,4].

This complex model is no longer deterministic, because more than one transition can be activated on the occurrence of one single event. It is however easily transformable to a deterministic automaton by classical methods. For the sake of visibility, the automata will nevertheless be shown in their non-deterministic form. One must consider that a set of states are current at any time instead of a single one.

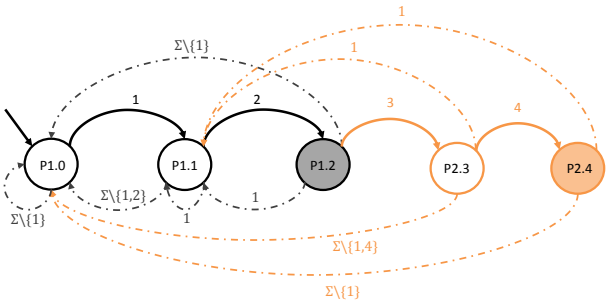

Fig. 3. EFA modelling the complex pattern P_2=[P_1,3,4]

Case 2 The contained pattern shares the last event with the complex pattern. In order to distinguish the patterns, the last state, which is a marked state, is duplicated. However, should only the contained pattern be played, the containing pattern must not be recognized, thus requiring the creation of a variable and a guard on this specific transition. The variable should be proper to the complex pattern, and indicate whether it has actually started and is currently being recognized. It is set to 1 when the patterns starts, and reset to 0 when the pattern is interrupted or ended.

See Figure 4 for an example with P_1=[3,4] being an elementary pattern contained in P_2=[1,2,P_1].

(1) Suppose that the sequence [1,2,3] has already been played. The set of current states of the automaton is $\{P_1.1\}$, and H2=1. Since $\delta(P_1.1, 4, ., .) = P_1.2$ and $\delta(P_1.1, 4, H2 == 1, .) = P_2.4$, if 4 occurs, the set of current states becomes $\{P_2.4, P_1.2\}$ and both patterns are recognized, which is the expected outcome.
(2) Suppose now that the sequence [2,3] has been played. The set of current states of the automaton is $\{P_1.1, P_2.0\}$, and H2=0. If 4 occurs, the new set of current states becomes $\{P_2.0, P_1.2\}$, and only P_1 is recognized, which is the expected outcome.

Case 3 The complex pattern contains two patterns. The previous cases apply their rules if the first and/or the last event is shared. Supplementary transitions have to be created between the two patterns. If exactly one event is expected between these patterns, a transition labelled

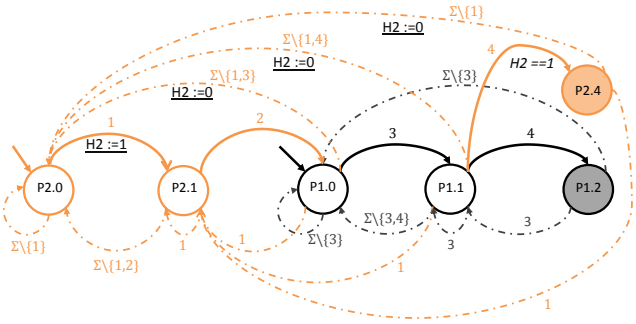

Fig. 4. EFA modelling the complex pattern P_2=[1,2,P_1]

with this event should be created from the last state of the automaton modelling the first pattern to the first state of the automaton modelling the second pattern. If more than one events are expected, additional states have to be added. If no events are expected, a transition labelled with the first event of the second pattern should be created from the last state of the first automaton to the second state of the second automaton.

See Figure 5 for an example of that latter case, with P_1=[1,2], P_2=[3,4] and P_3=[P_1,P_2]. A transition is created such that $\delta(P_1.2, 3, ., .) = P_2.1$.

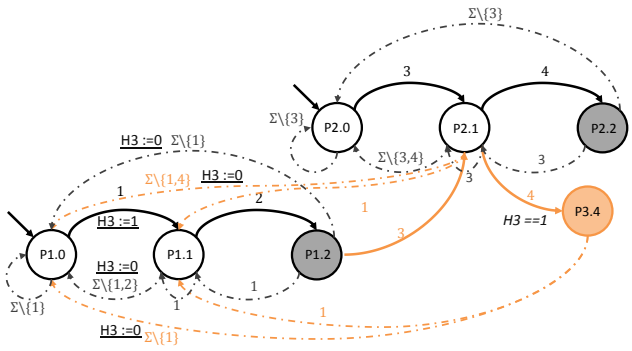

Fig. 5. EFA modelling the complex pattern P_3=[P_1,P_2]

4.4 Adding prediction knowledge to the complex models

Subsection 3.4 defined the *pred* operator. Let us say that $pred(H_i \rightarrow e)$ has been calculated. If H_i has been played, the automaton modeling H_i should be in its marked state, and a transition labelled with event e should leave this state in the model of the longer pattern H_j. This transition can be associated with the value of the prediction.

Case 1 See Figure 3. The transition $\delta(P_1.2, 3, ., .) = P_2.3$ can be associated with $pred(H_1 \rightarrow 3)$

Case 2 See Figure 4. No transition can be associated with a prediction value, because both habits end with the same event

Case 3 See Figure 5. The transition $\delta(P_1.2, 3, ., .) = P_2.3$ can be associated with $pred(H_1 \rightarrow H_2\ (3)\ )$

4.5 Building a map of the habits

A complex pattern, whatever its form, contains shorter patterns, and can be modelled by a complex EFA contain-

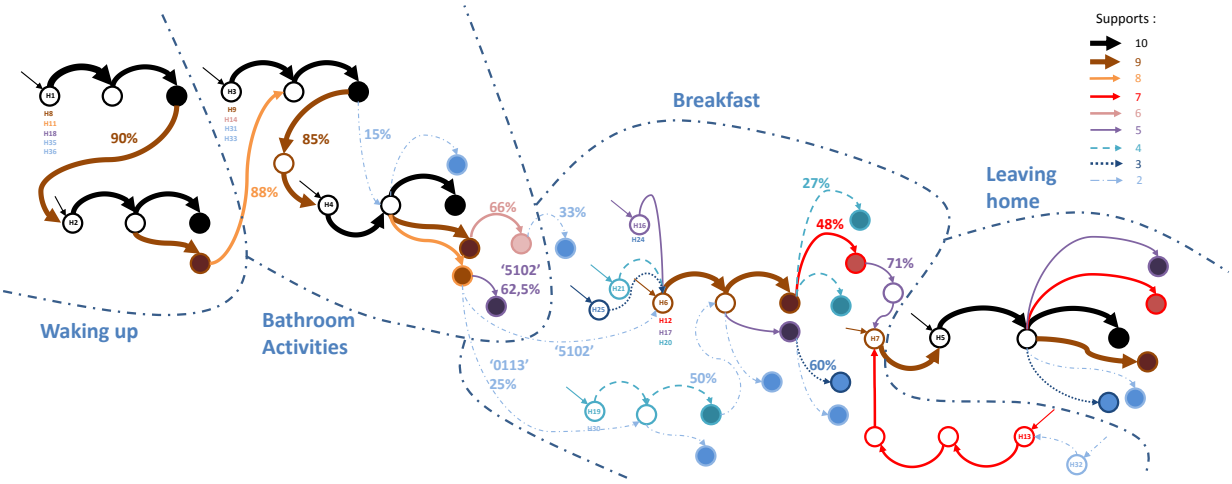

Fig. 6. Map of the habits of User 1 of the Domus Database

ing the EFAs of the shorter patterns. Since the operations presented in this section can be repeated, it is possible to build one single automaton representing all the links between the patterns discovered by the data mining step. Such a map can be used for some practical applications, that will be illustrated in the following section.

5. EXAMPLE OF APPLICATION

5.1 Domus database

The Domus Smarthome (Kadouche et al., 2010)(Chikhaoui et al., 2010) is an apartment of the University of Sherbrooke (domus.usherbrooke.ca) equipped with various sensors. Different users have been asked to perform the morning routine in the apartment (*ie* from waking up to leaving the apartment), while registering the evolution of the sensors values.

One user (the first one) has be chosen for the rest of the study. He performed the routine ten times, thus ten sequences of events can be compared in order to find frequent patterns. The events generated by the infra-red sensors have not been taken into account, because of their intrusiveness (a lot of irrelevant events generated), and they are mainly concerned by the localisation of the inhabitant.

5.2 Data mining results

The application of the method of section 3 led to the discovery of 36 patterns, summarized in Table 1

It can be noted that more and more complex patterns are discovered when the support gets lower. A few elementary patterns can still be found, but might be of little relevance in the behaviour of the inhabitant (for instance, an elementary pattern of length 2 and support 2 depicts a succession of two events that happened only twice, and can merely be considered an habit).

5.3 Mapping the habits of a person

The application of the method presented in section 4 led to the construction of the automaton presented in Figure 6. For a visibility purpose, only the main transitions, *i.e.* not the interruptions, have been represented. The automata representing low support elementary patterns not contained in complex patterns have as well not been represented.

Table 1. Patterns discovered for User 1

Support	Elementary	Complex	Min Length	Max Length
10	5	-	2	2
9	1	3	2	6
8	1	1	3	11
7	-	2	3	6
6	-	1	7	7
5	1	3	2	12
4	1	2	2	4
3	2	2	2	9
2	3	8	2	15

Real-time identification When the real-time observation of the inhabitant begins, the set of active states will be the set containing all the initial states. When an event occurs, the set of active sets is updated according to the transition function. When one or more marked states become active, one or more habits have been recognized.

Prediction When a marked state becomes active, a prediction might be available for each main transition issuing of this state. Some transitions leaving the same marked state can be labelled with the same event, and be assigned with a prediction value. For instance, when in the marked state of H11, the transition leading to the marked state of H18 and the transition leading to H6 are labelled by the same event '5102'. Thus, only the highest probability is relevant (here 62,5%). This issue is due to the non-uniqueness of the decomposition of the complex patterns. The decomposition of H36 did not take H18 into account, although it contains it.

Identifying activities In order to identify activities, it is possible to expertly analyse the map, and delimit areas. If there are active states that are not initial in such a delimited area (*i.e.* patterns being currently played), then the inhabitant is probably currently performing such an activity. Five activities are expected in the Domus dataset:

Waking up, Use toilet, Preparing Breakfast, Having Breakfast, Washing Dishes. The first two can be associated to areas of the map. The third and the fifth are associated to the kitchen activities area. The fourth would need the information on the location to be distinguished, thus requiring a location tracking model to be coupled.

Adaptation of the map After the observation is over, the observed sequence of events can be used to recompute the map. The patterns that have been recognized will see their support strengthened, confirming which are the fundamental habits of the inhabitant.

6. CONCLUSION

Using low-level knowledge issued out of a sensor network, this work proposed a method to build an automaton that models the behaviour of the monitored inhabitant, namely by representing his frequent habits that have been extracted by data mining. Such a model could be used for on-line recognition of the habits and activities; it might be of huge help in order to detect interruption of activities symptomatic of health problems. It could even further predict the expected behaviour of the inhabitant, leading to proactive actions in a smart-home and an improvement of the comfort.

The current perspectives of this promising work are twofold. On one hand, the accuracy of the estimation of the current activity should be evaluated, according to performance criteria, to validate the proposed model. On the other hand, some diseases could lead to a slow deviation of the behaviour of the inhabitant. Two separate maps built a few weeks apart could be different, and thus, a method to automatically detect major and light changes between the maps, mirroring behavioural variations, should be built. Such a method could also be used to distinguish multiple inhabitants.

REFERENCES

Agrawal, R. and Srikant, R. (1994). Fast algorithms for mining association rules. In *Proc. of 1994 Int. Conf. Very Large Data Bases (VLDB94), Santiago, Chile*, 487–499.

Botia, J., Villa, A., and Palma, J. (2012). Ambient assisted living system for in-home monitoring of healthy independent elders. *Expert Systems with Applications 39*, 8136–8148.

Cheng, B., Tsai, Y., Liao, G., and Byeon, E. (2010). HMM machine learning and inference for activities of daily living recognition. *J. Supercomput.*, 54, 29–42.

Chernbumroong, S., Cang, S., Atkins, A., and Yu, H. (2013). Elderly activities recognition and classification for applications in assisted living. *Experts systems with Applications 40*, 1662–1674.

Chikhaoui, B., Wang, S., and Pigot, H. (2010). A new algorithm based on sequential pattern mining for person identification in ubiquitous environments. *KDD Workshop on Knowledge Discovery from Sensor Data*, 19–28.

Chikhaoui, B., Wang, S., and Pigot, H. (2011). A frequent pattern mining approach for ADLs recognition in smart environments. In *Proc. of 2011 International Conference on Advanced Information Networking and Applications*, 248–255.

Danancher, M., Lesage, J., Litz, L., and Faraut, G. (2013). Indoor location tracking based on a discrete event model. In *Proc. of the IEEE International Conference on Systems, Man, and Cybernetics - SMC 2013*.

Dimitrov, T., Pauli, J., and Naroska, E. (2010). Unsupervised recognition of ADLs. In *Proc. of the 6th Hellenic conference on Artificial Intelligence: theories, models and applications*, SETN'10, 71–80.

Han, J., Pei, J., and Yin, Y. (2000). Mining frequent patterns without candidate generation. In *Proc. of 2000 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD00), Dallas, TX*, 1–12.

Intille, S., Tapia, E., Rondoni, J., Beaudin, J., Kukla, C., Agarwal, S., Bao, L., and Larson, K. (2003). Tools for studying behavior and technology in natural settings. In *Proc. of UBICOMP 2003*, 157–174.

Kadouche, R., Pigot, H., Abdulrazak, B., and Giroux, S. (2010). Support vector machines for inhabitant identification in smart houses. *UIC 2010*, 83–95.

Kleinberger, T., Becker, M., Ras, E., Holzinger, A., and Muller, P. (2007). Ambient intelligence in assisted living: Enable elderly people to handle future interfaces. In *Universal Access in Human-Computer Interaction*, 103–112. Springer.

Magnusson, M. (2000). Discovering hidden time patterns in behavior: T-patterns and their detection. *Behavior Research Methods, Instruments, & Computers*, 93–110.

Mannila, H., Toivonen, H., and Verkamo, A.I. (1995). Discovering frequent episodes in sequences. In *Proc. of KDD'95*, 210–215.

Nehmer, J., Becker, M., Karshmer, A., and Lamm, R. (2006). Living assistance systems: an ambient intelligence approach. In *Proc. of the 28th international conference on Software engineering*, ICSE '06, 43–50.

Patterson, D., Fox, D., Kautz, H., and Philipose, M. (2005). Fine-grained activity recognition by aggregating abstract object usage. In *Proceedings of the Ninth IEEE International Symposium on Wearable Computers*, 44–51.

Ros, M., Cuellar, M., Delgado, M., and Vila, A. (2013). Online recognition of human activities and adaptation to habit changes by means of learning automata and fuzzy temporal windows. *Information Sciences 220*, 86–101.

Sköldstam, M., Åkesson, K., and Fabian, M. (2007). Modeling of discrete event systems using finite automata with variables. In *Proc. of the 46th IEEE Conference on Decision and Control, New Orleans, LA, USA*, 3387–3392.

Yu, X., Wang, X., Kittipanya-Ngam, P., Eng, H., and Cheong, L. (2009). Fall detection and alert for ageing-at-home of elderly. In *Proc. of the 7th International Conference on Smart Homes and Health Telematics (ICOST'09), Tours, France*, 209–216.