Constraint-Based Activity Recognition with Uncertainty

Masoumeh Mansouri
Technology
Constraint-Based Activity Recognition with Uncertainty
Studies from the Department of Technology at Örebro University

Masoumeh Mansouri

Constraint-Based Activity Recognition with Uncertainty
To my father
with whom every moment of these days was shared,
and this is the most true sentence of this thesis.
Abstract

In the context of intelligent environments with the ability to provide support within our homes and in the workplace, the activity recognition process plays a critical role. Activity recognition can be applied to many real-life, human-centric problems such as elder care and health care. This thesis focuses on the recognizing high level human activity through a model driven approach to activity recognition, whereby a constraint-based domain description is used to correlate sensor readings to human activities. An important quality of sensor readings is that they are often uncertain or imprecise. Hence, in order to have a more realistic model, uncertainty in sensor data and flexibility and expressiveness should be considered in the model. These needs naturally arise in real world applications where considering uncertainty is crucial.

In this thesis, a previously developed approach to activity recognition based on temporal constraint propagation is extended to accommodate uncertainty in the sensor readings and temporal relations between activities. The result of this extension is an activity recognition system in which each hypothesis deduced by the system is also weighted with a possibility degree.

We validate our solutions to activity recognition with uncertainty both theoretically and experimentally, describing some explanatory examples.
Acknowledgements

I would like to thank my supervisors, Federico Pecora and Alessandro Saffiotti for the patient guidance during this project.
Contents

1 Introduction 21
  1.1 Activity Recognition ............................................. 21
  1.2 Outline ............................................................. 22

2 Background 23
  2.1 Activity recognition approaches .................................. 23
  2.2 Constraint-Based reasoning ...................................... 25
  2.3 Temporal Constraint Network .................................... 27
    2.3.1 Temporal Constraint-Based activity recognition model ..... 29
    2.3.2 Simple temporal problem .................................... 33
  2.4 Constraint Network for Value Checking ......................... 33
  2.5 Overall Motivation of This Work ............................... 35

3 Methods 37
  3.1 Soft Constraints .................................................. 37
  3.2 Fuzzy Constraint Networks ...................................... 37
    3.2.1 Fuzzy Constraint Network for Value Checking ............ 39
      3.2.1.1 Fuzzy Arc Consistency ............................... 40
      3.2.1.2 Greedy Search ...................................... 42
      3.2.1.3 Specially-Structured Constraint Networks .......... 45
    3.2.2 Fuzzy temporal Constraint Networks ....................... 46
      3.2.2.1 Conceptual Neighbors ................................. 48
      3.2.2.2 Fuzzification of Allen Relations .................... 48
      3.2.2.3 Fuzzy Composition of Allen Relations .......... 51
      3.2.2.4 Applying Path Consistency to Fuzzy relations .... 51
      3.2.2.5 Temporal Tractability ................................ 53
  3.3 Determining the Possibility Degree of an Activity .......... 54

4 Implementation and experiments 57
  4.1 Activity Recognition Process With Uncertainty ............... 57
  4.2 Occurrence Interval Extracting ................................ 59
List of Figures

2.1 Allen’s Composition table. The entry at row $r_1$ and column $r_2$ in the table denotes the possible relations between $O_1$ and $O_3$ assuming that $O_1 \leq r_1 O_2$ and $O_2 \leq r_2 O_3$. 28

2.2 Representing qualitative relations between intervals: (a) an example of temporal constraint network (b) a consistent scenario extracted from the network, and (c) a consistent scenario shown in a timeline. 30

2.3 An example of inconsistent scenario. 31

2.4 A possible timeline for the three state variables. 32

2.5 Temporal constraint graph. 32

2.6 An example of temporal constraint translation. 34

2.7 Value constraint network. 35

3.1 A fuzzy CSP. 39

3.2 A constraint network (a) before fuzzy arc consistency (b) after fuzzy arc consistency. 46

3.3 A fuzzy CSP with two state variables, three constraints ($c_1$, $c_2$, $c_3$) and domain [ONN, OFF]. 0.7 for ON and 0.3 for OFF are data obtained from sensor. 47

3.4 A nontree constraint network sample. 48

3.5 Allen thirteen relations arranged according to their conceptual neighborhood. 49

3.6 The atomic Allen relations and their membership grades with respect to the relation $m$. 50

3.7 Translation of an $IA^{fuz}$ relation to a $PA^{fuz}$ relation. 54

3.8 Constraint based activity recognition with uncertainty. The question marks represent uncertainty on the constraints. Arcs are the binary constraint and the square is indicator of unary constraints. 55

4.1 Timelines relevent to the hypothesis that the human user is cooking. 59

4.2 Consistent temporal constraint network. 60
4.3 Thirteen Allen relations between hypothesis and activity. The information provided by each relation are mentioned on the interval of activity. 64
4.4 Timeline reflecting uncertainty on the sensors. 65
4.5 Six possible combinations of state variables Stove and Location for the example scenario till $t = 15$. 65
4.6 Two greatest degrees for each hypothesis at time points $t = 1, 5, 18, 30$. 66
4.7 Patterns used in the performance test in Figure 4.8 and Figure 4.9. 66
4.8 CPU time required to provoke process of inferring an hypothesis using a rule with two requirements. 67
4.9 Propagation performance for the pattern defined in the Figure 4.7. 67
4.10 CPU time required to provoke process of inferring an hypothesis using three rules with two requirements. 68
4.11 Propagation performance for the pattern in the scenario defined in the Figure 4.4. 68
## List of Tables

2.1 The basic relation between a pair of intervals and their inverses . 27  
2.2 Two rules in a possible domestic activity recognition model. . . 31  

3.1 Unary constraint membership grades before fuzzy arc consistency 45  
3.2 Fuzzy arc consistency sample . . . . . . . . . . . . . . . . . . . 45  
3.3 Unary constraint membership grades after fuzzy arc consistency 46  
3.4 Two rules in a possible domestic activity recognition model. . . 47  

4.1 The rule defined in our domain for the human activity like cooking. 58  
4.2 Domain consists of three rules in a possible domestic activity  
recognition model. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 61  
4.3 Cooking hypotheses and possibility degree associated with each  
hypothesis at t = 15 . . . . . . . . . . . . . . . . . . . . . . . . . . . 62
List of Algorithms

1 Algorithm Qualitative-Path-Consistency ....................... 29
2 Fuzzy arc consistency ........................................... 41
3 Greedy Search .................................................... 43
4 Path consistency for fuzzy relations ............................. 53
Chapter 1
Introduction

Having interaction with robots, living in an intelligent environment, does not seem to be an unreachable dream for us any more. Imagine you are sitting on a sofa in your living room and watching your favorite series. Your living room is fully equipped with different kinds of sensors such as cameras or RFID readers, robotic devices, etc. Meanwhile, an intelligent moving table brings your daily drinks. Making this imaginary scenario to happen in every day life, is a challenging research in the field of robotics. In fact, the aim is to cooperate between robots and humans in complex levels of tasks in daily life and to enhance the welfare of human being.

1.1 Activity Recognition

Through the scenario described above, we address the issue of building cooperative smart environments which would be capable of understanding what people are doing and what could be their intentions. Understanding the state of the human and his future plan is a key capability for a smart robotic environments, which is called human activity recognition.

Activity recognition provides wide applications in personalized support including medical diagnosis [20], health monitoring [3], [13], etc. A typical scenario of activity recognition is in assisting the sick and disabled; for example, providing adaptive personalized reminders of daily activities for older adults [16]. Consider the scenario in which, a girl monitors her father’s activities in a secure web site by looking at the specific check list filled by a intelligent system embedded in her father’s apartment.

Activity recognition has also emerged a decisive research issue related to intelligent pervasive environments. Approaches undertaken for this problem must be relevant to the home settings equipment and information processing system. This relates to the fact that activities in a pervasive environment provide important contextual information and any intelligent behavior of such an environment must be relevant to the user context.
Various approaches have been used for activity recognition, including data-driven and model-driven, each having their own strengths and weaknesses. In the next chapter, these approaches are discussed with particular emphasis on a model-based approach. This thesis addresses the issue of enhancing this approach to take into account uncertainty.

In this thesis, we employ a constraint based model within an existing framework. Although this framework has many advantages, it cannot deal with uncertainty. Fuzzy inference methods are used to resolve this deficiency. The main focus of this thesis is fuzzy constraint based reasoning within the context of model based activity recognition. Fuzzy constraint based reasoning can be used in other applications as well, however enhancing the activity recognition model is the main purpose in this thesis.

1.2 Outline

The rest of this thesis is organised as follows.

Chapter 2 Gives an overview of different approaches for the activity recognition problem. This chapter also describes the underlying mechanism of the architecture which is extended in this thesis. Finally, the limitation of this architecture is going to be addressed.

Chapter 3 Proposes a solution to fuzzify the constraint networks which are underneath the activity recognition model. This solution contains two separate parts which are fused together to solve the problem.

Chapter 4 Includes implementation of the proposed methods and also contains an evaluation of a real world scenario.

Chapter 5 Presents conclusions and discusses future work.
Chapter 2
Background

The purpose of this chapter is to give a background and a motivation for the research that has been done in this thesis, which is grounded on extending a model-based activity recognition approach. This extension is accomplished by accommodating uncertainty on the underlying model through fuzzy inference methods.

The chapter begins by giving an overview of the different approaches of recognizing human activity. Also, The reasons which led to the preference of the model based approach are explained. The chapter then proceeds to describe the application of the model and its limitations. It also gives a short description of a framework in which the mentioned model is applied.

2.1 Activity recognition approaches

Current approaches for solving the problem of recognizing human activities can be categorized as data driven or model driven approach. Data driven approaches are based on machine learning techniques which use probabilistic and statistical reasoning, for instance, Hidden Markov Models (HMMs) [18], [6]. These approaches use large amounts of data retrieved from homogeneous sensors placed in the environment. In data driven approaches, sensor data must be aggregated to make a classifier for recognizing human activities. This process is done by training the retrieved sensor data to make different patterns for low level human behaviors. For example, by setting up an accelerometer sensor on a person’s body, it is straightforward to learn human state patterns such as walking, running or falling. Moreover, data driven approaches are driven by probabilistic learning models which are capable of handling noisy and incomplete sensor data. However, They are usually highly activity dependent. The resulting models are often not reusable and scalable due to the variation of the individual’s behavior and their environments, in other words, it requires retraining when the application context changes. More importantly, data-driven approaches require annotated data which is often difficult to obtain. Using a
data-driven approach makes the system sensitive to the changes in the environment and quantity of the sensors data. The other drawback of applying data driven approach is often the computational burden associated with it. As an illustration, [11] proposed a partially observable Markov decision process (POMDP) models for a handwashing task. These models are computationally expensive in case of supporting multiple and heterogeneous sensors and generating sufficient policies to aid people with dementia.

Consider the case that a data driven approach would be able to recognize human activities like cooking. First, input data features and its annotation should be determined to enable recognition algorithm, basically classification algorithm, to train aggregated data from multiple data sources and transform them into the application dependent features. If the human changes their habits for cooking even in terms of duration or the order of tasks, the system should retrain to find other suitable features for the cooking as a human activity. The system confronts with this situation quite often, since people usually have different habits for their activities.

On the contrary, in the case of recognizing high level human activity such as cooking and sleeping, model driven approaches are often used. Model driven approaches are useful when the criteria for recognizing human activities are given by rules that are clearly identifiable. Consider the example above, there is a clear correlation between sensor readings and observed pattern. For example, if the sensor data tell us the human is in the kitchen and stove is on, there is enough evidence to infer that the human is cooking.

In this thesis, model-driven activity recognition is employed. The main reason to choose this model is the clear correlation between sensor reading and pattern observation, in other words, we know what to be expected from the real environment based of the observations.

Model-driven approaches to activity recognition follow a complementary strategy in which patterns of observations are modeled from first principles. These approaches typically employ an abductive process, whereby sensor data is explained by hypothesizing the occurrence of specific human activity and testing this hypothesis repeatedly [15].

Abductive reasoning is a way of providing an explanation which is sufficient to explain a sentence’s being true [2]. Imagine \( p \) is a cause (for example, "it is raining") and \( q \) is an effect (for example, "the grass is wet"). If it rains, the grass is wet. Deductive reasoning would be used to predict the effects of rain, that is, wet grass, among others; abductive reasoning would be used to hypothesize the cause of wet grass, that is, rain, among others. In fact, abductive reasoning is in some sense the converse of deductive reasoning.

Abduction based activity recognition works as following: Given a set of rules, the abductive reasoning process iteratively hypothesizes whether the head of each rule is justified given sensor readings. This is done through different kinds of inference methods existing in the literature such as temporal reasoning. All the existing temporal reasoning used in the context of activity recognition...
2.2. CONSTRAINT-BASED REASONING

and also plan recognition are based on crisp temporal reasoning process. For example, the NASA, model-based planner/scheduler uses the crisp inference method for the crisp rules defined in the domain of the infrastructure [12] and the chronicle recognition approach also applied crisp temporal reasoning [5].

Human activities can be recognized just by determining the identifiable crisp rules and by performing the abductive process iteratively. Applying the crisp rules just give us a yes or no answer and we are not able to know that to what degree the rules are applied. If a model is precisely defined, abductive reasoning will find evidence for the current activity of the human. In the case that the model does not reflect the human behavior exactly which is more likely to happen, then we will simply not recognize the occurrence of an activity. Moreover, analogously to data driven approaches, model driven approaches should also be able of handling noisy, uncertain and incomplete sensor data. For these reasons, we investigate how to incorporate uncertainty in the model. In this thesis, fuzzy activity recognition system based on constraint based reasoning is used. In fact, this thesis aims to improve the existing activity recognition model in terms of applicability and easy-of-use by accommodating uncertainty in the model.

2.2 Constraint-Based reasoning

The activity recognition approach applied in this thesis, is modeled as a constraint based network and temporal constrained based reasoning techniques are used inside the model. Hence, we need to briefly introduce Constraint-Based reasoning and the concept of classical constraint networks [21].

Given a sequence of distinct variables $V = \{x_1, ..., x_k\}$ and their associated domains $D_1, ..., D_k$, a relation $R$ on $V$ is a subset of $D_1 \times ... \times D_k$. The arity of the relation is $k$ and the scope of the relation is $V$. To make scopes explicit, we will often denote a relation $R$ over variables $V$ as $R_V$ and an element of $R_V$ as a tuple $t_V$. Such a tuple $t_V$ is called an assignment of the variables in $V$. The projection of a tuple $t_V$ over a sequence of variables $W, W \subseteq V$, is the tuple formed by the values in $t_V$ corresponding to variables in $W$, denoted as $t_V[W]$.

A classical constraint network (classical CN) is a triple $X, D, C$ defined as follows:

- $X = \{x_1, ..., x_k\}$ is a finite set of $k$ variables.
- $D = \{D_1, ..., D_k\}$ is the set of the domains corresponding to variables in $X$, such that $D_i$ is the domain of $x_i$; $d$ bounds the domain size.
- $C$ is a finite set of constraints. A constraint $c \in C$ is defined by a relation $R$ on a sequence of variables $W \subseteq V$. $W$ is the scope of the constraint. The relation specifies the assignments allowed by $c$ for the variables of $V$. Thus, a constraint $c$ can be viewed as a pair $R, W$ also noted $R_W$. 

Given an assignment \( t_V \) and a constraint \( R_W \), we say that \( R_W \) is completely assigned by \( t_V \) when \( W \subseteq V \). In such case, we say that \( t_V \) satisfies \( R_W \) when \( t_V[W] \in R_W \). If \( t_V[W] \notin R_W \), \( t_V \) violates \( R_W \). An assignment \( t_V \) is consistent if it satisfies all constraints completely assigned by it. An assignment \( t_V \) is complete if \( V = X \). A solution of a classical CN is a complete consistent assignment. The task of finding a solution in a classical CN is known as the constraint satisfaction problem (CSP), which is known to be NP-complete.

One way to solve a CSP, is to enumerate each \( n \)-tuple and test if it is a solution. This blind enumeration can be improved by using backtracking search combined with inference [21]. Through searching, subspaces with a single failure can be eliminated. In inference techniques, from a subset of the constraints and the domains, more restrictive constraints or more restrictive domains are inferred. The inferences are accomplished by local consistency properties that characterize necessary conditions on values or set of values to belong to a solution. The level or scope of consistency, the size of the set of variables involved in the local context, can be adjusted as a parameter from 1 up to \( n \). If we increase the level of consistency, more computation is needed. Time complexity grows polynomially by increasing the level of consistency.

Arc consistency is a form of approximate inference which is a technique for tightening the domains. In arc consistency, each edge in the constraint network is considered as a directed arc. A directed arc associated to the variables \( X_1 \) and \( X_2 \) \((X_1 \rightarrow X_2)\) is consistent if, for every value \( v \) of \( X_1 \), there exists a value of \( X_2 \) that is consistent with \( v \). Arc consistency checking can be applied iteratively until no more inconsistencies remain. In each iteration, a value of the domain variable is removed if it is found inconsistent. The inconsistency which were found on each iteration of arc consistency may propagate to cause inconsistencies in neighboring arcs that were previously consistent.

The next level of consistency to consider, would be Path consistency. In arc consistency, we tighten the domain of each variable using local binary constraints. In path consistency, domains are also tightened by using the implicit induced constraints on triples of variables \( X_i, X_m \) and \( X_j \). A path of length two from node \( i \) through node \( m \) to node \( j \) is path consistent if, for every pair of values \( \langle a, b \rangle \) allowed by the explicit relation \( R_{ij} \) there is a value \( c \) for \( X_m \) such that \( \langle a, c \rangle \) is allowed by \( R_{im} \) and \( \langle c, b \rangle \) is allowed by \( R_{mj} \). Path consistency is repeated applied to ensure path consistency for each length 2 path within the entire network.

In summary, approaches for reasoning in CSP are based on inference and search, and also various combinations of them. For every choice generated by backtracking search, inference techniques compute the consequence of this generated choice on the other variables. This propagation diminishes the number of possible value in the domain and therefore, the branching factor of the search algorithm is reduced. There are always tradeoffs to evaluate the effort required to avoid search by making the algorithm more informed and the reduction in search effort obtained.
2.3 Temporal Constraint Network

The temporal reasoning system used in our approach to activity recognition is grounded on the temporal relations between sensor readings. In the current approach, temporal information are time intervals. Intervals correspond to the time periods during which the event occurs. Relations between paired intervals are formulated in terms of qualitative statement regarding the relative location of them. This formalism is called Interval Algebra (IA) [1].

Temporal reasoning can be viewed as a CSP. Intervals are variables in our CSP and temporal relations are treated as constraints. There are seven basic (atomic) relations that can hold between intervals: before, meet, overlaps, start, during, finishes and equal, as depicted in Table 2.1. Moreover, each one of these relations is associated with an inverse relation. For instance, the inverse of relation before is the relation after. Consider the following example, P occurred before Q, the relative temporal information can be expressed as CSP in which P and Q are variables and before is the constraint.

<table>
<thead>
<tr>
<th>Relation</th>
<th>Symbol</th>
<th>Inverse</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>X before Y</td>
<td>&lt;</td>
<td>&gt;</td>
<td></td>
</tr>
<tr>
<td>X equal Y</td>
<td>=</td>
<td>=</td>
<td></td>
</tr>
<tr>
<td>X meets Y</td>
<td>m</td>
<td>mi</td>
<td></td>
</tr>
<tr>
<td>X overlaps Y</td>
<td>o</td>
<td>oi</td>
<td></td>
</tr>
<tr>
<td>X during Y</td>
<td>d</td>
<td>di</td>
<td></td>
</tr>
<tr>
<td>X starts Y</td>
<td>s</td>
<td>si</td>
<td></td>
</tr>
<tr>
<td>X finishes Y</td>
<td>f</td>
<td>fi</td>
<td></td>
</tr>
</tbody>
</table>

Having the knowledge in IA framework, we are interested to determine whether the given information is consistent, that is, whether it is possible to arrange intervals along the time line according to the given information. A solution to interval algebra constraint network can be associated with a consistent labeling, that is assigning an atomic relation to each constraint which is consistent with the other assigned relations. Finding a solution to the interval constraint problem is NP-hard and it can be solved with exponential time algorithms like those seen for general CSP search [23].

Generally, path consistency is used to improve exponential search algorithms and detect inconsistency. Although performing path consistency does
not necessarily lead to have a minimal network, there are some cases that path consistency is exact. For example, when there is one relation per constraint, path consistency is enough for deciding the consistency. An algorithm is exact for a class of input if it, depending on the version of the problem, correctly finds the minimal labels between all pairs of variables or between one variable and every other variable, for all instances in that class.

To apply path consistency for IA networks, two operations on constraints are defined:

Intersection: \( R' \oplus R'' = \{ r : r \in R' \cap r \in R'' \} \)
Composition: \( R' \otimes R'' = \{ r' \otimes r'' : r' \in R' \cap r'' \in R'' \} \)

---

**Figure 2.1:** Allen’s Composition table. The entry at row \( r_1 \) and column \( r_2 \) in the table denotes the possible relations between \( O_1 \) and \( O_3 \), assuming that \( O_1 r_1 O_2 \) and \( O_2 r_2 O_3 \).


2.3. TEMPORAL CONSTRAINT NETWORK

Algorithm QPC-1

input : An IA network T
output: A path consist IA network

repeat
    \( S \leftarrow T; \)
    for \( k \leftarrow 1 \) to \( n \) do
        for \( i, j \leftarrow 1 \) to \( n \) do
            \( C_{ij} \leftarrow C_{ij} \oplus C_{ij} \otimes C_{kj}; \)
        end
    end
until \( S = T; \)

Algorithm 1: Algorithm Qualitative-Path-Consistency

Composition of individual Allen relations \((r' \otimes r'')\) is given by a composition table shown in the Table 2.1. The path consistency method depicted in Algorithm 1 is a polynomial algorithm having time complexity \(O(n^3)\) [4] in which \(n\) is the number of variables. It employs relaxation operation until either a fixed point is reached or some constraints become empty.

Consider the intervals of times over the following events which may be performed by a person: reading a paper, drinking coffee, having breakfast and walking. Figure 2.2(a) expresses all feasible relations between all pairs of intervals. One possible consistent scenario extracted from part (a) is shown in 2.2(b) which can be a solution for this constraint network. This solution corresponds to the 2.2(b) is depicted in part (c) along a timeline.

In Figure 2.3 lies a constraint network with an inconsistent scenario. According to this constraint network, it is obviously impossible to drink coffee at the same time with walking whereas the person starts walking just after (relation meet) eating breakfast and drinks coffee during the eating breakfast. In this example, Coffee during Breakfast, Breakfast meet walk and Coffee equal walk. A composition of the relation between Coffee and Walk in the composition table (i.e, the relation at row d and column m) yields \(\{\rangle\}\), which means that \(=\) has to be deleted from initial relation between Coffee and Walk, resulting in making the relation between the them empty.

2.3.1 Temporal Constraint-Based activity recognition model

The model driven approach adopted for this thesis is based on the activity recognition model in SAM [15]. SAM\(^1\) is an architecture which provides the capabilities of the activity recognition and planning for controlling the actuation devices in a smart environment. In SAM, both recognition and actuation

\(^1\)SAM stands for "SAM the activity Manager"
are integrated at the reasoning level as well as being modeled in the same formalism. SAM is built on the OMPS\textsuperscript{2} framework [8], both using constraint representation language and temporal propagation algorithms. In this thesis, we concentrate on the extension of the underlying mechanisms of SAM.

In SAM, abductive reasoning process is used to infer human behavior. Through this reasoning, activity recognition process employs a knowledge representation formalism based on temporal constraints. The temporal constraints are modeled inside the rules. Within the rules, temporal relation between the head of the rule as human activity and its requirements are defined. The elements of each rule including the head and the requirements are represented by notion of state variable. Each rule contains several state variables. State variables are used to represent the parts of real world that are relevant for SAM decision process. For example, a state variable can represent the state of human being like eating or correspond to the possible sensor readings e.g., stove whose value changes over time and the values can be \{on, off\}. In fact, the rules specify the temporal constraints between state variables. A collection of these rules makes the domain. The rules in the domain constitute a set of "temporal queries" used by SAM to ascertain whether a particular pattern of sensor readings holds. These temporal queries are modeled in a constraint network. SAM ensures the temporal capabilities of the model by maintaining temporal requirement in a

\textsuperscript{2}Open Multi-component Planner and Scheduler
2.3. TEMPORAL CONSTRAINT NETWORK

Temporal constraints are formulated as relations in Allen interval algebra in a temporal constraint network. As a case in point, Table 2.2 consists of two rules, each rule shows how temporal constraints can be used to model the requirements for the specific human behavior. These rules describe possible condition under which human activities of **Cooking** and of **Eating** can be inferred.

Consider the first rule in the table, it involves three so called state variables: first, a state variable representing human state, second, state variable representing the stove state sensor with value **ON** or **OFF** and another state variable representing the location of the human as it is determined by the corresponding sensor. In the abduction process, the occurrence of **Cooking** as a human activity is hypothesized and stove states and location state are considered as its explanations. Cooking can be inferred if the user being located (DURING as a temporal constraint) in the **KITCHEN** and at the same time (EQUAL as a temporal constraint) when the stove is **ON**. This inference process is operated at the same time with sensing process. In the sensing process, the value of state variables are updated over time by querying real sensors. Since the sensor measurement changes over time, the value of the state variable also changes over time. The state variables can be instantiated in a given period of time (see Fig-
The stove state is changed from being on to off at the certain time and the human location similarly changes its state from kitchen to kitchentable.

For example, Consider the timeline depicted in figure 2.4, we can create the temporal constraint network based on both information acquired from the sensors and the rule defined in the model (e.g, first rule in the Table 2.2). The constraint graph representing the scenario in the time line 2.4 is depicted in Figure 2.5. This temporal constraint network involves a set of variables \{HumanCooking, StoveOn, LocationKitchen\}, where each variable represents a temporal interval.

![Figure 2.4: A possible timeline for the three state variables](image1)

![Figure 2.5: Temporal constraint graph](image2)
Overall, the constraint network for the activity recognition process ensures temporal capabilities through *temporal propagation*. The constraint network determines whether the temporal rules apply. This temporal checking is done iteratively. If a rule applies, there is a consistent scenario which means that the hypothesized human behavior can be deduced from the applied rule. Figure 2.4 shows the possible scenarios in which the information acquired during the activity recognition process employed to infer different states of the human.

### 2.3.2 Simple temporal problem

Instead of performing path consistency on the network of IA constraints, in SAM, the problem of ascertaining temporal consistency is reduced to a simple temporal problem (STP). Within a network of STP, the temporal propagation of SAM is done through Floyd-Warshall algorithm [7]. High level temporal constraints, formulated as Allen interval can be translated to STP level by computing quantitative bounds on the interval’s start and end times of an activity.

Simple Temporal Problems (STPs) are a restriction of the framework of Temporal Constraint Satisfaction Problems. As noted earlier, temporal constraints can be quantitative (distance between time points) or qualitative (relative position of temporal objects). STP is a quantitative temporal constraint satisfaction problem in which variables represent time points (events) and constraints represent relations between them. The restriction is to have at most one interval in each temporal constraint which entails that a STP can be solved in polynomial time. By solved, we mean that consistency is decided and the minimal network obtained [4]; applying path consistency suffices for this. In contrast, the general TCSP is NP-hard. The underlying temporal propagation of SAM is based on Floyd-Warshall algorithm whose computational cost is $O(n^3)$. Floyd-Warshall basically finds the shortest path between all pairs of nodes in a distance graph created by simple temporal network. This temporal network named by STP level is the result of a translation in which high level temporal network is translated to the STP level. An overview of this translation is depicted in figure 2.6.

Translation from Allen’s interval algebra to the quantitative temporal network does not necessarily lead to have an STP. However, since the temporal constraint network created based on one rule, has one relation per constraint, the constraints specify a single interval in the quantitative temporal network. In this case, checking temporal consistency is tractable. On the contrary, the problem of finding which subset of the rules applies is still exponential.

### 2.4 Constraint Network for Value Checking

As indicated earlier, SAM checks the temporal requirements by using the rules defined in the domain. Through the activity recognition process, SAM also checks for the presence of particular values of sensors within the network. For
instance, SAM first checks whether the stove is on and the human is in the kitchen, and then checks temporal constraints regarding these state variables. This checking is a very simple form of constraint checking. Specifically, there is no constraint based reasoning involved in the determination as to whether values are present in the network. We introduce a form of constraint reasoning for the purposes of checking whether value requirements are met. This allows us, as we will see, to check for both equality of value (what is done in SAM) as well as inequality of values. To do this, we model value requirements of each human activity within a constraint network, which we call the "value constraint network". The value constraints, like the temporal constraints, are specified in the domain.

A formal constraint network specification for the current problem is the following: the finite set of variables contains the current state variables of the model, for example, stove and human activity, the domains of variables are all states of variables such as ON, OFF for stove and constraints are equality (=) or inequality (≠) defined by the model. Figure 2.7 shows the constraint network associated with the first rule (Human: Cooking) in the domain shown in the Table 2.2. In this figure, there are two variables \( X = \{ X_1, X_2 \} \) with the domain \( X_1 = \{ \text{ON} \} \) and \( X_2 = \{ \text{ON, OFF} \} \). Note that both variables stand for stove, however \( X_1 \) is obtained from the given model and \( X_2 \) is obtained from sensor data which support multiple value possibilities on state variables. This
representation of value constraint network is applied in this thesis to be able to model the activity recognition with uncertainty.

Overall, Constraint satisfaction problems on finite domains are typically solved using a form of search. The most used techniques are variants of backtracking, constraint propagation, and local search. More details about the this CSP and how to solve it with regarding to the extra features added in this thesis are explained in the next chapter.

Figure 2.7: Value constraint network

2.5 Overall Motivation of This Work

Recognizing and understanding the activities of people from sensor readings is an important task in ubiquitous computing. Activity recognition is also a particularly difficult task due to the inherent uncertainty and complexity of the data collected by the sensors. In the activity recognition model of SAM, uncertainty in the sensor data were not considered. Considering uncertainty in sensor data brings the need of supporting multiple value possibilities on state variables while this possibility does not exist in SAM.

Furthermore, crisp temporal reasoning, once exposed to the difficulties of real life problems, can be found lacking both expressiveness and flexibility. Planning and scheduling for service providing architecture to assist human being, for example, involves not only qualitative temporal constraints between events, but also soft temporal relations; as an illustration, monitoring activities should not overlap, but may if necessary.

To address the lack of expressiveness of hard constraints, preferences can be added to the framework; to address the lack of flexibility, handling of uncertainty can be added. Some real world problems, however, need both.

This limitation motivates us to consider uncertainty by fuzzifying the underlying mechanism of SAM. In fact, SAM like other similar approaches, e.g., chronicle recognition approach [5], relies on crisp constraints and consistency checking returns true or false. Let us make an example to clarify these limitations. For instance, consider the case that sensor stove goes off after human left the kitchen. In order to infer human activity like cooking, necessary constraints (see table 2.2) are imposed on the given data and of course, in this case, the cooking inference will be fail and result for consistency checking is no. This failure is not just because of the sensor imprecision; it is also a consequence of
the fact that model can not hold all the temporal requirements. In fact, domain
can not recognize all the possible states that human is going to do, however, it
can estimate. This estimation is based on some methods which is explained in
the next chapters.

In brief, our model is based on the same principle as SAM, which is tempo-
ral constraint reasoning. However, we develop a new approach. This approach
combines fuzzy temporal constraint reasoning and so-called fuzzy value con-
straint reasoning to achieve an activity recognition architecture that can deal
with uncertainty in both required values of sensor readings, and in the tempo-
ral placement of these readings.
Chapter 3
Methods

As previously mentioned in Chapter 2, there were two constraint networks on the activity recognition model. In this chapter, we focus on the fuzzification of these two constraint networks.

3.1 Soft Constraints

In many practical cases, crisp constraint reasoning is not enough to solve a CSP. For example, it is possible that there is no way to satisfy all constraints among the variables, the instance is said to be over constrained; on the other hand, there may be several solutions to an under constrained problem. Crisp constraints reasoning give us no way to discriminate between them. Soft constraints provide different ways to model above cases. In some scenarios, constraints represent the desired properties rather than requirements that can not be violated, hence, preferences whose violation should be avoided as far as possible, is a better definition for such constraints.

In our model, we define soft unary constrain to model the uncertain sensor readings. Moreover, temporal requirement can be modeled as preferences to every atomic relation of Allen interval algebra. These preferences are obtained by the concept of Freksa neighborhood which is detailed in following sections.

3.2 Fuzzy Constraint Networks

In this section, we describe a framework which has been proposed in the literature for modeling soft constraints [17], [19]. This framework is focused on the specific interpretation of soft constraints, in terms of possibility theory.

Based on the notation of constraint network expressed in Section 2.2, we briefly describe fuzzy constraint network. A classical constraint can be seen as the set of value combinations for the variables in its scope that satisfy the constraint. In a fuzzy framework, a constraint is no longer a set, but rather a fuzzy set. This means that, for each assignment of values to its variables, we do
not have to say whether it belongs to the set or not, but how much it does so. In other words, we have to use a graded notion of membership. The membership function \( \mu_E \) of a fuzzy set \( E \) associates a real number between 0 and 1 with every possible element of \( E \).

A fuzzy constraint network (fuzzy CN) is a triple \( X, D, C \) where \( X \) and \( D \) are the set of variables and their domains, as in classical CNs, and \( C \) is a set of fuzzy constraints. A fuzzy constraint is a fuzzy relation \( R_V \) on a set of variables \( V \). This relation, that is a fuzzy set of tuples, is defined by its membership function

\[
\mu_{R_V} = \prod_{x_j \in V} D_j \rightarrow [0, 1]
\]

The membership function of the relation \( R_V \) indicates to what extent an assignment \( t \) of the variables in \( V \) belongs to the relation and therefore satisfies the constraint.

In classical constraint satisfaction problem, when we have a set of constraints we want all of them to be satisfied. Thus, we combine constraints by taking their conjunction. In the fuzzy framework, constraints are naturally combined conjunctively. The conjunctive combination \( R_V \otimes R_W \) of two fuzzy relations \( R_V \) and \( R_W \) is a new fuzzy relation \( R_{V \cup W} \) defined as

\[
\mu_{R_{V \cup W}}(t) = \min(\mu_{R_V}(t[V]), \mu_{R_W}(t[W])) \quad t \in \prod_{x_i \in V \cup W} D_j
\]

We can now define the preference of a complete assignment, by performing a conjunction of all the fuzzy constraints. Given any complete assignment \( t \), its membership degree, also called satisfaction degree, is defined as

\[
\mu_t = (\bigotimes_{R_V \in C} R_V)(t) = \min_{R_V \in C} \mu_{R_V}(t[V])
\]

A solution of a fuzzy CN is a complete assignment with satisfaction degree greater than 0. When we compare two complete assignments, the one with the highest preference is considered to be better. Thus, the optimal solution of a fuzzy CN, \( \hat{t} \), is the complete assignment whose membership degree is maximum over all complete assignments, that is,

\[
\hat{t} = \arg \max_{t \in \prod_{x_i \in X} D_i} \min_{R_V \in C} \mu_{R_V}(t[V])
\]
3.2. FUZZY CONSTRAINT NETWORKS

3.2.1 Fuzzy Constraint Network for Value Checking

Uncertainty in the given sensor data is modeled as a soft unary constraint on each value of the variable’s domain. For example, in the crisp case, the domain associated with the stove is a set \{ON, OFF\}, in the fuzzy case, the set can be represented as \{<ON, \alpha_1>, <OFF, \alpha_2>\}. \alpha_1 and \alpha_2 are membership grades that express soft unary constraints. If \(V = \{x_1, ..., x_k\}\) is a finite set of \(k\) variables, the membership function for the unary fuzzy constraints imposed on every variable \(x_i \in V\) is expressed as below

\[
\mu_{R_{x_i}}(t) \rightarrow [0, 1] \quad t \in D_i
\]

As explained in the previous chapter, we want to unify the value requirements of each rule with current sensor readings. For example, suppose we are looking for the value ON for the stove to hypothesize an activity like Cooking. This means that the state variable representing the stove state should have the value ON. When adding the notion of soft constraint to the model, we still want these value requirements to hold, although each value is associated with corresponding possibility degree. For this reason, we leverage the concept of hard binary constraints in the model. In fact, a set of binary constraints, \(\{=, \neq\}\), are defined as hard constraints in our constraint network. The membership function for the binary hard constraint over variables \(w_{ij} = \{x_i, x_j| x_i \neq x_j, x_i, x_j \in V\}\) is defined as

\[
\mu_{R_{w_{ij}}}(t) \rightarrow \{0, 1\} \quad t \in D_i \times D_j
\]

Figure 3.1 shows the graph representation of a fuzzy CSP. Variables are X and Y, and constraints are represented by nodes and undirected (unary for c1 and c3 and binary, equality constraint, for c2) arcs. The domain \(D\) of the variables contains only elements a and b. Since we define binary constraints as hard constraints, the tuple \(<a, b>\), for example, has the value 0 and \(<a, a>\) has the value 1 as membership grades.

![Figure 3.1: A fuzzy CSP](image)
3.2.1.1 Fuzzy Arc Consistency

Considering the definition of optimal solution indicated in the previous sections; we have to find the maximum satisfaction degree of all complete assignments. Therefore, to prune the search space containing all the k-tuples, constraint propagation should be performed. Constraint propagation consists of evaluating the implications that a constraint has on one variable onto another variable. As noted earlier, arc consistency is a form of approximate inference which transforms the original network into a tighter representation [4]. In this thesis, fuzzy arc consistency is applied for the fuzzy constraint network. Through this constraint propagation, the membership grade of unary constraints is updated, more specifically, it can be only decreased and some of them become zero after the propagation. The fuzzy arc consistency depicted in algorithm 2 follows the notation of constraint network already explained in Section 2.2 and 3.2. The algorithm works as follows.

Let \( R = (X, D, C) \), where \( X \) is a set of variables, \( c_{ij} \in C \) are binary constraints, and \( D \) are the domains of the variables. For each \( c_{ij} \in C \), the membership grades for every value \( a_i \in D_i \) and \( a_j \in D_j \) are updated based on the type of binary constraint (i.e., either equality or inequality) which they are involved in. If \( c_{ij} \) is an equals constraint, for every \( a_i \in D_i \), the algorithm updates the membership degree of the unary constraint on \( a_i \) to be the minimum of \( \mu_{a_i} \) and \( \mu_{a_j} \). This is done only if there exists an \( a_j \in D_j \) such that \( a_i = a_j \), otherwise the corresponding membership function becomes zero. In other words, crisp constraint propagation would delete the inadmissible tuples; here, we also delete inadmissible tuples, but by assigning 0 to the membership grade of value associated to that tuple. The difference between crisp arc consistency and our fuzzy arc consistency arises when we have the admissible tuple (e.g., \( (a_i, a_j) \mid a_i = a_j \)). In this case, instead of assigning value 1 to each value involved in the tuple, we update the associated membership value by taking the membership grade minimum between them. This is because we still want both values to satisfy the constraint, and the fuzzy logic, a t-norm is employed for this case (in our specific implementation, the min operator).

If \( c_{ij} \) is an inequality binary constraint, then for every \( a_i \in D_i \), the membership grade of \( a_i \) is the minimum of \( \mu_{a_i} \) and maximum membership grades of the set \( \{a_j \in D_j \mid a_j \neq a_i\} \). This corresponds to the following logical expression which defines inequality between two variables with domains \( D_i \) and \( D_j \),

\[
\forall a \in D_i, \exists b \in D_j \mid a \neq b
\]

Applying fuzzy arc consistency decreases the amount of feasible tuples. It can be also applied to calculate an upper bound for the satisfaction degree of the constraint network. This upper bound is not necessarily the satisfaction degree for the whole network. As indicated earlier, the optimal solution of a fuzzy CN is the complete assignment whose membership degree is maximum over
**Algorithm Fuzzy arc consistency**

**input**: a triple $< X, D, C >$ as a fuzzy CN

**output**: A constraint network with updated membership grades

```
repeat
  foreach $c_{ij} \in C$ do
    // Constraint $c_{ij}$ is defined over variables $x_i$ and $x_j$
    foreach $t_i \in D_i$ do
      foreach $t_j \in D_j$ do
        if $c_{ij}$ is " = " then
          if $t_i = t_j$ then
            $\mu_{R_{x_i}}(t_i) = \min(\mu_{R_{x_i}}(t_i), \mu_{R_{x_j}}(t_j))$
            $\mu_{R_{x_j}}(t_j) = \min(\mu_{R_{x_i}}(t_i), \mu_{R_{x_j}}(t_j))$
          else
            $\mu_{R_{x_i}}(t_i) = 0$
            $\mu_{R_{x_j}}(t_j) = 0$
        end
        end
      end
    end
  end
until no membership grade is changed;
```

Algorithm 2: Fuzzy arc consistency
all complete assignments and these complete assignments must have the satisfaction degree greater than 0. However, the complete assignments we obtain after applying arc consistency are not necessarily feasible assignments. In fact, arc consistency is not enough to solve the problem, we need to perform search. The mentioned upper bound is defined as below

$$\text{USD}^1 = \max_{t \in \prod_{x \in X} \min_{d \in D} \min_{r \in C} \mu_{r} (t[V])}$$

The complexity of the non fuzzy arc consistency, for example, naive arc consistency algorithm is $O(ek^3)$ for a given constraint network $R$ having $n$ variables, with the domain sizes bounded by $k$, and $e$ binary constraints [4]. This fuzzy arc consistency algorithm does not increase the time complexity of crisp case of arc consistency. Considering algorithm 2, one cycle through all the binary constraint takes $O(ek^2)$. Update of all unary constraints over variables in the scope of each binary constraint takes $O(k^3)$. Since, in the worst case, one cycle may cause the update of just the membership grade imposed on one value from one domain, and since there are $nk$ values, the maximum number of such cycle is $nk$, resulting in the overall bound of $O(n.ek^3)$.

### 3.2.1.2 Greedy Search

Since arc consistency is not enough for deciding the globally consistent network, we have to perform search to find the maximum satisfaction degree of feasible tuples.

First, we sort the value members of each domain in the constraint network. This sorting is based on the membership degree of unary constraints imposed on each value of the domain. Then, for each variable, we select the value which has the highest membership degree defined by its unary constraint. This selection generates a tuple. If this tuple is a feasible assignment for the constraint network, we calculate the minimum of all unary constraints involved in this assignment and consider this value as maximum satisfaction degree. If the tuple is not a feasible assignment, we continue to create feasible tuples with satisfaction degree equal to USD that is expressed in the previous section. For those feasible tuples which have the satisfaction degree less than USD, we select a feasible tuple with the highest value. All possible assignments are generated if first generated assignment is not feasible or the rest of generated assignments have satisfaction degree less than USD. In the worst-case, the algorithm generates all possible assignments. Consequently, the complexity of the greedy search is $O(k^n)$, where $k$ bounds the domain size and $n$ is the number of variables. In fact, this algorithm is a complete method which solves the problem in exponential time. The pseudocode of the greedy search method is express in algorithm 3.

---

1 Upper bound for Satisfaction Degree
Algorithm 3: Greedy Search

**input**: A CN as a tuple \( <X, D, C> \) after applying fuzzy arc consistency, USD as an upper bound for satisfaction degree.

**output**: Maximum satisfaction degree of constraint network

```
foreach \( D_i \in D \) do
  \( \text{OD}_i \leftarrow \text{sort}(D_i) \) // based on membership grades
end
ODR = \{\text{OD}_1 \times \ldots \times \text{OD}_k\}
sort(ODR) // sorting Lexicographically
// max-ft is the maximum satisfaction degree associated with the feasible tuples
max-ft = 0;
foreach \( t \in \text{ODR} \) do
  if \( \mu_{RX}(t) = \text{USD} \) and \( t \) is a feasible tuple then
    return \text{USD}
  else
    max-ft = \text{max}(\text{max-ft}, \mu_{RX}(t) \mid t \text{ is a feasible tuple})
  end
end
return max-ft
```

Algorithm 3: Greedy Search
Example: Consider a three variable network: \( X, Y, Z \) with \( D_X = \{A, B, C\} \), \( D_Y = \{B, C, D\} \) and \( D_Z = \{C, D, E\} \). There are three constraints: \( R_{XY} \), specifying that \( X \) should be equal to \( Y \), \( R_{YZ} \), specifying that \( Y \) should be unequal to \( Z \), and \( R_{XZ} \), specifying that \( X \) should be unequal to \( Z \). The constraint network of this problem is depicted in Figure 3.2(a) and unary constraints associated with each value in the domains are shown in table 3.1. First, for simplicity, we define the unary constraint on each value \( a \) in the domain of variable \( V \) as \( \mu_a^V \). In order to apply fuzzy arc consistency to the network, we put \( (X, Y) \), \( (Y, Z) \) and \( (X, Z) \) onto the queue. Consider \( R_{XY} \) which is an equal constraint. We are going to update every value in the domains of \( X \) and \( Y \). In the first iteration, \( \mu_A^X \) becomes zero as a result of minimum 0.4 and 0. As there is no value \( A \) in the domain of \( Y \), \( \mu_A^Y \) is zero. \( \mu_B^X \) and \( \mu_B^Y \) are also updated by taking minimum of their membership grades which is 0.6. Similarly, membership grades of the values in the \( D_X \) and \( D_Y \) are updated (see table 3.2, the row indicates the relation \( X = Y \)). Processing the pair \( (X, Y) \) changes the problem, since the domains \( X \) and \( Y \) are not arc-consistent relative to \( R_{XY} \). Now, the domain of both \( X \) and \( Y \) are shrunk such that \( D_X = \{B, C\} \) and \( D_Y = \{B, C\} \). Then, we process \( (Y, Z) \) with regard to \( R_{YZ} \) which is unequal constraint. Having \( \mu_C^Y = 0.7 \) from the processing \( X = Y \), we take the minimum of this \( \mu_C^Y \) and maximum value of set \{0.8, 0.5, 0.2\} \setminus \{0.8\}. The final membership grade of \( \mu_C^Y \) in this step is 0.5, that is the result of minimum of 0.7 and 0.5. Note that, we exclude \( \mu_C^Z \) from the set, because we are in the process of unequal constraint \( R_{YZ} \) in order to update \( \mu_C^Y \). The similar procedure is applied for updating the other membership grades in \( Z \). Processing \( (X, Z) \) is still needed since the membership grades of values in \( Z \) are changed. Consequently, the updating process may need to be applied more than once to each constraint until there is no change in membership grades of values in the domain of any variable in the network. Fuzzy arc consistency is applied iteratively until there is one full cycle with no change. As shown in Table 3.2, fix point is reached after one cycle. Table 3.2 contains three constraints. To calculate the upper bound for satisfaction degree (USD), the maximum of membership grades regarding to each constraint is taken (for example, 0.6 for \( X \neq Z \) constraint). Then, by taking the minimum over set of all constraints, we determine the upper bound for satisfaction degree (in this example, USD is 0.6).

In order to perform search, first an ordered set for each updated variable domain is created. Each ordered set is sorted based on the membership grade of unary constraint on each value of domain. According to the updated values depicted in table 3.3, domains are \( D_X = \{B = 0.6, C = 0.5\} \),\( D_Y = \{B = 0.6, C = 0.5\} \) and \( D_Z = \{D = 0.6, C = 0.5, E = 0.2\} \). Now, selecting the maximum value from each set, allows the assignment \( X = B \), \( Y = B \), \( Z = D \). Since the tuple \( (B, B, D) \) is a feasible tuple and has the maximum satisfaction equal to calculated upper bound, there is no need for further search. Maximum satisfaction degree is obtained in the first iteration.
3.2. FUZZY CONSTRAINT NETWORKS

Table 3.1: Unary constraint membership grades before fuzzy arc consistency

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>0.4</td>
<td>0.6</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Y</td>
<td>0</td>
<td>0.8</td>
<td>0.7</td>
<td>0.3</td>
<td>0</td>
</tr>
<tr>
<td>Z</td>
<td>0</td>
<td>0</td>
<td>0.8</td>
<td>0.5</td>
<td>0.2</td>
</tr>
</tbody>
</table>

Table 3.2: Fuzzy arc consistency sample

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>0</td>
<td>0.6</td>
<td>0.7</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Y</td>
<td>0</td>
<td>0.6</td>
<td>0.7</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Z</td>
<td>0</td>
<td>0</td>
<td>0.6</td>
<td>0.5</td>
<td>0.2</td>
</tr>
</tbody>
</table>

3.2.1.3 Specially-Structured Constraint Networks

So far, we solve the general fuzzy CSP problem. However, the constraint network built based on the rules sometimes has a special property. In most common cases, this constraint network is a tree. Having the constraint tree make the CSP solvable in polynomial time.

If there is at most one path between any two variables and no loop in the constraint network, the constraint network is a tree. Based on a theorem for a loop free graph, the CSP can be solved in $O(nk^2)$ time [4]. This is done by ordering variable from root to leaves such that every node’s parent precedes it in the ordering. Then, by applying arc consistency twice in different directions, the fully arc consistent network is obtained. Since we apply the fuzzy arc consistency iteratively and in both directions till we find no more inconsistency, we will achieve the full arc consistency for our constraint tree. Therefore, we do not need to perform any search to find the maximum satisfaction degree of the constraint network.

For example, consider the rule defined for cooking in the domain (see first rule in 2.2). The constraint network which is built to check state of the stove sensor for cooking is shown in 3.3. This network is a tree and in this case, there is no need to search to obtain maximum satisfaction degree.

In contrast with the previous example, in case of having a disjunction of inequality constraints, the resulting constraint network is not a tree. A constraint graph like Figure 3.4 is created for the rule defined in the table 3.4. In this rule, human behavior A is hypothesized, X1 and X2 stand for the requirement X which should be different than b or c and variable Y represents retrieved sensor data. Fuzzy arc consistency is not enough to solve this kind of constraint
Table 3.3: Unary constraint membership grades after fuzzy arc consistency

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>0</td>
<td>0.6</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Y</td>
<td>0</td>
<td>0.6</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Z</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td>0.6</td>
<td>0.2</td>
</tr>
</tbody>
</table>

Figure 3.2: A constraint network (a) before fuzzy arc consistency (b) after fuzzy arc consistency

network and greedy search needs to be applied to determine the maximum satisfaction degree.

In general, we solve the constraint network in a general case to make the framework flexible for more complicated rules in the domain.

3.2.2 Fuzzy temporal Constraint Networks

So far, we have addressed how to check the eligibility of values. In addition to value constraints, we have to consider the temporal requirements in the rules. For instance, Cooking depends on being in the kitchen while the stove is on. Overall, the model contains well defined rules which expresses a question about the value of sensor readings and about the relations in time. In the previous section we have explained how the "value" query is answered. Now, we focus on temporal queries. As explained in Section 2.3.1, the temporal relationships between sensor readings are formulated in Allen’s Interval Algebra and the current model expresses the precise relationships between sensory intervals. In fact, the
Table 3.4: Two rules in a possible domestic activity recognition model

<table>
<thead>
<tr>
<th>Human: A</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>DURING X DIFFERENT</td>
<td>OR</td>
<td>DURING X DIFFERENT</td>
</tr>
</tbody>
</table>

Figure 3.3: A fuzzy CSP with two state variables, three constraints \{c1, c2, c3\} and domain \{ONN, OFF\}. 0.7 for ON and 0.3 for OFF are data obtained from sensor.

rules involved in the model represent the temporal information in the form of qualitative relation between objects. Lack of flexibility addresses us to deal with accommodating imprecision in temporal relation and combining the qualitative relation with quantitative information. For example, two objects should meet each other, but if not, in what degree these two objects meet each other. To build a flexible model considering uncertainty in temporal relations, fuzzy temporal reasoning is proposed. As a case in point, [14] uses fuzzy theory approach for temporal model-based diagnosis. In this diagnostic process, the temporal component is modeled by fuzzy temporal constraints networks, which makes the representation of quantitative and qualitative imprecise temporal information possible.

The idea of applying fuzzy Allen’s Algebra in this thesis is based on [10] which is detailed in the following sections. In this chapter, we are going to show how fuzzy temporal reasoning can be used in our model by means of a composition table in Allen’s Algebra.
3.2.2.1 Conceptual Neighbors

To adopt the composition table for imprecise reasoning, the notion of conceptual neighborhood is proposed [9]. Assume that two objects $O_1$ and $O_2$, are in relation $m$ (meet), then by moving or deforming the objects slightly we can change this relation to $<$ or $o$. Therefore, $<$ and $o$ are conceptual neighbors of $m$. Depending on the types of changes, deformation or moving the events, we obtain different neighborhood structures.

The case of allowing movement of objects with no deformation is considered in this thesis which refers to the B-neighbor relation in Freksa structures [9]. The topological view of conceptual neighborhood, B-neighbor, is shown in the figure 3.5.

As the topological view shows clearly, some relations are closer to others. For example, $m$ is very close to $o$, whereas $<$ is not as close as $m$ to $o$. In this case, we say that $<$ is the neighbor of $o$’s neighbor.

3.2.2.2 Fuzzification of Allen Relations

In order to fuzzify Allen relations, first, a characteristic function for the atomic relation should be introduced. $r$ stands for a crisp atomic relation

$$\mu_r : A \rightarrow \{0, 1\}$$

The domain of $\mu_r$ is the set of atomic Allen relation, i.e.:

$$A = \{<, m, o, fi, di, si, =, s, d, f, oi, mi, > \}$$

$\mu_r$ yields a value of 1 if and only if the argument is equal to the atomic relation denoted by the characteristic function:
The next step towards the introduction of imprecision is to transform the atomic Allen relations into fuzzy sets. For that purpose, we represent each atomic relation as a set of pairs, each pair consisting of an element of $\mathcal{A}$ and the value of the characteristic function of the relation applied to that element [10]. For example, if two objects $O_1$ and $O_2$ are adjacent to each other, i.e., $O_1 \ m \ O_2$, we use the characteristic function of the relation $m$ to convert this statement into the following:

$$O_1 \{(r, \mu_m) \mid r \in \mathcal{A}\} \ O_2 \rightarrow O_1 \{(m, 1), (<, 0), (s, 0),...\} \ O_2$$

Instead of having two classes, one with the accepted relations where $\mu_m$ results in 1 and another with the discarded relations where $\mu_m$ results in 0, we now assign acceptance grades (or membership grades, to use the term from fuzzy set theory) with the relations. If the relation is $m$, we assign the membership grade 1; if the relation is a neighbor of $m$, we choose a membership grade $\alpha_1$ with $1 \geq \alpha_1 \geq 0$; if the relation is a neighbor of a neighbor of $m$, we assign a grade $\alpha_2$ with $\alpha_1 \geq \alpha_2 \geq 0$; and so on. Figure 3.6 illustrates this example.

The conceptional distance between the relation $r$ and the relation $r'$ is defined by a function $\Delta$ such that $\Delta$ results in 1 if $r$ is a neighbor of $r'$, in 2 if $r$ is a neighbor of a neighbor of $r'$, and so on:

$$\Delta : \mathcal{A} \times \mathcal{A} \rightarrow \{0, 1, 2, ...\}$$
Figure 3.6: The atomic Allen relations and their membership grades with respect to the relation m

\[ \Delta(r, r') = \begin{cases} 
0, & \text{if } r = r' \\
\min\{\Delta(r', r'')} + 1 | r''' \text{ neighbor of } r'\}, & \text{otherwise}
\end{cases} \]

Given a sequence of membership grades, 1 = \( \alpha_0 \geq \alpha_1 \geq \alpha_2 \geq \ldots \geq 0 \), the function \( \Delta \) can be used to associate Allen relations with membership grades, depending on some given Allen relation \( r \). In particular, we can define a membership function \( \mu_{\tilde{r}} \) as follows:

\[ \mu_{\tilde{r}} : A \rightarrow [0, 1] \]

\[ \mu_{\tilde{r}}(r') = \alpha_{\Delta(r,r')} \]

With this definition, the fuzzy Allen relation \( \tilde{r} \) of an Allen relation \( r \in A \) is given by the following:

\[ \tilde{r} = \{(r', \mu_{\tilde{r}}(r')) | r' \in A\} \]

The special case of \( \alpha_1 = \alpha_2 = \ldots = 0 \) leads to traditional, crisp Allen reasoning.
3.2. FUZZY CONSTRAINT NETWORKS

3.2.2.3 Fuzzy Composition of Allen Relations

As indicated in section 2.3, Allen’s composition table is applied to propagate the Allen interval relations. In order to fuzzify the composition of Allen relations, again, we should start by crisp relations and continue with the fuzzy relation. In the crisp case, Allen’s composition table (see Table 2.1) can be represented as a set of characteristic functions of the following form:

\[ \mu_{r_1 \circ r_2} : A \to \{0, 1\} \]

The domain of \( \mu_{r_1 \circ r_2} \) is the set of atomic Allen relations, i.e.:

\[ A = \{<, m, o, fi, di, si, =, s, d, f, oi, mi, >\} \]

\( \mu_{r_1 \circ r_2} \) yields a value of 1 for arguments that are elements of the corresponding entry in the composition table (row \( r_1 \times \) column \( r_2 \), denoted by \( r_1 \circ r_2 \)); otherwise, a value of 0:

\[
\mu_{r_1 \circ r_2}(r) = \begin{cases} 
1, & \text{if } r \subseteq r_1 \circ r_2 \\
0, & \text{else}
\end{cases}
\]

We can now define the fuzzy composition \( \tilde{r}_1 \circ \tilde{r}_2 \) of two fuzzy Allen relations \( \tilde{r}_1 \) and \( \tilde{r}_2 \) as the fuzzy Allen relation \( \{(r, \mu_{\tilde{r}_1 \circ \tilde{r}_2}(r)) | r \in A\} \), where \( \tilde{r}_1 \circ \tilde{r}_2 \) is given by the following:

\[
\mu_{\tilde{r}_1 \circ \tilde{r}_2} = \max_{r_1', r_2' \in A | \mu_{\tilde{r}_1 \circ \tilde{r}_2} = 1} \{\min[ \mu_{\tilde{r}_1}(r_1'), \mu_{\tilde{r}_2}(r_2')]\} \quad (3.1)
\]

In order to make a simple and explicable example, we consider fuzzy relations including not all thirteen atomic relations. Consider the constraint \( O_i R_{ij} O_j \) and \( O_j R_{jk} O_k \), where \( R_{ij} = \{(o, 0.5), (m, 0.7)\} \) and \( R_{jk} = \{(<, 0.9)\} \). According to the formula 3.1, composition is defined as

\[
(o, 0.5) \circ (<, 0.9) = (<, 0.5) \quad \text{where } 0.5 = \min(0.5, 0.9)
\]
\[
(m, 0.7) \circ (<, 0.9) = (<, 0.7) \quad \text{where } 0.7 = \min(0.7, 0.9)
\]

The greatest degree to which both \( R_{ij} \) and \( R_{jk} \) can be satisfied is 0.7. In fact, \( R_{ij} \circ R_{jk} = \max\{ (<, 0.5), (<, 0.7)\} \).

3.2.2.4 Applying Path Consistency to Fuzzy relations

As shown in section 2.3, a path consistent network is achieved through an iterative process that looks at three objects at a time and applies the composition table to those objects. Any relation that is not justified by the composition table is inconsistent with the rest of the given relations and therefore is deleted by
the algorithm. In the case of having fuzzy relations, we apply path consistency concept not only to have a path consistent network, but also to define the satisfaction degree of the soft constraint. In fact, we want to apply the extension for the crisp Allen’s algorithm, based on the path consistency concept, to decide the degree of consistency.

Input to the extension of Allen’s algorithm [10] for the fuzzy relation is a set of objects and a set of (not necessarily atomic) fuzzy Allen relations. If there is no relation specified for a pair of objects, it is assumed that the relations between objects is the set of all thirteen atomic Allen relations. The fuzzy path consistency algorithm does not make a yes/no decision about whether a relation is an element of the composition of two other relations, rather it computes membership grade for that relation. This membership grade is compared with the initial membership grade of the relation. If the new grade is smaller than the initial grade, the membership grade of the relation is updated with the new grade. Analogously to non-fuzzy Allen relations, this step can be formulated as follows:

$$\tilde{r}(O_1, O_3) \leftarrow \tilde{r}(O_1, O_3) \cap [\tilde{r}(O_1, O_2) \circ \tilde{r}(O_2, O_3)]$$

The intersection of two fuzzy Allen relations $$\tilde{r}(O_1, O_3)$$ and $$\tilde{r}'(O_1, O_3)$$ is defined by minimizing membership grades:

$$\tilde{r}(O_1, O_3) \cap \tilde{r}'(O_1, O_3) = \{ (r, \min\{\mu_{\tilde{r}(O_1, O_3)}(r), \mu_{\tilde{r}'}(O_1, O_3)(r)\}) \mid r \in A \}$$

Figure 4 shows pseudocode for the extended algorithm. Unlike in the crisp version of Allen’s algorithm, no elements are deleted from the fuzzy Allen relations during the steps of the algorithm, however, their membership grades are updated.

To determine the degree of satisfaction of the temporal constraint network which we are interested in, let us consider the fuzzy interval network where $$n$$ is the number of nodes representing the intervals, $$\alpha_{ij}^*$$ is the maximum value of the membership degrees of the edge $$(i, j)$$ representing a fuzzy Allen relation. The satisfaction degree is defined as

$$\text{Sat-deg} = \min\{\alpha_{ij}^*\} \quad i, j \in \{1, \ldots, n\}$$

The value of Sat-deg shows the degree of consistency of the network. In fact, if the value of Sat-deg is 1, we have a fully consistent network, otherwise, our network is consistent by the degree of Sat-deg value. In the next section, we will explain that fuzzy path consistency is enough for deciding the consistency of the defined temporal network; hence, there is no need to perform a search to find maximum satisfaction degree of the temporal network.
3.2. FUZZY CONSTRAINT NETWORKS

Algorithm 4: Path consistency for fuzzy relations

input: Given a set of objects \( \{O_1, O_2, ..., O_n\} \)
output: Given a set \( \tilde{R} \) of fuzzy allen relation between these objects

while \( \tilde{R} \) is not empty do
    Select a relation \( \tilde{r}(O_i, O_j) \in \tilde{R} \)
    \( \tilde{R} \leftarrow \tilde{R} - \{\tilde{r}(O_i, O_j)\} \)
    for \( k \in \{1, ..., n\} \) with \( k \neq i, j \) do
        \( r(O_k, O_j) \leftarrow r(O_k, O_j) \cap [(O_k, O_j) \circ (O_i, O_j)] \)
        if \( \tilde{r}(O_k, O_j) \) then
            \( R \leftarrow R \cup \{\tilde{r}(O_k, O_j)\} \)
        end
    end
    \( r(O_i, O_k) \leftarrow r(O_i, O_k) \cap [(O_i, O_j) \circ (O_j, O_k)] \)
    if \( \tilde{r}(O_i, O_k) \) then
        \( R \leftarrow R \cup \{\tilde{r}(O_k, O_j)\} \)
    end
end

3.2.2.5 Temporal Tractability

It is well known that Allen Interval Algebra is non-tractable. In particular, the problem of finding a consistent scenario and computing the minimal network are both NP-hard. However, these problems are solvable in polynomial time by taking approaches limiting the expressive power of the temporal temporal language. As a case in point, if we restrict the temporal constraint network to have one relation per edge, Allen algebra can be formulated in terms of a point algebra (PA) in which path consistency decides the consistency in \( O(n^3) \) steps [23].

Our interval network consists of one Allen relation per edge with membership grade of one. A fuzzy network is then obtained by applying the concept of Freksa neighborhood, by which we compute a possibility degree for each of the thirteen relation for every edge. During constraint propagation (see Allen’s algorithm described in section 3.2.2.4), membership grades of each fuzzy relation are updated and, of course, they can only decrease; therefore, in case the value of Sat-deg is 1, we obtain a consistent scenario, i.e., one in which each edge represents the (only) relation with possibility degree 1. This IA network can be translated into a PA network in polynomial time \( O(n^2) \) [4]; path consistency decides the consistency of a PA network in \( O(n^3) \) steps. In terms of our activity recognition application, this means that path consistency is complete for the
case of consistent scenarios. Now, we want to show that the path consistency algorithm applied to fuzzy relations is also complete for inconsistent scenarios.

In case of inconsistent scenarios, i.e., Sat-deg is less than 1 and greater than zero, we want to obtain the highest amount of satisfaction degree of the temporal network. This means that we want to select the relations on each edge that lead to the highest overall possibility degree. Analogously to the case in which Sat-deg is equal to 1, we can consider the fuzzy Allen network (IA$^{fuz}$) obtained by choosing the relation with the highest value of membership grade for every edge. In the crisp case, a PA network was used to prove the completeness of path consistency; similarly, we can use a fuzzy PA (PA$^{fuz}$) in the fuzzy case of Allen network. A PA$^{fuz}$ can be defined in the same way as IA$^{fuz}$. The IA$^{fuz}$ can be translated to the fuzzy point network PA$^{fuz}$ as shown in figure 3.7. It can be shown that the completeness of the path consistency algorithm for PA can be extended to PA$^{fuz}$ networks [22]. For this reason, path consistency is complete for PA$^{fuz}$ which is obtained from IA$^{fuz}$ with one fuzzy relation per edge. Therefore, also in inconsistent scenarios, path consistency will decide whether the network has a non-zero Sat-deg.

![Figure 3.7: Translation of an IA$^{fuz}$ relation to a PA$^{fuz}$ relation](image)

### 3.3 Determining the Possibility Degree of an Activity

So far, we have studied the underlying mechanism of our activity recognition model. We explained two fuzzy constraint networks which ascertain temporal satisfiability and value satisfiability (see Figure 3.8). Then, we applied fuzzy inference techniques in both dimensions. Now, in order to grade the activity inferred through abductive reasoning process, we are facing to the problem
3.3. DETERMINING THE POSSIBILITY DEGREE OF AN ACTIVITY

of multi criteria aggregation to obtain overall possibility degree. Different operators can be employed to achieve this aggregation. A primary factor in the determination of the structure of such aggregation is the relationship between the criteria involved. Consider the cases of wanting, for example, "all" or "at least one of them" or "most" of the criteria to be satisfied. For each of these cases and many of which are not contemplated here, a specific operator is proposed (e.g., t-norm, co-t-norm and ordered weighted averaging (OWA) operators) [24]. In our problem, we desire that both evaluating criteria be satisfied; for this reason, a t-norm operator can be an appropriate choice, e.g., combining two possibility degrees by taking the minimum between them. The investigation of these operators and the other possible methods for fusion remains as a future work.

For the purposes of this work, we employ the minimum possibility degree of the two constraint networks as the final belief of the corresponding human behavior.

\[
\text{Possibility Degree for Human Activity} = \min(\text{USD}, \text{Sat} - \text{deg})
\]

![Figure 3.8: Constraint based activity recognition with uncertainty. The question marks represent uncertainty on the constraints. Arcs are the binary constraint and the square is indicator of unary constraints.](image-url)
Chapter 4
Implementation and experiments

In this thesis, a simulation program for activity recognition with uncertainty is implemented. This program is in Java within the framework developed in Örebro university [15], [8]. In this simulation program, we aim to model the activity recognition process to understand user contexts like [Cooking, Eating, WatchingTV].

4.1 Activity Recognition Process With Uncertainty

So far, we have explained the model which enables us to recognize the activities. This model was based on the concept of fuzzy constraint satisfaction problem. Through Chapter 3, we showed how we can propagate the fuzzy constraint networks. Both fuzzy constraint networks including temporal and value constraint networks are created based on the rules defined in the domain. In fact, the activity’s requirements are modeled in the domain through the rules (see Table 2.2). Now, by knowing all the underlying methods, we focus on the activity recognition process which is implemented as a module. This module contains two concurrent processes: Sensing processes and an Inference process. In sensing processes, we model the current sensor reading in a constraint network. The sensor values are updated over time. Throughout this process, the state variables associated with sensor readings and temporal relation between them are inserted in the constraint network. The inference process which is also a continuous process, adds the state variables and constraints, based on current hypothesis of human activity. While the sensing process is performing, the inference process is called iteratively. The inference process selects a hypothesis of human activity from the domain and adds the state variable which represents the current state of human user, thus, the constraint network is updated over time as a result of changing the hypotheses or changing the sensor data observation. For instance, suppose that the domain contains two rules [Sleeping,
Chapter 4. Implementation and Experiments

Table 4.1: The rule defined in our domain for the human activity like cooking.

<table>
<thead>
<tr>
<th>Human : Cooking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equal Stove EQUAL ON</td>
</tr>
<tr>
<td>During Location EQUAL KITCHEN</td>
</tr>
</tbody>
</table>

Cooking, the inference would be performed once with the hypothesis Sleeping and once with hypothesis Cooking. As we impose constraints into networks, requirements consisting of value constraints and temporal constraints are propagated through the fuzzy arc consistency (see algorithm 2) and path consistency for fuzzy relations (see algorithm 4). If propagation does not fail, in other words, if it has the satisfaction degree more than zero, we say that the hypothesis is possible to degree.

For example, we monitor human activities and the domain contains only one rule shown in Table 4.1. We aim to assess if the human is cooking or not, more precisely, the possibility degree of him cooking. It means that the current hypothesis is Cooking. This hypothesis consists of two requirements, that are, the user should be located in the Kitchen (DURING constraint) and is temporally equal to sensed activity of Stove sensor. The sensor reading retrieved from sensing process module is a value between the interval [0, 1] which shows the degree of belief about the existence of a particular value for the related sensor. For instance, \{<ON, \alpha_1>, <OFF, \alpha_2>\} for Stove state variable and \{<KITCHEN, \beta_1>, <LIVINGROOM, \beta_2>, <BATHROOM, \beta_3>, <BEDROOM, \beta_4>\} for Location state variable.

Let us look more in details at how the constraint network is updated. Suppose that at time \(t = 10\), sensors representing the location of the person have the values depicted in the time line (see Figure 4.1). Inference process is invoked by arrival of a new sensor reading. Abduction based inference process tries to add constraints and requirements regarding to Cooking hypothesis, but obviously this attempt fails (there is no Stove state variable as a necessary requirement). At \(t = 20\), the temperature sensor deployed in the stove starts to show us the value \{<ON, 0.7>, <OFF, 0.3>\}. As the new sensor reading is retrieved, the inference process attempts to match the state variables representing sensor readings with the states variables representing the requirements of current hypothesis. For each requirement, a constraint network is created (see Figure 3.3 related to Stove state variable). As you see in this figure, the constraint network is consistent to the degree of 0.7, regarding the unary constraints modeling uncertainty in the sensor reading.

Apart from the constraint network for value checking, a temporal constraint network is built to assess temporal eligibility of current hypothesis. In this example, expansion of Cooking rule will identify all the pairs of state variables
4.2. OCCURRENCE INTERVAL EXTRACTING

[Location, Stove], here is just one combination, and attempt to constraint them with the current hypothesis of Cooking with the constraint defined in the rule (i.e., equal and during). The temporal network is shown in Figure 4.2. Notice that both StoveOn and LocationKitchen are ongoing activities according to this scenario. There are no new updating for stove and location since the monitoring started. We want to keep the network updated and of course we are not able to know which pattern is going to finish sooner or later. For this reason, three possible conditions \{Finishes \cup During \cup OverlapedBy\} are considered for every pair of ongoing activities. This disjunction of temporal relation is updated by the actual relation between activities when at least one of ongoing activities becomes fixed. One state variable is fixed when new sensor reading with updated value of the same state variable are retrieved form the sensing process. In fact, the disjunction of temporal relation is replaced with one temporal constraint per edge. If the maximum satisfaction degree resulted by applying temporal propagation is 1, the applying of path consistency to fuzzy Allen relations is enough to guarantee the consistency of the temporal network, otherwise, in this specific case (disjunction in temporal relation between ongoing activities), we obtain the upper bound for satisfaction degree.

The Cooking hypothesis can be inferred with satisfaction degree equal to 0.7 which is the minimum of two satisfaction degrees of underlying constraint networks: 0.7 for value checking and 1.0 for temporal requirement.

Figure 4.1: Timelines relevant to the hypothesis that the human user is cooking

4.2 Occurrence Interval Extracting

To draw a timeline from graded activities recognized during the process, we determine the minimum and maximum intervals in which the hypothesis can occur. These intervals are calculated based on following information: the temporal relation between a hypothesis and its requirements and start time and end
time of the occurrence interval of each requirement. Allen relation between the hypothesis and the requirement gives us certain information of start time and end time of the hypothesis. For example, if a hypothesis contains (contain as a temporal relation) an activity, the start time of the activity is the maximum start time of hypothesis and end time of activity would be minimum end time of hypothesis. We are not able to provide any information about minimum start time and maximum end time of the hypothesis based on the Contain relation. Figure 4.3 shows all possible thirteen Allen relations between hypothesis and an activity. On the start time and end time of each activity four kinds of notations are defined: sm representing the minimum start time, em representing the minimum end time, sM representing the maximum start time and eM representing the maximum end time. To define maximum interval of the hypothesis, the minimum start time and maximum end time are considered and for the minimum interval, the maximum start time and minimum end time are used.

This is a preliminary approach to extract occurrence interval of an activity. Since the information provided by each relation does not cover both minimum and maximum start time and end time of an activity, many problem arise in different occasion of temporal relations. There are cases in which we can not say anything about minimum interval or maximum interval, or the length of minimum interval is greater than of maximum interval. We implement an approach to handle these cases, but finding an appropriate way of extracting an occurrence interval is beyond the scope of this thesis.

4.3 A More Realistic Test Case

To study a more complex case, we will introduce another scenario. In this scenario, the system will recognize the Cooking and the Changing clothes and the Sleeping. We define a domain with four sensors and three rules shown in the ta-
Table 4.2: Domain consists of three rules in a possible domestic activity recognition model

| Human: Cooking | Contains Stove EQUALS ON | During Location EQUALS KITCHEN |
| Human: ChangeClothes | Contains Bed EQUALS OFF | StartedBy Location EQUALS Bedroom |
| Human: Sleeping | During Bed EQUALS ON | Contains NightLight EQUALS OFF |

ble 4.2. These sensors can be, for example, a pressure sensor placed beneath the bed, a luminosity sensor placed close to the night light, some cameras employed in different rooms and a temperature sensor for stove.

We provide an example of sensing process depicted in the Figure 4.4. The data is generated based on following scenario: The person enters the kitchen and turns on the stove, but it takes time to become warm enough. At t = 3, the stove sensor shows the value ON in a degree of belief 0.5. This degree changes at t = 5 to 1. Once the user entered the kitchen, the location detector system has also some beliefs about the location of human being in the living room and bedroom. This may happen as a result of some errors in image processing system. For example, some objects block parts of the camera’s view. After a few minutes, the person turns off the stove and directly goes to the bedroom. The stove is still warm and the temperature gradually decreases. When the person enters to the bedroom, she/he starts changing clothes. While this activity is being done, some clothes and the other stuff placed on the bed causing the Bed sensor having a belief of being occupied (it is noted by value ON) in a degree of 0.2. Afterwards, the person lies on bed and turns off the night light at t = 30. It is noticeable that, although the person left the kitchen, the location detector has some idea of the person’s location being in kitchen while this can be because of a dog sitting on the kitchen table.

As explained earlier, for each rule, the inference process must attempt to impose the required constraints between the hypothesis and a number of possible state variables. Consider t = 15, when values of Location are changed; another iteration of trying all hypotheses is started. Cooking is hypothesized by considering all combination of states variable {Location, Stove} existing till the current time of process. Six possible combinations of state variables Stove
and Location until t = 15, are shown in Figure 4.5. The resulting hypothesis of each combination are depicted in Table 4.3. In this table, the minimum and maximum interval in which the hypothesis occurs calculated. Temporal consistency value depicted in this table is the result of applying path consistency to fuzzy relations algorithm. The membership grades defined for the conceptional distance $\Delta$ in this example, are as following

$$\alpha_{\Delta=0} = 1, \quad \alpha_{\Delta=1} = 0.4, \quad \alpha_{\Delta=2} = 0.2, \quad \alpha_{\Delta>2} = 0$$

We are not usually interested in monitoring hypothesis with low possibilities. It is possible to set a desire threshold for monitoring hypotheses higher than the threshold each time. The membership grade for the conceptional distance $\Delta$ can be manually changed regarding to the amount of preciseness in temporal relations we wish to have.

<table>
<thead>
<tr>
<th>Cooking</th>
<th>TC</th>
<th>VC</th>
<th>(minInterval, maxInterval)</th>
<th>Total Consistency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hypothesis 1</td>
<td>1</td>
<td>0.0</td>
<td>([1,3], [1,15])</td>
<td>0.0</td>
</tr>
<tr>
<td>Hypothesis 2</td>
<td>0.2</td>
<td>0.0</td>
<td>([1,3], [15,INF])</td>
<td>0.0</td>
</tr>
<tr>
<td>Hypothesis 3</td>
<td>1.0</td>
<td>0.5</td>
<td>([3,5], [1,15])</td>
<td>0.5</td>
</tr>
<tr>
<td>Hypothesis 4</td>
<td>0.2</td>
<td>0.5</td>
<td>([3,5], [15,INF])</td>
<td>0.2</td>
</tr>
<tr>
<td>Hypothesis 5</td>
<td>0.4</td>
<td>0.9</td>
<td>([5,INF], [1,15])</td>
<td>0.4</td>
</tr>
<tr>
<td>Hypothesis 6</td>
<td>0.4</td>
<td>0.5</td>
<td>([5,INF], [15,INF])</td>
<td>0.4</td>
</tr>
</tbody>
</table>

We provide the two greatest degrees for each hypothesis at time points t = 1, 5, 18, 30 in Figure 4.6. In this figure, We omit the hypothesis with 0 possibility degree if that is the second largest. For each hypothesis, the possibility degree resulted from minimum of temporal degree and value degree and minimum and maximum possible interval are shown.

4.4 Performance

In this section, we intend to evaluate performance of the activity recognition procedure with a specific focus on the fuzzy propagation module. This module presented in this thesis does not significantly decrease the performance of the existing framework which was based on crisp propagation. All tests described in this section were carried out on an Intel Core2 Duo processor @ 2.26 GHz.

We perform one test with a domain containing only one rule. This rule states that an activity should be recognized if it occurs During value A and should contain value B. The sensory input shown in Figure 4.7 was fed to the system over a period of 50 seconds. Over time, number of combinations of
state variables that need to be explored increases. Figure 4.8 shows the CPU
time required by activity recognition process. As shown, the performance of
system grows exponentially with the number of sensory events.

In previous chapter, we proved that the time complexity of the fuzzy propa-
gation both for value checking and temporal network is polynomial regarding
to the size of the constraint network. To analyze the performance of fuzzy prop-
gagation module which is the contribution of this thesis, we record the CPU time
required for the propagation process at each update of the constraint network.
The result is presented in Figure 4.9. The required CPU time grows polynomi-
ally. As it is shown in this figure, a polynomial curve in the degree of 3 is nicely
fitted into the results (with variance 0.0061).

Figure 4.10 shows that how the system performs in a more realistic scenario
which is explained in the Section 4.3. The required CPU time for propagation
is also depicted in Figure 4.11.
Figure 4.3: Thirteen Allen relations between hypothesis and activity. The information provided by each relation are mentioned on the interval of activity.
4.4. PERFORMANCE

Figure 4.4: Timeline reflecting uncertainty on the sensors

Figure 4.5: Six possible combinations of state variables Stove and Location for the example scenario till $t = 15$. 
Figure 4.6: Two greatest degrees for each hypothesis at time points $t = 1, 5, 18, 30$

Figure 4.7: Patterns used in the performance test in Figure 4.8 and Figure 4.9
4.4. PERFORMANCE

Figure 4.8: CPU time required to provoke process of inferring an hypothesis using a rule with two requirements.

Figure 4.9: Propagation performance for the pattern defined in the Figure 4.7
Figure 4.10: CPU time required to provoke process of inferring an hypothesis using three rules with two requirements.

Figure 4.11: Propagation performance for the pattern in the scenario defined in the Figure 4.4
Chapter 5
Conclusions

5.1 Summary

This thesis aimed at extending the activity recognition model and reasoning infrastructure to deal with uncertainty. Considering uncertainty in sensor readings is very intuitive, for example, when we deal with the sensors embedded in a domestic environment. In this thesis, we accommodate uncertainty in the sensor readings and temporal relations between activities. The reason is that the occurrence of an activity is recognized if one or more certain sensors have certain values and there exists a specific temporal relation between this activity and corresponding sensor readings. These requirements for each activity are defined with temporal and value rules in what we call a domain. Checking the eligibility of sensor requirements for each activity is modeled in a so-called value constraint network. The previously developed approach to activity recognition was based on the temporal constraint propagation and there was no value constraint network defined within the model. In the value constraint network, variables are sensory requirements containing all possible values and their corresponding possibility degrees. This possibility degree shows how reliable the value is. The need to consider all possible values for a sensor model arises when we take into account uncertainty in the sensor readings (e.g., the value of stove can be ON with a certain possibility degree and also OFF with another possibility degree).

Dealing with uncertainty in the constraint network led us to use the notion of fuzzy constraint satisfaction problem. In fact, we use fuzzy reasoning techniques in both temporal and value constraint networks. In the value constraint network, the possibility degrees obtained from the uncertain sensor readings are modeled as soft unary constraints. The constraint propagation of this fuzzy network gives us the maximum degree of satisfiability. The maximum degree of satisfiability determines how much current sensor readings can support the value requirements of the hypothesized activity. The maximum satisfaction de-
CHAPTER 5. CONCLUSIONS

greater than zero shows that there is at least one feasible assignment for the state variables.

Moreover, our activity recognition inference process is also based on temporal constraint propagation and temporal knowledge is represented as relations in Allen’s interval algebra. Since the temporal relations among the uncertain sensor readings are very important in our inference process, the model cannot be limited to have crisp temporal constraints. For this reason, the temporal aspect of the rules are also fuzzified. The concept of Freksa neighborhood is used to fuzzify Allen relations. To propagate fuzzy temporal constraints, path consistency is applied on the fuzzy temporal constraint network. The result of constraint propagation would be a graded consistent network in which the maximum satisfaction degree represents the degree of consistency.

Finally, we fuse two obtained satisfaction degrees to indicate that in what degree the activity recognition system believes in deduced activity. It is noticeable that, the time complexity of propagation algorithms for fuzzy constraint networks remains in the same category as it was in the crisp case.

To summarize, the important contributions of the work done during this thesis are the following:

• For each rule, the inference process attempts to impose required constraints between the hypothesis and a number of possible state variables whose values unify with the requirement sensor values. The problem of unification is extended to solve a value constraint network. In fact, we model the problem of checking the requirements of each activity into a constraint network. Value constraints include both equality and inequality whereas existing constraint-based recognition systems cannot deal with inequality constraints.

• An existing activity recognition model based only on crisp temporal constraints was replaced by an activity recognition model based on fuzzy constraint networks. The problem of accommodating uncertainty into a crisp model is solved, using the notion of soft unary constraint and conceptual neighborhood.

• Two state-of-the-art algorithms for propagation of fuzzy constraints are applied and also implemented in the preexisting constraint-based reasoning infrastructure developed at Örebro university.

5.2 Future Work

The current system has a lot of space for improvement. For instance, it would be interesting to use the obtained combined possibility degrees to prune the space of possible timelines and to combine multiple hypotheses into candidate timelines. Finding a solution to deduce temporal bounds for the recognized activities is also an important aspect to be investigate in future work. Furthermore, future
work can include a study of different, potentially more sophisticated strategies for combining temporal and value possibility degrees.

It is essential to improve the performance of the current overall activity recognition algorithm. As described in this thesis, the performance of the system is mainly governed by propagation cost, and the number of combinations that needs to be tried during the inference process. The current implementation maintains those state variables which are fixed and already used for inferring an specific hypothesis in order to avoid re-inferring hypotheses that have been already inferred. However, this is a very basic form of optimization, and further work should be done to increase the amount of information that is carried between iterations. For instance, the performance of the approach could be improved by reducing the number of combinations that need to be tried by implementing some more advanced form of filtering.

In this thesis, the developed approach has been evaluated in a synthetic scenario (although inspired by a real-world application). As the next step, we are interested in studying the performance of this system in real scenarios with real sensors.
References


