Learning about preferences and common behaviours of the user in an intelligent environment

Asier AZTIRIA\textsuperscript{a,1}, Alberto IZAGUIRRE\textsuperscript{a}, Rosa BASAGOITI\textsuperscript{a} and Juan Carlos AUGUSTO\textsuperscript{b}

\textsuperscript{a} University of Mondragon, Spain
\textsuperscript{b} University of Ulster, United Kingdom

Abstract. Intelligent Environments are supposed to act proactively anticipating user’s needs and preferences in order to support people in their daily lives. Therefore, the capability of an Intelligent Environment to learn user’s habits and common behaviours becomes an important step towards allowing an environment to provide such personalized services. In this chapter, we use two scenarios, we highlight the opportunities given by learned patterns in order to interpret and understand the habits of the user which allows to automate services in an environment for the benefit of its user. We propose a system which learns user’s patterns of behaviour and an interaction system, based on speech recognition, which facilitates the use of such patterns in real applications.

Keywords. Intelligent Environments, pattern learning, interaction based on speech

1. Introduction

Intelligent Environments (IE) are digital environments that proactively, but sensibly, assist people in their daily lives [1]. They offer an opportunity to blend a diversity of disciplines, from the more technical to those more human oriented, which at this point in history can be combined to help people in different environments, at home, in a car, in the classroom, shopping centre, etc. Ambient Assisting Living (AAL) [2] refers to the potential use of an IE to enhance the quality of life of people, that is emphasizes the applications of IE for healthcare and wellbeing in general. For example it is well known that the majority of the elderly people prefer to live in their own houses carrying out an independent life as far as possible [3]. AAL systems aim at achieving that by:

- Extending the time people can live in their preferred environment.
- Supporting maintaining health and functional capability of the elderly individuals.
- Promoting a better and healthier lifestyle for individuals at risk.
- Supporting carers, families and care organizations.

Let us consider two scenarios which illustrates potential applications of IE for AAL:

\textsuperscript{1}Corresponding Author. E-mail: aaztiria@eps.mondragon.edu
Scenario 1: Michael is a 60 years-old man who lives alone and enjoys an assisting system that makes his daily life easier. On weekdays Michael’s alarm goes off few minutes after 08:00AM. He gets dressed and on Tuesdays, Thursdays and Fridays approximately 10-15 minutes later he usually steps into the bathroom. At that moment the lights are turned on automatically. The temperature of the water in the shower is already set according to Michael’s preferences (around 24-26 degrees in winter and around 21-23 degrees in summer). After he left the bathroom all blinds are opened. When he goes into the kitchen the radio is turned on so that he can listen to the headlines while he prepares his breakfast. Just before having breakfast the system reminds him that he has to measure his blood pressure and heart rate and this data is sent to his GP by using the broadband internet connection. He has breakfast and in 15-20 minutes he leaves the house. At that moment all lights are turned off and safety checks are performed in order to prevent hazardous situations in his absence (e.g. checking if cooker is turned on) and if needed the house acts accordingly (e.g. turning the cooker off).

Scenario 2: Sarah is a 75 years-old woman who is frail and hence needs help in order to carry out some daily tasks such as dressing up or having a shower. Fortunately, she lives in a modern building where people live in independent apartments share communal services of nursing-care, rehabilitation, entertainment, and so forth. The staff members in that accommodation know (by means of previous reports generated by the intelligent environment) that Sarah usually likes to have a shower just after getting up so that when her alarm goes off, nurses are ready to help her. In relation to her rehabilitation, Sarah has the freedom of choosing what type of exercises she wants to do. Specialized staff members, after monitoring and detecting Sarah’s preferences, design a personalized treatment much more suitable to her needs and preferences. At nights, on Tuesdays and Thursdays she likes watching her favourite sitcom so that she goes to bed around 11:00PM whereas the rest of the days she goes around 10:00PM.

Staff members are concerned about Sarah’s recent behaviour because the system has detected that although she takes pills everyday, she takes them just before having lunch, which is not desirable. Finally, doctors are concerned because there are some indications which show that she could be in the first stage of Alzheimer’s disease. In order to confirm or rule out these suspicions they are going to check if she carries out repetitive tasks in short periods of time or shows signs of disorientation (e.g. going back repetitively to places where she has been).

These scenarios show desirable environments that make the life of the user’s easier and safer. It is clear that knowing the users’ common habits or preferences gives either the environment (scenario 1) or staff members (scenario 2) the opportunity to act more intelligently according to each situation.

Discovering these habits and preferences demands a previous task of learning. In an Intelligent Environment, learning means that the environment has to gain knowledge about the user’s preferences, the user’s common behaviour or activity pattern in an unobtrusive and transparent way [4,5]. For example the environment of scenario 1 had to learn Michael’s morning habits, which are represented in Figure 1.

1.1. Advantages of learning patterns

Michael’s example shows how the environment, knowing his most common behaviour, can act proactively in order to make his life easier and safer. In this case acting proac-
tively means the environment can automatically turn on and off the lights, set the temperature of the water, open the blinds, turn on the radio and so on. Automation of actions and/or devices can be considered as positive side effect which can be obtained once the environment has learned his common habits and preferences.

In Sarah’s case, patterns are not used to automate actions or devices, but they are used to understand her behaviour and act in accordance with it. From Sarah’s perspective, the staff members are always at the correct place and time. On the other hand, for the staff members, the knowledge of habits of different patients allows them to organize their time in an efficient way as well as giving more personalized services to patients. The understanding of usual patterns also allows the detection of bad or unhealthy habits (e.g. she takes pills before having lunch).

The learning of patterns is not merely an optional aspect of the system which may bring some advantages to an intelligent environment, rather we consider that an essential contribution to the idea that an environment can be intelligent. It supports an environments which adapt itself to its users in an unobtrusive way and one where the users are released from the burden of programming any device [6]. Therefore the ability to learn patterns of behaviour is of paramount importance for the successful implementation of Intelligent Environments.

The remainder of this chapter is organized as follows. Section 2 describes the special features of Intelligent Environments which have to be considered when performing the learning process. Section 3 provides a literature review of different approaches suggested so far. In Section 4 we propose a new approach in order to learn patterns. Finally, Section 5 provides some overarching reflections on this topic.

2. Intelligent Environments’ special features

We overview the special features which make these environments different from others in the process of acquiring new knowledge.

2.1. Importance of the user

One of the hidden assumptions in Intelligent Environments is that unlike other current computing systems where the user has to learn how to use the technology, a fundamental
axiom in Intelligent Environments requests that the environment adapts its behaviour to
the user. Thus, the user gains a central role. The learning process has to be accomplished
as unobtrusively as possible, being transparent to the user. This implies that:

- Data have to be collected by means of sensors installed either on standard devices
  or in the environment (e.g. temperature sensors).
- System actions in relation to the user have to be performed maximizing the user’s
  satisfaction.
- User’s feedback has to be collected either through the normal operation of stan-
  dard devices (e.g. switch of lights) or through friendly interfaces such as mul-
  timodal user interfaces (e.g., through voice and image processing technologies)
  [7,8,9,10].

2.2. Collected data

As we suggested in the previous section, necessary data must be collected from sen-
sors installed in the environment. All patterns will depend upon the data captured. In
Michael’s case an example of collected data could be:

<table>
<thead>
<tr>
<th>Devices’ activations</th>
<th>Other sensors</th>
</tr>
</thead>
<tbody>
<tr>
<td>(date;device;status;value)</td>
<td>(date;device;status;value)</td>
</tr>
<tr>
<td>2008-10-20</td>
<td>2008-10-20</td>
</tr>
<tr>
<td>08:02:12;Alarm;on;100</td>
<td>08:01:53;TempBedroom;on;21</td>
</tr>
<tr>
<td>08:15:54;MotionCorridor;on;100</td>
<td>08:04:16;TempBathroom;on;18</td>
</tr>
<tr>
<td>08:15:55;BathroomRFID;on;Michael</td>
<td>08:12:26;TempBedroom;on;22</td>
</tr>
<tr>
<td>08:15:55;MotionBathroom;on;100</td>
<td>08:13:49;HumBathroom;on;51</td>
</tr>
<tr>
<td>08:15:57;SwitchBathroomLights;on;100</td>
<td>08:16:04;TempBathroom;on;20</td>
</tr>
<tr>
<td>08:31:49;MotionBathroom;on;100</td>
<td>08:19:04;TempBathroom;on;21</td>
</tr>
<tr>
<td>08:31:50;BathroomRFID;on;Michael</td>
<td>08:26:42;HumBathroom;on;53</td>
</tr>
<tr>
<td>08:31:50;MotionCorridor;on;100</td>
<td>08:28:12;TempBathroom;on;20</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>2008-10-21</td>
<td>2008-10-21</td>
</tr>
<tr>
<td>08:10:50;Alarm;on;100</td>
<td>08:05:16;TempBedroom;on;22</td>
</tr>
<tr>
<td>08:23:18;MotionCorridor;on;100</td>
<td>08:19:10;HumBathroom;on;50</td>
</tr>
<tr>
<td>08:23:19;BathroomRFID;on;Michael</td>
<td>08:22:42;TempBathroom;on;19</td>
</tr>
<tr>
<td>08:23:19;MotionBathroom;on;100</td>
<td>08:23:58;TempBathroom;on;20</td>
</tr>
<tr>
<td>08:23:21;SwitchBathroomLights;on;100</td>
<td>08:31:30;HumBathroom;on;53</td>
</tr>
<tr>
<td>08:30:52;shower;on;24</td>
<td>08:32:52;TempBathroom;on;22</td>
</tr>
<tr>
<td>08:48:33;MotionBathroom;on;100</td>
<td>08:38:10;HumBathroom;on;54</td>
</tr>
<tr>
<td>08:48:32;BathroomRFID;on;Michael</td>
<td>08:45:39;TempBathroom;on;21</td>
</tr>
<tr>
<td>08:48:32;MotionCorridor;on;100</td>
<td>08:49:02;HumBathroom;on;52</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2.2.1. Need of pre-processing

Data will be typically collected in a continuous way from different information sources.
Integrating data from different sources usually presents many challenges, because differ-
different sources will use different recording styles and also different devices will have different possible status [11]. Finally, as in other areas of computing, ‘noisy’ data, with missing or inaccurate values, will be common and finding out how to appropriately deal with that is another important challenge.

2.2.2. Nature of collected data

As far as the nature of information provided by sensors is concerned, different classifications can be used. But an interesting classification from the point of view of learning is that one which distinguishes between information about the user’s actions and information about the context.

Direct information of the user’s actions is usually provided either by sensors installed on devices which indicate when the user has modified the status of that device, or by movement sensors which detect where the user is. A simple example of it can be a sensor installed in a switch which indicates when the user turns on that light.

On the other hand there will be sensors which provide general information about the context. Sensors such as temperature, light or humidity do not provide direct information of the user’s actions, but they provide information about the context.

Finally, it is worth noting that due to the complexity of Intelligent Environments, external knowledge in combination with the collected data could be useful to carry out a successful learning process. Externally gathered knowledge will typically be domain-related knowledge such as:

- Patient’s medical background.
- Preferences specified in advance by the user.
- Calendar information (e.g. when the user goes on holiday)

2.2.3. Spatiotemporal aspects

Every act of a person situated in an environment is performed in a spatiotemporal dimension. The importance of considering spatial and temporal aspects in Intelligent Environments has been pointed out before by various authors [12,13,14].

As shown in the example given in Section 2.2 spatial information of different actions is given by the location of either devices (e.g. switch of the light) or user (e.g. motion in the bathroom) whereas temporal information is given by the timestamp of each action. As we will see later on, spatiotemporal aspects are always present in every step of the learning process.

2.3. Scheduling the learning process

Discovering common patterns of behaviour in recorded data is part of a complex process involving several stages. On one hand it is desirable the system acts as intelligently as possible from the very beginning. Typically these actions will not be as intelligent and efficient as those performed once the patterns of the user have been learnt, and we can expect minimal services at this stage.

On the other hand, once patterns have been discovered, it seems clear that those patterns will have to be continuously revised and updated because:

- The user can change his/her preferences or habits (e.g. Sarah now prefers other types of exercises for her rehabilitation).
• New patterns could appear (e.g. Sarah has started receiving some visits at the
weekends).
• Previously learned patterns were incorrect (e.g. the system wrongly learned that
Sarah likes going to bed at 9:00PM).

This adaptation process could mean the modification of parameters in a pattern we
have learnt previously, adding a new pattern or even deleting one. This is a sustained
process which will last throughout the lifetime of the environment. To do this effectively
the user's feedback is essential.

We can conclude that at least three learning periods seem to be necessary. The first
one is to act as intelligently as possible without patterns while the system is starting
to gather data. The second and main process deals with learning the user's common
behaviours and habits. Finally, while the system is acting in accordance with patterns
previously learned, it is necessary to update those patterns in a continuous way. This
chapter and the approach we suggest in Section 4 are mainly focused on the process of
learning patterns from collected data, that is the first stage.

2.4. Representation of the acquired patterns

Depending on the objectives of each environment, different representations can be used.
For example if the only goal is to provide an output given the inputs (e.g. switch the light
on given the current situation) it does not require a user comprehensible representation.
However, most of the times the representation of the patterns is relevant. In this cases a
human-understandable representation of the patterns is an essential feature for the suc-
cess of the system. Unlike the previous example where the internal representation is not
important, most of the times the internal representation of patterns is as important as the
final output. For instance, in order to understand the behaviour of a user, a comprehen-
sible representation is essential. Moreover, it maybe necessary for the system to explain
the decisions made to the user. Thus, the system could explain to the user that it turns
on the light of the bathroom because it has strong evidence that Michael usually do that.
Sometimes the output of the learning process has to be integrated into a bigger system or
has to be combined with other types of knowledge in order to make sensible high level
decisions.

Representing common behaviours by means of sequences of actions seems to be a
promising approach. Figure 1 shows Michael’s habits in a sequenced way. It is worth
mentioning that this type of representation allows to inter-relate actions among them (e.g.
'go into the bathroom’ and ‘turn on the light’). At the same time it allows to represent
the time relations using relative time references instead of absolute times (e.g. 'go into
the bathroom'; 2 seconds after; 'turn on the light’). Finally, conditions are necessary to
further specify the occurrence of this sequences. General conditions help to contextualize
the whole sequence (e.g. ‘On weekdays between 8AM and 9AM’ or ‘takes a shower on
Tuesdays, Thursdays and Fridays’).

3. State of the art

Intelligent Environments as a technological paradigm has attracted a significant number
of researchers and many applications are already being deployed, with different degree of
success. The complexity of these systems is due to the combination of hardware, software and networks which have to cooperate in an efficient and effective way to provide a suitable result to the user. Due to this complexity, up to now each project has focused upon different aspects of such complex architectures. In that sense, it is understandable, and even logical in some way, that the first developments have been focused upon the needs associated with hardware and networking as supporting infrastructure. This has resulted in simple automation that implements a reactive environment. Although to date, many researches [3,13,15,16] noted the importance of providing the environment with intelligence, little emphasis has placed in general upon the subject of learning ‘per se’. There are some notable exceptions and next we provide an overview of those focused on the Machine Learning techniques they use for learning the user’s patterns.

3.1. Artificial Neural Network (ANN)

Mozer et al. [17] and Chan et al. [18] were amongst the first groups that developed applications for Intelligent Environments where the user’s patterns were involved.

Mozer et al. designed an adaptive control system for the environment named Neural Network House, which considered the lifestyle of the inhabitants and the energy consumption. For that, they use a feed-forward neural network which predicted where the user would be in the coming seconds. Based on these predictions they controlled the lighting. Chan et al. also used ANNs for similar objectives. They developed a system that predicted the presence or absence of the user and his/her location. The system compared the current location with the prediction made by the system and said if the current situation was normal or abnormal.

Other authors have used ANNs in order to learn patterns related to users. A similar application which calculated the probability of occupation of each area of the house was developed by Campo et al. [19]. Boisvert et al. [20] employed ANNs in order to develop an intelligent thermostat. Monitoring the use of the thermostat the system automatically adapted it to the user’s preferences, reducing the number of interactions as well as reducing the energy consumption. See [21] for a survey focused on ANNs for Smart Homes.

3.2. Classification techniques

Classification techniques such as decision trees or rule induction have been used by other groups. The group that works in the environment named ‘SmartOffice’ [22] developed, using decision trees, an application which generated rules splitting situations where examples indicated different reactions. Considering Michael’s case, an example of it could be that when he goes into the bathroom he sometimes has a shower and sometimes he does not. Thus, the application developed by SmartOffice group would be able to discover the rules to separate these different reactions. In our case, the conditions would be that he has a shower on Tuesdays, Thursdays and Fridays.

Stankovski and Tmkoczy [23] generated a Decision Tree based on the training data. They considered that the training data set described normal events and the induced decision tree would therefore describe the normal state of the environment. Thus, they tried to detect abnormal situations, which would be out of the tree. Let us consider that we create a decision tree based on Michael’s normal behaviour. Thus, when Michael forgets
to switch the cooker off and he goes out the house the system would detect that this situation is out of the tree, so that it would be labelled as an abnormal situation and an alarm would be issued.

3.3. Fuzzy rules

Researchers at Essex’s iDorm lab have given prominence to the problem of learning, being one of the most active groups in this area. They represented the user’s patterns by means of fuzzy rules. Their initial efforts [6] [24] were focused on developing an application that generated a set of fuzzy rules that represented the user’s patterns. Recording the changes caused by the user in the environment, they generated membership functions as well as fuzzy rules which mapped those changes. Besides, they defined a strategy to adapt such rules based on negative feedback given by the user.

Vainio et al. [25] also used fuzzy rules to represent the user’s habits. In contrast to the approach followed in the iDorm project they manually constructed the membership functions and they used reinforcement learning to replace old rules in order to avoid that a single override event had large impact unless it lasts for a significant amount of time.

3.4. Sequence discovery

The group that is working in ‘MavHome’ and ‘CASAS’ environments is one of the most active groups. The first applications developed by this group were oriented to build universal models, represented by means of Markov Models, in order to predict either future locations or activities [26]. In that sense, they carried out notable improvements developing applications to discover daily and weekly patterns [27] or to infer abstract tasks automatically, with the corresponding activities that were likely to be part of the same task [28].

One of the major contributions of this group is the association of time intervals between actions [29]. They were one of the first groups that considered relations between actions, representing these relations using Allen’s temporal logic relations [30]. Once time relations between actions were defined, they tried to discover frequent relations by means of sequence discovery techniques. Considering Sarah’s case, this approach would be able to relate the actions of ‘getting up’ and ‘having a shower’, establishing that ‘having a shower’ comes ‘after’ ‘getting up’.

3.5. Instance based learning

Researches at Carnegie Mellon University’s MyCampus lab developed a message filtering application using Case-Based Reasoning (CBR), which can be defined as an Instance based learning technique [31]. Based on the user’s preferences, showed in previous interactions, the system filtered the messages. They validated the system without and with the CBR module and participants’ satisfaction increased from 50% to 80%.

Another example of the use of CBR techniques in order to learn the user’s preferences is the UT-AGENT [32]. Recording the set of tasks that user carried out, the UT-AGENT tried to provide the information that user needed based on the information he/she used to ask in similar situations.
3.6. Reinforcement learning

Some of the groups we have previously mentioned, e.g. Neural Network House and SmartOffice, have developed a module based on reinforcement learning in order to add the capacity of adaptation to the environment.

Mozer et al. [33] used Q learning for lighting regulation. Taking as starting point that the user has no initial preferences, the system tried to minimize the energy consumption as long as the user did not express discomfort. Zainderbeg et al. [34], starting from a pre-defined set of actions, progressively adapted them to the user by giving rewards to the system associated with good decisions.

3.7. Summary of related work

As we have seen in the previous sections, different learning techniques have been used for developing different applications in Intelligent Environments. Analysing different applications, it seems clear that the use of different techniques is firmly conditioned by the specific needs of each environment or application.

Muller [35] pointed out that

‘In many research projects, great results were achieved ... but the overall dilemma remains: there does not seem to be a system that learns quickly, is highly accurate, is nearly domain independent, does this from few examples with literally no bias, and delivers a user model that is understandable and contains breaking news about the user’s characteristics. Each single problem favours a certain learning approach’.

The current state of the art shows that there is no a global or holistic approach yet. In that sense, given the strengths and weaknesses of different techniques, combining different techniques seems a promising approach. Next section shows a first approach of a system that combining different techniques to learn user patterns.

4. Sequential Patterns of User Behaviour System

Sequential Patterns of User Behaviour System (SPUBS) is a system that discovers the user’s common behaviours and habits (e.g. Michael’s common behaviour shown in Figure 1). It is mainly focused on discovering common sequences of actions from recorded data. Due to the complexity of the Intelligent Environments the architecture of the system demands an exhaustive analysis. Thus, we have created a three-layered architecture which allows us to distinguish those aspects related to particular environments from those aspects that can be generalized for all environments. Figure 2 shows the global architecture of SPUBS.

4.1. Transformation Layer

The objective of this first layer is to transform raw data, i.e. information collected from sensors, into useful information for the learning layer. As we are going to see in this section, most of the transformations carried out in order to get useful information are dependant on each environment. Therefore, although some general transformations can be defined, different environments will demand different transformations.
Considering the data shown in Section 2.2, the following sections suggest some basic transformations.

4.1.1. Inference of simple actions

Once data from sensors have been collected an important task is to infer meaningful information from these raw data. Sometimes the information provided by sensors is meaningful, for example:

```
from

2008-10-20T08:15:57, SwitchBathroomLights, on, 100

we infer

2008-10-20T08:15:57, BathroomLights, on, 100
```

In this case, the action itself is meaningful because we can directly infer from it the action of the user. But there are other actions that are quite difficult to infer from a simple activation of a sensor. Let us consider that the inference of the simple action ‘Go into the Bathroom’ is not possible from the activation of a simple sensor, so that it must be inferred combining different actions. The following example shows that there is a motion in the corridor, then the RFID tag installed in the door of the bathroom detects that it is Michael and then there is a motion in the bathroom. We can logically infer that Michael has entered the bathroom. Thus, the transformation of those three actions into only one meaningful action allows to annotate the sequence of raw data items with meaning.
from

2008-10-20T08:15:54, Motion Corridor, on, 100
2008-10-20T08:15:55, Bathroom RFID, on, Michael
2008-10-20T08:15:55, Motion Bathroom, on, 100

we infer

2008-10-20T08:15:55, Bathroom, on, 100

The most basic way of inferring these actions is by means of templates. Templates define what actions must be combined as well as which constraints are to be considered. Actions can be defined either as mandatory or as optional. As far as constraints are concerned, they can be related to the order of the actions or to durational aspects. ‘Go into bathroom’ action’s template is defined as:

‘Go into Bathroom (Bathroom, on,100)’

Actions:

- Motion Corridor (Mandatory)
- RFID Detection (Mandatory)
- Open Door (Optional if already open)
- Motion Bathroom (Mandatory)

Constraints:

Order

Motion Corridor < RFID Detection < Open door < Motion Bathroom

Time

\[ T_{\text{MotionBathroom}} - T_{\text{MotionCorridor}} < 3\text{ seg.} \]

At the end of this first step, all actions should be meaningful. It is clear that the definition of templates depends on particular environments because it is mainly influenced by sensors installed in the environment. It is worth mentioning that this step of inferring meaningful actions is important because once we have defined such actions the rest of the learning process will depend upon them.

4.1.2. Inference of complex actions

Inference of simple actions makes sure that, from that point on, all considered actions are meaningful. Once simple actions have been inferred, a similar process can be carried out in order to infer complex actions such as ‘prepare a coffee’ or ‘take a pill’. This inference could be necessary because simple actions do not always represent the type of actions we want to analyse.

As in the inference of simple actions, the most basic method is the use of templates, with only one difference. Whereas the former consists in combining raw data, in order to infer complex actions we will combine simple actions. ‘Prepare a coffee’ action’s template could be defined as:
‘Prepare a Coffee (PrepareCoffee, on,100)’

Actions:
- Put Kettle on (Optional)
- Open Cupboard (Optional)
- Get Coffee (Mandatory)
- Take a cup (Mandatory)
- Open fridge (Optional)
- Get Milk (Optional)

Constraints:
- Time
  \[ T_{\text{FirstAction}} - T_{\text{LastAction}} < 5\text{ min}. \]

Combining different actions into only one action does not mean the impossibility of defining its internal structure. For example, retrieving all cases that were labelled as ‘Prepare a coffee’ we can carry out a particular learning process and detect if there exists a pattern which defines how Michael prepares a coffee.

4.1.3. Nature of information

Once different actions have been identified, it is important to determine what type of information has been collected. As stated in Section 2.2.2, different sensors provide information of different nature, so that they will be used for different purposes in the learning process. For this first approach two different types of information are considered:

- Information related to the user’s actions. Simple actions detected by sensors (e.g. ‘turn the light on’), inferred simple actions (e.g. ‘go into bathroom’) or complex actions (e.g. ‘prepare a coffee’) that represent the user’s actions.
- Information related to context. Temperature, humidity, luminosity and some other type of sensors indicate the state of the context in each moment. Besides, temporal information will also be considered as context information. For instance, time of the day, day of the week or season information are interesting to contextualize actions.

4.1.4. Splitting actions into sequences

Data will be collected in a continuous way from sensors, so that they will be represented as a string of actions, i.e. without any structure or organization. The aim of this task is to structure the data collected from sensors, using that organization in order to add some meaning to the collected data.

In that sense, many different organizations can be suggested. The organization we have used is based on the facts that the user usually carries out actions in a sequenced way and actions are mainly influenced by previous and next actions. Thus, we split the string of actions into sequence, but instead of using a quantitative window-width, we use a relative and more flexible criteria that determine the end of one meaningful sequence and the beginning of a new one. For instance, going into bed and staying there for more than 2 hours or going out and staying out for more than 30 minutes are considered as ‘landmarks’ demarcating sequences.

This task is dependant on each environment because different environments will demand different criteria. For example, a criteria defined in a Smart Home will not have any sense in a Smart Car.
4.2. Learning Layer

The objective of this layer is to discover common behaviours and habits of the user, taking as starting point the information provided by the transformation layer.

First of all it is necessary to define what type of patterns we are trying to discover. In that sense, the objective of this first approach is to discover patterns whose representation allows their use in most of the applications that can be proposed in Intelligent Environments. Considering different applications (See Section 4.3), a comprehensible representation of patterns was defined as an essential aspect. It was also considered that the user’s patterns are best represented by means of sequences of actions, where actions are related temporally among them and conditions are defined in order to get more accurate patterns. An example of this type of representation was shown in Figure 1.

The core of this layer is $\mathcal{A}_{SPUBS}$, an algorithm that discovers patterns using the information coming from the Transformation Layer. The language $\mathcal{L}_{SPUBS}$ included within this layer provides a standard framework to represent patterns with a clear syntax and which facilitates the application of them. The essential components of this layer are shown in Figure 3. It is worth mentioning that unlike the Transformation and Application layers the components of this one are not dependant on particular environments, so that its design allows its use in other environments too.

4.2.1. Representing patterns with $\mathcal{L}_{SPUBS}$

Defining a language that allows to represent patterns of user behaviour in Intelligent Environments is necessary to have a clear and non ambiguous representation. The language (See Appendix A) integrated into our system, $\mathcal{L}_{SPUBS}$, is an extension of the language defined in [36] and it is also based on ECA (Event-Condition-Action) rules [37]. ECA rules allow to define what action has to be carried out when an event occurs under relevant conditions. Thus, the sequence would be represented as a string of ECA rules, named $ActionPatterns$, contextualized by means of general conditions. Thus, the sequence showed in Figure 1 would be represented as:
General Conditions

IF context (day of week is (=, weekday) & time of day is ((>,08:00:00) & (<,09:00:00)))

Sequenced Actions

(ActionPattern 1)
ON occurs (Alarm, On, t₀)
IF context()
THEN do (On, Bathroom, t) when t=t₀+[10min,15min]

(ActionPattern 2)
ON occurs (Bathroom, On, t₀)
IF context()
THEN do (On, BathroomLights, t) when t=t₀+2seg

(ActionPattern 3)
ON occurs (BathroomLights, On, t₀)
IF context (day of week is (=,(Tuesday, Thursday, Friday)))
THEN do (On, Shower, t) when t is after t₀

(ActionPattern 4)
ON occurs (BathroomLights, On, t₀)
IF context (day of week is (≠,(Tuesday, Thursday, Friday)))
THEN do (Off, Bathroom, t) when t is after t₀

(ActionPattern 5)
ON occurs (Shower, On, t₀)
IF context (day of week is (=,(Tuesday, Thursday, Friday)))
THEN do (Off, Bathroom, t) when t is after t₀

As well as providing a way of representing patterns, it makes sure patterns are clearly
specified and enable other technologies that can check their integrity [38].

General Conditions

The general condition part defines the conditions under which the whole sequence oc-
curs. These general conditions allow to contextualize the whole sequence using either
context or calendar information. In this sense, calendar information seems more promis-
ing than context information, but some sequences can also demand the use of context
information such as temperature or humidity.
**Event Definition**

The event part defined by the ON clause defines the event(s) that occurred and triggered the relation specified by the pattern. In the event definition, the event together with the status is defined. As patterns relate the user’s behaviours, the ON event(s) will contain the user’s actions. For instance, in Michael’s case the event of *ActionPattern1* is that the alarms has gone off.

\[
ON \text{ occurs (Alarm, On, } t_0) \quad \text{(Event 1)}
\]

**Condition Definition**

The IF clause defines the necessary condition under which the actions specified in the THEN clause is the appropriate reaction to the occurrence of events listed in the ON clause. Unlike global conditions, these conditions are specific for each *ActionPattern*, defining necessary conditions in order to relate the actions defined in the ON and THEN clauses. Due to the fact that an event-action relation is rather unlikely to be true under any circumstance, the identification of appropriate conditions is necessary in order to represent accurate patterns. In Michael’s example, the third *ActionPattern* shows that he only has a shower on Tuesdays, Thursdays and Fridays. Different types of conditions can be found and \( \mathcal{L}_{SPUBS} \) allows to represent all them. Below we provide some examples:

\[
\begin{align*}
\text{IF context (Day of week is } =, \{ \text{Tuesday, Thursday, Friday} \} \text{) (Condition 1)} \\
\text{IF context (LivingRoom temperature is } <, 20^\circ C \text{) (Condition 2)} \\
\text{IF context (Time of day is } >, 13:00:00 \text{ and } <, 14:15:00 \text{) (Condition 3)}
\end{align*}
\]

Conditions are defined using an attribute and a value. Attributes can be either information coming from context sensors (e.g. temperature in Condition 2) or calendar information (e.g. day of week in Condition 1 or Time of day in Condition 3). Values can be either qualitative (e.g. Tuesday, Thursday, Friday in Condition 1) or quantitative (e.g. 20 °C in Condition 2). It is even possible to define a range of values (e.g. [13:00:00-14:15:00] in Condition 3).

**Action Definition**

Finally, the THEN clause defines the action that a user usually carries out given the General Conditions, the ON clause and IF clause. As well as defining the action and the status, it defines the time relation between the Event and the Action, being that relation either quantitative (e.g. 2 seconds in Action 1) or qualitative (e.g. ‘after’ in Action 2). The usefulness of each type of relation is different. Both define the user’s habits, but whereas quantitative relations can be used to automate action defined in the THEN clause, qualitative relations are hardly of any use for this purpose.

\[
\begin{align*}
\text{THEN do (On, BathroomLights, } t) \text{ when } t = t_0 + 2 \text{seg} \quad \text{(Action 1)} \\
\text{THEN do (On, Shower, } t) \text{ when } t \text{ is after } t_0 \quad \text{(Action 2)}
\end{align*}
\]
4.2.2. Learning patterns with \( \mathcal{A}_{SPUBS} \)

In accordance with \( \mathcal{L}_{SPUBS} \), we have developed an algorithm (\( \mathcal{A}_{SPUBS} \)) to discover patterns. The architecture of this algorithm is summarized in Figure 4.

Next, using Michael’s morning habits as an example, we will see how patterns are discovered using \( \mathcal{A}_{SPUBS} \).

**Identifying Frequent Sequences**

Once data have been collected from sensors and the user’s actions have been defined and split into sequences (See Section 4.1), the first step is to identify the set of actions that frequently occur together. If the user carries out a set of actions repetitively, i.e. that set of actions appear together in many sequences, it will be considered as a *Frequent Sequence*.

In order to get these *Frequent Sequences* we use the Apriori algorithm [39] for mining association rules. Taking as starting point the set of actions (split in sequences) coming from the Transformation Layer, this approach discovers the maximal set of actions that occur frequently. If a sequence is frequent or not is established considering the minimum coverage and support values required in each case.

In Michael’s case this first step would discover that many times (more than minimum coverage and support values demand):


occur together, so that they would be defined as a *Frequent Sequence*.

**Identifying Topology of Frequent Sequences**

The set of actions discovered in the first step only define the actions involved in the pattern. It seems logical that the user most of the times carries out that set of actions in a specific order. Although that order can sometimes be random, the objective of this step is to discover the most common order of each *Frequent Sequence*. Hence for each *Frequent Sequence*: 
We collect the sequences where actions included in the Frequent Sequence happened.
We represent the actions in the same order as happened.

Let us consider that following sequences represent Michael’s actions corresponding to the Frequent Sequence discovered in the first step:

Sequence 1:
'Alarm on' ; 'Bathroom on' ; 'BathroomLights on' ; 'Bathroom off';

Sequence 2:
'Alarm on' ; 'Bathroom on' ; 'BathroomLights on' ; 'Shower on' ; 'Bathroom off';

Sequence 3:
'Bathroom on' ; 'BathroomLights on' ; 'Bathroom off' ; 'Alarm on';

Sequence 4:
'Alarm on' ; 'Bathroom on' ; 'BathroomLights on' ; 'Shower on' ; 'Bathroom off';

Sequence 5:
'Alarm on' ; 'Bathroom on' ; 'BathroomLights on' ; 'Shower on' ; 'Bathroom off';

Once actions have been collected and represented we discover the topology of Frequent Sequences using an approach based on Workflow mining algorithms [40,41,42]. It is worth mentioning that, unlike the proposed example, the topology discovering process can be very complex. Unordered subsets of actions are one of the most important source of complexity. The user does not always carry out the same actions in the same order. Let us consider that the user is preparing a coffee. Sometimes he/she puts coffee into the mug before pouring hot water, whereas some other times he/she does it the opposite way. Our approach identifies these unordered sub-sequences and groups them without specifying any order.

Exceptions are another aspect to consider. In that sense, considering that the representation of all exceptions can make the topology too complex, we rule out those relations that are weaker than a required threshold. The possibility of varying that threshold allows to get topologies of any complexity. In Michael’s case, ‘Alarm on’ appears first in all sequences, except in the third one. Although that exception does not modify the topology, it modifies the frequency of relations, which indicates the strength of each relation. Once the topology has been discovered, Michael’s morning habit can be represented as shown in Figure 5 (frequency are defined only considering Sequences defined above):

Once the topology has been discovered it can be translated into $L_{SPUBS}$, defining the ON clause and part of the THEN clause of ActionPatterns. Thus, the topology shown in Figure 5 can be represented as (notice the ‘when ...’ part of the clause will defined in the next step of the algorithm):
Identifying time relations

In order to get complete and accurate patterns, time relations and conditions (general and specific) must also be discovered. The aim of this step is to identify the time relations between actions related in the previous step. Let us consider the following time distances (hh:mm:ss) between actions:

Sequence 1:
‘Alarm on’; ‘Bathroom on’; ‘BathroomLights on’; ‘Bathroom off’;
00:13:42 00:00:02 00:15:53

Sequence 2:
‘Alarm on’; ‘Bathroom on’; ‘BathroomLights on’; ‘Shower on’; ‘Bathroom off’;
00:11:29 00:00:02 00:07:31 00:17:42

Sequence 3:
‘Bathroom on’; ‘BathroomLights on’; ‘Bathroom off’; ‘Alarm on’;
00:00:01 00:03:12 00:10:42

Sequence 4:
‘Alarm on’; ‘Bathroom on’; ‘BathroomLights on’; ‘Shower on’; ‘Bathroom off’;
00:14:12 00:00:03 00:19:20 00:05:16

Sequence 5:
‘Alarm on’; ‘Bathroom on’; ‘BathroomLights on’; ‘Shower on’; ‘Bathroom off’;
00:12:38 00:00:02 00:02:58 00:24:06
Considering each ActionPattern and taking as starting point the time distances between actions, the algorithm will make groups taking into account the similarities among them and check. It will also check if there is any time distance that groups enough instances to consider it as ‘interesting’. The technique to make groups could be as complex as we can imagine. In this case the technique we have used is based on joining values that are within a range of established by (1):

\[
[min, max] = \bar{x} \pm (\bar{x} \ast tolerance) \quad \text{where} \quad \bar{x} = \frac{\sum_{i=1}^{n} a_i}{n}
\]  

(1)

with: tolerance = tolerated deviation from \(\bar{x}\) (%); \(a_i = \) time distance of an element; and \(n = \) number of elements

Let us consider ActionPattern2 and ActionPattern3. In both cases we will consider a tolerance of 50% and minimum level of 25%. In ActionPattern2’s case, the time distances between ‘Bathroom on’ and ‘BathroomLights on’ are depicted in Figure 6.

Grouping such values only one group, which covers all instances, is created with mean value ‘2 seconds’. In this case, it has been possible to define a quantitative value because \(mean\ value \pm tolerance\) groups 6/6 instances (100%<25%).

Figure 7 shows the time distances between ‘BathroomLights on’ and ‘Shower on’ (ActionPattern3). In this case there is no group with more than one instance. As neither of them is above the required minimum level (1/6 < 25%), it has not been possible to define a quantitative value. Thus, instead of defining the relation by means of a quantitative value we define it using a qualitative relation. As the topology represents the order of actions the only qualitative relation we consider is ‘after’.

Once the time relations have been discovered the ActionPatterns would be defined as follows:
(ActionPattern 2)

\textit{ON occurs (Bathroom, On, t_0)}

\textit{IF context()}

\textit{THEN do (On, BathroomLights, t) when t = t_0 + 2seg}

(ActionPattern 3)

\textit{ON occurs (BathroomLights, On, t_0)}

\textit{IF context()}

\textit{THEN do (On, Shower, t) when t is after t_0}

Identifying conditions

In the previous steps we have generated patterns relating, in terms of time, two actions (represented by the \textit{ON} and the \textit{THEN} clauses). Although \textit{ActionPattern 2} represents a time relation which groups all instances, it is very difficult to find relations based on only one relation. For instance in \textit{ActionPattern 3}, the relation ‘after’ defines well the sequences s2,s4,s5 whereas s1,s3 are not well-defined by that relation. The aim of this step is to find out, if possible, under what conditions a pattern appears or not. It will also try to identify general conditions in order to contextualize the whole sequence. Although in both cases calendar and context information will be used, different approaches are necessary due to their different nature. The discovery of specific conditions will be considered first.

For the purpose of discovering specific conditions, two tables, covered and non-covered, are generated. In the covered table there will be instances classified well by the pattern together with the calendar and context information collected when such actions happened, whereas the same information of instances where the patterns fails is registered in the non-covered table. Figure 8 shows context and calendar information related to \textit{ActionPattern 3}.

Dividing both tables, using the information they contain, allows to know when the pattern defines properly the relation between both actions. Considering our example, an option of separating both tables is considering the calendar information ‘day of week’. Thus, \textit{ActionPattern 3} can be defined as:
ON occurs (BathroomLights, On, t₀)

IF context (day of week is (=,(Tuesday, Thursday, Friday)))

THEN do (On, Shower, t) when t is after t₀

It is worth mentioning that adding these conditions does not increase the number of instances the pattern classifies well (it still covers the same instances s₂, s₄, s₅), but we make it more accurate, making sure that it does not include instances that do not have that pattern. The task of separating both tables has been considered as a classification problem. For that, we have used the JRip algorithm [11] which we have to modify as JRip provides rules with the only objective of separating both classes (covered and non-covered), whereas in our case it is desirable to obtain rules about the covered class. In this way we always get a set of conditions that indicates when a pattern defines well the relation, instead of a mix that indicates when it defines it well and when it does not.

Identifying general conditions demands a different approach as it is not a classification problem where there are two different classes to separate. In this case we only have context and calendar information collected when different instances of the sequence happened. The approach we are using is a very basic approach which creates ranges of values considering the parameters ‘day of week’ and ‘time of day’. Thus, in Michael’s case it discovers that all sequences occur on weekdays and between 08:00AM and 09:00AM, defining the following general condition:

IF context (day of week is (=, weekday) & time of day is ((>,08:00:00) & (<,09:00:00)))

4.3. Application Layer

Once patterns about the user’s common behaviour have been learned, they can be used for different purposes. Following we will discuss some of the most promising applications. But before we will see the interaction system we have developed and which can be used in any type of applications.

Given the importance of the user for the success of an Intelligent Environment, we considered essential for this application layer that it included allowing a friendly and easy way of interaction between the environment and the user. Thus, we have developed a Human-Computer Interaction system based on speech that allows user to interact with the patterns, based on the $L_{SPUBS}$ representation, discovered by $A_{SPUBS}$. The functionality we have developed is aimed at getting the user’s feedback in order to use his/her acceptance to automate actions (See Section 4.3.1 for further details). But it can be easily evolved in many different ways to suit the needs of different users in different environments.

As far as technical aspects are concerned, this interaction system has been developed using a speech synthesizer and a speech recognizer. In order to facilitate the integration with the rest of the system (developed in Java), we have chosen a synthesizer and recognizer written entirely in Java. The chosen speech synthesizer has been FreeTTS 1.2 whereas Sphinx-4 has been the chosen speech recognizer. Both FreeTTS and Sphinx-4 make the interaction with the user easier providing easy to use tools. Complications come

---

2 http://freetts.sourceforge.net/docs/index.php
3 http://cmusphinx.sourceforge.net/sphinx4/
mainly due to changing nature of Intelligent Environments. For example, the interaction system cannot know beforehand what devices are in the environment, so that grammars for the recognizer must be created and loaded dynamically to tie the interaction module with a specific environment.

4.3.1. Automation of actions

Scenario 1 suggested that the user’s patterns can be used in order to automate actions. For instance, if the environment knows that on weekdays between 8AM and 9AM, Michael turns on the lights of the bathroom 2 seconds after he goes into the bathroom, it can act proactively anticipating Michael’s actions in those cases. In that sense, it is worth noting that standard and comprehensible representation of patterns allows to translate them into any type of model, such as Markov models or finite state machines that can be used to automate actions.

The interaction system developed for this case shows, based on $L_{SPUBS}$ and by means of speech, the patterns discovered by the algorithm. Let us consider the environment which shows Michael’s habits (represented by ActionPattern2).

System: Occurs Bathroom is on Then turn on the Bathroom lights 2 seconds after
System: Do you want to accept, refine or delete it?

Accepting a pattern means that the user accepts a pattern as useful and therefore the environment will use it to automate the action defined in the THEN clause. If the user chooses to delete the pattern, it will not be used to automate the action. Finally, a user can choose to refine a pattern if he/she considers it is a useful pattern but some aspect needs modification. In this case, the interaction system must guide the user through the pattern in order to find out exactly what needs to be changed. Let us consider that Michael wants to turn the lights on a bit longer after entering to the bathroom, then the interaction could go along the following lines:

System: Choose the part of the pattern you want to refine: Event, Condition or Action
User: Action
System: The Action part was: Then turn on the Bathroom Lights 2 seconds after
System: The device was ‘Bathroom Lights’. Do you want to change it?
User: no
System: The action was ‘on’. Do you want to change it? (yes/no)
User: no
System: The time relation was ‘2 seconds after’. Do you want to change it?(yes/no)
User: yes
System: Please, say the new value
User: five
System: Do you want to confirm ‘five’ as the new value?
User: yes
4.3.2. Understanding of the user’s behaviours

Another interesting application of patterns, as scenario 2 suggested, is the understanding of the user’s common behaviour in order to detect unhealthy habits or provide help and support his/her daily tasks. In Sarah’s case, although the house will be looking after her, the patterns will not be shown to her but to one of the carers. In this sense, it is clear the need of a comprehensible representation of patterns and sequential representation makes their understanding easier.

Knowing when she usually has a shower or when she likes to go to bed allows staff members to act in an unobtrusive and more efficient way. Staff members can also detect possible bad or unhealthy habits. Let us consider that one of the patterns shows that Sarah usually takes pills before having lunch. Staff members know that she should take the pills just after having lunch instead of before. Thus, they can persuade Sarah to change her habit.

It is clear the need of an interaction system for this case as well. On the one hand, the system must show the learned patterns and interacting by means of speech seems a useful way. On the other hand, although understanding of the user’s behaviour seems to be one-way interaction, staff members could label the patterns as ‘unhealthy’, ‘normal’ or ‘healthy’, so that the system can issue an alarm when ‘unhealthy’ actions are detected.

4.3.3. Specific Learning processes

Up to now, the learning process has been focused on discovering the user’s most common behaviours without focusing the learning process on particular aspects. But, some applications can demand specific learning processes. Let us consider that Sarah’s carers are worried because they have detected some indications which could show that she is in the first stage of Alzheimer’s disease. Repetitive tasks and disoriented behaviours could be interpreted as its signal but they are not discovered by the system because they are not very frequent. In these cases the learning process should be modified in order to identify these particular aspects. For instance, in this case two modifications are needed:

- The threshold to consider a pattern as interesting must be lower.
- As repetitive tasks and disoriented behaviours entail repetition of either tasks or visited places, patterns could be selected or ruled out checking the existence of loops in the topology of the sequence. A loop can mean that user carries out the same tasks or visits the same places many times.

Sarah’s carers also want to know if she uses the computer to keep in touch with her son who lives in another country. Her use of computers is not frequent so that it does not appear in any frequent pattern, but a specific learning process, focused on her use of computers, can discovery it. Thus, other types of specific learning processes can be specialized in specific actions or use of devices. In Sarah’s case the specific action is her use of computers. To address this case the following modifications will be needed:

- The threshold to consider a pattern as interesting must be lower.
- Only patterns that include the use of computers would be considered, ruling out the rest of the patterns.

In general, these specific learning processes allow to carry out learning processes focused on specific aspects that would have not been discovered if we had only considered
frequent patterns. In Sarah’s case for instance, discovering such type of patterns helps staff members to confirm or rule out their suspicions about Sarah’s behaviour in aspects as important as Alzheimer’s disease or contact with relatives.

5. Conclusion

In this chapter we highlighted the importance of learning the user’s common behaviours and preferences in order to provide the environment with intelligence. First we defined the specific characteristics of Intelligent Environments to be considered in the process of learning. We provided evidence that an increasing effort is being made to adapt learning techniques to the development of Intelligent Environments but it has also been highlighted that further work is required.

Once patterns have been learned they can be used for many different purposes. Considering different applications and being aware of a natural interaction between the system and the user is necessary. To support that a speech-based interaction system was developed.

Learning in Intelligent Environments is full of challenges. As far as the system described in this chapter is concerned, we have identified the following which needs improvement:

- the learning of values within the process (e.g. discover that Michael likes to have a shower around 24-26 degrees).
- the consideration of other types of information (e.g. medical information, allowing patterns like ON ... IF heart rate is ... THEN ...)
- making the interaction system more flexible.

At a more general level, some of the ongoing challenges to accomplish suitable learning for Intelligent Environments are:

- being able to handle environments where multiple inhabitants co-exist.
- being able to manage the complexity and richness of human’s everyday life.

Finally, we are aware a holistic approach has not been achieved yet. In that sense the system we proposed in this chapter can be understood as a step in that direction. It has been designed in order to satisfy important desirable aspects for Intelligent Environments with emphasis in unobtrusive learning, accessible representation and natural refinement through spoken-dialogue.

Acknowledgements

Validations of the system presented in this chapter were conducted over artificial data generated at the University of Ulster and environment data from MavPad provided by Diane Cook from the University of Washington. Craig Wootton and Michael McTear provided initial guidance on available technologies for voice processing. This work was partially supported by Basque Government grant PC2008-28B.
References


A. Language Specification

Sequenced_Pattern ::= Sequenced_Actions, General_Conditions
Sequenced_Actions ::= ActionPattern & ... & ActionPattern

ActionPattern ::= ON (Event_Definition)
  IF (Condition_Definition)
  THEN (Action_Definition)

Event_Definition ::= Primitive_Event | Composite_Event
Primitive_Event ::= User_Presence | User_Action
  User_Presence ::= user_is_at(Location)
    Location ::= home | bedroom | living room | ...
  User_Action ::= occurs(Action, Action_Status, time)
  Action ::= Type, SetActions
    Type ::= simple | unordered
    SetActions ::= SimpleActions | UnorderedActions
      SimpleActions ::= action_1 | action_2 | ... | action_n
      UnorderedActions ::= SimpleActions & ... & SimpleActions
  Action_Status ::= on | off
Composite_Event ::= Primitive_Event & ... & Primitive_Event

Condition_Definition ::= Primitive_Condition | Composite_Condition
Primitive_Condition ::= Context_Condition
  Context_Condition ::= context (Attribute, Quantitative_Condition | Qualitative_Condition)
    Attribute ::= Calendar | Sensor
      Calendar ::= time of day | day of week | ...
    Sensor ::= sensor_1 | sensor_2 | ... | sensor_n
  Quantitative_Condition ::= (Symbol, Quantitative_Value)
    Symbol ::= = | < | > | => | =<
    Quantitative_Value ::= real_number
  Qualitative_Condition ::= qualitative_value
Composite_Condition ::= Primitive_Condition & ... & Primitive_Condition

Action_Definition ::= Primitive_Action | Composite_Action
Primitive_Action ::= do (Action_Status, Action, time) when Relation
  Action_Status ::= on | off
  Action ::= Type, SetActions
    Type ::= simple | unordered
    SetActions ::= SimpleActions | UnorderedActions
      SimpleActions ::= action_1 | action_2 | ... | action_n
      UnorderedActions ::= SimpleActions & ... & SimpleActions
  Relation ::= Qualitative_Relation | Quantitative_Relation
    Quantitative_Relation ::= (Symbol, Quantitative_Value)
      Symbol ::= = | < | > | => | =<
      Quantitative_Value ::= real_number
    Qualitative_Relation ::= qualitative_value
Composite_Action ::= Primitive_Action & ... & Primitive_Action

General_Conditions ::= Condition_Definition