UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

VÍTOR FORTES REY

An Ontology-Driven Evidence Theory Method for Activity Recognition

Thesis presented in partial fulfillment
of the requirements for the degree of
Master of Computer Science

Advisor: Prof. Dr. Edson Prestes

Porto Alegre
January 2016

CIP — CATALOGING-IN-PUBLICATION

Fortes Rey, Vítor

An Ontology-Driven Evidence Theory Method for Activity Recognition / Vítor Fortes Rey. – Porto Alegre: PPGC da UFRGS, 2016.

109 f.: il.

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2016. Advisor: Edson Prestes.

1. Dempster–shafer theory. 2. Ontology. 3. Evidence theory. 4. Activity Modelling. 5. Activity recognition. 6. Smart home. I. Prestes, Edson. II. Título.

“It is really quite impossible to say anything with absolute precision, unless that thing is so abstracted from the real world as to not represent any real thing.”

— Richard Feynman

ACKNOWLEDGEMENTS

I would like to thank Edson Prestes, who was fundamental to this work. I thank him for his enthusiasm, encouragement, and resolute dedication to humanitarian causes. In fact, I would like to thank all the members of the Φ Robotics Research Lab, for the discussions, feedback and camaraderie. Specially, I would like to thank Vitor Jorge and Renan Maffei, who were fellows in many interesting projects. I would also like to thank Mariana Kolberg for her revisions and encouragement

In fact, I would like to thank many of my colleagues, even out of the reseach group. Specially, Joel Luis Carbonera and Sandro Rama Fiorini, for the discussions and assistance. And, of course, Mara Abel, for years of ontology training. I would also like to thank the members of the LiSSi laboratory, who were very kind to me. In special, Yacine Amirat and Abdelghani Chibani.

I would like to thank my friends and, specially, my girlfriend, Adriana, who was always there for me when I needed.

Finally, I would like to thank my family, who I love most in the world.

ABSTRACT

Activity recognition is a vital need in the field of ambient intelligence. It is essential for many internet of things applications including energy management, healthcare systems and home automation. But, even with the many cheap mobile sensors envisioned by the internet of things, activity recognition remains a hard problem. This is due to uncertainty in sensor readings and the complexity of activities themselves. Evidence theory models provide activity recognition even in the presence of uncertain sensor readings, but cannot yet model complex activities or dynamic changes in sensor and environment configurations. This work proposes combining knowledge-based approaches with evidence theory, improving the construction of evidence theory models for activity recognition by bringing reusability, flexibility and rich semantics.

Keywords: Dempster–shafer theory. ontology. evidence theory. activity Modelling. activity recognition. smart home.

Uma Abordagem baseada em Ontologias e Teoria da Evidência para o Reconhecimento de Atividades

RESUMO

O reconhecimento de atividaes é vital no contexto dos ambientes inteligentes. Mesmo com a facilidade de acesso a sensores móveis baratos, reconhecer atividades continua sendo um problema difícil devido à incerteza nas leituras dos sensores e à complexidade das atividades. A teoria da evidência provê um modelo de reconhecimento de atividades que detecta atividades mesmo na presença de incerteza nas leituras dos sensores, mas ainda não é capaz de modelar atividades complexas ou mudanças na configuração dos sensores ou do ambiente. Este trabalho propõe combinar abordagens baseadas em modelagem de conhecimento com a teoria da evidência, melhorando assim a construção dos modelos da última trazendo a reusabilidade, flexibilidade e semântica rica da primeira.

Keywords: ontologias,teoria da evidência,modelagem de atividades, reconhecimento de atividades, casas inteligentes.

LIST OF ABBREVIATIONS AND ACRONYMS

ADL	Activities of Daily Living
bba	basic belief assignment
DAG	Directed Acyclic Graph
DS	Dempster-Shafer
fod	frame of discernment
TBM	Transferable Belief Model
DOLCE	The Descriptive Ontology for Linguistic and Cognitive Engineering
SSN	The Semantic Sensor Network ontology

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

The rise of cheap mobile devices, be them sensors, personal computers or both, as is the case with smart phones, is on its way to realize the vision of ubiquitous computing (ARNDT et al., 2013). This vision of computers any and everyware has the potential to provide essential services such as ambient assisted living and healthcare for elderly persons living alone, smart security systems, among others (YE; DOBSON; MCKEEVER, 2012).

This services rely on the behavior of the users, and so they need to recognize what are the activities they are engaged in. For example, a smart cooking application needs to know if a person is still actually cooking when the stove was left on.

In general, we can define activity recognition as the process of inferring what are the activities being performed by an agent based on sensor readings and other contextual information (OKEYO et al., 2012) such as the time of the day.

Today, even with the rise of cheap sensors, activity recognition remains a hard problem because:

- scenarios are dynamic, with sensors added, lost or replaced (ROGGEN et al., 2013).
- sensors can misfire or provide uncertain or even conflicting information (YE; DOBSON; MCKEEVER, 2012).
- some activities require inter-activity dependencies that require temporal reasoning (OKEYO; CHEN; WANG, 2014).
- activities can be complex and may take many forms depending on user or the context.

In this work we will focus on activity recognition in smart home scenarios, where activities of daily living (ADLs) of a subject living alone are detected. An activity of daily living is a deliberate self care activity, such as showering, eating, using the toilet and moving around the house. Detecting activities in this type of scenario is important in the healthcare domain as those activities are used to monitor the autonomy of elderly subjects (NOELKER; BROWDIE, 2013). In other words, automating activity recognition has the potential to improve the autonomy and safety of the aging global population, by tracking user habits for further study or detecting abnormal or even harmful situations in real time.

We assume that the subject is performing only one activity at a time. While a person surely can perform two activities at the same time, as when one reads a newspaper while having breakfast, those scenarios are not prominent in the ADL domain. For example, in a popular guide for detecting elderly autonomy (SHELKEY; WALLACE, 1998), no concurrent activi-

ties are cited. Moreover, few real world datasets containing concurrent activities are available, making testing such a model hard.

This work, as many others, focuses in simple, binary sensors like the ones shown in Figure 1.1. The limiting of the sensors to only simple status sensors, such as movement sensors, excluding intrusive monitoring devices such as cameras also reflects the subjects aversion to constant surveillance (FABIEN et al., 2011). Whats more, wireless status sensors are easy to deploy, being basically "tape and forget", since their expected battery life is measured in years.

Figure 1.1: Example of binary sensors deployed in smart homes.

Source: (TAPIA,)

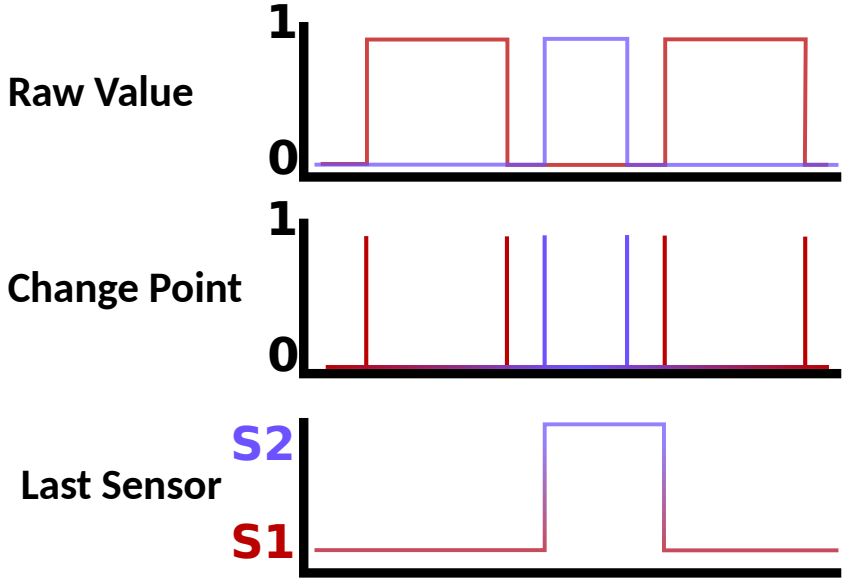
1.0.1 Features for Activity Recognition

Sensor information must be represented in some form, in order to be used to perform activity recognition. The most straightforward choice is to gather, from each sensor, the *raw value* it provides through time. But, as we see in Figure 1.2, this is not the only feature that can be extracted from sensors. For example, for binary sensors, we can also work with sensor

change points, where activations are generated every time a sensor value changes. Features can also capture global information, such as the *last sensor* that was fired in the home.

Those are not the only feature types possible for activtity recognition. Many other feature types can be designed based on domain knowledge. Moreover, if some semantic information is available regarding the sensors, their values can be used to build higher level features, such as subject location.

Figure 1.2: Example of features that can be generated based on sensor firings.



Source: The author

1.0.2 Problem Formal definition

Given a smart environment, habitated by a single subject and equipped with simple status sensors, activity recognition will be defined as the problem of estimating, for each feature assemble at time t in the form $f_t = \{f_t^1, \ldots, f_t^n\}$, the activity a_t that is happening at that time, from a set of possible activities $a_t \in \{a_1, \ldots, a_m\}$. The lenght of the evaluated timeslices, $\Delta_t = t_{i+1} - t_i$, is usually set as one minute, as this duration is considered long enough to discriminate short activities and short enough to provide high accuracy in the user labeling process (KASTEREN et al., 2008).

In order to generate f_t, an activity recognition method must process the series of raw sensor readings $\vec{S}_{0:t}$ in the form

$$\vec{S}_{0:t} = \begin{pmatrix} s_0^0 & \cdots & s_0^{|s|} \\ \vdots & \ddots & \vdots \\ s_t^0 & \cdots & s_t^{|s|} \end{pmatrix}$$

where s_t^i represents the raw value of sensor s^i at time t. As a convention, the first time for the activity recognition process will always be $t = 0$.

Of course, as an activity recognition method receives sensor values in real time, it is clear that not all lines of $\vec{S}_{0:t}$ are relevant for recognizing a_t. For example, the fact that a fridge sensor was activated yesterday may not be relevant to recognizing a sleep activity in the present. For this reason, activity recognition methods will choose a subset of $\vec{S}_{0:t}$, in the form $\vec{S}_{t':t}$ with $t' < t$ for generating the feature vector f_t. The process of choosing t' for every t is named *activity segmentation*, as we are segmenting which sensor values are relevant for which activitites. Segmentation strategies are diverse, ranging from fixed windows to dinamic approaches. If, for every activity instance a^i, starting at time ts_a, $\vec{S}_{t':t}$ is defined as $\vec{S}_{ts_a:t}$, we denominate the segmentation *perfect segmentation*.

Evaluating activity recognition assumes the existence of some ground truth information regarding activity occurrences. This means that we must have a function mapping every t to an activity $a_t \in \{a_1, \ldots, a_m\}$. More formally, ground truth information defines a function GT

$$GT : t_v \in [t_S, t_E] \rightarrow a_t \in \{a_1, \ldots, a_m\}$$

where t_S and t_E represent the start and end times of the ground truth labeling, respectively.

Naturally, in any activity recognition process, we usually want to detect certain activities from all the possible ones the user can perform. For that reason, when building the set of possible activities $\{a_1, \ldots, a_m\}$ we will list all the ones we want to recognize and add another representing "none of those". This activity reprents the times the user is not performing any of the relevant activities, and is usually named "idle".

1.0.3 Overview of Activity Recognition Approaches

Approaches to activity recognition can be divided into three categories: knowledge-based, data-driven and hybrid approaches (OKEYO; CHEN; WANG, 2014).

1.0.3.1 Data-driven Approaches

Data-driven approaches apply machine learning techniques to match sensor and contextual data to activities. They can recognize activities in complex environments even in the presence of uncertainty, but first require annotated datasets representing labeled instances of the activities present in the environment. For most environments, such as smart homes, it is unrealistic to assume such datasets are present before deployment. Besides, data-driven techniques learn to recognize activities based on the training data, which represents a **specific** system configuration and so cannot account for changes in sensor placement, sensor availability and user activity patterns. Finally, such approaches also have to consider the overfitting problems that may arise in cases where labeled instances of some activity are scarce while the context information parameters related to it may be diverse and numerous. This is a well-known issue and was pointed out in (LESTER et al., 2005).

1.0.3.2 Knowledge-based Approaches

Knowledge-based approaches focus on creating high level descriptions of activities. They perform activity recognition by finding the best activity description that matches the current sensor and context information.

The idea is to describe activities by making explicit what are the situations in the system context that characterize them, making it possible to create activity descriptions that work independent of sensor configuration. Moreover, those approaches provide, by definition, intelligible description of what constitutes each activity and can even encode user preferences when performing activities as was done in (CHEN; NUGENT; WANG, 2012).

Current knowledge-based approaches, however, are not without their faults. Activity models are represented (normally in ontologies) using some logic language such as description logics. This logical formalisms were not intended to handle all the aspects of the activity recognition problem. For example, they don't natively incorporate reasoning under uncertainty or models that evolve over time. In fact, due to trade-offs between expressibility and reasoning complexity, they may not be expressive enough to represent some aspects of the activity model (RIBONI et al., 2011). This has led to two different approaches: The first is extending the languages to include time (LUTZ; WOLTER; ZAKHARYASCHEV, 2008), uncertainty (HELAOUI et al., 2012) and other aspects (ROY et al., 2011). The second is using external tools to do part of the reasoning. Using external tools to handle aspects such as time has been successful as in approaches such as (RIBONI et al., 2011), while the extension of the languages

is still ongoing and may not be feasible for real world setups due to performance issues and/or modeling complexity.

1.0.3.3 Hybrid Approaches

Hybrid approaches try to combine the flexibility and re-usability of knowledge-based modeling with the resilience of data-driven algorithms. They may incorporate domain knowledge in the model before the learning process takes place, as is the case in Bayesian networks where the structure of the graph can be created by a specialist while the weights can be learned from the training data. Other approach, as seen in (ROGGEN et al., 2013), is to use ontologies for configuring a recognition-chain composed of data-driven methods. Work on unifying both approaches is ongoing, and no approach has proven itself accurate and scalable in performing activity recognition in real world scenarios.

1.1 Ontologies for Activity Recognition

Ontologies are the most promising and expressive models fulfilling the requirements for modeling context information (RODRÍGUEZ et al., 2014). In this section, we will shortly explain what is an ontology and talk about some of the ontologies developed for activity recognition.

1.1.1 Ontology

The term “ontology” itself has different meanings in different fields. In this work we are refering to an ontology as “an explicit specification of a shared conceptualization”(STUDER; BENJAMINS; FENSEL, 1998).

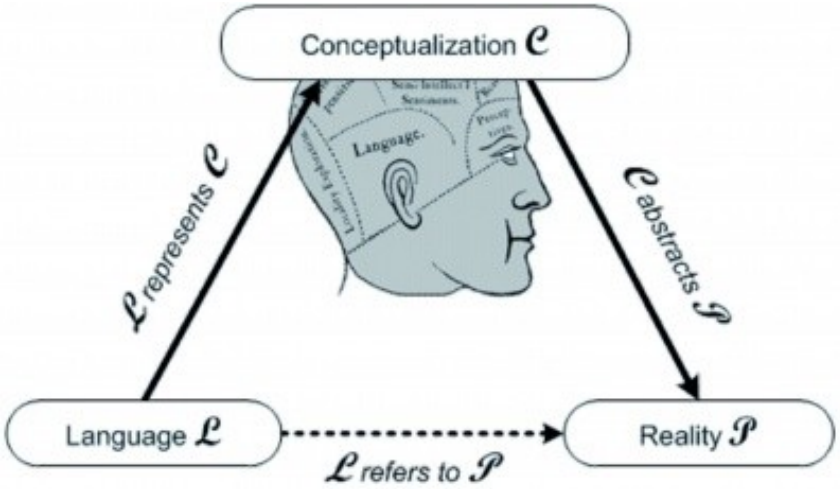
In more details:

- The specification is explicit, meaning that all the information is formally defined in the ontology. In this definition lies one of the basic strengths of ontologies, as being explicit means they can be understood (and therefore used) by both machines and men.
- The ontology is a specification of a conceptualization, meaning it models the concepts we have of things rather than the things themselves. This means that an ontology about persons might have a concept for “Man”, but it never possesses a concept for “Socrates”.

- The conceptualization is also shared, meaning it complies with the intended meaning a community shares about the concepts and not that of a single individual. This means the ontology constitutes a shared meaning on a vocabulary.

An ontology is usually made of a taxonomy of concepts, a set of relations and axioms (PRESTES et al., 2013) represented in some logic-based language, such as first order logics or description logics. The ontology does not directly model the objects of reality. Instead, it models the shared conceptualization a community uses to abstract those aspects of reality, as depicted in Figure 1.3.

Figure 1.3: The Ullman triangle shows the relationships between conceptualization, language and reality.

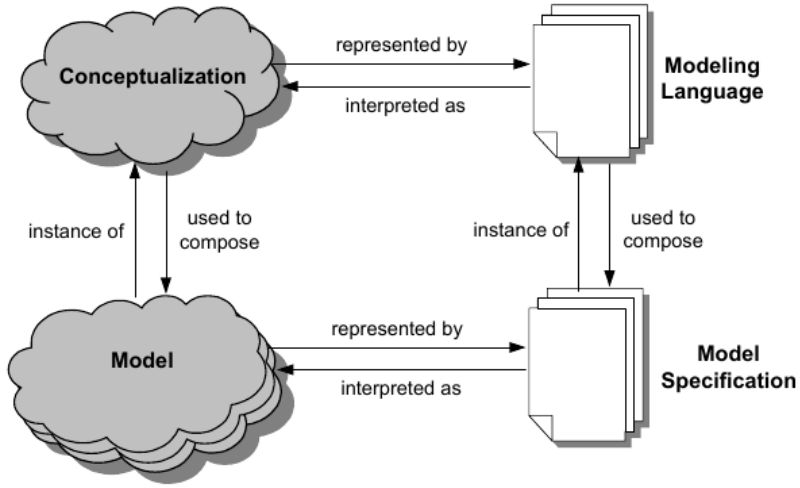


Source: (THE...,)

A conceptualization is an abstract entity, existing only in the mind of a user or community of users (GUIZZARDI, 2005). It captures the concepts of the domain, representing the main abstractions needed to represent the state of affairs in it. A specific state of affairs possible in the conceptualization is named a model (GUIZZARDI, 2005). The relationship between conceptualization, model, modeling language and model specification can be seen in Figure 1.4.

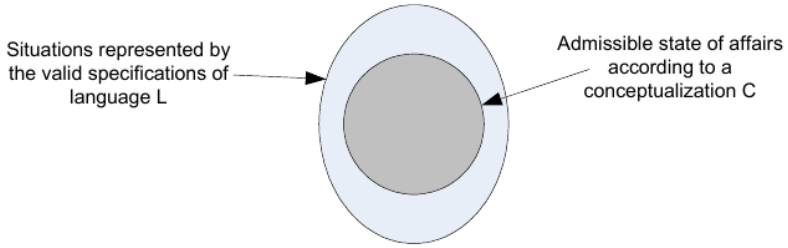
The separation between the conceptualization and its implementation in some logic language is also a very important in ontological design. A conceptualization C defines restrictions on reality for the domain, constraining the possible states those entities may be in. If C is represented using a specific language L, it means we should be able to build all models that conform to C using L. Thus, the quality of the ontology can be measured by evaluating how close L comes to C. It is possible that L was made too general, allowing models not valid in C, as represented in figure 1.5. On the other hand, if L is not generic enough, it will leave out some

Figure 1.4: The relationship between conceptualization, model, modeling language and model specification.



Source: (GUIZZARDI, 2005)

Figure 1.5: A language may be too general, allowing for models that don't conform to the targeted conceptualization.



Source: (GUIZZARDI, 2005)

valid states of affairs of C. Of course, both situations can happen at the same time. It may be the case that L is, in some areas, too generic, while in others being too restrictive.

1.1.1.1 Top, Core and Domain ontologies

In order to enhance compatibility between two related ontologies that were developed separately or simply to avoid rework, ontologies can be built in layers of descending level of generality. According to (PRESTES et al., 2013), an ontology can be classified regarding its generality level as:

Top-level ontologies define very general concepts like time, space, events and matter. Examples of top-level ontologies are SUMO(NILES; PEASE, 2001) and Dolce (GANGEMI et al., 2002).

Core ontologies define the concepts of some generic domain according to a top-level ontology. For example, it defines that, in the robotics domain, a robot is a physical object. So a core ontology defines terms that are not so general as space and time, but are common enough in their area that they see a lot of reuse. For example, the concept of robot in the many domains related to robotics and automated systems and the notion of gene biology. Examples of core ontologies are CORA(PRESTES et al., 2013) and the Gene Ontology (ASHBURNER et al., 2000).

Domain ontologies define the concepts of the domain at hand according to a core ontology. For example, model the domain of car-washing robots or Cyanobacteria.

This hierarchy, while useful, is not mandatory. Semantic web applications, for example, often skip the first two levels and simply start defining the domain concepts. Domain ontologies can also not use a core ontology and use the top-level directly instead.

1.1.2 Ontologies for the Activity Recognition Domain

An extensive number of ontologies that describe context information for representing and recognizing human behavior can be found in the literature (RODRÍGUEZ et al., 2014).

Regarding activity recognition scenarios, those ontologies describe:

- Spatial notions such as the places activities take place. This may include house layout, latitude and longitude, among others.
- Temporal notions such as the times an activity takes place. This can include the time right now, the time of the day an activity happens, among others.
- Information regarding the subjects, that is, the agents who perform activities. This may include their age, their friends and family, among others.

Table 1.1: Analysis of some of the ontologies that can be used for human activity recognition.

Ontology	Location	Time	User	Service	Privacy	Devices
CoBrA-Ont: Chen, Finin and Joshi (2003)	X	X	X	X	X	X
CoDAMoS: Preuveneers et al. (2004)	X	X	X	X		X
Delivery Context Ontology: Cantera-Fonseca and Lewis (2009)	X			X		X
SOUPA: Chen, Finin and Joshi (2005)	X	X	X		X	X
mIO!: Villalon et al. (2010)	X	X	X	X		X
PalSPOT: Riboni et al. (2011)	X	X	X			X
CONON: Wang et al. (2004)	X	X	X			X
PiVOn: Hervás, Bravo and Fontecha (2010)	X	X	X	X		X
Situation Ontology: Yau and Liu (2006)	X	X	X			X

Source: The author

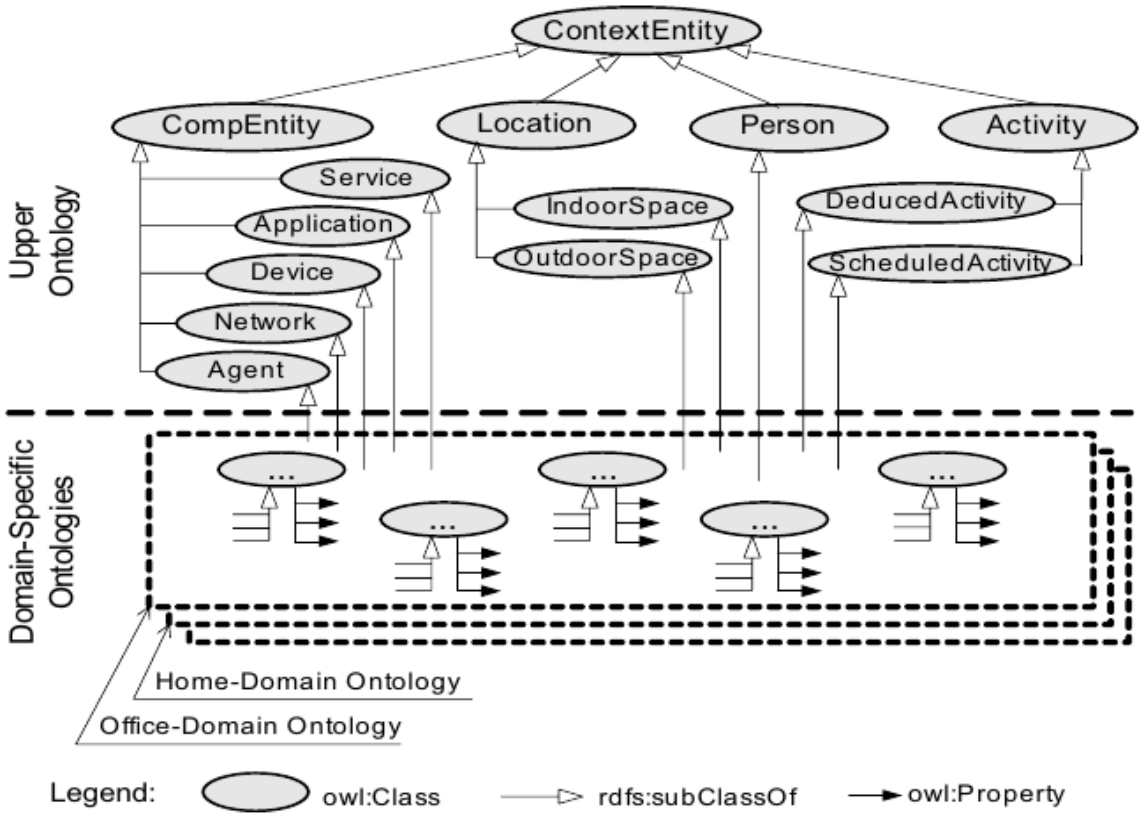
- Software agents and services available at smart spaces. This may include the description of smart applications/services present in the house, among other things.
- Privacy policies regarding the context information it describes. That is, who should have access to this information, among other things.
- The devices present in the smart space. This may include device information such as battery range, type, screen size, among others.

Table 1.1 lists some of the ontologies in the literature that can be used to represent context information relevant to activity recognition.

Some of those ontologies, such as CONON, have a separated, upper ontology for describing location, time and other upper level concepts. Part of said ontology can be seen in Figure 1.6. Those upper level ontologies are not well a stablished ones, such as SUMO(NILES; PEASE, 2001) or DOLCE(GANGEMI et al., 2002), but specific ontologies designed based on their project's needs. Using them, specific home ontologies are defined as can be seen in Figure 1.7. The SOUPA ontology is divided between its core and extensions. In this case, it blurs the line between core and upper level ontology, since it has, in its core part, subontologies for time, space and policy.

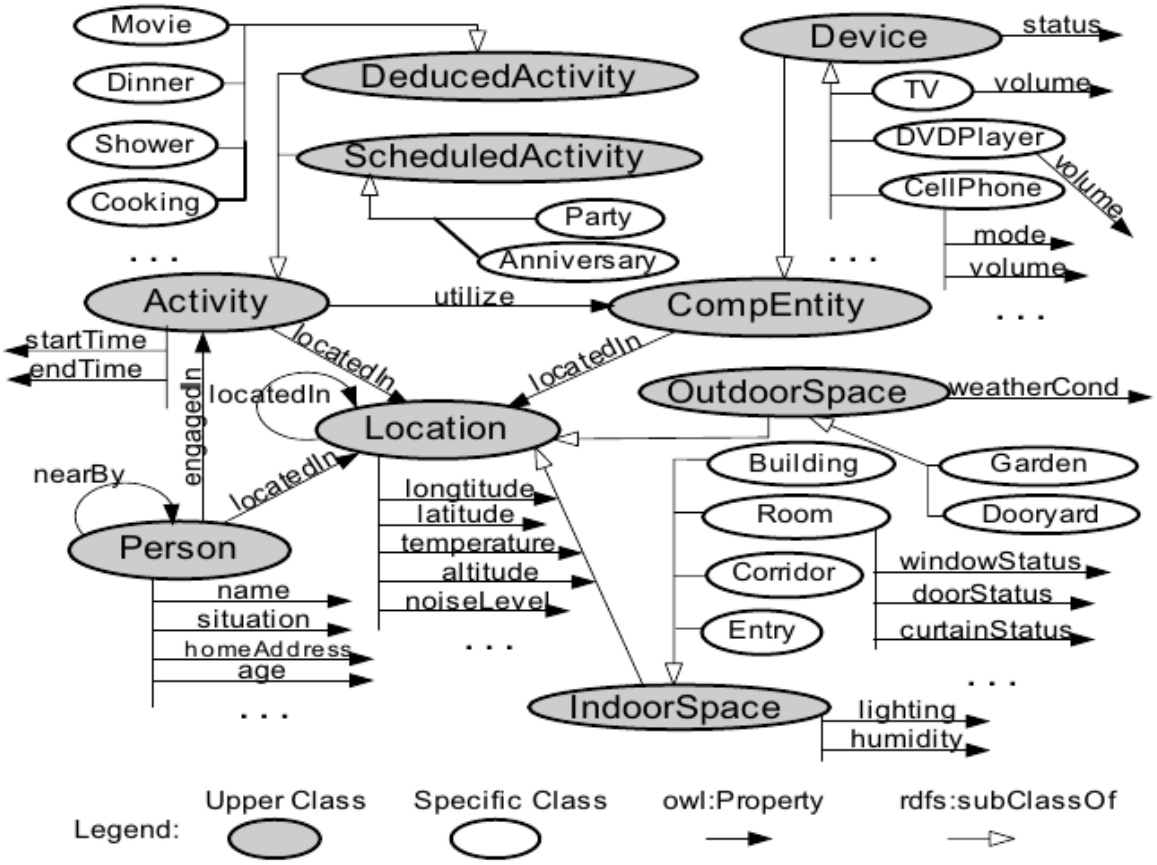
We will not go into more detail about the differences in the activity recognition ontologies found in the literature. This is because, in this work, instead of choosing an existing ontology, we will develop ours using DOLCE, a well stablished upper level ontology. This way, our approach can be applied with any of the reviewed ontologies, as long as a mapping from said ontology to DOLCE is provided.

Figure 1.6: Part of the CONON upper level ontology.



Source: (WANG et al., 2004)

Figure 1.7: Partial definition of a specific home domain ontology using the CONON upper ontology.



Source: (WANG et al., 2004)

1.1.3 The Descriptive Ontology for Linguistic and Cognitive Engineering

The Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) is a top level ontology. DOLCE has a cognitive bias, meaning that it is aimed at capturing common sense (GANGEMI et al., 2002). This does not mean that it only describes physical things. A plan, for example, is an abstract entity that can be instantiated, since, for example, I may refer to the specific plan I have in mind right now.

In fact, in DOLCE, everything is an *entity*, which is either concrete or abstract. An entity is said to be *abstract* if it does not exist in space/time. For example, numbers and functions are abstract, since they are not located anywhere is space or time (GANGEMI et al., 2002). A *region* is also *abstract*, even if it is a spatial one. For example, the Euclidean space is not located in any real spatial region, and so is *abstract*. For the same reason, regions in time are also *abstract*, just as regions in color spaces.

In DOLCE, there are two categories for things that are concrete. They are the *endurant* and *perdurant*. A *perdurant* is something that extends in time, and thus not all of its parts may be present at any single time (GANGEMI et al., 2003). For example, a birthday party is a *perdurant*, since it is an event that happens over time. In other words, not all stages of the party are present at the same time. On the other hand, an *endurant* is always present as a whole. For example, whenever a sculpture is said as present in a room, it is wholly present. The sculpture is not extended in time, since it is the same sculpture at all times, but it can participate in a *perdurant*. For example, a sculpture can be part of a specific exhibition at one time, later being moved to another. The exhibition genuinely changes in time, losing or gaining sculptures, but the sculptures themselves endure. Even if a specific sculpture has its color changed, it retains its identity, after all, we are still referring to the same individual. On the other hand, a *perdurant* can genuinely change in time. For example, a discussion may change its topic and participants in time, until no resemblance to its former properties exists.

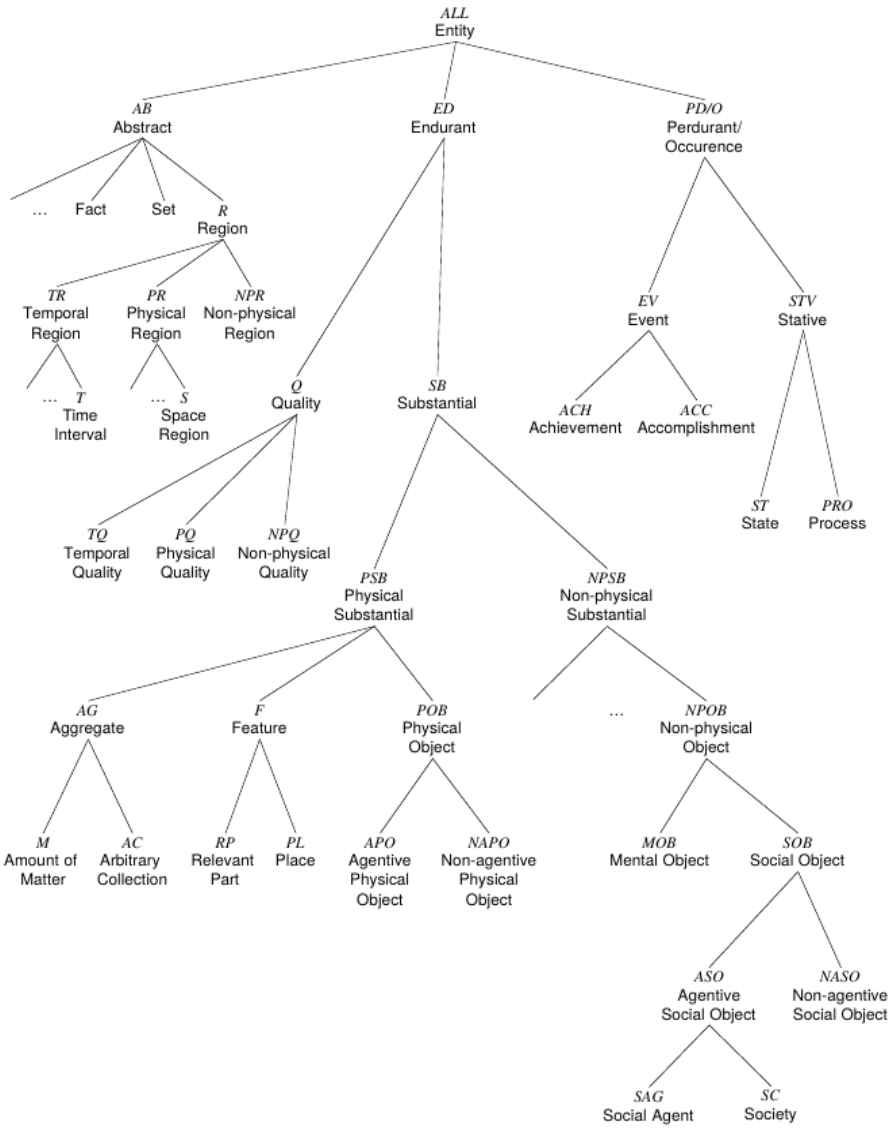
An *endurant* can be either be a *substantial* or a *quality*. A *substantial* is that which can have qualities without being one. A *quality* is something we can measure or perceive, such as the length of an object, its shape or color. It is always inherent to the thing that possesses it. For example, a piece of rock's length cannot exist without the piece of rock itself. There is also a basic difference between the rock's length and its value. The length is the *quality* itself, which means that, after sculpting the piece of rock it remains with the same length *quality*, which has only changed in value. So, if its length is now 30 centimeters, where it previously was 60, the rock did not acquire a new length *quality*, but its current one changed its value. In fact, its value

just moved down in the *abstract* partial length *region*.

*Substantial*s are divided between physical and non-physical, where physical *substantial*s are those that have direct spatial *qualities*. For example, a car is located somewhere, but parking laws are not. Other examples of non-physical *substatials* include laws, political organizations and companies.

Of course, we haven't covered here all of DOLCE, but only some of its most general parts and others relevant to this work. There is much more to be said about it, as we can see in Figure 1.8, which shows the taxonomy for DOLCE's basic categories.

Figure 1.8: Taxonomy for DOLCE's basic categories.



Source: (GANGEMI et al., 2003)

As we have seen in this section, DOLCE makes clear ontological choices. Using an ontology such as DOLCE gives us a good basis for the modeling process and can later help us compare between different ontological choices (OBERLE et al., 2007).

1.1.4 Situation Observation Design Pattern

The description and situation (D&S) ontology is an addition to DOLCE that allows for a separation between facts and their interpretations. It works with two primitives: Situations and Descriptions.

- A situation is constituted by the entities and relations belonging to some state of affairs. Thus, a situation is a DOLCE non-physical *substantial* that represents the aggregated information.
- A description is an entity that partly represents a theory that can be conceived by some agent. Thus, a description is a DOLCE non-physical *substantial* that represents a conceptualization.

A state of affairs (situation) can have different interpretations under different theories (descriptions). For example, the same set of symptoms are explained differently by different diagnosis.

Using D&S, we can represent, in a single ontology, several possible interpretations to a same set of facts. Evenmore, we can define a mapping (even if partial) of how assertions in the situation are interpreted according to each theory.

In this work, we will represent every activity class as a description, and, for each activity possibly happening, we will use the mappings from the current situation to its description to determine if the activity is happening.

Some other recent works represent activities as Descriptions in D&S(MEDITSKOS; DASIOPOULOU; KOMPATSIARIS, 2015) (MEDITSKOS et al., 2013). Our approach differs from them in two important ways:

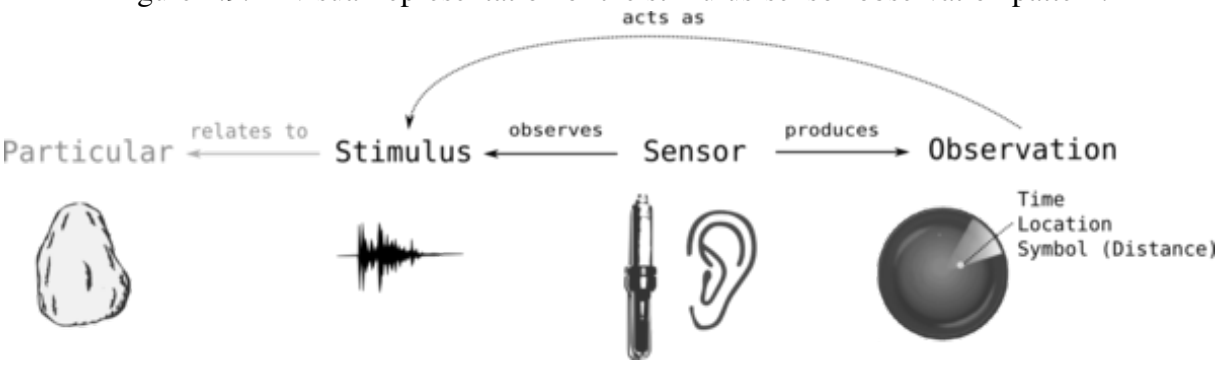
- We represent description as theories, evaluating the mappings from situations to their descriptions using ontological reasoning.
- Our situation evaluation models the uncertainty present in the sensor data.

1.1.5 The Semantic Sensor Network Ontology

The Semantic Sensor Network ontology (SSN) is an ontology focused in representing sensors, from many points of view. Its main objective is to semantically describe sensors. This include a sensor's capability, its different modes of operation, operating restrictions, current deployment (including placement), outputs, among other aspects.

It implements the stimulus-sensor-observation ontology design pattern. That is, a sensor, which is a physical object, detects a stimulus, that is some detectable change in the physical world, providing some result in the form of an observation (which is a situation in the D&S sense). For example, a pressure sensor in a chair seat is a sensor that detects pressure stimuli, generating observations that describe the measured pressure. A visual representation of this pattern can be seen in Figure 1.9.

Figure 1.9: A visual representation of the stimulus-sensor-observation pattern.



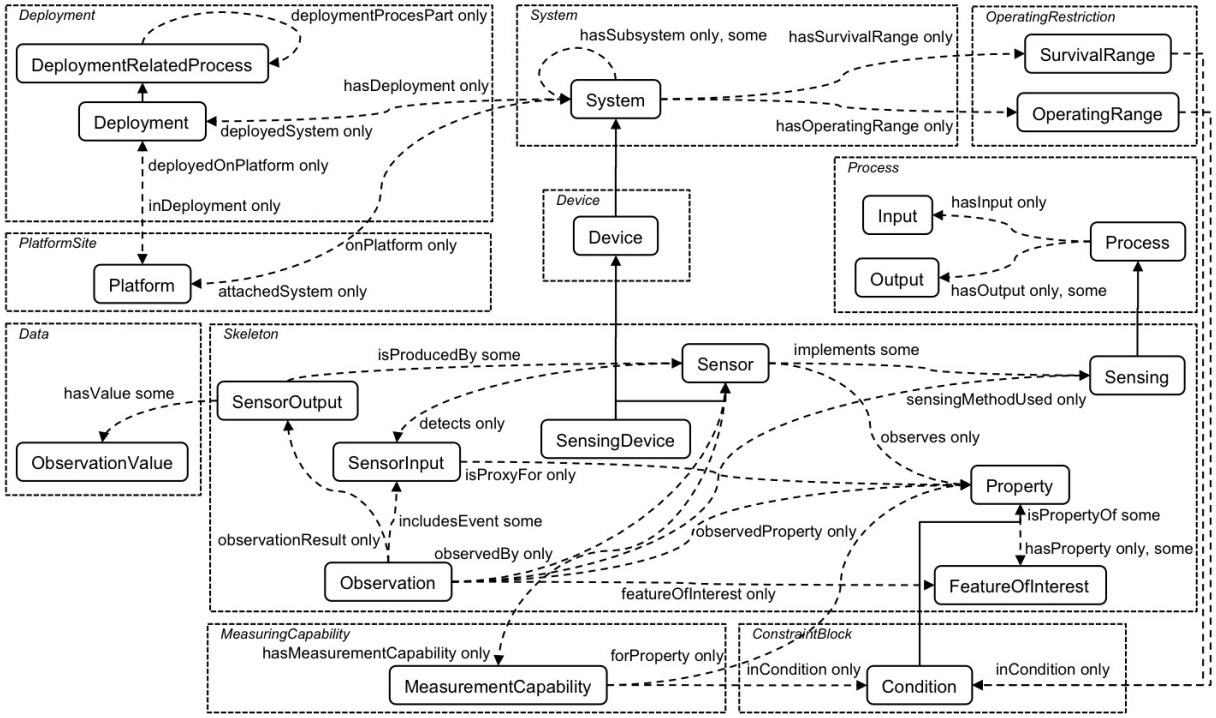
Source: (LEFORT et al., 2011).

Another advantage of SSN is that it is aligned with DOLCE, which facilitates reuse and interoperability (LEFORT et al., 2011). For example, a sensor in SSN is a physical *substantial*. A sensor detects a *Property*, which is a DOLCE *Quality*. For SSN, sensors are not exclusively electronic devices. For example, a room with people performing calculations to estimate some object's *Quality* is also a sensor. The concept of device (a system in a box) is also modeled in SSN. Those devices that are also sensors are instances of the *SensingDevice* concept.

The SSN ontology also uses the D&S extension for DOLCE. Sensors provide *observation*s, which are *situation*s in the D&S sense. This can help us model sensor data, as each sensor can provide not only a raw reading value, but the situation it detected and its interpretation.

An overview of SSN's classes and properties can be seen in Figure 1.10.

Figure 1.10: Overview of the Semantic Sensor Network ontology classes and properties.



Source: (LEFORT et al., 2011)

2 EVIDENCE THEORY BASICS

2.1 Motivation

In many real world applications, activity recognition included, decisions must be made under uncertainty. It is usual to represent uncertainty using probability distributions, for example, when assessing the result of a balanced coins toss. In this case, while the outcome of the toss is uncertain, its probability is certainly 0.5 and such information can be used in the decision process. Difficulties arise when probability distributions are not fully known. For example, if one buys an unfair six sided die, which has as its only guarantee that it lands a 6 with probability 0.8, what should be the probability that it will land a 5? One option is to consider equiprobable all unfavored numbers and so assume a value of $0.2/5 = 0.04$ for each. While useful, this assumption may not lead to realistic choices for the probability estimates. This can affect decisions when using standard probabilistic decision theories such as Bayesian decision theory, which cannot model the fact that we do not know yet how the remaining 0.2 is distributed.

Evidence theory is a framework for reasoning under uncertainty where uncertainty is represented by distributing evidence to *sets* (instead of elements) of mutually exclusive hypothesis. Just as probabilistic decision theories such as Bayesian decision theory deal with subjective and frequentist paradigms, evidence theory also has various interpretations to such distributions. In this chapter, we will cover some of those interpretations, along with some models that work under them.

2.1.1 Basic Concepts

Analogous to the way probability distributions assign probabilities to events of some event space, evidence theory basic belief assignments distribute masses to *subsets* of a frame of discernment. While interpretations and sometimes even terminology differ, many basic concepts are common in evidence theory models. For this reason, those basic concepts are presented here.

A frame of discernment (*fod*) is a finite set of mutually exclusive elements

$$\Omega = \{H_0, H_1, ..., H_n\}$$

where each element H_i is a hypothesis that represents one of a number of mutually exclusive

states of affair in the real world. For example, for the result of a single coin toss, the *fod* is the set $\Omega_{toss} = \{heads, tails\}$.

A basic belief assignment (*bba*) m^{Ω} is a function that assigns a numeric value in the interval $[0, 1]$, called a *mass*, for each member of the power set of some *fod* Ω. Formally

$$m^{\Omega} : 2^{\Omega} \rightarrow [0, 1]$$

where

$$2^{\Omega} = \{A | A \subseteq \Omega\}$$

Most models also add the restriction that zero must be assigned to the empty set, which is also called the conflict set.

$$m^{\Omega}(\emptyset) = 0$$

Just like the probabilities of every possible event in a probability distribution, masses of a *bba* must sum up to one.

$$\sum_{A \subseteq \Omega} m^{\Omega}(A) = 1$$

For example, since

$$2^{\Omega_{toss}} = \{\{heads, tails\}, \{heads\}, \{tails\}, \emptyset\}$$

a *bba* for the coin toss of a coin with unknown distribution can be represented by a *bba* $m^{\Omega_{toss}}$

$$m^{\Omega_{toss}}(\{heads, tails\}) = 1.0$$

$$m^{\Omega_{toss}}(\{heads\}) = 0.0$$

$$m^{\Omega_{toss}}(\{tails\}) = 0.0$$

$$m^{\Omega_{toss}}(\emptyset) = 0.0$$

while the *bba* for a fair coin toss can be represented by $m^{\Omega_{toss}}$

$$m^{\Omega_{toss}}(\{heads, tails\}) = 0.0$$

$$m^{\Omega_{toss}}(\{heads\}) = 0.5$$

$$m^{\Omega_{toss}}(\{tails\}) = 0.5$$

$$m^{\Omega_{toss}}(\emptyset) = 0.0$$

It is usual in evidence theory works, when showing *bba*s, to leave out mass asignments with zero values. From this point forward, we will follow the same practice, representing, for example, the *bba* of a fair coin toss as

$$m^{\Omega_{toss}}(\{heads\}) = 0.5$$

$$m^{\Omega_{toss}}(\{tails\}) = 0.5$$

for brevity.

2.1.1.1 Belief and Plausability

Given a certain *bba* on some *fod* Ω , the belief function Bel represents the amount of justified belief a *bba* assigns in some set A, or, in other words, how much mass we know to be gathered in all subsets of A. Formally

$$\begin{gathered} Bel : 2^{\Omega} \rightarrow [0,1] \\ Bel(A) = \sum_{X \subseteq A, X \neq \emptyset} m^{\Omega}(X) \end{gathered} \tag{2.1}$$

Intuitively, we can think of $Bel(A)$ as a lower bound for the mass a *bba* has assigned in some set A. Coming back to the problem of the unfair six sided die, which we only know that it lands a 6 with probability 0.8, we can represent its *fod* as $\Omega_{die} = \{1, 2, 3, 4, 5, 6\}$ and our knowledge regarding a throw as the *bba* $m^{\Omega_{die}}$

$$m^{\Omega_{die}}(\{6\}) = 0.8$$

$$m^{\Omega_{die}}(\{1, 2, 3, 4, 5\}) = 0.2$$

In this example, $Bel(\{1, 5\}) = 0$ as it is possible that a 5 or 1 will never happen in a toss, while $Bel(\{1, 6\}) = 0.8$ as, even if a 1 is not possible, we know for sure that the mass of 6 is 0.8.

The Plausibility function, on the other hand, can be seen as the upper bound for the mass a *bba* has assigned in some set A. It represents the degree to which one fails to doubt A (YAGER, 1983). That is, it considers the best case scenario for A, which is equivalent to

answering the following question: “If all masses assigned to a set that shares some element(s) with A were to migrate to those elements of A, how much mass would A have?” Formally

$$\begin{aligned} Pl : 2^{\Omega} \rightarrow [0,1] \\ Pl(A) = \sum_{X \cap A \neq \emptyset} m^{\Omega}(X) \end{aligned} \tag{2.2}$$

Using the unfair die example *bba* we can see that $Pl(\{6\}) = 0.8$, as there is no uncertainty regarding its mass and $Pl(\{1\}) = 0.2$ since there is 0.2 of mass to be divided among $\{1, 2, 3, 4, 5\}$.

2.1.1.2 Discounting and the unreliable truth machine interpretation

Discounting is a common operation introduced in (SHAFER et al., 1976). It changes a *bba*, moving a percentage of each belief assignment on a *fod* Ω from all sets 2^{Ω} to the total uncertainty set Ω.

Formally, a discount of $d \in [0, 1]$ changes m^{Ω}, generating m^{Ω}_{d} so that:

$$m^{\Omega}_{d}(A) = \begin{cases} (1-d)m^{\Omega}(A) & A \neq \Omega \\ d + (1-d)m^{\Omega}(\Omega) & A = \Omega \end{cases} \tag{2.3}$$

An easy way to understand the role of discounting is by seeing its role in the unreliable truth machine interpretation (SHAFER, 1982). Suppose you possess a truth machine that, when asked, informs the user about the state of some aspect of the real world. While the machine works fine with probability p, the rest of the time it answers the question using some unreliable technique unrelated to the desired aspect of reality. In this case the message the machine gives is certain, but not its reliability.

Consider the output options $\Omega_{res} = \{a, b\}$. If a machine that informs us about the truth of Ω_{res} is perfectly reliable ($p = 1.0$), when it tells us the truth is a, it generates a *bba*

$$m^{\Omega_{res}}(\{a\}) = 1.0$$

If, on the other hand, the machine can operate in the unreliable mode ($p < 1.0$), telling that the truth is in a should be discounted $1 - p$, generating the *bba* $m^{\Omega_{res}}$

$$m^{\Omega_{res}}_{d}(\{a\}) = p$$

$$m_d^{\Omega_{res}}(\{a,b\}) = 1 - p$$

to account for the cases where the machine is operating in the unreliable mode.

2.1.1.3 The randomly coded message interpretations

The unreliable truth machine is not the only interpretation for the meaning of uncertainty in mass functions. In fact, the *randomly coded message* interpretation(SHAFER, 1981) can help us understand another source of uncertainty that can be encoded using evidence theory.

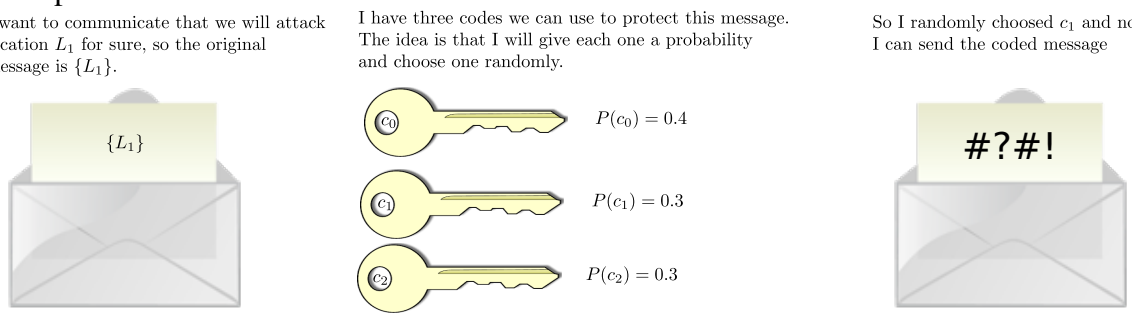
Suppose you know the enemy will attack one, and only one, city of a list of cities tomorrow. You intercept a coded message, regarding the name of the city the enemy will attack. The message is not readable, but you know it has been coded using one of n codes $c_1, \ldots, c_n$. Your spies are also certain that the choice of code was picked at random, with every code c_i having a known probability p_i of being picked. You are interested in knowing the original message, but, after decoding the message using every available code, you find that each of them gives a possible set of targets, such as "we will attack either location L_1 or L_2". The information gathered by reading those messages can be represented by a *bba* m^Ω

$$m^\Omega(A) = \sum_{loc(c_i)=A} p_i$$

where Ω is the set of all locations possible $\Omega = \{L_1, L_2, \ldots, L_n\}$ and $loc(c_i)$ is the set of locations mentioned as possible by decoding the coded message using c_i.

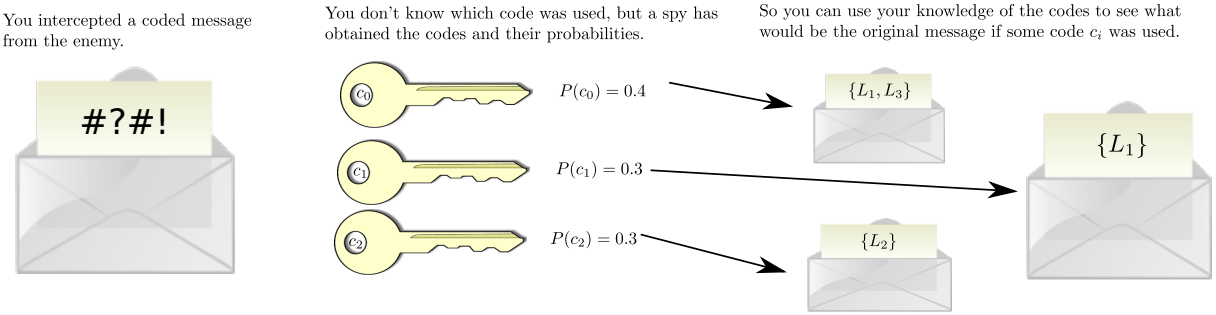
For example, if there are three possible locations for the attack, we have the *fod* $\Omega_{attack} = \{L_1, L_2, L_3\}$. If the enemy will attack location L_1, using the randomly coded message idea, he will encode a message following the procedure seen in Figure 2.1.

Figure 2.1: The procedure for producing a randomly coded message for the three location attack example.



Source: The author

Figure 2.2: The result when decoding the randomly coded message of the three location attack example.



Source: The author

When you intercept the randomly encoded message, you have the information shown in Figure 2.2. Of course, you don't know that the enemy chose code c_1, and every code yields information regarding the location of the attack. But, since one of the three codes was used, you can represent the information obtained by intercepting this message as a *bba* m^Ω in the form

$$m^\Omega(\{L_1\}) = 0.3$$

$$m^\Omega(\{L_2\}) = 0.3$$

$$m^\Omega(\{L_1, L_3\}) = 0.4$$

When deciding which location to protect based only on the decoding of the randomly coded message, one can analyze the belief and plausibility of each hypothesis.

$$Bel(\{L_1\}) = 0.3 \quad Pl(\{L_1\}) = 0.3 + 0.4 = 0.7$$

$$Bel(\{L_2\}) = 0.3 \quad Pl(\{L_2\}) = 0.3$$

$$Bel(\{L_3\}) = 0.0 \quad Pl(\{L_3\}) = 0.4$$

and decide to bet on L_1, as it has as much belief as the others and the best plausibility.

In this interpretation, the message content in itself is reliable, but ascertaining its true value is subject to some uncertainty. This example also helps us understand why only zero is allowed as mass value for the empty set for some evidence theories. Let's consider a code c_w that, when used to decode some randomly coded message, gives us some unreadable message. This code can be thought as providing mass to the empty set, as an unreadable decoded message does not specify any location. But, in this case, since we know that the message contains information regarding which of the hypotheses is true, this code could not have been used to

encode the message in the first place. Thus, one should normalize on the probability of c_w, so that the probability of all other codes sums up to one, leaving zero at the empty set.

2.1.1.4 The Transferable Belief Model and other interpretations

The original Dempster-Shafer (DS) theory always deals with probability values. That is, they work with some mapping with a underlying probability distribution (SMETS; KENNES, 1994) related to the unreliable truth machine or randomly coded message interpretation. This probabilistic interpretations are not the case for all evidence theory models. In the Transferable Belief Model (TBM), belief masses may represent the subjective beliefs of rational agents (SMETS; KENNES, 1994). This does not mean the probabilistic examples given before are not valid under TBM, since a rational agent may often choose a probability of a propositional as its belief when it is available (SMETS; KENNES, 1994).

TBM works in the open world assumption (SMETS, 2007), which means that it is possible that not all relevant hypothesis are present in the *fod*. In other words, TBM allows the assignment of non-zero masses to $m^{\Omega}(\emptyset)$. For example, regarding the previous tale of the randomly coded messages, a positive mass in the empty set indicates a belief that none of the listed codes was used to encode the message. Another major difference between standard DS and TBM lie on the way they combine evidences (SMETS; KENNES, 1994), as we will see in Section 2.1.2.

There are other interpretations and models for evidence theory including random sets (NGUYEN, 1978), hints (KOHLAS; MONNEY, 2008), probability of provability (PEARL, 2014), among others. In this work, we will explain and apply evidence theory using DS and TBM, so only those interpretations will be discussed.

2.1.2 Combinig Evidence Sources

Combining evidences of different sources is a vital part of any decision theory. In the Bayesian decision theory, for example, new information updates world knowledge using conditioning in the form:

$$P(A|B) = \frac{P(A,B)}{P(B)} = \frac{P(A)P(B|A)}{P(A)P(B|A) + P(\neg A)P(B|\neg A)}$$

When dealing with conditioning on events B_1, B_2, this becomes

$$P(A|B_1,B_2) = \frac{P(A)P(B_1|A)P(B_2|A,B_1)}{P(A)P(B_1|A)P(B_2|A,B_1) + P(\neg A)P(B_1|\neg A)P(B_2|\neg A,B_1)}$$

Note that, when B_1 and B_2 are independent, this formula assumes the simpler form

$$P(A|B_1,B_2) = \frac{P(A)P(B_1|A)P(B_2|A)}{P(A)P(B_1|A)P(B_2|A) + P(\neg A)P(B_1|\neg A)P(B_2|\neg A)}$$

but independence is not *obligatory* for combining new evidence using bayes rule.

This aspect differs from the one used in evidential theories such as DS and TBM. In order to combine two *bba*s, they *must* be doxastically independent(SMETS; KENNES, 1994).Saying that two *bba*s are doxastically independent means that their knowledge does not interact, meaning that knowledge of one does not concern the other. This is different, but related to, stochastic independence, which needs $P(A,B) = P(A)P(B)$ for A and B to be independent events. For example, in the case of the randomly coded message, if we were to receive a second encoded message, the evidences would only be doxastically independent if the probability of choosing c_i in the first message is stochastically independent from choosing it again in the current one(SHAFER, 1981).

A major difference between combination as done by the Bayesian theory is that, in DS and TBM, evidence combination is done using a specific rule, and not by conditioning, which plays other roles in the theories (SMETS; KENNES, 1994).

In TBM, given two independent basic belief assignments, m_1^Ω and m_2^Ω, on the same *fod* Ω, their combined *bba* $m_{1\textcircled{\Upsilon}2}^\Omega$ by the TBM combination rule is the result of the operator $\textcircled{\Upsilon}$ so that

$$m_1^\Omega \,\textcircled{\Upsilon}\, m_2^\Omega = m_{1\textcircled{\Upsilon}2}^\Omega$$

where

$$m_{1\textcircled{\Upsilon}2}^\Omega(A) = \sum_{X\cap Y = A} m_1^\Omega(X) m_2^\Omega(Y) \tag{2.4}$$

An easy way to visualize how the masses of m_1^Ω and m_2^Ω become masses in m_{12}^Ω is the diagram shown in Figure 2.3. In it, we can see that masses in one *bba* are distributed in m_{12}^Ω according to their intersection with masses in the other. It is also easy to see that masses in m_{12}^Ω sum up to one, as they are made from all the fractions of the unit square. For example, consider two independent unreliable truth machines that answer questions regarding the state of the *fod* $\Omega = \{a, b\}$. If each one is reliable with a probability of 0.2, and both tell us the truth is a, we

have m_1^Ω and m_2^Ω equal to

$$m_1^\Omega(\{a\}) = 0.2 \quad m_1^\Omega(\{a,b\}) = 0.8$$

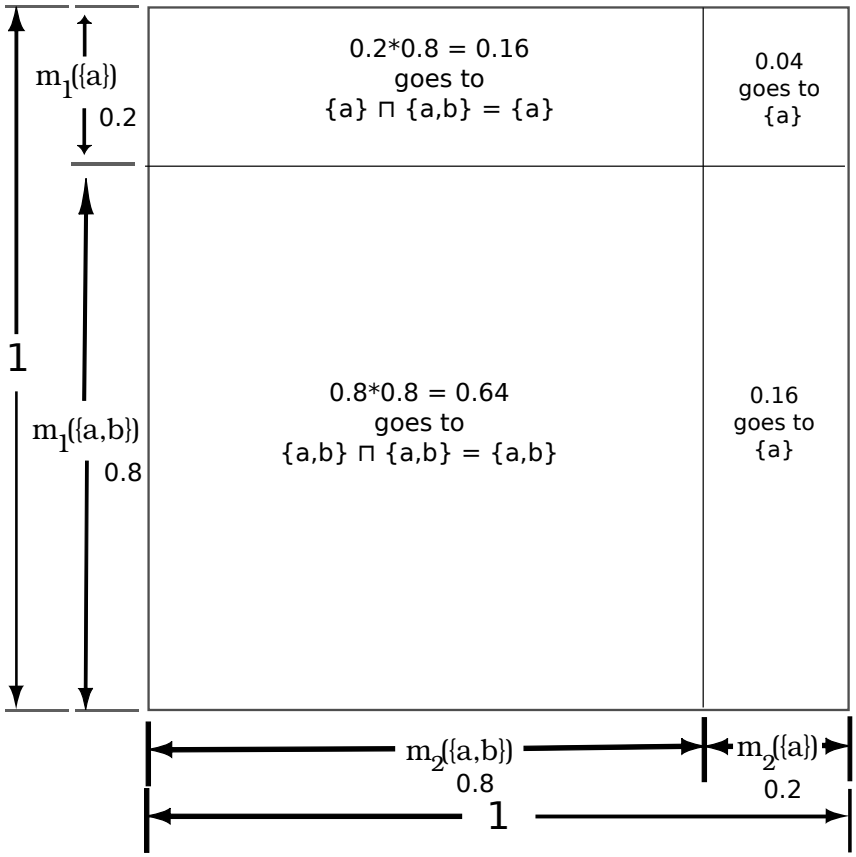
$$m_2^\Omega(\{a\}) = 0.2 \quad m_2^\Omega(\{a,b\}) = 0.8$$

Their combination is performed, as shown in Figure 2.3, resulting in the combined *bba* m_{12}^Ω

$$m_{1\textcircled{Y}2}^\Omega(\{a\}) = 0.16 + 0.16 + 0.04 = 0.36$$

$$m_{1\textcircled{Y}2}^\Omega(\{a,b\}) = 0.64$$

Figure 2.3: The unit square is a good visualization for where masses move when combining *bba*s. This one shows the combination of *bba*s from two independent unreliable truth machines that answer questions regarding the state of the *fod* $\Omega = \{a, b\}$, when each one is reliable with a probability of 0.2, and both tell us the truth is a.



Source: The author

If one machine had told us a while the other said b, some of the set intersections would result in the conflict set $\emptyset$. In TBM, there is no restriction on assigning non-zero mass values to the conflict set. The value of $m_{1\textcircled{Y}2}(\emptyset)$ simply represents the confict between the *bba*s. In

DS theory, on the other hand, the mass of the conflict set must be zero, so we must normalize TBMs result on the conflict mass.

Formally, the combined mass of m_1^Ω and m_2^Ω by the Dempster-Shafer combination rule is the result of an operator $\textcircled{\Psi}$ so that

$$m_1^\Omega \textcircled{\Psi} m_2^\Omega = m_{1\textcircled{\Psi}2}^\Omega$$

where

$$m_{1\textcircled{\Psi}2}^\Omega(A) = \begin{cases} \frac{m_{1\textcircled{\Upsilon}2}^\Omega(A)}{1-m_{1\textcircled{\Upsilon}2}^\Omega(\emptyset)} & \forall A \subseteq \Omega, A \neq \emptyset \\ 0 & \text{if } A = \emptyset \end{cases} \tag{2.5}$$

Both TBM and DS rule are associative and commutative(SMETS, 2007). In fact, in (SMETS, 2007) it is argued that those are the only rules with theoretically well defined justifications for the combination of independent *bba*s.

Many other combination rules, without such guarantees, exist for evidence theory. In special, Murphys combination rule (MURPHY, 2000), where two *bba*s are averaged before being combined with the DS combination rule. Formally, the resulting *bba* obtained by applying Murphys is given by applying an operator $\textcircled{\varpi}$ so that

$$m_1^\Omega \textcircled{\varpi} m_2^\Omega = m_a^\Omega \textcircled{\Psi} m_a^\Omega \tag{2.6}$$

where m_a^Ω is the averaged mass of m_1^Ω and m_2^Ω

$$m_a^\Omega(A) = \frac{m_1^\Omega(A) + m_2^\Omega(A)}{2}$$

In this case, the motivation is that no one evidence source can "dominate" all others, since we are always combining averaged masses. This makes sense when thinking of combination as a measure of agreement between the beliefs of different evidences, which is not a valid interpretation for combination in DS or TBM.

2.1.3 Mapping evidence through frames of discernment

Combination rules assume that evidences are encoded in the same frame of discernment. Evidences, on the other hand, can come from different sources as, for example, when two distinct context informations such as location and time are used to detect some activity.

In order to combine evidences on two context *fod*s Ω_{ctx1} and Ω_{ctx2} that are related to Ω_{act}, all *bba*s must be at the same *fod*. A possible solution is to assign masses on their Cartesian product $\Omega_{ctx1} \times \Omega_{ctx2} \times \Omega_{act}$ instead. This is fine in theory, but, in practice, due to the size of the aggregated *fod* and complexity of assigning values for its masses, mapping strategies were developed to translate mass from one *fod* to the other directly.

The basic idea of such mappings is to define how evidence on some *fod* Ω_{ctx} becomes evidence on some other *fod* Ω_{act} by making explicit only the relationships present in $\Omega_{ctx} \times \Omega_{act}$. The idea is to represent this relationship using some mapping strategy able to derive a *bba* in Ω_{act}, which can be combined with other *bba*s on Ω_{act} that were obtained in the same manner.

The existing mapping strategies are divided between *Multivaluated* and *Evidential*. A *Multivaluated* mapping $\Gamma : 2^{\Omega_1} \rightarrow 2^{\Omega_2}$ is a function that maps evidence from a *fod* Ω_1 to another *fod* Ω_2. Γ is used in the process of translating a *bba* in Ω_1 to another *bba* $m^{\Omega_1 \rightarrow \Omega_2}$ in Ω_2 in the following manner:

$$m^{\Omega_1 \rightarrow \Omega_2}(A) = \sum_{\Gamma(B)=A} m^{\Omega_1}(B)$$

Notice that every mass of 2^{Ω_1} goes completely to a single element of 2^{Ω_2}. If a single mass in 2^{Ω_1} needs to be split into masses for more than one element of 2^{Ω_2}, one can use an *Evidential* mapping Γ^*

$$\Gamma^* : 2^{\Omega_1} \times 2^{\Omega_2} \rightarrow [0, 1]$$

where $\forall A \in 2^{\Omega_1}$

$$\sum_{B \in 2^{\Omega_2}} \Gamma^*(A \times B) = 1$$

and generate a *bba* in Ω_2 using

$$m^{\Omega_1 \rightarrow \Omega_2}(B) = \sum_{A \in 2^{\Omega_1}} m^{\Omega_1}(A) \Gamma^*(A \times B)$$

Notice that Γ^* defines where each fraction of the mass goes, allowing modeling the uncertainty of the mapping itself.

2.1.4 Making decisions

Once all available information regarding some aspect of reality is gathered and combined, one must, by some method, decide which of the hypothesis, or set of hypothesis, of the *fod* in question is most likely the truth. In evidence theory models this decision is not as

straightforward as choosing the highest probability option.

One method for weighting the likelihood of a set of hypotheses is the Pignistic Probability, which is the decision method of choice for TBM (SMETS, 2007). It is a middle ground between Plausibility and Belief where masses are normalized on the empty set and then weighted by how close they are to containing the H being evaluated. Formally, the Pignistic Probability of a set of hypothesis A in some *bba* m^{Ω} is

$$BetP : 2^{\Omega} \rightarrow [0, 1]$$

$$BetP(A) = \begin{cases} \sum_{X \subseteq \Omega} \frac{|X \cap A|}{|X|} \frac{m^{\Omega}(A)}{1 - m^{\Omega}(\emptyset)} & \forall A \subseteq \Omega, A \neq \emptyset \\ 0 & \text{if } A = \emptyset \end{cases} \tag{2.7}$$

This value represents the rate a rational agent should use when forced to make bets on some aspect of reality (SMETS; KENNES, 1994). For example, in the case of an unknown coin, we have $\Omega_{toss} = \{heads, tails\}$ and $m^{\Omega_{toss}}(\{heads, tails\}) = 1.0$. So, if we are forced to choose betting rates for $heads$ and $tails$, we can use their Pignistic probability

$$BetP(\{heads\}) = \frac{|\{heads, tails\} \cap \{heads\}|}{|\{heads, tails\}|} m^{\Omega_{toss}}(\{heads, tails\}) = \frac{1}{2}$$

$$BetP(\{tails\}) = \frac{|\{heads, tails\} \cap \{tails\}|}{|\{heads, tails\}|} m^{\Omega_{toss}}(\{heads, tails\}) = \frac{1}{2}$$

and bet as much money in $heads$ as in $tails$.

In short, the likelihood score for each hypothesis set H, can be evaluated

- Using the mass of H, which represents the support for H alone.
- By the Belief of H, which can be thought of as a lower bound on the support of H.
- By the Plausibility of H, which can be thought of as an upper bound on the support of H.
- By the Pignistic Probability of H, which can be thought as the betting rate one should assign to H.

3 EVIDENTIAL THEORY METHODS FOR ACTIVITY RECOGNITION

Some evidential theory methods for recognizing ADLs were developed over the years (HONG et al., 2009) (MCKEEVER et al., 2010) (SEBBAK et al., 2014). The work in this field has been mostly incremental, starting with (HONG et al., 2009), obtaining temporal aspects in (MCKEEVER et al., 2010) and receiving other changes in (SEBBAK et al., 2014). In this section we will discuss their use of the evidential theory concepts for activity recognition.

3.0.1 The Frames of Discernment

Every method uses *fod*s for sensors, contexts and activities. The basic idea is that sensors, such as a fridge contact sensor, provide information regarding some context, such as fridge door status, which is related to some activity such as preparing a drink. In other words, every sensor is represented by a *fod* $\Omega_s = \{s_{v1}, \ldots, s_{vn}\}$ where $s_{v1}, \ldots, s_{vn}$ represent all values this sensor can provide. For example, for a fridge door's contact sensor,

$$\Omega_{fridgeSensor} = \{sensor_{On}, sensor_{Off}\}$$

By the same logic, the fridge's door context is

$$\Omega_{fridgeDoorContext} = \{fridge_{Opened}, fridge_{Closed}\}$$

and the activity *fod* is

$$\Omega_{getDrink} = \{getDrink, \neg getDrink\}$$

Notice that this is not the only choice for representing the activity *fod*. In fact, one could use a single *fod* for all activities $\Omega_{act} = \{a_1, \ldots, a_n\}$ where $a_1, \ldots, a_n$ represent the possible activities for the scenario. For example, if there are only the eat and sleep activities, the only *fod* needed for dealing with them is $\Omega_{act} = \{sleep, eat\}$. This option would reduce the number of *fod*s needed by the method, but it increases the number of hypothesis in the activity *fod*. Since the objective of the models is to recognize a single activity, out of disjoint options, a single *fod* seems to be the appropriate, on account of it represents automatically this restriction. Although this choice is never explicitly discussed in the cited works, we assume that activities are not represented by a single *fod* for the following reasons:

- If each activity has a separated *fod*, one could envision a modular system, where old

activity structures can stay the same as a new activity is added to the system.

- It facilitates migration to a scenario with concurrent activities. With multiple *fod*s, the model can be easily extended for multiple activities happening at once if an activity is set as happening if its support reaches a certain threshold (MCKEEVER, 2011).

3.0.2 Sensor discounting

Every time a sensor is activated, providing one of its possible values, this information is discounted by that sensor's accuracy. This means that, if a sensor is accurate with probability p, the existing evidence theory models for activity recognition assume that it works as an unreliable truth machine that gives the correct feature value with probability p. So, for example, if we have a binary sensor $\Omega_s = \{s_{on}, s_{off}\}$ with accuracy 0.8, when it provides an "on" value, the *bba* associated to this firing would be m^{Ω_s}

$$m^{\Omega_s}(\{s_{on}\}) = 0.8$$

$$m^{\Omega_s}(\{s_{on}, s_{off}\}) = 0.2$$

Thinking about accuracy in this binary case, we can see the possible matchings between the sensor and the feature it detects in Table 3.1. Accuracy is defined as $\frac{TP+TN}{FP+FN+TP+TN}$, and can be thought as the quality of the sensor information, but it does not represent a direct mapping to unreliable truth machine reliability as defined in evidence theory. If we consider that a sensor really behaves as an unreliable truth machine, values from the TP and TN may be coming from unreliable operation. In fact, it is possible that the machine is *always* operating in its unreliable mode, and the accuracy value observed is meaningless. On the other hand, all the FP and FN firings could only have come from the unreliable mode, and so its probability should at least support those cases. Since the probability of unreliable behavior should range from $\frac{FP+FN}{FP+FN+TP+TN}$ up to one, it is clear that the evidence models discussed chose discounting values assuming the best case scenario regarding sensor reliability.

Table 3.1: Possible outcomes for the reading of a binary sensor that detects some feature F.

	F **is present**	F **is not present**
Sensor gives F	True Positive (TP)	False Positive (FP)
Sensor gives $\neg F$	False Negative (FN)	True Negative (TN)

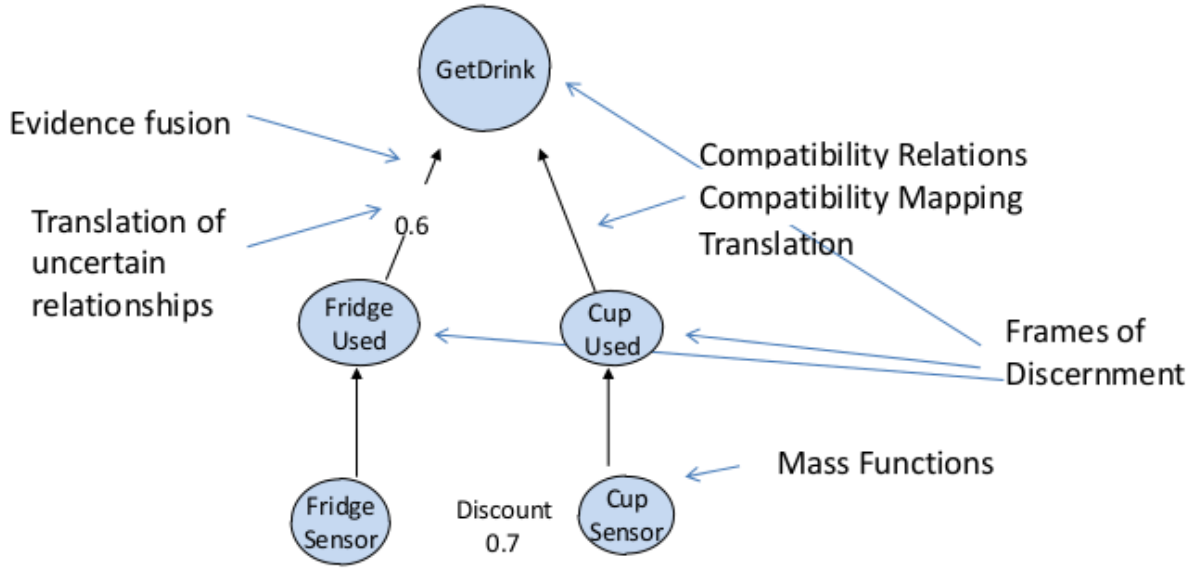
Source: The author

Table 3.2: Comparison network type for the evaluated evidence theory metods for activity recognition

Method	Network Type
(HONG et al., 2009)	Evidential network
(TOLSTIKOV et al., 2011)	Evidential network
(MCKEEVER et al., 2010)	DAG
(MCKEEVER, 2011)	DAG
(SEBBAK et al., 2014)	DAG

Source: The author

Figure 3.1: Example of a DAG for activity recognition.



Source: (MCKEEVER, 2011)

3.0.3 From sensor/context to activity

Once we have *bba*s on sensor information, those must be mapped to evidences in context and later activities. The current evidential theory techniques employ graphical models to show this relationships. In (HONG et al., 2009), evidential networks such as the one in Figure 3.2 map sensors to objects and objects to activities. In (MCKEEVER et al., 2010) and (SEBBAK et al., 2014), a Directed Acyclic Graph (DAG) is used for the same objective, as is shown in Figure 3.1. Table 3.2 shows the network types for all of the evidence theory activity recognition methods.

Table 3.3: Sensor *bba*s are converted to context *bba*s using associations as defined by an multivaluated mapping.

Original in Ω_s	**Mapped by Γ to Ω_{ctx} as**
$\{s_{on}\}$	$\{ctx_\top\}$
$\{s_{off}\}$	$\{ctx_\bot\}$
$\{s_{on}, s_{off}\}$	$\{ctx_\top, ctx_\bot\}$

Source: The author

In both Evidential networks and DAGs, connections represent mappings between *fod*s in the form of evidential or multivaluated mappings. While, in theory, this mappings can represent both positive and negative relationships between sensors and their relative activities, in practice, sensors are connected through context nodes to their related activities. For example, a toilet sensor will not be linked to a meal activity, because this mapping is a negative one and the toilet, as an object, is not thought as being "indicating" a meal activity.

With this in mind, and knowing the evaluated methods favor binary sensors, we can flesh out the general evidential and multivaluated mappings created by their networks. Given some binary sensor s, with which reports on some binary context ctx positively related to some activity a, we have the *fod*s

$$\Omega_s = \{s_{on}, s_{off}\}$$

$$\Omega_{ctx} = \{ctx_\top, ctx_\bot\}$$

$$\Omega_a = \{a, \neg a\}$$

for all the nodes involved. In (HONG et al., 2009) and (MCKEEVER et al., 2010), the discounted sensors *bba*s are translated to context values using a multivaluated mapping which can be seen at Table 3.3.

After this procedure is done, context values are translated to activities using either evidential or multivaluated mappings. Since multivaluated mappings can be thought as a specific version of evidential mapping (where every mapping is exact), we can say without any loss of generality that the context ctx will be mapped to a using the evidential mapping which can be seen at Table 3.4.

The value v_1 is the confidence s gives to activity a. This value could arguably be determined by some expert, but, as is the general case due to the infeasibility of such approach, this value is calculated by estimating $P(s|a)$ on the training data. Either way, this type of mapping shows the general interpretation of evidence theory those activity recognition models use, which is, that of an unreliable truth machine that reports on the current activity. We can see

Table 3.4: Sensor *bba*s are converted to context *bba*s using associations as defined by an evidential mapping.

From / To	$\{a\}$	$\{\neg a\}$	$\{a, \neg a\}$
$\{ctx_\top\}$	v_1	0	$1.0 - v_1$
$\{ctx_\bot\}$	0	v_1	$1.0 - v_1$
$\{ctx_\top, ctx_\bot\}$	0	0	1.0

Source: The author

Table 3.5: For (SEBBAK et al., 2014), sensor *bba*s are converted to context *bba*s using associations as defined by this multivaluated mapping.

Original in Ω_s	**Mapped by Γ to Ω_{ctx} as**
$\{s_{on}\}$	$\{ctx_\top\}$
$\{s_{off}\}$	$\{ctx_\top, ctx_\bot\}$
$\{s_{on}, s_{off}\}$	$\{ctx_\top, ctx_\bot\}$

Source: The author

this as we follow the path of a sensor event to the activity evidence it generates in the model. Because every hypothesis indicates only some specific state for the activity *fod*, the sensor is either operating in an unreliable mode, or the activity is really happening. So, for example, if a sensor related to *eat dinner* is set as on, the *bba* generated on the *eat dinner fod* will not have any mass on $\neg dinner$. Analogously, if this same sensor is off, it generates a *bba* that has no mass in $dinner$.

3.0.3.1 Changes introduced in the mapping strategy

In (SEBBAK et al., 2014), mappings were changed in two ways, revealing a slightly different interpretation of evidence theory.

First, sensors that are not firing now map to total uncertainty in their respective context. This means that sensor to context mappings follow the form of which can be seen in Table 3.5.

Thus, this changed mapping means that now there is no contextual information to be gathered from inactive sensors. To understand the effect of this change, lets consider a *prepare dinner* activity that has as its related sensors a cooktop sensor and a plate sensor. If there is a cooktop activation, but not yet a plate one, the lack of plate activation will be translated to negative evidence. Even worse, at a later stage, as the cooktop is turned off and the plate is used, now the cooktop sensor is sending negative evidence. If an inactive sensor maps to a totally uncertain context, both of those problems disappear, but, on the other hand, inactive sensors missing from some activity will not reduce the activity's support and thus it may be difficult to differentiate similar activities. For example, the *get drink* and *prepare dinner* activities may share the fridge

Table 3.6: Comparison of combination rules used for the evaluated evidence theory metods for activity recognition

Method	**Sensor favoring activity is inactive**
(HONG et al., 2009)	Means evidence of $\{\neg a\}$
(TOLSTIKOV et al., 2011)	Means evidence of $\{\neg a\}$
(MCKEEVER et al., 2010)	Means evidence of $\{\neg a\}$
(MCKEEVER, 2011)	Means evidence of $\{\neg a\}$
(SEBBAK et al., 2014)	Means evidence of $\{a, \neg a\}$

Source: The author

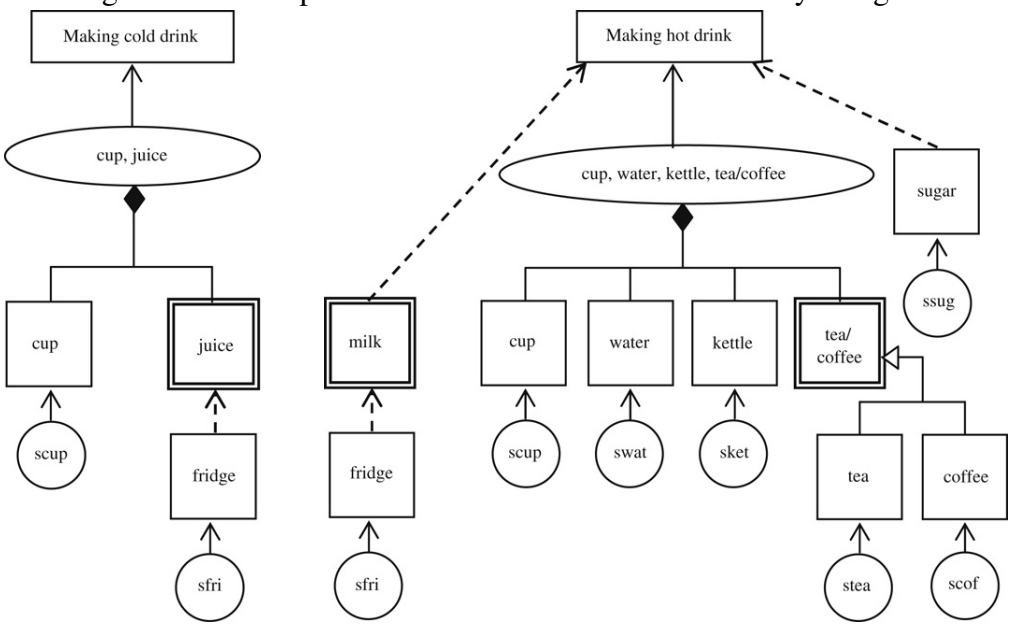
and cup sensors, but not the cooktop sensor. When only fridge and cup sensors fire, *prepare dinner* may be preferred over *get drink*, since an inactive cooktop sensor will bring no context information.

Second, to better differentiate activities that share sensors, some negative mass mappings are allowed. Specifically, if two activities A and B share sensors, but B has more sensors than A, those extra sensors are mapped to $\neg A$ when they fire. So, if we are deciding between *get drink* and *prepare dinner* from the previous example, a cooktop sensor is not just evidence for dinner, but negative evidence for drink.

Those differences in the mappings don't change much the interpretation of evidence theory used by the model. Every sensor still reports either positive or negative evidence for the activity, with inconsistencies still being attributed to operating in the unreliable mode. In other words, if we think about sensor firings as randomly coded messages, there will only be two codes available, one where the message provides no information (the unreliable option) or one where it decides surely the presence of the activity.

Table 3.6 shows the interpretations of an inactive sensor for all of the evidence theory activity recognition methods.

Figure 3.2: Example of an Evidential network for activity recognition.



Node	Context	Link	Relation
sensor	Sensor	A → B	Sensor A is associated with object B
object	Object (associated with a sensor)	A ⇢ B	Object A derives object B
object	Object (derived from other object)	A, B ◆— C	A and B are compulsory to C; A, B and C can be objects or activities
composite object	Object (a set of compulsor objects)	A, B —▷ C	A and B are alternative to C; A, B and C can be objects or activities
activity	Activity	A → B	A is compulsory to activity B; A can be an object, a compound object, or an activity
		A ⇢ B	A is optional to activity B; A can be an object, or an activity

Source: (HONG et al., 2009)

Table 3.7: Comparison of combination rules used for the evaluated evidence theory metods for activity recognition

Method	**Rules used**
(HONG et al., 2009)	$\textcircled{\Psi}$, $\oplus$, OR
(TOLSTIKOV et al., 2011)	$\textcircled{\Psi}$, $\oplus$, OR
(MCKEEVER et al., 2010)	$\textcircled{\varpi}$, OR
(MCKEEVER, 2011)	$\textcircled{\varpi}$,OR
(SEBBAK et al., 2014)	$\textcircled{\varpi}$ or $\textcircled{\Psi}$,OR

Source: The author

3.0.4 Other Rules Used

As we can see in Figure 3.2, arrows in an evidential network have other different meanings, which relate to the mappings. It is possible to represent nodes as *compulsory* or *alternative*.

For example, if two objects are alternatives to an upper node, as is the case of the tea and coffee nodes to the "tea / coffee" node in Figure 3.2, those two evidences will not be mapped and combined in the "tea/coffee" node. Instead, from the result of two individual mappings, the one with the highest belief will be chosen.

For combining compulsory nodes, such as cup and juice for cold drink in Figure 3.2, or combining dependent evidences, a different combination rule is used (HONG et al., 2009). It is named the Equally Weighted Sum (EWS), and it combines n *bba*s $m_1^\Omega, \ldots, m_n^\Omega$ generating a new *bba* $m_\oplus^\Omega$. This *bba* is generating by applying the $\oplus$ operator where

$$m_1^\Omega \oplus \ldots \oplus m_n^\Omega = m_\oplus^\Omega(A)$$

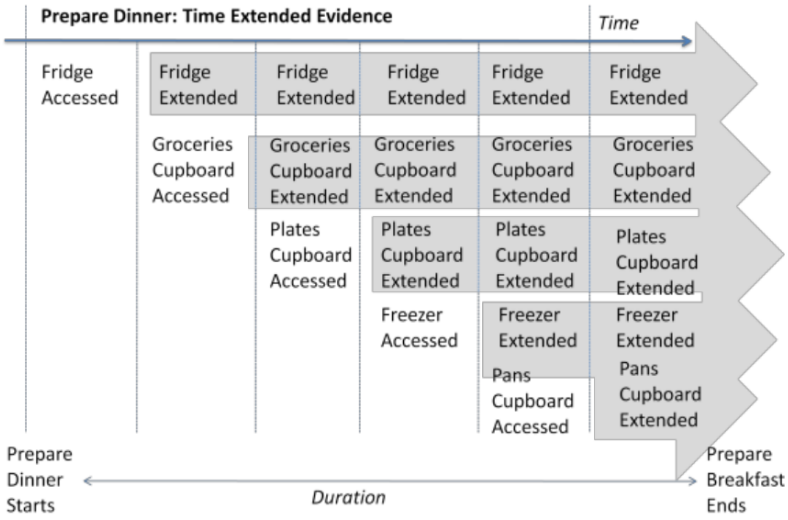
with

$$m_\oplus^\Omega(A) = \frac{1}{n}\sum_{i=1}^{n} m_i^\Omega(A) \tag{3.1}$$

This notion of compulsory evidence is not used in methods that employ DAGs, but alternative evidences, while not represented directly in the DAG, are still allowed. For example, the sleep activity is described as "bedroom door OR no sensor active at night", while the leave house activity is described as "Front door OR no sensor active during the day".

Table 3.7 shows all the rules used by all the evaluated evidence theory methods for activity recognition.

Figure 3.3: Visual representation for the extension of evidence.



Source: (MCKEEVER et al., 2010)

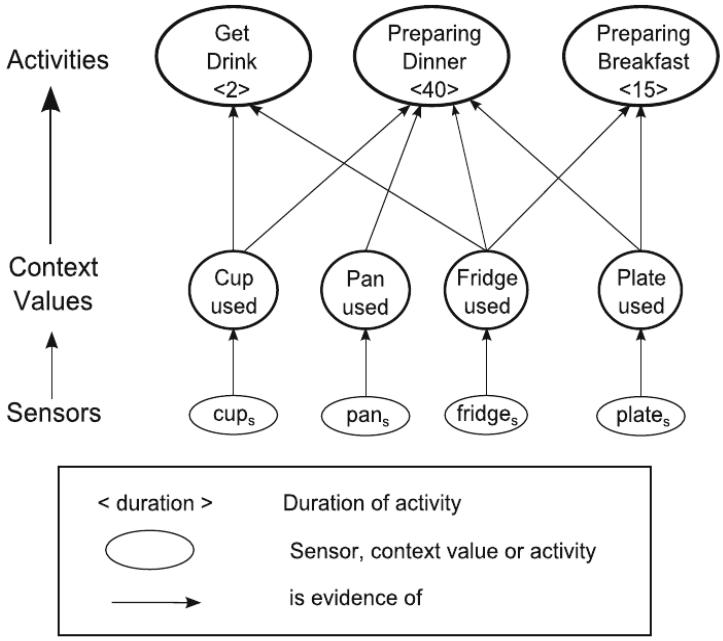
3.0.5 Temporal Aspects

In (HONG et al., 2009), only sensor information from the current minute was used for activity recognition. From (MCKEEVER et al., 2010) on, this was remedied by including temporal aspects to evidence theory. They are the extension of evidence and the time cut.

3.0.5.1 Extending Evidence

Some activities were marked in the network as being influenced by evidences from past minutes, and thus given a fixed window size len_{a_i}. The main idea is that, once an activity receives the first sensor evidence supporting its existence, all evidence regarding it will be kept ("extended") until len_{a_i} minutes have passed. Once the fixed time is over, the window is cleaned and all past evidences are discarded. Notice that every activity has its *own* window size, so there can be multiple windows active at any time. And, of course, those activities that were not set as having a window size use only the information available this minute. The window lenght for each activity can be seen as a number on top of activity nodes as can be seen in Figure 3.4. You can also see the extension of evidence graphically in Figure 3.3.

Figure 3.4: Example of a DAG as used in (SEBBAK et al., 2014) and (MCKEEVER et al., 2010) for activity recognition. Numbers in nodes represent their maximum window size in minutes.



Source: (SEBBAK et al., 2014)

3.0.5.2 Performing Time Cuts

A time cut is a general time restriction on some activity. For example, one cannot have breakfast in the afternoon, so this activity should not be even considered at this time of the day. This type of restriction is represented as the interval of the day (in absolute time) that the activity is possible. If an activity has an interval, it is discarded ("cut") or not even started whenever the time of the day does not match its interval.

All the evaluated evidence theory methods for activity recognition that use evidence extension and time cuts can be seen in Table 3.8.

Table 3.8: Comparison of available temporal techniques for the evaluated evidence theory metods for activity recognition.

Method	**Temporal Extension**	**Time Cut**
(HONG et al., 2009)	No	No
(TOLSTIKOV et al., 2011)	No	No
(MCKEEVER et al., 2010)	Yes	Yes
(MCKEEVER, 2011)	Yes	Yes
(SEBBAK et al., 2014)	Yes	Yes

Source: The author

Table 3.9: Comparison of testing details for existing evidence theory methods for activity recognition

Method	**Decision**	**Skips unlabeled minutes**	**Tested in ADL Dataset(s)**
(HONG et al., 2009)	Not tested	Not tested	None
(TOLSTIKOV et al., 2011)	Bel($\{a\}$)	No	Only (KASTEREN et al., 2008)
(MCKEEVER et al., 2010)	Bel($\{a\}$)	Yes	Only (KASTEREN et al., 2008)
(MCKEEVER, 2011)	BetP($\{a\}$)	Yes	Only (KASTEREN et al., 2008)
(SEBBAK et al., 2014)	Bel($\{a\}$)	Yes	Only (KASTEREN et al., 2008)

Source: The author

3.0.6 Decision Strategies

Once the sensor *bba*s have been translated to some activity *fod*, it is time to combine them using some rule. In (HONG et al., 2009), this combination rule is the Dempster-Shafer combination rule, while in (MCKEEVER et al., 2010), it is Murphys combination rule. In (SEBBAK et al., 2014), results are shown using both rules, since allegedly none of them is superior to the other (SEBBAK et al., 2014).

As we can see in Table 3.9, the different methods decide using different strategies. Each activity a, after having all evidence sources combined, has a *bba* m^{Ω_a} where $\Omega_a = \{a, \neg a\}$. In this situation, following Equation 2.1, we have

$$Bel(\{a\}) = m^{\Omega_a}(\{a\})$$

and so we are actually making decision based on a single mass value. It is interesting to notice that Mckeever, later on her thesis (MCKEEVER, 2011), changed the evaluation procedure to the Pignistic Probability as defined in Equation 2.7. Thus,

$$BetP(\{a\}) = m^{\Omega_a}(\{a\}) + \frac{m^{\Omega_a}(\{a, \neg a\})}{2}$$

Another thing worth noticing is that decision strategies may involve other factors. For example, in (MCKEEVER, 2011), ties are broken by minimum uncertainty and, when no activity is detected, a default activity is chosen based on the current time.

3.0.7 Evidential Theory Advantages and Drawbacks

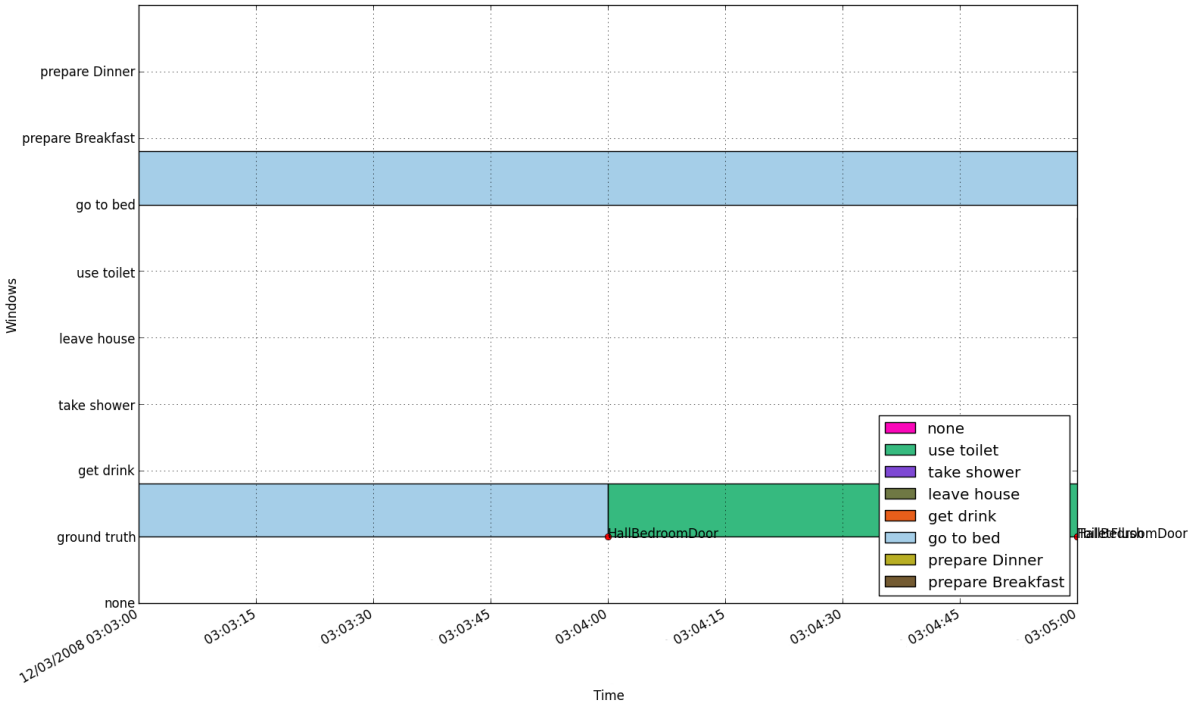
Evidential theory models are hybrid models, since the scenario description is knowledge-based, but weights for the model are learned in a data-driven fashion. While we cannot state that they work abstracting sensor configuration, they provide explicit information regarding their activity recognition decisions. They can also incorporate and quantify the uncertainty present in the decision process, bringing some of the data-driven advantages to standard logical approaches, without the need for any general purpose reasoning technique.

Activity recognition using evidence theory has shown that it can outperform classic data-driven approaches in scenarios with little training data (MCKEEVER et al., 2010) by modeling the degree of lack of knowledge and relying on temporal information to determine the relevant evidences for some activity at a given time.

But, as with any activity recognition method, the existing methods for activity recognition based on evidence theory have some drawbacks. They include:

- A DAG or evidential network needs to be created by some expert before the model can be deployed. This is true for all methods, as they do not yet provide any learning procedure for network structure. This means that those methods cannot be easily tested in arbitrary houses and thus may be hard to deploy in the real world.
- Most methods ignore the presence of unlabeled minutes in the datasets, as can be seen in Table 3.9. This makes them unsuitable for real life use, as they ignore the important problem of determining if there is in fact any real activity happening. Even more, detecting inactivity can be very important for healthcare scenarios, where such situations can represent life threatening situations as household accidents or sudden death by a heart attack.
- As can be seen in Table 3.9, they were only tested in a single dataset, (KASTEREN et al., 2008), which does not provide sensor accuracy information. For this reason, all methods of table 3.9 assume sensors are perfectly accurate, leaving uncertainty only in the mappings from context/object to activity.
- General problems related to the use of multiple fixed windows. When using multiple windows, some method must be chosen to determine window sizes. In (MCKEEVER, 2011) the recommended methods to determine activity duration are user interviews and estimating the duration of activities using their mean duration in the training data. Either way, regardless of the duration chosen, some problems may arise.

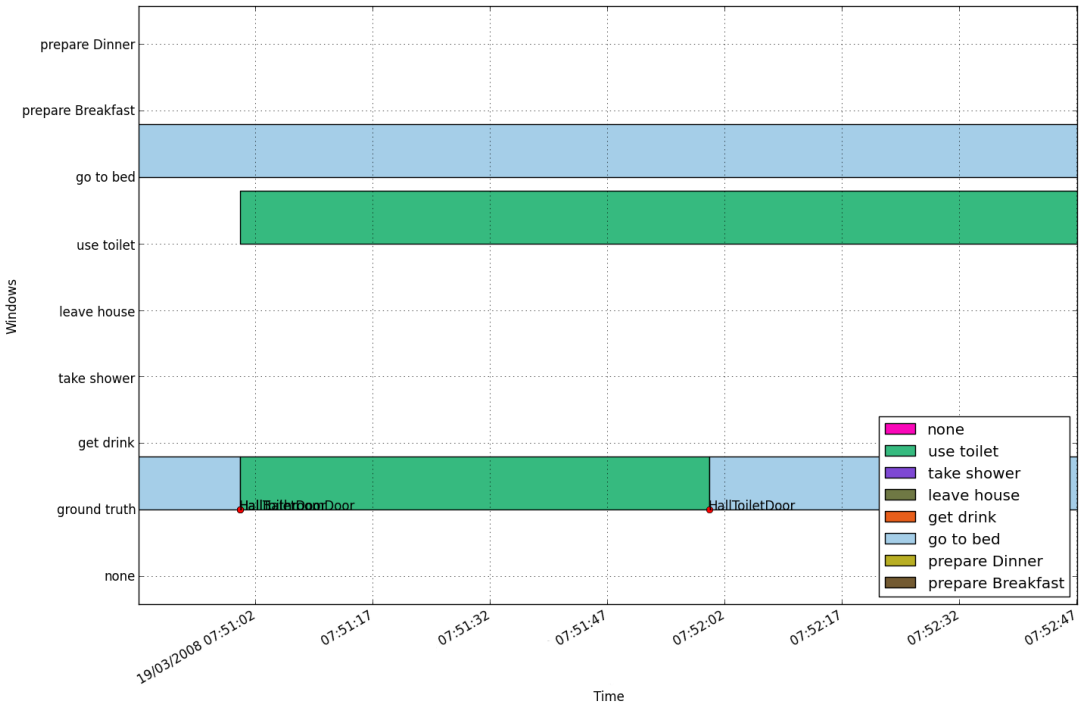
Figure 3.5: Activities may start before any firing that indicates it. In this case, using the toilet started as the subject opened the bedroom door, leaving for the toilet in the middle of the night. But, because this sensor is not related to the use toilet activity, its window will be started only later. Example generated from real life activity data from (KASTEREN et al., 2008).



Source: The author

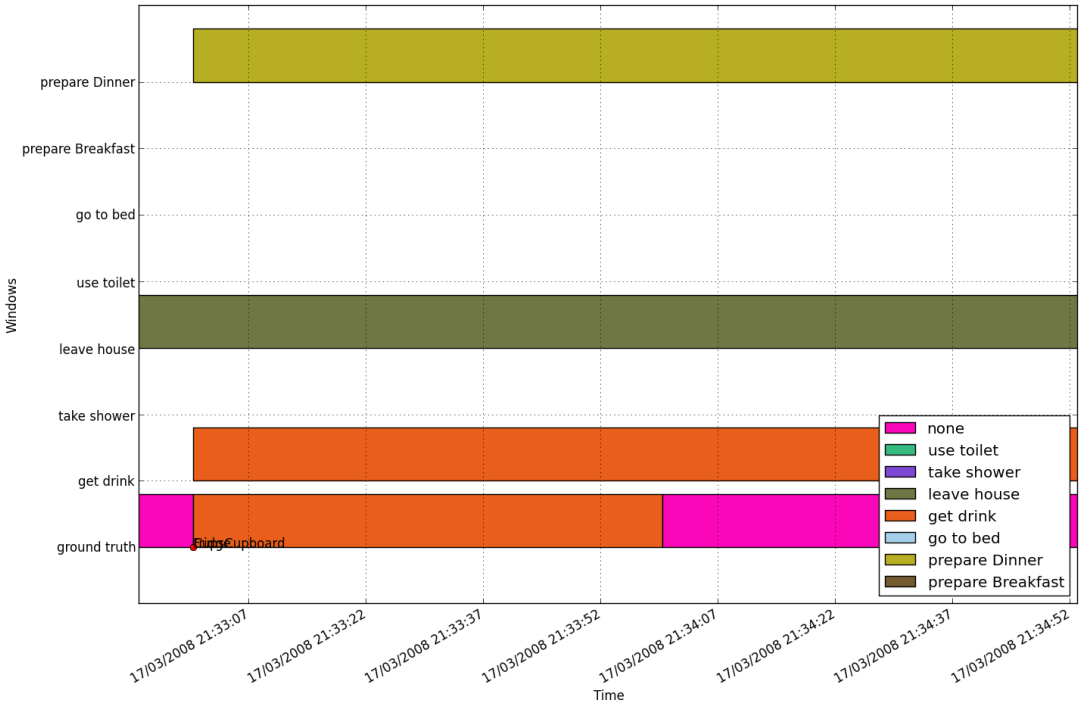
For example, a window for an activity a is started when some sensors generates positive evidence for a. Of course, a may start before a sensor indicates its presence, as we can see in Figure 3.5. Windows can also be too small or too big. An example of window overshoot can be seen in Figure 3.6. In this cases, as all old evidence is kept in the window, when an activity window is bigger than the activity it was correctly detecting, it can lead the model in the wrong direction. These problems can be alleviated by the fact that the evidential models skip minutes that are not annotated with activities. For example, since it is usual for an activity to be followed by some idle time, and those minutes are not evaluated by those methods, they can mask overshooting errors. This can be seen in Figure 3.7. Regarding idle minutes, which are discarded by some methods, it is also important to note that, while they may not be labeled with any activity, they may contain

Figure 3.6: Activities windows can be too big, continuing after the activity finishes. Example generated from real life activity data from (KASTEREN et al., 2008).



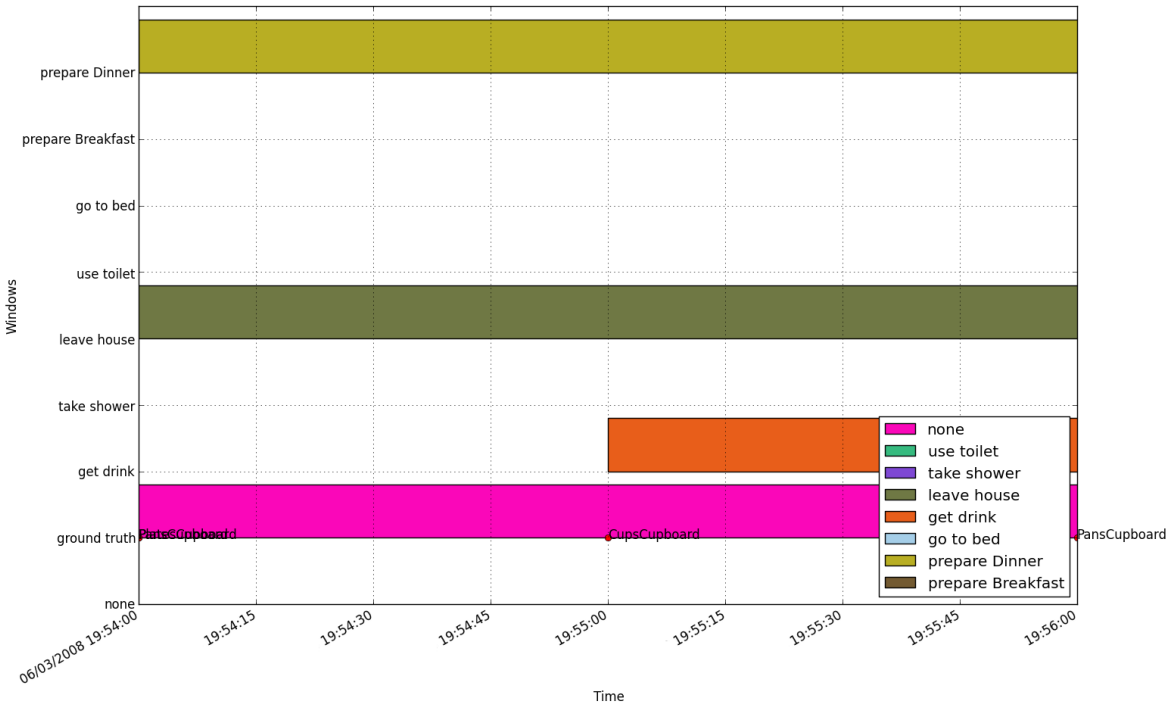
Source: The author

Figure 3.7: The get drink window matches its activity perfectly, as its excess ends up in the unlabeled minutes, which are not evaluated. Example generated from real life activity data from (KASTEREN et al., 2008).



Source: The author

Figure 3.8: In a real life scenario, activity windows will start at idle minutes. Example generated from real life activity data from (KASTEREN et al., 2008).



Source: The author

sensor firings. In a real world scenario for activity recognition, as there is no ground truth information to be queried regarding idle minutes, windows would simply start at those minutes as shown in Figure 3.8.

4 AN ONTOLOGY-DRIVEN EVIDENCE THEORY METHOD FOR ACTIVITY RECOGNITION

In this Chapter we present our proposed activity recognition technique that combines ontology-driven activity modelling with evidence theory.

Its main contributions to evidence theory activity recognition methods are:

- It provides an evidence theory model that represents better the process of activity recognition.
- It eliminates static networks, defining scenarios using an ontology that can be automatically adapted to different houses.
- It eliminates static mappings, substituting them for generic activity modelling and ontological reasoning.

4.1 On the correct intepretation for activity recognition

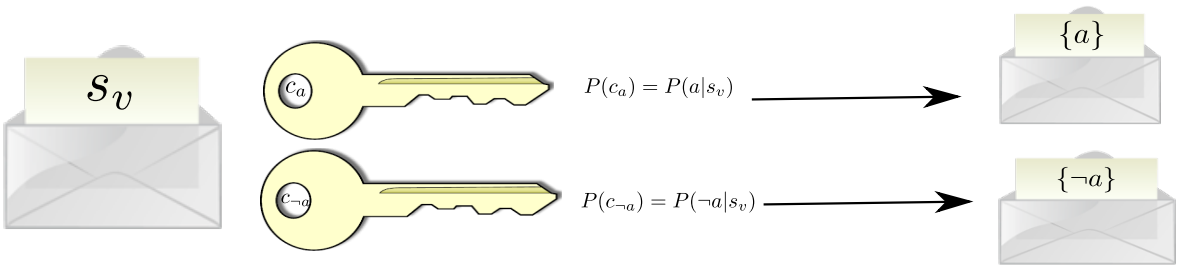
As we have discussed in Chapter 3, current evidence theory activity recognition methods work under the unreliable truth machine interpretation of evidence theory. They have contexts either translating to evidence for or against an activity evidence, with non-conformity meaning operation in the unreliable mode. This approach is unrealistic. Sensors should provide, at the same time, both positive and negative evidence for an activity, the same way an imperfect unfair coin is still expected to land on both sides.

We can arrive at the same conclusion if we think of sensors using the randomly coded message interpretation. In this case, we can view a specific sensor status s_v, provided by a sensor s, as a coded message regarding the state of some activity a denoted $\Omega_a = \{a, \neg a\}$. Therefore, we can think that the message s_v has been encoded with two possible codes c_a or $c_{\neg a}$, representing, respectively:

- The case where a is happening. This code decodes s_v as $\{a\}$ and, as we know s_v was received, has a probability of $P(a|s_v)$ of being chose.
- The case where a is not happening. This code decodes s_v as $\{\neg a\}$ and, since we know s_v was received, has a probability of $P(\neg a|s_v)$ of being chose.

This situation can be seen graphically in Figure 4.1.

Figure 4.1: You intercept, from a totally reliable sensor, a coded message about the activity a, in the form of a sensor value s_v. Therefore, you know the s_v message was generated either during a by c_a or not during a by $c_{\neg a}$.



Source: The author

Now, if we know s is in fact an unreliable truth machine, operating with probability p_u in the unreliable mode, we will have three codes, c_u, c_a and $c_{\neg a}$, representing, respectively:

- The case where the sensor is operating in the unreliable mode. This code c_u will have a probability of p_u, since this is the probability of operating in the unreliable mode. By definition, this code's decoded message will be $\{a, \neg a\}$, since s_v is unrelated to the state of Ω_a.
- The case where a is happening, and s_v is reliable. Since we know s_v was received, this code c_a will have a probability of $P'(a|s_v)$ of being chose, where P' is a distribution where every s_v is reliable. Of course, this code decodes s_v as $\{a\}$.
- The case where a is not happening, and s_v is reliable. Since we know s_v was received, this code $c_{\neg a}$ will have a probability of$P'(\neg a|s_v)$ of being chose, where P' is a distribution where every s_v is reliable. Of course, this code decodes s_v as $\{\neg a\}$.

Finally, we are left with the problem of estimating P' from P. Following the unreliable truth machine interpretation, $P(c_u)$ is independent to both $P(c_a)$ and $P(c_{\neg a})$ and so we can obtain P' by discounting P by p_u. Thus,

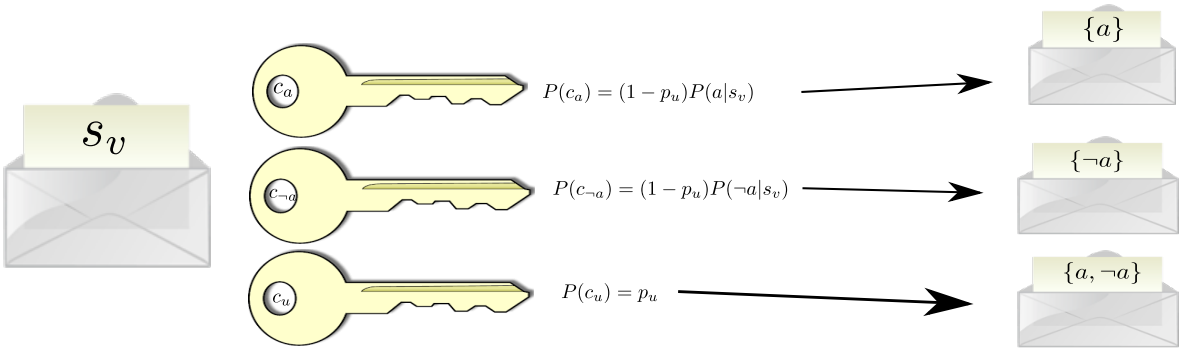
$$P'(a|s_v) = (1 - p_u)P(a|s_v)$$

$$P'(\neg a|s_v) = (1 - p_u)P(\neg a|s_v)$$

The final situation for the codes can be seen in Figure 4.2.

For example, let's consider a fridge sensor that is activated 60% of the times a get drink activity happens, with its remaining 40% activations happening in other activities. If

Figure 4.2: You intercept, from an unreliable sensor, a coded message about the activity a, in the form of a sensor value s_v. Therefore, s_v has been generated in one of three different ways: in the unreliable mode (c_u) , in the reliable one by a (c_a), or in the reliable mode by some activity that is not a ($c_{\neg a}$).



Source: The author

we consider the sensor as totally reliable, the *bba* the fridge sensor generates on $\Omega_{drink} = \{drink, \neg drink\}$ would be

$$m^{\Omega_{drink}}(\{drink\}) = 0.6$$

$$m^{\Omega_{drink}}(\{\neg drink\}) = 0.4$$

If we knew, for sure, that this sensor is an unreliable truth machine, operating in its unreliable mode with a probability of 20%, the correct procedure is to discount $m^{\Omega_{drink}}$ by 0.2, generating

$$m^{\Omega_{drink}}(\{drink, \neg drink\}) = 0.2$$

$$m^{\Omega_{drink}}(\{drink\}) = (1 - 0.2)0.6 = 0.48$$

$$m^{\Omega_{drink}}(\{\neg drink\}) = (1 - 0.2)0.4 = 0.32$$

So, for any sensor that does not operate always in the unreliable mode, or is linked exclusively to a single activity, it should provide evidence both in favor and against the activity.

There is a practical problem with the model as presented here. Sensor information is too low level, being relative to a specific sensor layout and not immune to changes in the environment. For example, if a movement sensor is changed from one room to another, we do not want it to favor activities related to its old location, but the new one instead. In fact, the user's location is the relevant information, and our model should reflect that. For this reason, we will work with messages that represent higher level context information, instead of simple sensor outputs. In order to represent this high level messages, we will use a generic ADL ontology, described in the following section.

4.2 An Ontological model for smart houses

If one wishes to use one of the evidence theory methods for activity recognition reviewed in Section 3, one needs to know, before applying the method, which sensors provide evidences for which activities. Arguably, this modeling process can incorporate the expert's knowledge and experience in ADL recognition. On the other hand, it should be as automatic as possible, as expert time is a limited resource and houses are numerous.

In fact, this task can become quite dull, since it relies more in common sense than technical expertise. For example, a smart expert could write rules such as "if there is a toilet seat sensor, link it to *use toilet* to automate some of the process. Even best, one can build an ontology describing activities of daily living, which will define what are the possible activity models. In this case, deciding if some information is compatible with an activity is simply checking if it fits any valid model of the conceptualization encoded in the ontology.

In this section we will show how this, to some extent, can be achieved using an ontological description of activity recognition. We can think of the approach receiving as inputs an ontology describing the concepts related to ADL scenarios and a specific instance of a smart house and returning a mapping from the existing house features to the activities that allow them.

4.2.1 Modelling Sensors

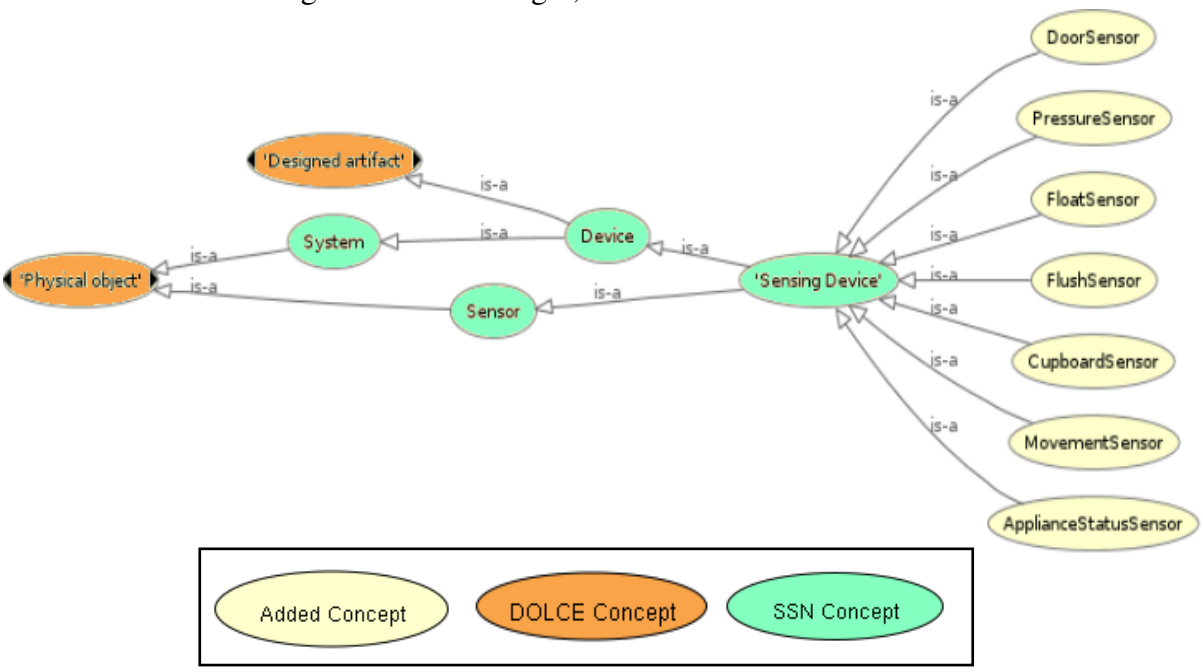
In order to best model sensors, we will use the SSN ontology (LEFORT et al., 2011). It is aligned with DOLCE (GANGEMI et al., 2002) ultra lite version, DUL (DUL...,) , a top level ontology that will also help us model other aspects of the activity recognition process. A sensor, in SSN, observes a *Property*, which is an observable *Quality* of an *Object* or *Event* (LEFORT et al., 2011). Sensors provide observations in the *region* of possible values regarding those properties.

While many types of smart sensors exist, we will model some of the most popular ones. So, let's shortly explain the modeled sensors, their properties and their possible observations

- A *DoorSensor* is a sensor attached to a door that observes if the door is opening or closing. This is usually achieved by attaching a magnet to the moving part of the door and the sensor itself in the door frame. So, every door sensor observes a Property *doorStatus*, generating a *DoorStatusObservationValue* in the *region* for *doorStatus*, which contains *DoorClosedObservationValue* and *DoorOpenedObservationValue*.

- A *PressureSensor* is a sensor attached to some surface, measuring the pressure that is being applied to it. This sensor is usually placed in a bed or chair, and is used to see if a person is sitting/lying on it. In this work, we are not interested in the specific pressure value obtained by the sensor, so we can divide pressure readings in two regions, one for (reliable) pressure detected and one for none. So, every pressure sensor observes a Property *pressure*, generating a *PressureObservationValue* in the *region* for *pressure*, which contains *PressureDetectedObservationValue* and *NoPressureDetectedObservationValue*. Notice that this modeling choice does not limit future use of more detailed pressure values, as *PressureDetectedObservationValue* is itself a *region* and can be divided as needed.
- A *FloatSensor* is a water level sensor attached normally to a toilet in order to detect changes in water level, such as what happens when one flushes the toilet. In this work, we are only interested in measuring the magnitude of the change in water level (as we don't want to account for a flushing twice). So, every float sensor observes a Property *waterLevel*, generating a *WaterFloatObservationValue* in the *region* for *waterLevel*, which contains *WaterLevelUpObservationValue* and *WaterLevelDownObservationValue*. Again, both *WaterLevelUpObservationValue* and *WaterLevelDownObservationValue* are *region*s and thus can be later detailed if needed.
- A *FlushSensor* is a sensor that can measure the flow of water. This sensor can be used to detect if a faucet is running, for example. In this work, we are interested only in knowing if the faucet is running or not. So, every flush sensor observes a Property *waterFlow*, generating a *WaterFlowObservationValue* in the *region* for *waterFlow*, which contains *WaterRunningObservationValue* and *WaterStillObservationValue*.
- A *MovementSensor* is a motion sensor, attached to a room's ceiling or some other platform, that detect motion in the envirorment. This is usually achieved using passive infrared, but other technologies are available. In this work, we are interested in detecting the user location and object interaction, so motion sensor in a room will detect user presence, but movement sensors attached to objects will detect not only presence, but object interaction. So, every movement sensor observes a Property *motion*, generating a *MovementObservationValue* in the *region* for *motion*, which contains *MovementDetectedObservationValue* and *StillnessDetectedObservationValue*.
- A *ApplianceStatusSensor* is a sensor attached to a home appliance, such as a microwave, that observes if said appliance is on or off. This is usually achieved by measuring the eletric current at the appliance's power plug. So, every appliance sensor observes a Property *applianceStatus*, generating a *ApplianceStatusObservationValue* in the *region* for *appli-*

Figure 4.3: To the right, the sensor classes modeled.



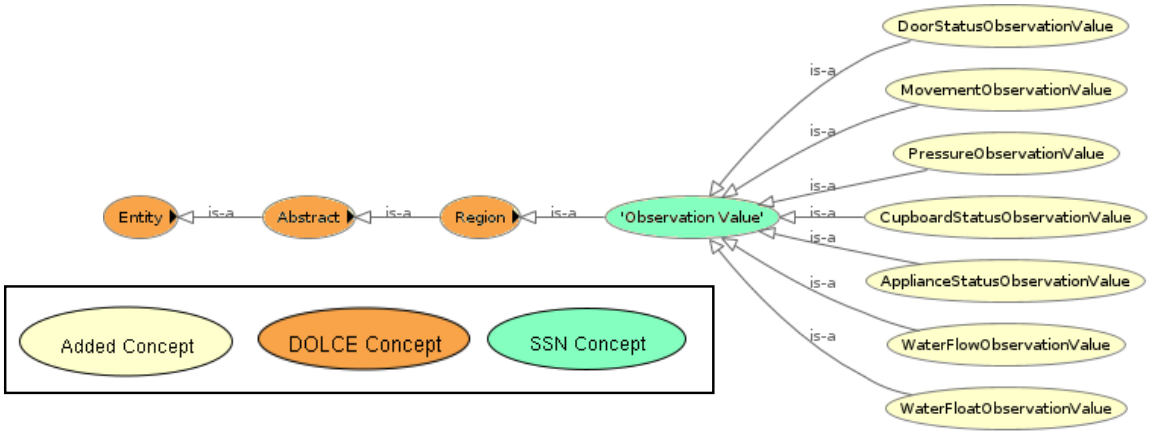
Source: The author

anceStatus, which contains *ApplianceStatusOff* and *ApplianceStatusOn*.

- A *CupboardSensor* is a sensor attached to a cupboard that observes if the cupboard is opened or closed. This is usually achieved by attaching a magnet to the moving part of the cupboard and the sensor itself in the cupboard's frame. So, every cupboard sensor observes a Property *cupboardStatus*, generating a *CupboardStatusObservationValue* in the *region* for *cupboardStatus*, which contains *CupboardOpenedObservationValue* and *CupboardClosedObservationValue*.

We can see the hierarchy for sensors in Figure 4.3, while Figure 4.4 shows the one for their corresponding observation values.

Figure 4.4: To the right, the observation classes for the sensors modeled.



Source: The author

4.2.2 Modelling Locations

Locations are essential to the activity recognition process, since some activities happen in specific locations. For example, preparing a meal happens typically in the kitchen. In this work, we model house regions as a *DOLCE SpaceRegion*, which is any spatial region used to localize an *Entity*.

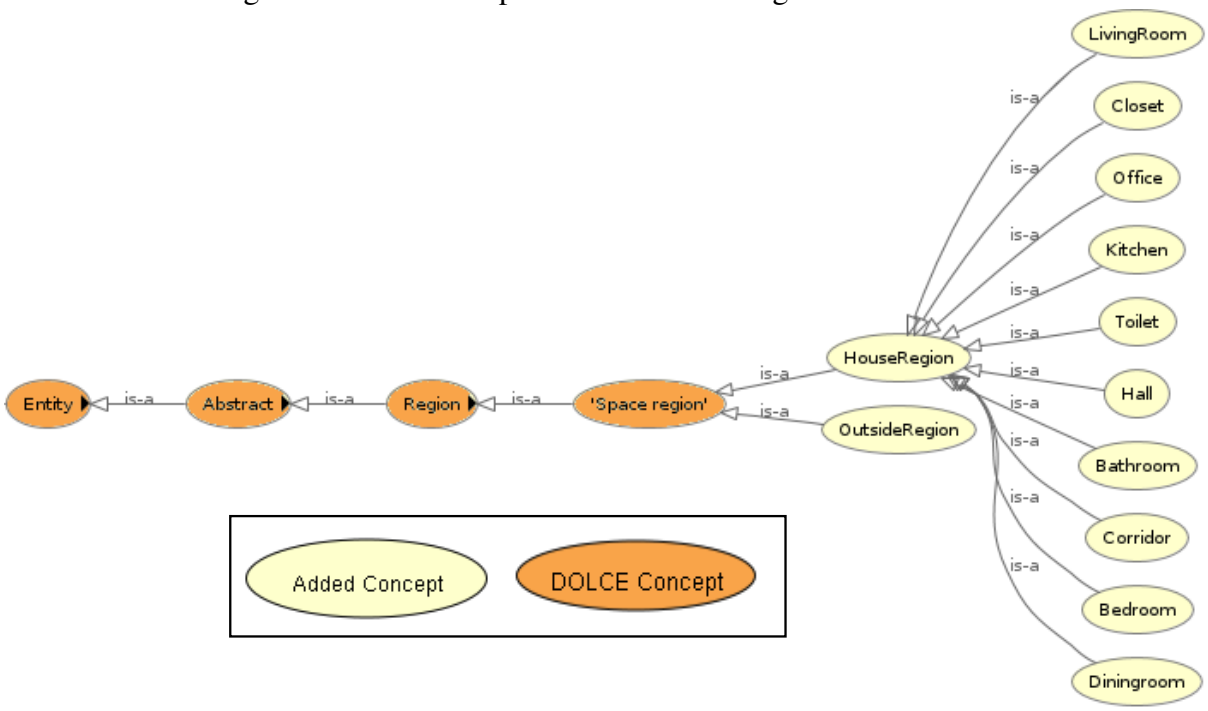
Each house has a *HouseRegion*, which is a region in space constituting of all that is considered inside the house. This region is composed of smaller *HouseRegion*s, representing relevant spaces for activity recognition inside the house. For example, a *Bedroom* is a class of *HouseRegion* that represents a room in the house which is used for sleeping. A house also has an *OutsideRegion*, which constitutes all areas outside the house, such as its garden, the sidewalk and also any other area out of the house's scope, such as a shop down the street.

The concepts used for describing house locations can be seen in Figure 4.5.

4.2.3 Modelling Time

Time itself is already modeled in *DOLCE* using the *TimeInterval* concept, which represents any region in a dimensional space used for representing time. The region could be the region of all the time in the universe, for example. The *TimeInterval* concept is generic enough to represent the notions of both instant and interval, as instants are just intervals of length zero.

Figure 4.5: The concepts used for describing house locations.



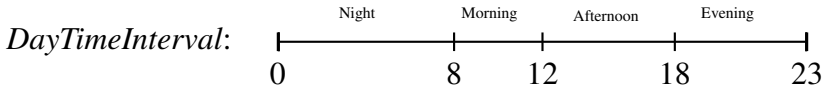
Source: The author

Absolute time can be very relevant to activity recognition, as some activities must happen in a specific time, such as medication intake. The idea of "time of the day", such as "morning" is also a very relevant concept in activity recognition. If we think of the region of all absolute time, it is easy to see that morning is a name we give to a specific subset of day time, such as from 8:00 to 12:00. While people may disagree on the start and end of such boundaries, since these concepts may have cultural or geographic roots, knowing the current time of the day is useful.

In this work, we will use the concept of *DayTimeInterval* to describe a class of *TimeInterval* that is part of a typical day. Thus, we define time regions for "morning", "afternoon", "evening" and "night". Such notions can be thought as fuzzy, as the "evening" does not suddenly become the "night". This notion is not explored in this work, but is still compatible with its time modeling, as an instant can belong to more than one *DayTimeInterval* class.

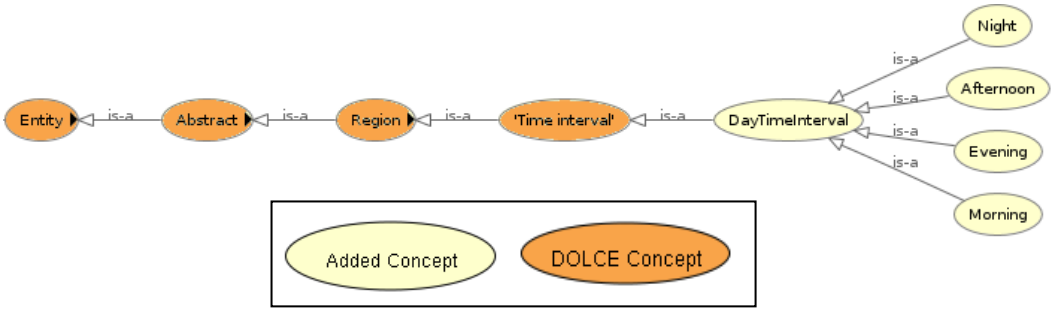
There is a simple reason to refrain from a fuzzy time representation. The evidence theory methods studied in Section 3, when recognizing ADLs, use a fixed scale for defining time of the day. So, for comparison's sake, all methods will use the same fixed intervals for defining time of the day: those shown in Figure 4.6. The concepts used for describing time of the day can also be seen in Figure 4.7.

Figure 4.6: Limits used to define time of the day regions for this work.



Source: The author

Figure 4.7: The concepts used for describing time of the day.



Source: The author

4.2.4 Modelling objects

Objects are also related to activity recognition. For example, going to sleep can include two obvious objects: the bedroom door and the person's bed. As some sensors are used to detect object interaction, the objects sensors are attached to are also relevant. For example, a pressure sensor attached to the toilet seat indicates that the person is using the toilet, while the same sensor in the living room's sofa can indicate that the subject is watching TV.

Household objects can be modeled in *DOLCE* as *DesignedArtifact*s. A *DesignedArtifact* is a *PhysicalArtifact*, which is a physical object that has some goal. Specifically, a *DesignedArtifact* also posesses some design, meaning that it is not natural object such as a simple rock. On the other hand, a rock that was manufactured to hold books in place is a *DesignedArtifact*.

There are a multitude of possible *DesignedArtifact*s in a house that can be possibly used in activity recognition. Including manually all of them is unfeasible. On the other hand, huge taxonomies of objects are available in Wordnet (MILLER, 1995) and other ontologies such as SUMO (NILES; PEASE, 2001). Moreover, web information can be gathered to enhance this classification. Alternatively, in the context of a smart sensor operating in a middleware, the sensor may inform its model, capabilities and location as it enters the smart system.

In this work, since our testing scope is limited, all the needed object classes for the evaluated datasets were manually created. The generated hierarchy can be seen in Figure 4.8.

Figure 4.8: The object classes used in this work, generated based on the available information about the tested houses.

Source: The author

4.2.5 Modelling Activities

The main idea, from an ontologic modeling perspective, is to define for each activity restrictions on what is the possible state of affairs for a situation of that type. Thus, one may face a trade-off between representing the common sense knowledge regarding an activity and its ontologic restrictions. The first is too specific to represent all possible activity instances, while the latter represents all instances, but is too generic to be of much use for activity recognition.

For example, we can restrict the *go to bed* activity, as seen in Figure 4.10b, to happen only in the night and morning. This does not represent a perfect restriction on going to bed, as one can occasionally go to bed in the afternoon due to jet lag, for example. But, if we leave out time restrictions, we may fail to recognize between going to bed and taking a power nap, for example.

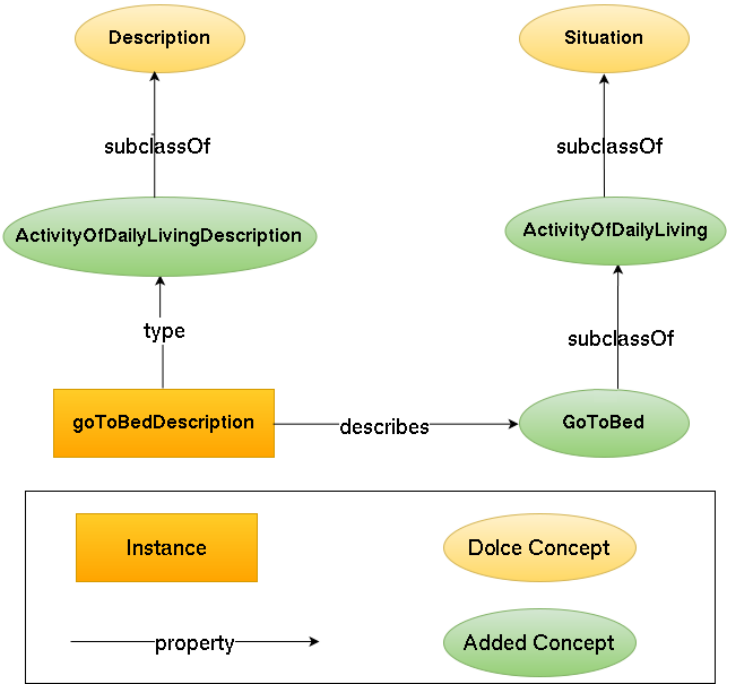
Moreover, in the real world, we also face two other problems:

- Sensors are unreliable. For example, if a movement sensor misfires, a wrong user location can be inferred. Thus, a real life instance of an activity may seem, to the system, to be incompatible with its conceptualization.
- Scenarios are heterogeneous, since they include different houses, habited by different people who perform activities in different ways. Even the same person may perform a specific activity in several different ways.

To account for all of these problems, we will consider an activity as a *Description*, that is, a theory explaining a *Situation*. For example, interacting with the kitchen's door and the cups cupboard can be explained in different ways using a description for the get drink and take medication activities. Just as a diagnose may not account for all the patient's symptoms, an activity description may not account for all that is present in the *Situation*. Following this idea, an activity conceptualization does not represent all the possible activity instances, but a correspondence between the characteristics of the current situation and their relative interpretations if we consider that the activity is happening. For example, if a movement sensor misfired, providing a wrong location information, an activity description that does not allow such location will map this location as a sensor error, instead of dismissing the activity as impossible.

In this work, this mapping is represented using *Situation* classes. That is, each activity type will have a *Description* instance, which will describe a specific *Situation* class as seen in Figure 4.9. Restrictions on the description's situation class represent what is explained by the activity theory. Everything that does not fit such restrictions is interpreted as either a sign that

Figure 4.9: An example of a mapping from a description to the class of situations it can explain. In our model, each activity has a specific description instance, representing a theory that explains a situation as being of specific activity type. Each description describes a specific class of situation, which specifies the expected restrictions on the activity. This representation of a property from an instance to a class is possible in OWL 2, and is denominated punning.



Source: The author

the activity is not happening, or a sign that the information is unreliable. Examples description's situation classes definitions can be seen in Figure 4.10.

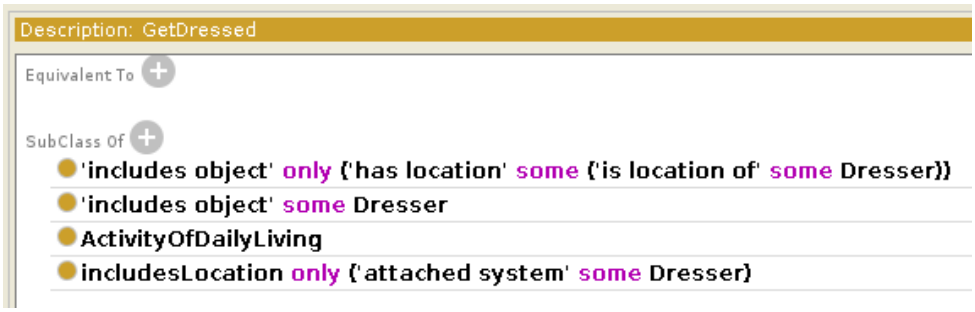
In the activity recognition process, we will not directly instantiate any description's situation class. Instead, we will create a generic *Situation* instance, of which we are not aware of its true description class. Indeed, choosing the most appropriate description for the aggregated situation is in itself a summary of what activity recognition entails. A taxonomy of all activities modeled can be found at 4.11.

By looking at the activity ontologies and what information is available in activity recognition datasets, we can see that four aspects characterize an activity *Situation*:

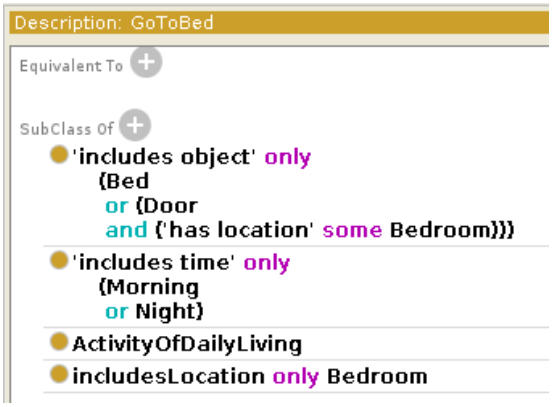
- The objects it includes. A drink activity can include the fridge, but not the shower. The current *Situation* will be linked with the object it includes using the *includesObject* DOLCE object property. In other words, a situation that was inferred as including a specific object has an object property assertion stating that it includes this object interaction.

Figure 4.10: Examples of *Situation*s classes described by an activity's *Description*. Everything that is valid in its described *Situation* class, an activity *Description* can explain.

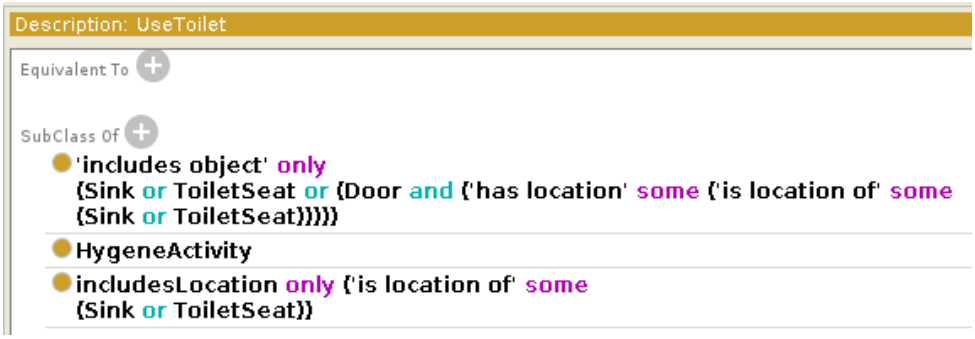
(a) Restrictions on the *Situation* class for the get dressed activity *Description*.



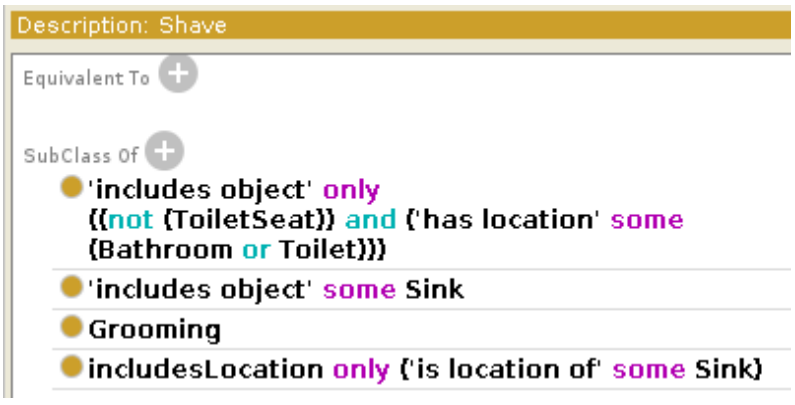
(b) Restrictions on the *Situation* class for the go to bed activity *Description*.



(c) Restrictions on the *Situation* class for the use toilet activity *Description*.

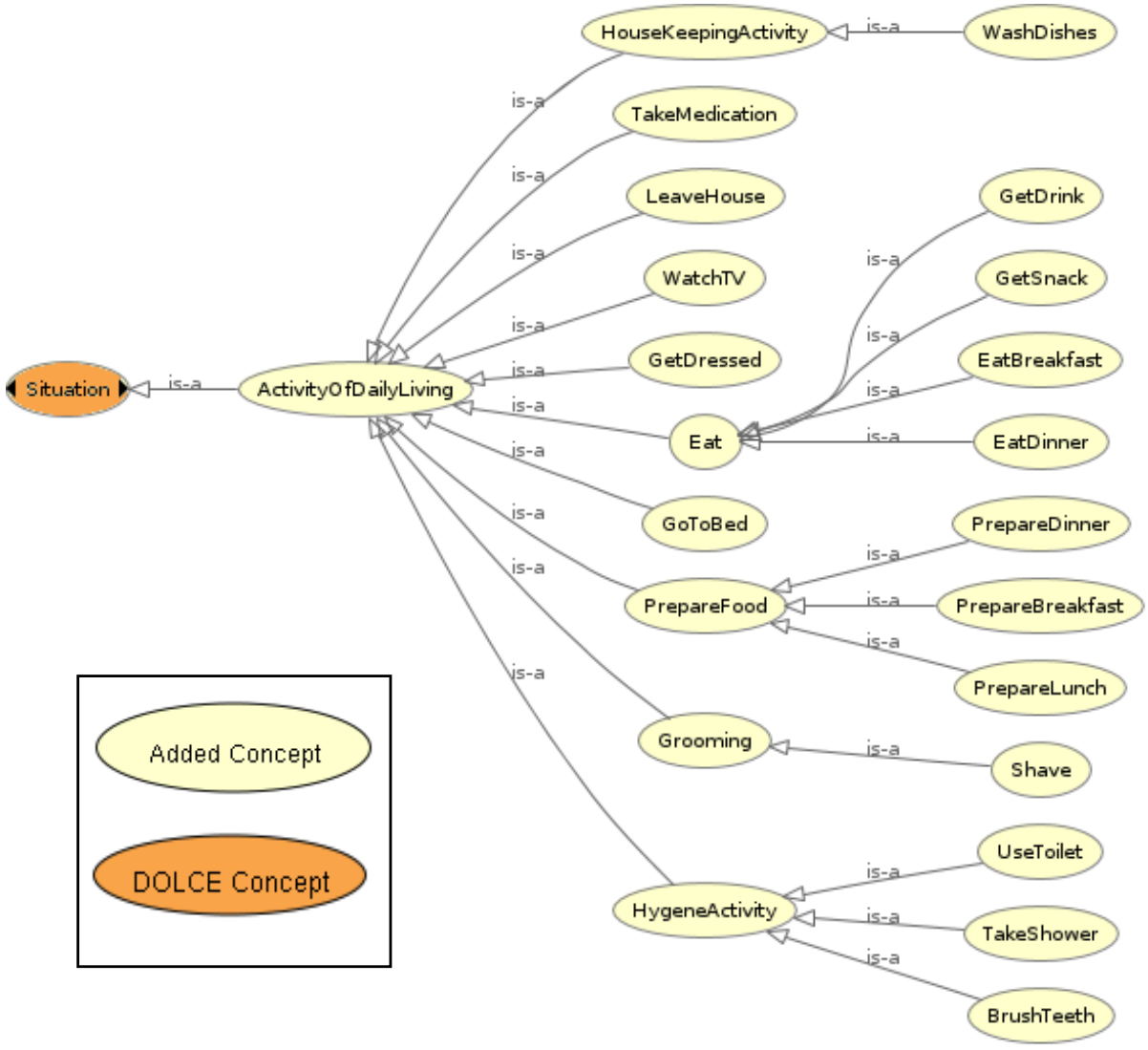


(d) Restrictions on the *Situation* class for the shave activity *Description*.



Source: The author

Figure 4.11: A taxonomy of all activities modeled.



Source: The author

- The time of the day it happens. A breakfast activity can only happen in the morning, but this is not true of a dinner one. Its use follows the same pattern seen in *includesObject*, but with time of the day being inferred instead of object.
- The locations the activity can happen. One is not expected to prepare dinner in the bathroom. This time, no DUL object property was found to represent this notion of a situation including a specific *SpaceRegion*. For this reason, a new object property, named *includesLocation* was created. Its use follows the same pattern seen in *includesObject* and *includesTime*, but with user location.
- The last object included. For example, if a subject has interacted with several objects, such as the sink and the oven, it is useful to know which one was the last one, so that we may differentiate between cleaning after and before cooking. Thus, to represent this, we will create a new object property, *includesLastObject*. The current *Situation* will be linked with the object it includes using the *includesLastObject* DOLCE object property. Of course, including an object as the last one also means including it. Thus, *includesLastObject* is subsumed by *includesObject*. That is, every *includesLastObject* assertion is also a *includesObject* assertion.

4.2.6 Implementing Situation Interpretation

The described ontology was developed in OWL 2(HITZLER et al., 2009) using Protégé(PROTéGé...,). It uses the OWL 2 model of SSN, which can be found at (LEFORT et al., 2011). Since SSN is already aligned with DUL, the lightweight DOLCE OWL 2 version, the modeling process was quite straightforward. The evaluation of a Situation can be easily done using the OWL API (HORRIDGE; BECHHOFER, 2011) and the Hermit reasoner (GLIMM et al., 2014).

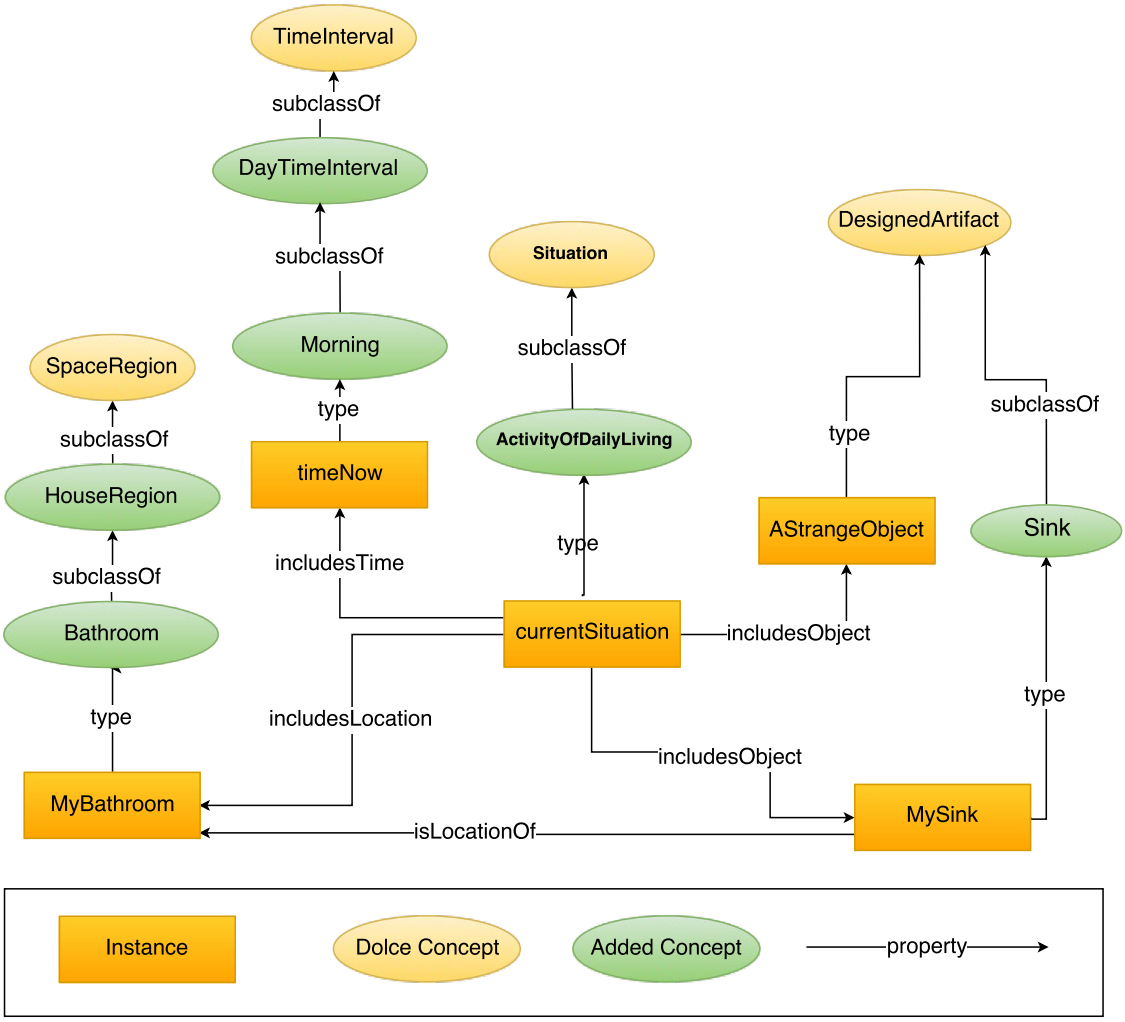
4.2.6.1 Evaluating a Situation

In order to understand how situation assertions are evaluated using descriptions, we will use a simple example:

"It is 9:00. In the house's bathroom, the user has turned on the sink. The user has also interacted with some other unknown object, which we don't yet know much about."

The representation of this example in the ontology can be seen in Figure 4.12.

Figure 4.12: An example of a situation for an Activity of Daily Living.



Source: The author

Now, let's see how this *Situation* is evaluated by the description for the *shave* activity. First, lets evaluate the assertion *includesTime timeNow*, seen in Figure 4.12. The shave description, which can be seen in Figure 4.10d, has no time of the day restriction. This means it can happen at any time of the day, and so shave can explain a time in the morning. On the other hand, shave has restriction on both the locations and objects involved. The assertion *includesLocation MyBathroom* is fine, since *MyBathroom* is the location of *MySink*.

On the object interaction side, things are not so straightforward. Shaving cannot include the toilet seat, and can only include objects located either in the bathroom or the toilet. Evaluating the assertion *includesObject MySink* is fine regarding the location, as the sink is located in *MyBathroom*. On the other hand, *MySink* cannot be a toilet seat. In OWL, a single instance can

have multiple classes, so this possibility needs to be checked. Of course, we know no sink can be a toilet seat, and so this problem can be solved adding an axiom stating that *ToiletSeat* and *Sink* are disjoint. This means that *includesObject MySink* is fine, since it fits the restrictions.

On the hand, the assertion *includesObject AStrangeObject* is not so easy. We are doing ontological reasoning in OWL. Since OWL reasoning works in the open world assumption, meaning that, just because there is no axiom stating that *AStrangeObject* is a *Sink*, does not mean it is not one. Even more, it can even be a *Sink* located at *MyBathroom*, meaning that it can possibly fit the shave description. In other words, *includesObject AStrangeObject* will be considered, by an OWL reasoner, as satisfiable for the *Shave* concept. This is not what we want. If we are not sure if an assertion fits some description, we cannot say that this assertion is explained by it.

For this reason, we don't work directly with satisfiability. We are interested in knowing if the assertion can be explained by the description, and thus we use entailment instead. That is, if we can prove the assertion fits the restrictions, it is considered as explained by the description.

4.2.6.2 Evaluating Missing Assertions

There are also the objects our current *Situation* does not include at the moment. A missing assertion for an activity *Situation* can be explained as valid by any activity description. This is because, for any missing possible assertion p, for any activity description d:

- If d explains the assertion p, p may be missing because it will come in the future, just as one washes one's hands after using the toilet. It is also possible that p did not happen in this particular activity instance, since there are more ways than one to perform an activity.
- If d cannot explain p, not having p is expected.

Thus, one may think missing assertions are meaningless. This is not the case. A missing assertion is still a message, in the evidence theory sense. If a particular object interaction is always missing for a certain activity, such as the toilet for shaving, not seeing it should increase our belief in a *shave* activity and decrease our belief in an *use toilet* activity. We will see more about how missing assertions provide evidences for activities in Section 4.3.

4.3 Combining Evidence Theory with Ontological Reasoning

Our proposal combines the evidence theory interpretation of activity recognition presented in Section 4.1 with the ontological modeling of activities presented in section 4.2.

- Ontological modeling will be used to define when sensors are operating as unreliable truth machines and how sensor information becomes activity evidence.
- The evidence theory part will combine evidences for activities and decide which activity is happening.

In this section, we detail, step by step, our approach. We will start with a general description of the algorithm, as can be seen in Algorithm 1. Our approach receives as its inputs:

- A specific scenario description, usually in the form of an XML file listing the sensors, rooms and activities present in the smart home.
- The generic ADL ontology discussed in Section 4.2.
- A sequence of ground truth activity instances.
- A segmentation strategy. Our approach is segmentation agnostic, relying on some other method for this task. Examples of available segmentation strategies are fixed time windows, segmenting on sensor change, etc.
- A evidence theory combination rule, as discussed in Section 2.1.2, such as the one for TBM or DS.
- A function for evaluating the combined evidences, as discussed in Section 2.1.4. Examples are belief, plausibility and the pignisitc probability.

Following the Algorithm 1, we start, by translating the available scenario information into the ontology's primitives (line 2), as detailed in Section 4.3.1. After this, we use the available training data to estimate both sensor reliability and the weight each situation feature should have for each activity (line 3), as detailed in Section 4.3.3. The following loop (line 4) is the actual activity recognition process. It starts with the segmentation step of sensor information (line 5).After that, the current situation is obtained from the segmented sensor information (line 6), as detailed in Section 4.3.2. The current situation is then evaluated using ontological reasoning and evidence theory. This process includes combining all evidence present in the current situation using an evidence theory combination rule (line 7) and evaluating this combined information using some scoring function (line 8) such as belief. For a complete description of this two steps, see Section 4.3.4.

Algorithm 1: Overview of activity recognition steps

Input:
A scenario specification $house_d$
An ADL ontology ont_{adl}
A sequence of ground truth activities $acts_{gt}$
A segmentation strategy seg_{st}
A combination rule $\odot$
An evaluation function ν

```
1  begin
2      initialize inst_h based on how house_d fits ont_adl ;
3      train a model for inst_h based on acts_gt and seg_st ;
4      while system is running do
5          use seg_st for segmentation ;
6          obtain the current situation from the segmented information ;
7          combine relevant evidences with ⊙ ;
8          choose best activity with ν ;
```

4.3.1 Preprocessing

Before we can start the activity recognition process, we must instantiate the current smart house scenario in our generic ADL ontology. The scenario specification must list the available sensors, locations and activities. It may be represented by a XML/JSON file, or may be obtained in real time from some middleware. The general process can be seen in Algorithm 2.

Algorithm 2: Instantiating a smart home

Input:
A scenario specification $house_d$
An ADL ontology ont_{adl}
Output: An instantiated smart house model

```
1  initialize inst_h with all ont_adl concept axioms ;
2  begin
3      foreach location l of house_d do
4          add corresponding ont_adl location instance for l in inst_h ;
5      foreach sensor s of house_d do
6          add corresponding ont_adl sensor instance for s in inst_h ;
7          create and locate a platform for the instantiated sensor ;
8      foreach activity a of house_d do
9          add corresponding ont_adl description instance for a in inst_h ;
10     return inst_h
```

As we can see, the current instantiating process is quite straightforward, requiring information normally available in smart house datasets.

4.3.2 From Sensors to Situations

In this work, sensor and context information will be translated to assertions regarding the current *Situation*. If individual sensor readings are added to the ontology as they happen, this process can be performed using SPARQL (PRUD'HOMMEAUX; SEABORNE et al., 2008) construct queries as seen in Figure 4.13. On the other hand, for faster performance, code can be written using the OWL API (HORRIDGE; BECHHOFER, 2011) that performs this translation. In either case, old information can be discarded after it is deemed irrelevant by the segmentation process. This puts a limit on the storage needed for the method and thus allows activity recognition for any arbitrary number of days.

Figure 4.13: Example of a SPARQL construct query that performs the translation between a sensor observation to a situation property assertion. In this case, if an appliance is sensed as "on", the situation is said to include such appliance.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX  dul: <http://www.loa-cnr.it/ontologies/DUL.owl#>
PREFIX ssn: <http://purl.oclc.org/NET/ssnx/ssn#>
PREFIX ont: <http://www.semanticweb.org/vitor/ontologies/2015/0/untitled-ontology-62#>

CONSTRUCT { ?situation ?property ?propertyValue . }

WHERE {
  ?situation                rdf:type                      ont:ActivityOfDailyLiving .
  ?value                    rdf:type                      ont:ApplianceStatusObservationValue .
  ?cap                      ssn:forProperty               ont:applianceStatus .
  ?s                        ssn:hasMeasurementCapability  ?cap .
  ?res                      ssn:hasValue                  ?value .
  ?s                        ssn:madeObservation           ?obs .
  ?obs                      dul:isObservableAt            ?obsInt .
  ?obsInt                   dul:hasIntervalDate           ?date .
  ont:currentTimeInterval   dul:hasIntervalDate           ?date .
  ?obs                      ssn:observationResult         ?res .
  ?s                        ssn:onPlatform                ?propertyValue .

  FILTER( ?value =  ont:ApplianceStatusOn ) .

  BIND( dul:includesObject AS ?property  )   .
   OPTIONAL {
     ?situation dul:isObservableAt ?sitInterv .
     ?sitInterv  ont:hasIntervalEnd  ?sitEnd .
   } .
   FILTER( !bound(?sitEnd) ) .
 }
```

Source: The author

Most translations can be simple, such as translating clock time to time of the day or movement in a specific region to user location. But one can think of more complex mappings, since we are operating with ontology primitives directly. For example, after inferring that the user is at the kitchen at midnight, the system can add to the current situation information regarding the intent of a midnight snack.

In this work, we consider the following simple mappings:

- Appliance, Cupboard, Flush and Float sensors provide information on object interaction.
- Movement sensors provide object interaction if attached to some object such as the shower.
- From all sensors that require user interaction, the last one activated provides the user's location.
- The last object interaction is also inferred.
- The current time provides information regarding time of the day.

4.3.3 Training

It is clear that different aspects of situations affect activity recognition with different weights. For example, using the sink should provide more weight to *use toilet* than to *take shower*, as washing one's hands is expected in the former. In order to estimate how each aspect affects each activity, we will use some labeled activity instances following Algorithm 3.

Algorithm 3: Training

Input:
An instantiated smart house model $inst_h$
A sequence of ground truth activities $acts_{gt}$
A segmentation strategy seg_{st}
Output: The trained model for $inst_h$

```
1  begin
2      foreach activity a_gt of acts_gt do
3          foreach minute m_a of a_gt do
4              f_min = feature present in m_a by seg_st ;
5              sit_t = translated f_min as situation in inst_h ;
6              a_d = description of a_gt ;
7              increase P_a(a_d) ;
8              foreach possible assertion p of sit_t do
9                  if p is present then
10                     increase P_w(a_d|p) ;
11                     if p is explained by the description a_d then
12                         increase P_r(reliable|p) ;
13                     else
14                         increase P_r(¬reliable|p) ;
15                 else
16                     increase P_r(reliable|p) ;
17                     increase P_w(a_d|¬p) ;
18         return (P_a, P_r, P_w)
```

First, we will translate, following some segmentation strategy, the available minute information to a situation in the ontology (lines 4 and 5). This is performed as seen in Section 4.3.2. At this point, we can add 1 to the activity count, used to estimate the prior probability of activities (line 7). This information is an evidence source for activity recognition since, even without any other context information, a rare activity in the past can be expected to continue being rare in the future. We will evaluate all possible assertions for the current situation in order to estimate their weight on the current ground truth activity (line 8).

If an assertion is present, it increases [1] the weight of its related activity (line 10). If this assertion can be explained by its activity's description, we will assume it is reliable. A present assertion found reliable not only increases the weight of its related activity, but also its reliability in the same manner (line 12).

The assertion may be present, but not compatible with the theory. For example, we may have an assertion of location in the kitchen during a shower activity. In this case, we will assume that this information came from operation in the unreliable mode, and so increase the weight of the assertion's unreliability (line 14).

The assertion may be missing. In this case, it can always be explained by any description, as we have seen in Section 4.2.6.2. For this reason, a missing assertion will increase the assertion's reliability (line 16). Just as a valid, present assertion, a missing one is also a randomly coded message following the evidence theory interpretation discussed in Section 4.1. Thus, we will also keep track of the times the activity is happening and the assertion is missing, (line 17).

The trained model for the house scenario is returned in line 18. It consists of:

- The prior distribution of activities P_a.
- The assertion reliability distribution P_r.
- The assertion weight on activity distribution P_w.

It is also important to mention that all distributions use the laplace estimator. We will see how these distributions are used to generate masses for evidences in the activity recognition process in Section 4.3.4.

[1] In this case, we won't proceed as with activity prior estimation. Since classes are unbalanced, we will take this into account, weighting instances by the size of the classes so that each class has the same summed instance weight. For example, if A happens twice as much as B, a B instance weights two times as much as A.

4.3.4 Activity Recognition

Given a segmented set of sensor information, used to generate some features, one must finally decide what is the activity that is happening. This step is performed using Algorithm 4. This algorithm uses the distributions learned in the training step, the instantiated smart house ontology obtained in the preprocessing step and other inputs related only to it. Those are, of course, the features deemed relevant by the segmentation step, but also includes some evidence theory primitives. They are:

- The evidence theory combination rule $\odot$ which will be used to combine all evidence regarding the current situation. This choice depends on which flavor of evidence theory is being used, as each one uses a specific combination rule as seen in Chapter 2. Alternatively, the methods allows the use of one of the many other combination rules proposed in the literature, even if they do not match a specific well defined evidence theory interpretation.
- The evaluation function ν that will calculate a score for this combined evidences. As we have seen in Chapter 2, if one is being optimistic, the Plausibility function can be used. The Belief function is the pessimistic choice, while the Pignisitc Probability function represents the betting rate one should apply for the activity.

Following Algorithm 4, we start by initializing a structure for storing each activity's score (line 2). Then, the current features are used to build the current situation (line 3), as we have seen in Section 4.3.2. Following, we iterate over all the possible activities present in the dataset, so that they are all evaluated (line 4).

In this method, for each activity, we will represent evidences about it in the *fod* $\Omega_a = \{a_d, \neg a_d\}$. The first evidence for any activity comes from its frequency. Following the interpretation presented in Section 4.1, we can think of frequency as a coded message. Thus at any time, for each activity a, its frequency message can be interpreted as in favor of a using c_a with probability $P_a(a_d)$ and against it using $c_{\neg a}$ with probability $P_a(\neg a_d)$. Consequently, the *bba* generated by the activity's frequency information will be

$$m^{\Omega_a}(\{a_d\}) = P_a(a_d)$$

$$m^{\Omega_a}(\{\neg a_d\}) = P_a(\neg a_d)$$

This can be seen, in a shorter form, in line 5.

Algorithm 4: Activity recognition

Input:
An instantiated smart house model $inst_h$
A feature set f_t
A combination rule $\odot$
An evaluation function ν
An activity distribution P_a
A property reliability distribution P_r
A property weight distribution P_w
Output: A prediction for the current activity

begin
  $score_a$ = [] ;
  sit_t = translated f_t as situation in $inst_h$;
  foreach *activity description* a_d *of* $inst_h$ **do**
    $ev_a = \{(\{a_d\}, P_a(a_d)), (\{\neg a\}, P_a(\neg a_d))\}$;
    foreach *possible assertions* p *of* sit_t **do**
      if p *is present* **then**
        if p *is explained by the description* a_d **then**
          $ev_p = \{(\{a_d\}, P_w(a_d|p)), (\{\neg a_d\}, P_w(\neg a_d|p))\}$;
        else
          $ev_p = \{(\{\neg a_d\}, 1.0)\}$;
      else
        $ev_p = \{(\{a_d\}, P_w(a_d|\neg p)), (\{\neg a_d\}, P_w(\neg a_d|\neg p))\}$;
      ev_p = discounted ev_p by $P_r(\neg realiable|p)$;
      $ev_a = ev_a \odot ev_p$;
    store $\nu(\{a_d\})$ of ev_a in $score_a$, as the score of a_d ;
  return $\arg\max score_a$

After, we will evaluate all possible assertions of the current situation (line 6), combining them as evidence on Ω_a. For every possible assertion, it is either present or not in the current situation.

If it is present and explained by a_d, it will provide a *bba* m^{Ω_a} proportional to its weight

$$m^{\Omega_a}(\{a_d\}) = P_w(a_d|p)$$

$$m^{\Omega_a}(\{\neg a_d\}) = P_w(\neg a_d|p)$$

This can be seen, in a shorter form, in line 9.

If, on the other hand, the assertion is present, but not compatible with a_d, it tells us that the activity is not happening, and so provides a *bba* m^{Ω_a}

$$m^{\Omega_a}(\{\neg a_d\}) = 1.0$$

This can be seen, in a shorter form, in line 11.

Assertions that are not present can always be explained by any activity description, as we have seen in Section 4.2.6.2. So, just as with explainable present assertions, those missing generate

$$m^{\Omega_a}(\{a_d\}) = P_w(a_d|\neg p)$$

$$m^{\Omega_a}(\{\neg a_d\}) = P_w(\neg a_d|\neg p)$$

as can be seen, in a shorter form, in line 13

Of course, the assertion itself may not be reliable, regardless of its presence or the interpretation a assigns it. For this reason, its *bba* is discounted according to its reliability (line 14). The discounting procedure is the evidence theory procedure presented in Equation 2.3. After the discounting is done, this *bba* is combined with the others using the combination rule (line 15).

As we are working with evidences on $\Omega_a = \{a_d, \neg a_d\}$, the score of a_d is the result of $\nu(\{a_d\})$, calculated on the *bba* ev_a, which represents the aggregated evidences regarding a_d. We will keep track of activity scores (line 16), so that, when all activity types have been evaluated, we return the one with the best score (line 17).

5 EXPERIMENTAL SETUP

5.1 Datasets

In this work, our objective is to develop a method for real world activity recognition. To evaluate our method, we have chosen datasets as close to reality as possible, following the criteria:

- The dataset should include ground truth activity information, so that evaluation is possible.
- A person must really live in a real world smart house. This leaves out smart labs and simulated scenarios.
- Activities in the dataset should not be scripted, since this hides their true complexity.
- The dataset must include basic ADLs, such as sleeping and using the toilet.
- The dataset must not include indoor cameras, since our focus is on smart, non-intrusive sensors.
- The dataset must have a single subject living alone, since this is a restriction of this work.
- The dataset should include at least 10 days, since the algorithms tested need training data.

Five smart house datasets were found that comply with our requirements for real world activity recognition. They are *houseA*, *houseB*, *houseC*, *ordonezA* and *ordonezB*. Houses *houseA*, *houseB* and *houseC* come from (KASTEREN; ENGLEBIENNE; KRÖSE, 2010), while *ordonezA* and *ordonezB* come from (ORDÓNEZ; TOLEDO; SANCHIS, 2013).

A dataset is usually comprised of:

- A list of the sensors present in the house, and files containing their outputs. The types of sensor present in each house can be seen in Table 5.1.
- Some information regarding the number of rooms and their types. This information can be seen in Table 5.2. Some datasets, such as the ones from (KASTEREN; ENGLEBIENNE; KRÖSE, 2010), come with a depiction of the house plan, as can be seen in Figure 5.1. This gives us information regarding room positioning and also sensor/object placement in the house. The datsets from (ORDÓNEZ; TOLEDO; SANCHIS, 2013) do not provide this information, listing only the room types and their respective sensors.
- A list of activities present in the house, and files containing their occurrences. Tables 5.3 to 5.7 show the activities performed in each house. The number of days present in each dataset can also be seen in Table 5.8

Table 5.1: Sensor types present in the houses.

	Door	**Toilet Flush**	**Cupboard**	**Movement**	**Sink Flow**	**Appliance Status**	**Pressure**
houseA	4	1	4	0	0	5	0
houseB	4	1	7	3	1	2	4
houseC	5	2	7	3	1	2	3
ordonezA	1	1	3	3	0	2	2
ordonezB	1	1	2	5	0	1	2

Source: The author.

Table 5.2: Locations present the dataset houses.

	Toilet	**Bathroom**	**Kitchen**	**Bedroom**	**Hall**	**Office**	**Living Room**
houseA	1	1	1	1	1	1	1
houseB	0	1	1	1	0	1	1
houseC	1	1	1	2	2	1	1
ordonezA	0	1	1	1	1	0	1
ordonezB	0	1	1	1	1	0	1

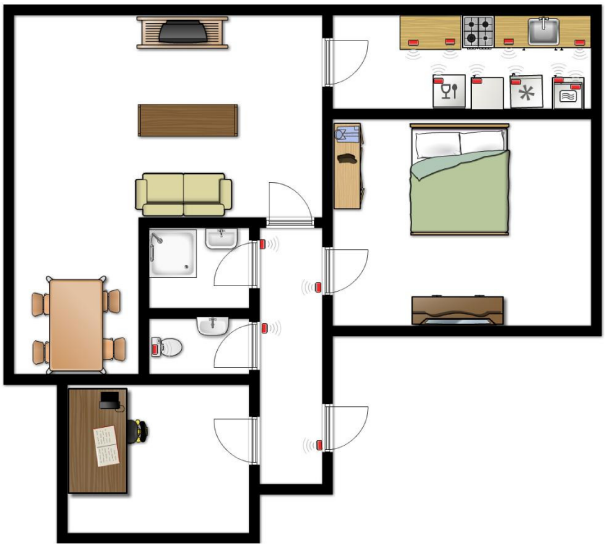
Source: The author.

Table 5.3: Activitties in houseA.

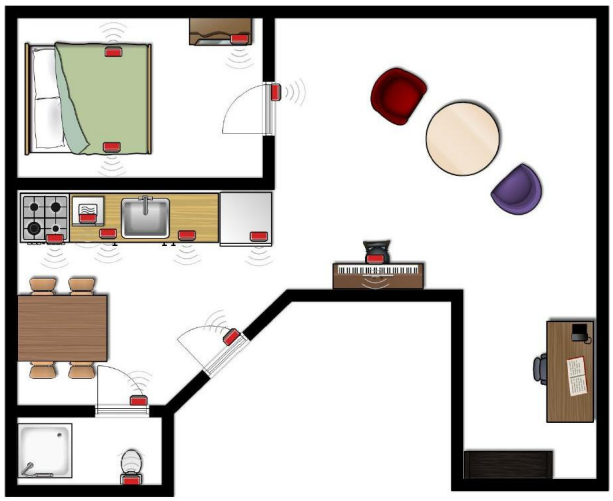
Activity	**Occurrences**	**Average Time (minutes)**	**From**	**To**
leaveHouse	33	509	8:34	23:43
useToilet	114	2	0:1	23:59
takeShower	23	9	8:16	17:22
brushTeeth	16	2	0:8	23:58
goToBed	24	486	0:16	18:5
prepareBreakfast	20	3	8:1	10:29
prepareDinner	9	36	17:58	20:59
getSnack	12	2	0:31	22:57
getDrink	20	1	1:11	23:52

Source: The author.

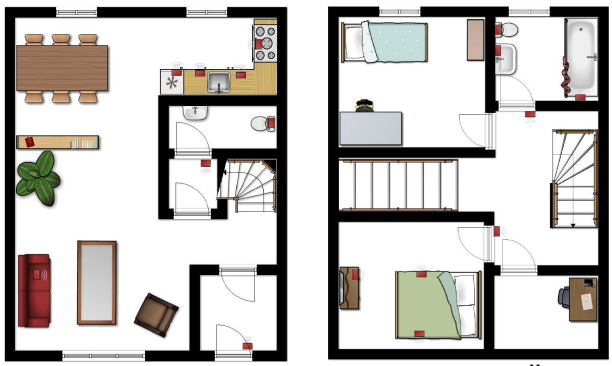
Figure 5.1: House plans for the datasets of (KASTEREN; ENGLEBIENNE; KRÖSE, 2010).
(a) *houseA*



(b) *houseB*



(c) *houseC*



Source: (KASTEREN; ENGLEBIENNE; KRÖSE, 2010)

Table 5.4: Activitties in houseB.

Activity	Occurrences	Average Time (minutes)	From	To
leaveHouse	24	449	4:27	23:47
useToilet	27	2	0:8	23:49
takeShower	11	10	10:27	18:58
brushTeeth	13	2	0:10	23:47
goToBed	14	432	0:11	12:40
getDressed	14	2	10:12	18:59
prepareBreakfast	9	9	9:46	12:51
prepareDinner	6	14	9:28	22:38
getDrink	8	1	6:55	22:22
washDishes	6	4	12:3	21:15
eatDinner	5	9	18:18	22:54
eatBreakfast	10	13	9:35	13:26

Source: The author.

Table 5.5: Activitties in houseC.

Activity	Occurrences	Average Time (minutes)	From	To
leaveHouse	47	254	6:48	23:48
eating	27	13	0:33	20:19
useToilet	52	1	0:11	23:56
takeShower	14	13	6:9	19:9
brushTeeth	26	3	0:4	19:33
useToilet	37	1	0:13	19:0
shave	7	8	0:1	18:35
goToBed	19	408	0:7	11:20
getDressed	23	3	6:14	19:13
takeMedication	5	2	0:12	18:24
prepareBreakfast	10	8	6:20	11:41
prepareLunch	8	7	7:14	14:28
prepareDinner	11	27	16:52	20:9
getSnack	9	1	0:8	23:39
getDrink	10	2	0:5	23:15

Source: The author.

Table 5.6: Activitties in OrdonezA.

Activity	Occurrences	Average Time (minutes)	From	To
goToBed	14	562	0:42	12:37
useToilet	44	32	0:39	23:54
takeShower	14	7	9:47	12:54
prepareBreakfast	14	107	9:55	12:59
prepareLunch	9	35	14:3	15:45
grooming	51	28	0:7	23:53
watchTV	77	111	0:8	23:53
leaveHouse	14	119	12:44	20:23
getSnack	11	1	13:5	20:38

Source: The author.

Table 5.7: Activitties in OrdonezB.

Activity	Occurrences	Average Time (minutes)	From	To
goToBed	29	371	0:0	12:35
useToilet	93	1	0:8	23:55
takeShower	11	6	9:22	13:53
prepareBreakfast	22	14	8:56	11:51
prepareLunch	13	30	13:37	15:43
grooming	113	3	0:3	23:56
watchTV	116	77	0:18	23:38
leaveHouse	38	138	9:53	22:23
getSnack	47	8	0:36	23:22
prepareDinner	11	11	21:37	23:35

Source: The author.

Table 5.8: Duration for the data collection process for each house.

House	Start	End	Duration in days
houseA	25-Feb-2008 00:19:32	21-Mar-2008 18:25:05	25
houseB	24-Jul-2009 16:46:19	07-Aug-2009 23:47:06	14
houseC	19-Nov-2008 22:49:00	08-Dec-2008 08:15:00	18
ordonezA	28-Nov-2011 02:27:59	11-Dec-2011 21:41:48	13
ordonezB	11-Nov-2012 21:14:00	03-Dec-2012 01:03:59	21

Source: The author.

5.2 Evaluating Activity Recognition

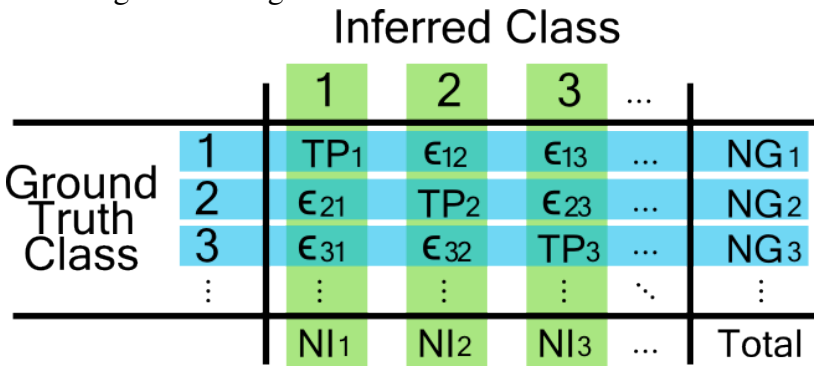
Just as in (SEBBAK et al., 2014) and (MCKEEVER, 2011), we will divide the days in segments of one minute size. This duration is considered long enough to discriminate short activities and short enough to provide high accuracy in the user labeling process (KASTEREN et al., 2008). Sensor values are also discretized so that every sensor provides only one raw value per minute.

Every minute will be labeled with what activity is happening according to the ground truth labels provided by the dataset. Minutes that don't have any ground trith label will be considered as belonging to the "idle" activity. Each method will have access to all sensor information up to the end of each minute and must predict its label.

5.3 Metrics

Popular metrics for activity recognition are Accuracy, Precision, Recall, as seen in (KASTEREN; ALEMDAR; ERSOY, 2011). All those metrics can be defined using confusion matrices, like the one shown in Figure 5.2.

Figure 5.2: A general multiclass confusion matrix.



Source: The author.

Accuracy is simply the percentage of instances correctly classified. So, using the notation of figure 5.2,

$$Accuracy = \frac{\sum_{i=1}^{i} TP_i}{Total}$$

Note that $Accuracy$ is a global measure for the activity recognition process. Precision and recall, on the other hand, are, by definition, measures of performance for a single class. The Precision, for a specific class i, is a measure of how relevant are the instances predicted to be in that class. So, still on the same notation, we have

$$Precision(i) = \frac{TP_i}{NI_i}$$

Recall, on the other hand, represents how many of the instances of that class were correctly classified. So, we have, for a class i,

$$Recall(i) = \frac{TP_i}{NG_i}$$

It is clear that it is possible to achieve perfect $Recall$ without perfect $Precision$ for i, by simply classifying all instances as i. On the other hand, it is possible to achieve a very good $Precision$ with a low $Recall$ by classifying as i only those instances that one is sure of being belonging to i. Since we have a trade-off between $Precision$ and $Recall$, its harmonic mean, the F-Measure, is often used as a metric.

$$FMeasure(i) = \frac{2 \cdot Precision(i) \cdot Recall(i)}{Precision(i) + Recall(i)}$$

In (KASTEREN; ALEMDAR; ERSOY, 2011), $Recall$ and $Precision$ values are averaged to generate a global metric for activity recognition. More formally, for any confusion

matrix, we have, for Q classes,

$$Precision = \frac{1}{Q}\sum_{i=1}^{Q}\frac{TP_i}{NI_i}$$

$$Recall = \frac{1}{Q}\sum_{i=1}^{Q}\frac{TP_i}{NG_i}$$

$$FMeasure = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

Another global measure is Cohen's Kappa, which is the measure of agreement between the ground truth label distribution and the inferred one. It is possible to achieve a high Accuracy score by just betting on the prior ground truth label distribution. In fact, we are interested in performance that surpasses this strategy, and this is given by the Cohen's Kappa. It discounts Accuracy by what would be obtained by change, so we have

$$CohenKappa = \frac{\frac{\sum_{i=1}^{Q} TP_i}{Total} - \sum_{i=1}^{Q}(\frac{NI_i}{Total} \cdot \frac{NG_i}{Total})}{1 - \sum_{i=1}^{Q}(\frac{NI_i}{Total} \cdot \frac{NG_i}{Total})}$$

5.4 Algorithms

In this work, we will compare our method to two of the evidence theory methods studied in chapter 3. They will be (SEBBAK et al., 2014) and (MCKEEVER, 2011), since they are the only evidence theory methods studied that use temporal information, extending evidence in time using multiple fixed windows.

We will also compare our method with other classic machine learning classifiers, such as the Naive Bayes classifier (JOHN; LANGLEY, 1995) and the C4.5 algorithm (QUINLAN, 2014). We will use their WEKA (WITTEN; FRANK, 2005) implementations. Since classes are unbalanced and those models allow for weighting instances, instances will be weighted according to the size of the classes in the training set. That is, the sum of instance weights for each class will be the same, so that, for example, if A happens twice as much as B, a B instance weights two times as much as A.

5.4.1 Features

For the evidence theory methods, we will follow the approach of (SEBBAK et al., 2014) and (MCKEEVER, 2011) and use only sensor raw data as features. For the Naive Bayes and J48 methods, we will use:

- The *ChangePoint* for each sensor. Just as is the case with the evidence theory windows, *ChangePoint* values will be extended in time. For example, once a value of 1 for a sensor is detected, this value will be kept until its time is deemed relevant by the segmentation strategy. If no *Changepoint* event happened for that sensor in segment's reach, the value goes back to the original 0.
- The *LastSensor*, defined as the last sensor to have a *ChangePoint* event.
- The location of the *LastSensor*.
- The time of the day, as defined in Section 4.2.3.

For our approach, we will use semantic information as described in Section 4.3.2. A sensor will be said as having fired for the current situation using the extended *ChangePoint* strategy described above.

5.4.2 Segmentation Strategies

Some activity recognition methods, such as the evidence theory methods described in Chapter 2, include a specific segmentation strategy (such as multiple fixed windows). On the other hand, other methods, such as our proposed approach, are segmentation agnostic. For the methods that do not provide a specific segmentation strategy, we will test the following:

- Single window, with a fixed size measured in minutes. That is, ignore sensor information that happened at more than a fixed amount of minutes ago.
- Segmentation by last sensor location. That is, ignore previous sensor information whenever a sensor in a different location fires.
- One can always argue that a method is superior to another, if only a better segmentation was applied. For this reason, we will also test all methods using the ground truth segmentation.That is, consider for each activity only the sensor information that belongs to it according to the ground truth activity labels.

5.4.3 Configurations for our method

In our method, we will use the DS combination rule, as defined in Equation 2.5. This rule was chose because our model fits the standard DS interpretation for evidence theory, using both the randomly coded messages interpretation and the unreliable truth machine one. For evaluating the combined evidences, we will use the belief function, as defined in Equation 2.1.

5.4.4 Evidence Theory Configurations

In this work, we implemented the evidence theory methods (SEBBAK et al., 2014) and (MCKEEVER, 2011). As those methods mandate, we built a DAG for every dataset, connecting the available sensors to the activities they favor. This was accomplished by, first, automatically generating a full DAG, and later removing all connections that do not conform to common sense.

5.4.4.1 Learning Temporal Aspects

In the chose evidence theory methods, two temporal aspects are relevant:

- Regarding the time duration for the multiple windows, we followed the approach used in (MCKEEVER, 2011) and calculated the maximum time for each activity using its average time in the training set. Of course, following their approach, not all activities have a fixed time window. The *leave house* and *go to bed* activities are left without a fixed time window, being, for every dataset, "no movement or bedroom door" and "no movement or front door", respectively.
- Regarding time cuts, we followed the approach used in (MCKEEVER, 2011) and provided the times restrictions for some activities as can be seen in Table 5.9. They were inferred using common sense only, without any look at the dataset's data. The intervals for the time of the day periods used are the ones introduced in Section 4.2.3. Using time cuts was found to improve performance for some datasets (MCKEEVER, 2011), but it is not mandatory. Thus, we will evaluate the methods both with and without time cuts.

Table 5.9: The time cuts for the evidence theory models. The intervals for the time of the day periods used are the ones introduced in Section 4.2.3.

Activity	**Time**
goToBed	Night
eatBreakfast	Morning
eatDinner	Evening
prepareDinner	Evening
prepareLunch	Afternoon
leaveHouse	not Night

Source: The author.

5.4.4.2 Rules and Decision Strategies used

We will also vary the combination rule used, choosing from the TBM, DS, EWS and MPY combination rules. Those rules are defined in Equations 2.4, 2.5 and 2.6, respectively. For the evaluation procedure, we will use the same approach proposed in (MCKEEVER, 2011). Thus, we will use the Pignisitic Probability, as defined in equation 2.7, and will break ties deciding for the least uncertain activities.

5.4.4.3 Handling Unlabeled Minutes

As we have seen in chapter 3, both (SEBBAK et al., 2014) and (MCKEEVER, 2011) do not deal with idle minutes. Those minutes are considered "hard to infer" in (MCKEEVER, 2011) and are simply skipped. But, in order to test those methods in a real world scenario, we have no option but to consider those idle minutes. This is not an easy task taking into account the way those methods deal with activity recognition. For example, what are the sensors and contexts related to the "idle" activity? Since the idle activity is composed of all unlabeled minutes, we can expect it to include all the sensors related to untracked activities, but also the absence of all sensor firings, since inactive moments (such as relaxing) will also be unlabeled in most datasets.

So, we arrive at a conundrum. We cannot know what activities lie in the unlabeled minutes, so we cannot choose its mappings in the design stage without looking at the data. For sure, we can infer "idle" when no other activity has a window. Additionally, we can link everything that appears in the training set as "idle" and train the network accordingly. In this work, we will evaluate both options.

5.4.4.4 Perfect Segmentation

In order to be fair, we will also test the perfect segmentation approach for the evidence theory methods. Since they use multiple fixed windows, we will represent the perfect segmentation approach as ending all windows when a ground truth activity finishes. This way, we ensure no previous activity evidence will enter the window of the current activity.

5.4.4.5 Other Aspects

Another important variation regards how sensor information is represented. For example, if a sensor is inactive for two minutes, does this counts as the sensor sending two "off" signals, one at each minute, or just a single one at the first minute? This is not made explicit in (SEBBAK et al., 2014) and (MCKEEVER, 2011), and thus, for fairness sake , we will test both approaches.

5.5 Evaluation Strategy

The leave one day out (LOO) strategy is a cross validation strategy where each instance is evaluated by a model trained with all the remaining ones. LOO ensures in a relatively unbiased classifier, as we are using almost all of the data for training (JAPKOWICZ; SHAH, 2011).

When performing activity recognition, there are some aspects one should take into account. In the case of activity recognition, we should not build a classifier for each minute, but for each day. It is easy to see why. If we created one classifier per minute, the task would be too easy, for the previous minute and the next would be in the training set, and they would most likely belongs to the same activity as the current one. Moreover, evaluating a whole day at a time is also more realistic, since many activities occur in a day and we are interested in how the method segments them too.

Another aspect we must take into account is the order of the tests. Minutes should be evaluated chronologically, so we must evaluate first day 1, followed by day 2 and so on. An important aspect of this process is that information from the past day can drive activity in the current one. For example, if the subject left the house yesterday, today (s)he should be considered out of the house until (s)he returns. This means that, if a sensor indicated that the front door was opened last night and this activity was not ended by the model when midnight was reached, we must keep this segmented information and provide it to our newly trained classifier.

We can expect classes to be unbalanced in all days, but this is a feature of human behavior, and therefore of real life activity recognition. This class imbalance can generate a high variance in day accuracy. Thus, we will not average day performance, since we do not wish to lose information by considering only means. Instead, for each dataset, we will calculate our metrics on the confusion matrix generated by unifying all day matrices.

5.5.1 Statistical Analysis

In this work, we will compare multiple algorithms on multiple datasets. Since each dataset follows a different distribution, we are, in fact, comparing multiple classifiers in multiple domains. So, we will follow the approach proposed in (DEMŠAR, 2006), and prefer non-parametric statistical ranking tests. For comparing multiple classifiers in multiple datasets, we will use the Friedman's test (FRIEDMAN, 1937) as described in (DEMŠAR, 2006), with the Nemenyi post-hoc test (NEMENYI, 1962) for finding the different group after they are deemed statistically significant by the Friedman's test.

In more detail:

- The Friedman test is a non-parametric alternative of the repeated-measures ANOVA. It ranks the algorithms for each dataset separately, so that the best algorithm has a rank of 1, a second of 2 and so on. If there are ties, average ranks are assigned. For example, if we have only two algorithms and they are tied, both will receive a ranking of 1.5. After ranks are assigned, the test calculates, for each algorithm, the average rank in all datasets. This average rank is used to calculate the Friedman statistic, which is

$$\chi_F^2 = \frac{12N}{k(k+1)}[\sum_{j=1}^{k} R_j^2 - \frac{k(k+1)^2}{4}]$$

  where N is the number of datasets, k is the number of algorithms and R_j is the average ranking of algorithm j over all datasets. Using this statistic, and a table for critical values, which can be found in (JAPKOWICZ; SHAH, 2011), we can determine if the test rejects the null hypothesis, which is that all algorithms perform the same.
- Once the null hypothesis is rejected, we can use the Nemenyi post-hoc test for finding which pairs of algorithms are different. The performance of two algorithms is considered statistically significantly different if their average rank over all datasets differs more than a critical difference

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}}$$

where N is the number of datasets, k is the number of algorithms and q_α is a constant relative to the desired p value. A table for q_α can be found in (DEMŠAR, 2006).

Each of the evaluated algorithms can work in different configurations. For example, the evidence theory methods can use different combination rules, decision strategies and other parameters. The same is true for the other methods, such as Naive Bayes, where we can choose between different segmentation strategies. For this reason, we will rank the possible models for each algorithm and find its best ranked configuration. We will compare their best versions so that we have a global overview of the results.

6 RESULTS AND DISCUSSION

6.1 Experimental Results

In order to evaluate our approach and compare it to others, we have performed experiments in five real world datasets, as explained in Chapter 5. Each approach has different configurations, producing different results. For brevity, we will show the results for the best configuration for each model.

Regarding the best models for the studied evidence theory approaches had the following characteristics:

- They used time cuts.
- They used the DS combination rule.
- They consider as evidence only changes in sensor values.
- They recognized the idle activity only when no other activity was possible.

For the Naive Bayes and J48 models, the best configuration was using fixed windows of size 15 and 12, respectively. On the other hand, for our approach, the best results were achieved using the perfect segmentation strategy. Having it as an option, one could think every model should performs the best using it. This is not the case for our results. For example, in the evidence theory models, the perfect segmentation did not improve the results compared to the best imperfect configuration. Also, the Naive Bayes and J48 models perform better on relatively small window sizes than on the perfect segmentation.

The perfect segmentation may be outranked because

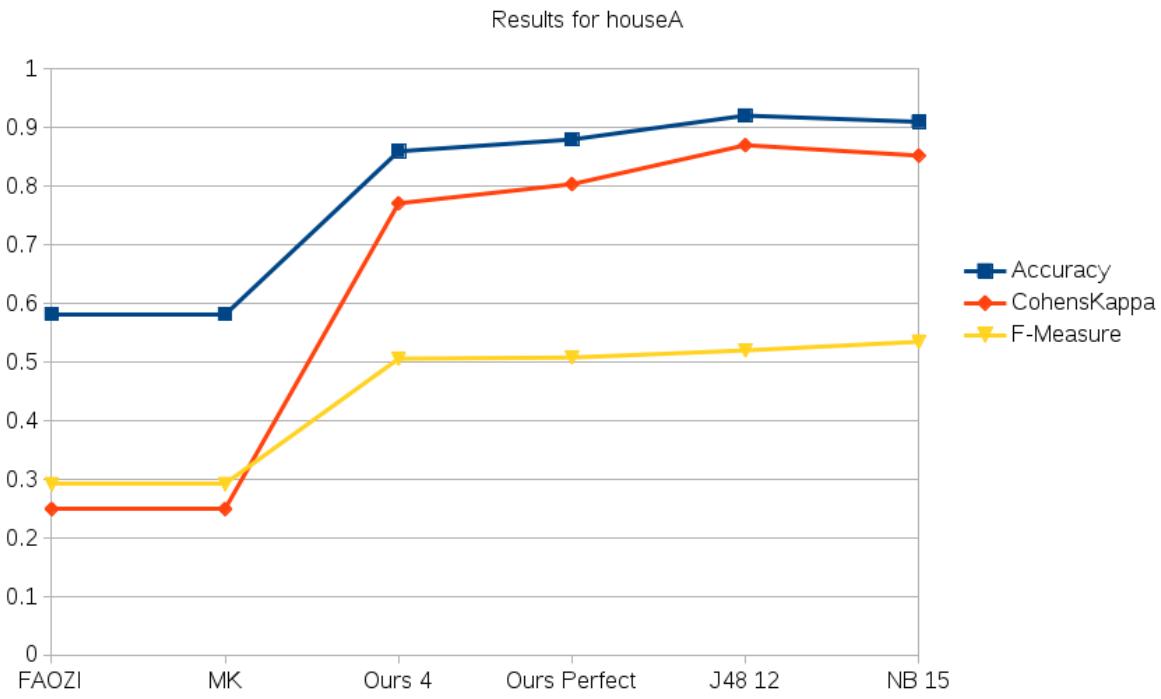
- Ground truth labeling is done by humans, and so is not always perfect. For example, if an activity is annotated as starting later than it actually did, relevant sensor firings will be missing in the "perfect" segmentation.
- The minutes right before the activity is starting are relevant to it.

On the other hand, our approach assumes that the segmentation is perfect, since it interprets an unexplainable sensor firing for an activity as either operation in the unreliable for that sensor or absence of said activity. Since one may not have a good segmentation algorithm, it is also interesting to know what is the best performance for our model. In our experiments, this was achieved under a fixed window segmentation strategy of size 4.

In the *houseA* dataset, as can be seen in Figure 6.1, the studied evidence theory method perform worse than our approach, which performs close, but still worse than the best J48 and

Naive Bayes models.

Figure 6.1: Performance for the best models in the *houseA* dataset.



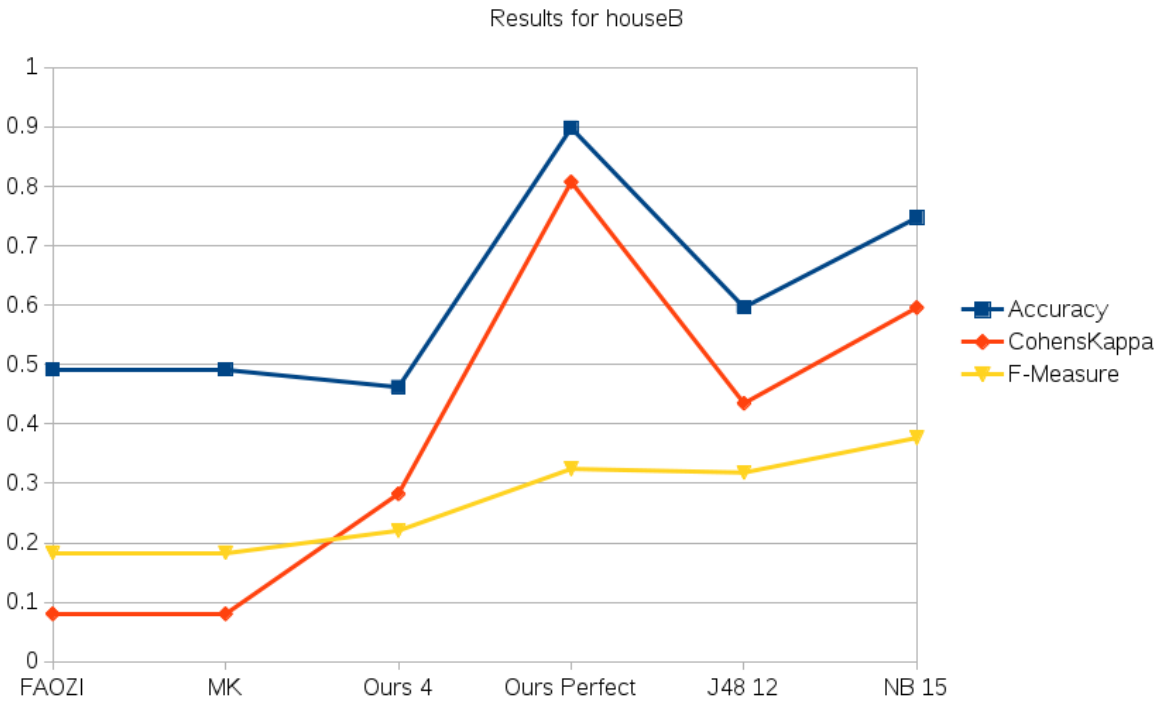
Source: The author

Regarding *houseB*, as we can see in Figure 6.2, our approach in the perfect segmentation has the highest Accuracy and Cohen's Kappa, but not the best average class F-Measure. Still, results follow the same pattern found in *houseA*: Even with fixed window segmentation, our approach performs better than previous evidence theory based approaches.

Results in the *houseC* also follow the pattern found in *houseA*, with our approach better than the previous evidence theory ones, but still in a close second place if we are using the Cohen's Kappa metric. But, in this house, no model performs exceptionally well, since even Accuracy values are below 80%. This may be due to many factors, one of which may be a lesser quality in the ground truth information for *houseC*, but is also related to its activities. For example, some activities, such as *take medication* don't have specific sensors related to it and we do not know beforehand its specific schedule.

In the *ordonezA* dataset, as we can see in Figure 6.4, again our method clearly outperforms the evidence theory approaches, both in the perfect segmentation and in the fixed window configurations. The perfect segmentation configuration also has the best results for F-Measure, but places third in the other two metrics. Regarding the *ordonezB* dataset, our method is again

Figure 6.2: Performance for the best models in the *houseB* dataset.

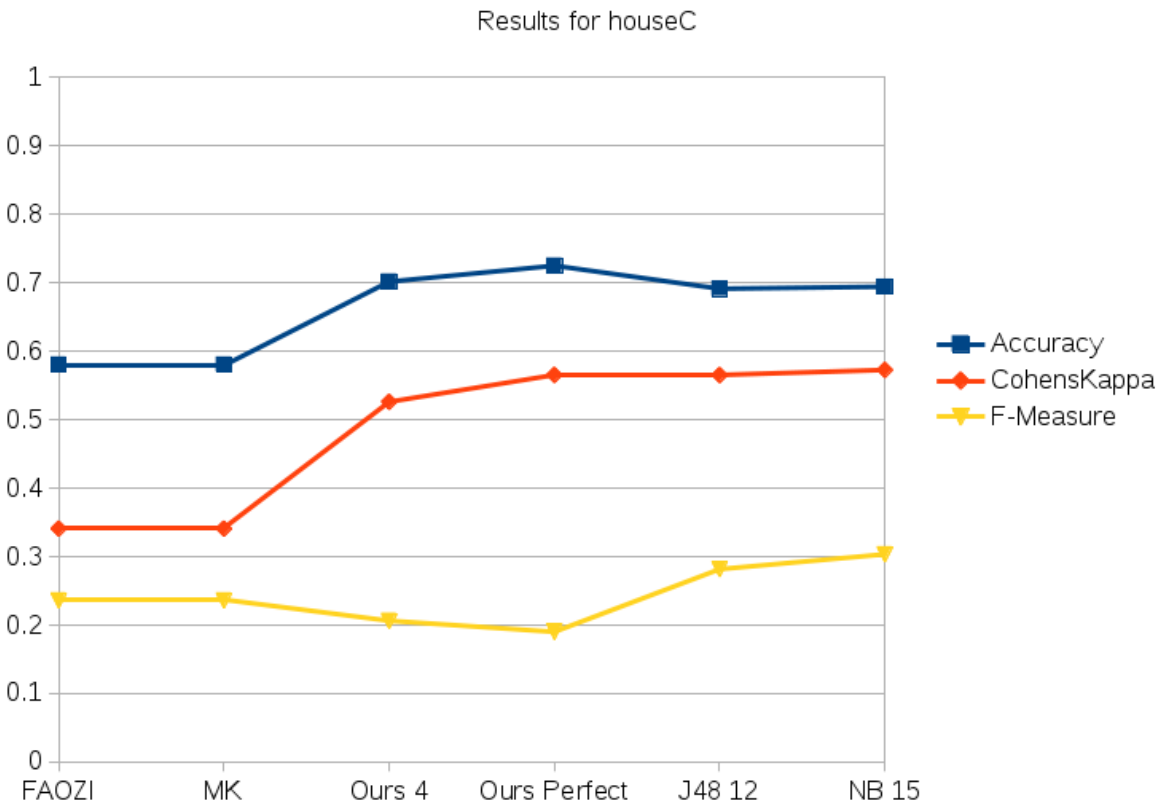


Source: The author

superior to the evidence theory ones, as can be seen in Figure 6.5. This time it has the best Accuracy and Cohen's Kappa values, but not the best F-Measure. It is interesting to notice that the mean class F-Measure does not account for class frequencies, as Cohen's Kappa does. On the other hand, Cohen's Kappa penalizes less the model for missing completely a small class, for example. Still, it is important to discover if the difference between the models is statisti-

cally significant. The Friedman test uses the mean ranking of a classifier in multiple domains to determine if at least one of the models performs statistically significantly better than some other. As we can see in Table 6.1, we can reject the null hypothesis regarding Accuracy and attest that at least one of the classifiers is better than the others with $p = 0.003233$. As proposed in (DEMŠAR, 2006), we will apply the Nemenyi post-hoc test to determine which of the classifiers in question perform differently. This can be visualized in the critical difference plot of 6.6. This diagram can be interpreted thus: the more to the left, the better ranked the model is. Additionally, if a model's line is not connected to another model's line, those models are said to be statistically significantly different. This means that the post-hoc test could not reject the null hypothesis, even if the Friedman test could. As explained in (DEMŠAR, 2006), this can happen, as the Nemenyi test is not as powerful as the Friedman test.

Figure 6.3: Performance for the best models in the *houseC* dataset.



Source: The author

Table 6.1: Result for the Friedman's rank sum test applied over accuracy values.

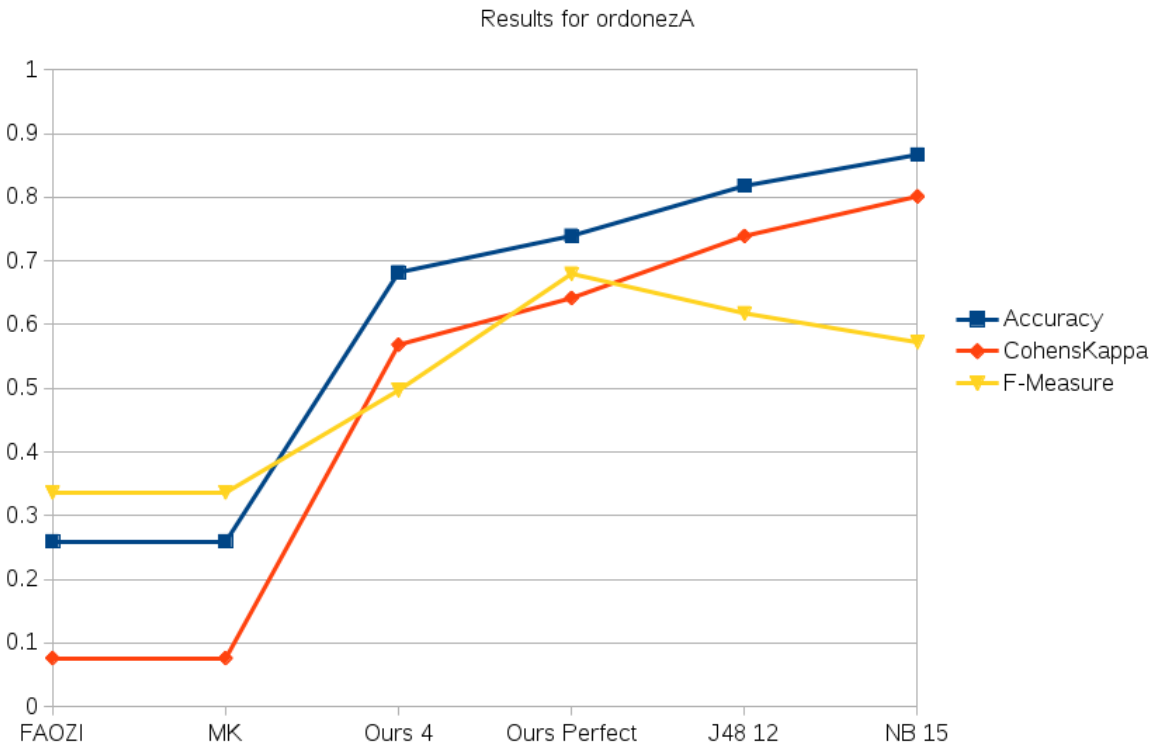
Friedman's rank sum test	**Result**
χ^2_F	17.7814
p-value	0.003233

Source: The author

Friedman test is a non-parametric ranking test, which means it makes no assumption on the distribution for the data. It relies only in the ranking of observations, and thus we may exchange the metric from Accuracy to Cohen's Kappa, since the latter deals with class imbalance. In this case, applying the Friedman test also rejects the null hypothesis, as we can see in Table 6.2. Now, one model is detected as statistically significantly different by the post-hoc test, as can be seen in figure 6.7. Namely, we can state that the Naive Bayes method is statistically significantly better than both evidence theory models. Still, no other pair of models is found to be statistically significantly different.

Still, different metrics may show different aspects of the results. For example, if we are interested in overall class balanced results, we can use the average class F-Measure metric. As

Figure 6.4: Performance for the best models in the *ordonezA* dataset.



Source: The author

Table 6.2: Result for the Friedman's rank sum test applied over Cohen's Kappa values.

Friedman's rank sum test	**Result**
χ^2_F	19.8562
p-value	0.00133

Source: The author

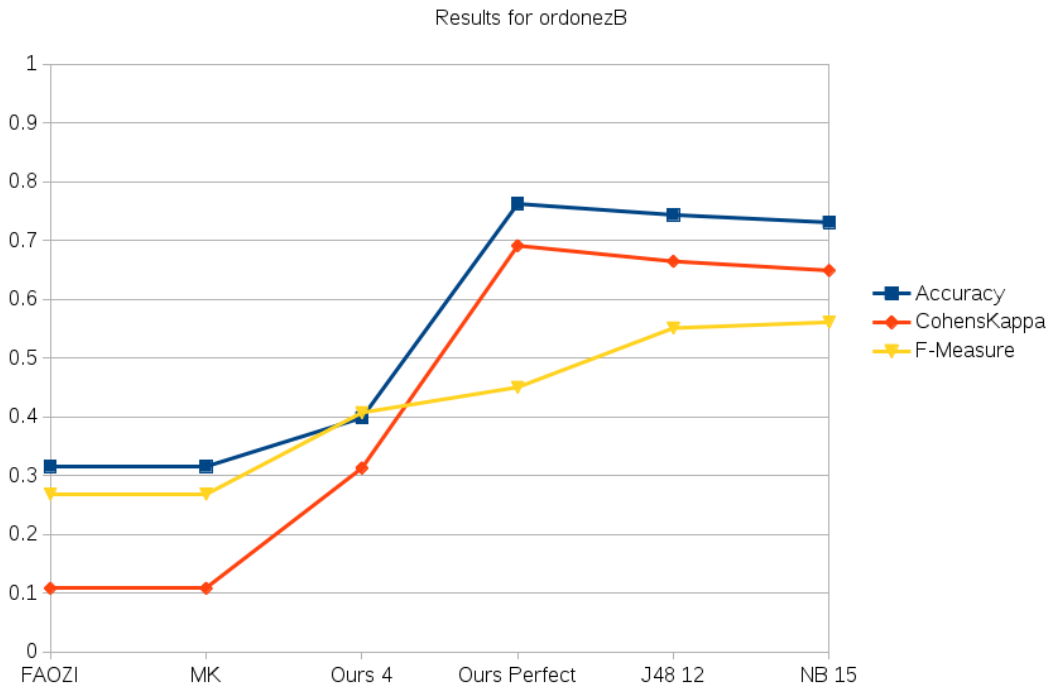
we can see in Table 6.3, results are still statistically significantly different for the models. Again the critical difference diagram, shown in Figure 6.8, detects only that the Naive Bayes model is statistically significantly better than the two evidence theory ones.

Table 6.3: Result for the Friedman's rank sum test applied over the average class F-Measure values.

Friedman's rank sum test	**Result**
χ^2_F	17.6485
p-value	0.003421

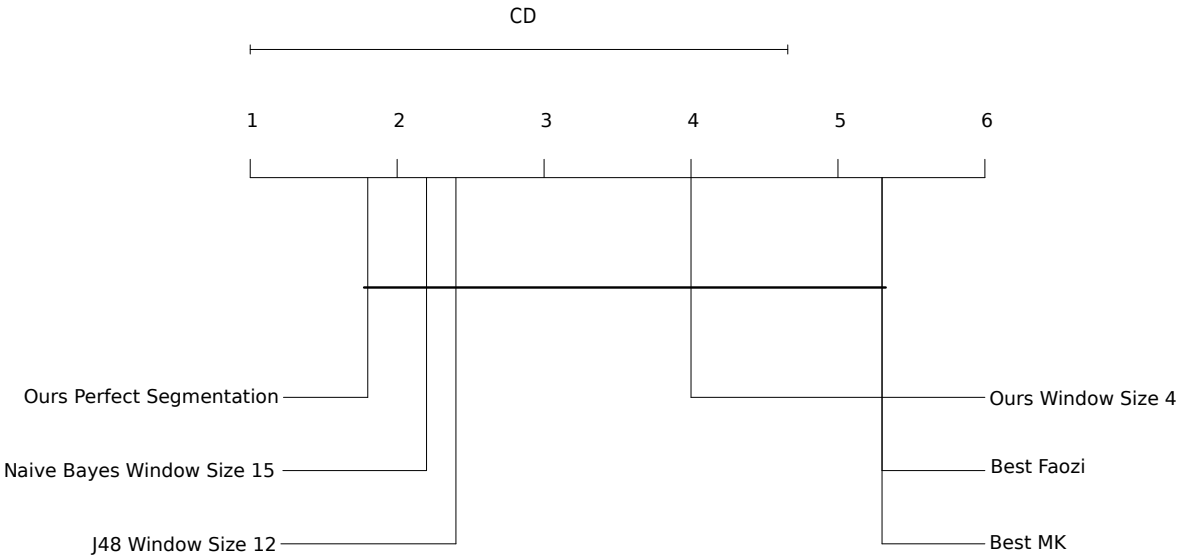
Source: The author

Figure 6.5: Performance for the best models in the *ordonezB* dataset.



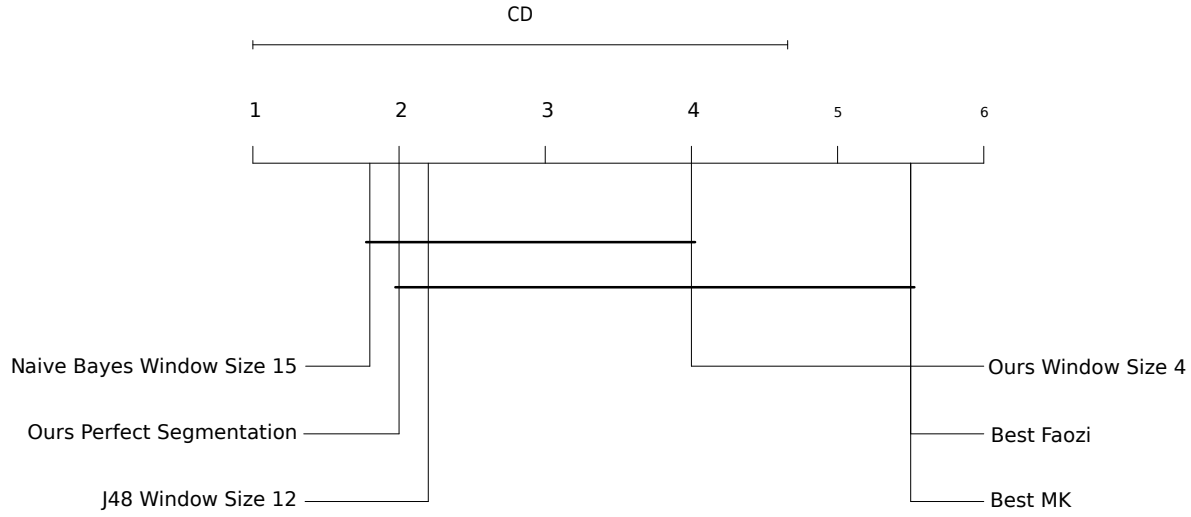
Source: The author

Figure 6.6: The critical difference plot for the results using accuracy and $p = 0.05$. This diagram can be interpreted thus: the more to the left, the better ranked the model is. Additionally, if a model's line is not connected to another model's line, those models are said to be statistically significantly different.



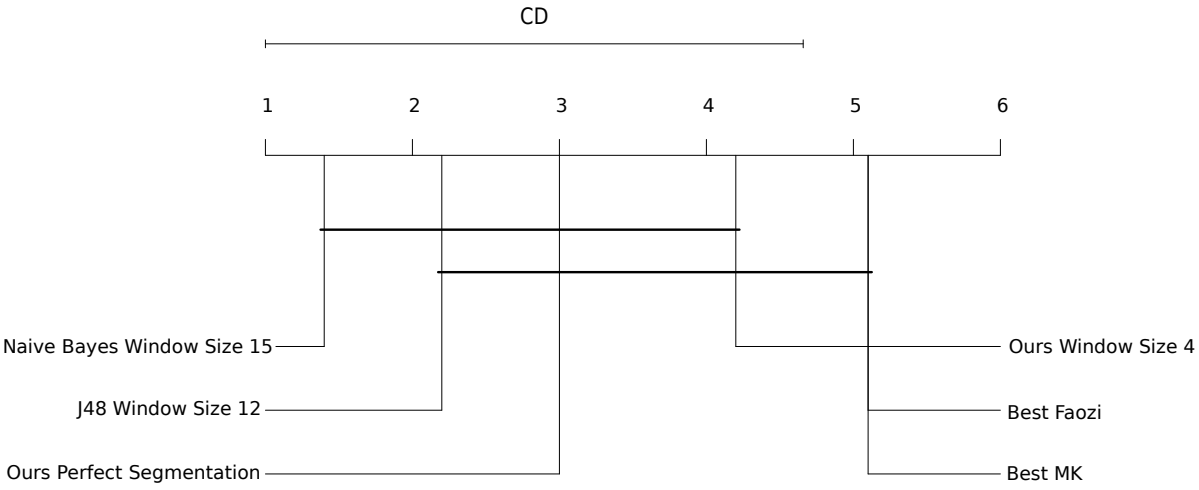
Source: The author

Figure 6.7: The critical difference plot for the data using Cohen's Kappa and $p = 0.05$. This diagram can be interpreted thus: the more to the left, the better ranked the model is. Additionally, if a model's line is not connected to another model's line, those models are said to be statistically significantly different.



Source: The author

Figure 6.8: The critical difference plot for the data using the average class F-Measure and $p = 0.05$. This diagram can be interpreted thus: the more to the left, the better ranked the model is. Additionally, if a model's line is not connected to another model's line, those models are said to be statistically significantly different.



Source: The author

6.2 Discussion

In this work we have presented a hybrid method for activity recognition that combines evidence theory with ontological reasoning. We have shown that, using ontological modeling, our model that can automatically adapt to different houses where a different set of activities is performed. That is, it can semantically interpret sensors, generate the current situation and evaluate it according to an activity description.

When the segmentation is done properly, it surpasses the current evidence theory models, achieving a level comparable to standard machine learning methods. Lacking a good segmentation approach, our method still arguably performs as good as or better than the evidence theory approaches.

Still, as we have seen in Section 6.1, more tests are needed in order to better compare all the methods.

Regarding evidence theory models, our approach has as its advantages:

- It can automatically interpret sensor information, doing away with the need to manually specify a DAG for each specific dataset. Moreover, new sensors can be added during the activity recognition process without the need manually place them in the DAG.
- It provides a clear ontological interpretation for context information, which allows for a richer definition of what can be evidence for or against some activity.
- It provides a clear evidence theory interpretation for the model and the activity recognition process itself.
- Its performance is arguably better, even when using fixed windows.

Regarding the other machine learning methods, our approach has as its advantages:

- It can automatically interpret sensor information, making it possible for new sensors to be included without the need to retrain the model, since we are weighting ontological assertions instead of raw sensor data.
- Its best case scenario performance is close or equivalent to the best machine learning models. Even better, this holds true using the leave one day out strategy, which provides plenty of training data.
- It provides a clear ontological explanation for each context information, and how it relates to each activity. This is a defining feature of knowledge-driven methods, which focus on explaining why a specific activity is detected as happening, instead of taking a more black box approach. This sentence may sound false, if we don't define carefully what

"explains" means. One may say, for example, that a tree-based method, such as J48, provides explanations for its predictions. A human certainly can understand its decisions, interpreting the path each instance takes in the model's tree. This does not mean each of the instance's features has a defined semantic, or that their relation the classes is well defined. If a user derives any interpretation for the path an instance classification takes in a tree model, this interpretation is present in the user's mind, and not fully encoded in the model.

6.3 Future Work

Even more, this work is only the first attempt at an ontology-driven evidence theory model for activity recognition. Thus, many relevant aspects can be improved. This includes temporal information, such as the sequencing of activities. Moreover, the model can be extended to recognize activities for multiple subjects, as well as multiple activities being performed in the same time by one subject. The semantic information described in the current model is also very simple, based solely on simple context information. Future work also includes developing even richer models, perhaps using datasets that include more ground truth information than the current activity the subject is performing.

REFERENCES

ARNDT, M. et al. Performance evaluation of ambient services by combining robotic frameworks and a smart environment platform. **Robotics and Autonomous Systems**, v. 61, n. 11, p. 1173 – 1185, 2013. ISSN 0921-8890. Ubiquitous Robotics. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S0921889013000614>.

ASHBURNER, M. et al. Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. **Nature genetics**, Nature Publishing Group, Department of Genetics, Stanford University School of Medicine, California, USA. cherry@stanford.edu, v. 25, n. 1, p. 25–29, may 2000. ISSN 1061-4036. Available from Internet: <http://dx.doi.org/10.1038/75556>.

CANTERA-FONSECA, J.; LEWIS, R. Delivery context ontology. **W3C Working Draft.[en línea] Disponible en: http://www. w3. org/TR/2009/WD-dcontology-20090616/[consulta 2010, 24 de abril]**, 2009.

CHEN, H.; FININ, T.; JOSHI, A. An ontology for context-aware pervasive computing environments. **The Knowledge Engineering Review**, Cambridge Univ Press, v. 18, n. 03, p. 197–207, 2003.

CHEN, H.; FININ, T.; JOSHI, A. The soupa ontology for pervasive computing. In: **Ontologies for agents: Theory and experiences**. [S.l.]: Springer, 2005. p. 233–258.

CHEN, L.; NUGENT, C. D.; WANG, H. A knowledge-driven approach to activity recognition in smart homes. **Knowledge and Data Engineering, IEEE Transactions on**, IEEE, v. 24, n. 6, p. 961–974, 2012.

DEMŠAR, J. Statistical comparisons of classifiers over multiple data sets. **The Journal of Machine Learning Research**, JMLR. org, v. 7, p. 1–30, 2006.

DUL: The DOLCE+DnS Ultralite ontology. <http://www.loa.istc.cnr.it/ontologies/DUL.owl>. Accessed: 2015-12-10.

FABIEN, C. et al. Video based technology for ambient assisted living: A review of the literature. **Journal of Ambient Intelligence and Smart Environments (JAISE)**, IOS Press, v. 3, n. 3, p. 253–269, 2011.

FRIEDMAN, M. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. **Journal of the American Statistical Association**, Taylor & Francis, v. 32, n. 200, p. 675–701, 1937.

GANGEMI, A. et al. Sweetening ontologies with dolce. In: GóMEZ-PéREZ, A.; BENJAMINS, V. (Ed.). **Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web**. Springer Berlin Heidelberg, 2002, (Lecture Notes in Computer Science, v. 2473). p. 166–181. ISBN 978-3-540-44268-4. Available from Internet: <http://dx.doi.org/10.1007/3-540-45810-7_18>.

GANGEMI, A. et al. Sweetening wordnet with dolce. **AI magazine**, v. 24, n. 3, p. 13, 2003.

GLIMM, B. et al. Hermit: an owl 2 reasoner. **Journal of Automated Reasoning**, Springer, v. 53, n. 3, p. 245–269, 2014.

GUIZZARDI, G. **Ontological foundations for structural conceptual models**. [S.l.]: CTIT, Centre for Telematics and Information Technology, 2005.

HELAOUI, R. et al. Towards activity recognition using probabilistic description logics. **Activity Context Representation: Techniques and Languages**, v. 12, p. 05, 2012.

HERVÁS, R.; BRAVO, J.; FONTECHA, J. A context model based on ontological languages: a proposal for information visualization. **J. UCS**, v. 16, n. 12, p. 1539–1555, 2010.

HITZLER, P. et al. Owl 2 web ontology language primer. **W3C recommendation**, World Wide Web Consortium W3C, v. 27, n. 1, p. 123, 2009.

HONG, X. et al. Evidential fusion of sensor data for activity recognition in smart homes. **Pervasive and Mobile Computing**, v. 5, n. 3, p. 236 – 252, 2009. ISSN 1574-1192. Pervasive Health and Wellness Management. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S157411920800045X>.

HORRIDGE, M.; BECHHOFER, S. The owl api: A java api for owl ontologies. **Semantic Web**, v. 2, n. 1, p. 11–21, 2011.

JAPKOWICZ, N.; SHAH, M. **Evaluating learning algorithms: a classification perspective**. [S.l.]: Cambridge University Press, 2011.

JOHN, G. H.; LANGLEY, P. Estimating continuous distributions in bayesian classifiers. In: CONFERENCE ON UNCERTAINTY IN ARTIFICIAL INTELLIGENCE. **Proceedings of the Eleventh conference on Uncertainty in artificial intelligence**. [S.l.]: Morgan Kaufmann Publishers Inc., 1995. p. 338–345.

KASTEREN, T. L. van; ALEMDAR, H.; ERSOY, C. Effective performance metrics for evaluating activity recognition methods. **ARCS 2011**, VDE VERLAG GmbH, 2011.

KASTEREN, T. V.; ENGLEBIENNE, G.; KRÖSE, B. J. Transferring knowledge of activity recognition across sensor networks. In: **Pervasive computing**. [S.l.]: Springer, 2010. p. 283–300.

KASTEREN, T. V. et al. Accurate activity recognition in a home setting. In: INTERNATIONAL CONFERENCE ON UBIQUITOUS COMPUTING. **Proceedings of the 10th international conference on Ubiquitous computing**. [S.l.]: ACM, 2008. p. 1–9.

KOHLAS, J.; MONNEY, P.-A. **Representation of evidence by hints**. [S.l.]: Springer, 2008.

LEFORT, L. et al. Semantic sensor network xg final report. **W3C Incubator Group Report**, v. 28, 2011.

LESTER, J. et al. A hybrid discriminative/generative approach for modeling human activities. In: **IJCAI**. [S.l.: s.n.], 2005. v. 5, p. 766–772.

LUTZ, C.; WOLTER, F.; ZAKHARYASCHEV, M. Temporal description logics: A survey. In: **TIME**. [S.l.: s.n.], 2008. p. 3–14.

MCKEEVER, S. **Recognising Situations Using Extended Dempster-Shafer Theory**. Thesis (PhD) — National University of Ireland, Dublin, 2011.

MCKEEVER, S. et al. Activity recognition using temporal evidence theory. **Journal of Ambient Intelligence and Smart Environments**, IOS Press, v. 2, n. 3, p. 253–269, 2010.

MEDITSKOS, G. et al. Ontology patterns for complex activity modelling. In: **Theory, Practice, and Applications of Rules on the Web**. [S.l.]: Springer, 2013. p. 144–157.

MEDITSKOS, G.; DASIOPOULOU, S.; KOMPATSIARIS, I. Metaq: A knowledge-driven framework for context-aware activity recognition combining sparql and owl 2 activity patterns. **Pervasive and Mobile Computing**, Elsevier, 2015.

MILLER, G. A. Wordnet: a lexical database for english. **Communications of the ACM**, ACM, v. 38, n. 11, p. 39–41, 1995.

MURPHY, C. K. Combining belief functions when evidence conflicts. **Decision support systems**, Elsevier, v. 29, n. 1, p. 1–9, 2000.

NEMENYI, P. Distribution-free multiple comparisons. In: INTERNATIONAL BIOMETRIC SOC 1441 I ST, NW, SUITE 700, WASHINGTON, DC 20005-2210. **Biometrics**. [S.l.], 1962. v. 18, n. 2, p. 263.

NGUYEN, H. T. On random sets and belief functions. **Journal of Mathematical Analysis and Applications**, Elsevier, v. 65, n. 3, p. 531–542, 1978.

NILES, I.; PEASE, A. Towards a standard upper ontology. In: INTERNATIONAL CONFERENCE ON FORMAL ONTOLOGY IN INFORMATION SYSTEMS. **Proceedings of the international conference on Formal Ontology in Information Systems-Volume 2001**. [S.l.]: ACM, 2001. p. 2–9.

NOELKER, L. S.; BROWDIE, R. Sidney katz, md: A new paradigm for chronic illness and long-term care. **The Gerontologist**, Oxford University Press, p. gnt086, 2013.

OBERLE, D. et al. Dolce ergo sumo: On foundational and domain models in the smartweb integrated ontology (swinto). **Web Semantics: Science, Services and Agents on the World Wide Web**, Elsevier, v. 5, n. 3, p. 156–174, 2007.

OKEYO, G.; CHEN, L.; WANG, H. Combining ontological and temporal formalisms for composite activity modelling and recognition in smart homes. **Future Generation Computer Systems**, v. 39, n. 0, p. 29 – 43, 2014. ISSN 0167-739X. Special Issue on Ubiquitous Computing and Future Communication Systems. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S0167739X14000399>.

OKEYO, G. et al. A hybrid ontological and temporal approach for composite activity modelling. In: TRUST, SECURITY AND PRIVACY IN COMPUTING AND COMMUNICATIONS (TRUSTCOM). **Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on**. [S.l.]: IEEE, 2012. p. 1763–1770.

ORDÓNEZ, F. J.; TOLEDO, P. de; SANCHIS, A. Activity recognition using hybrid generative/discriminative models on home environments using binary sensors. **Sensors**, Multidisciplinary Digital Publishing Institute, v. 13, n. 5, p. 5460–5477, 2013.

PEARL, J. **Probabilistic reasoning in intelligent systems: networks of plausible inference**. [S.l.]: Morgan Kaufmann, 2014.

PRESTES, E. et al. Towards a core ontology for robotics and automation. **Robotics and Autonomous Systems**, v. 61, n. 11, p. 1193 – 1204, 2013. ISSN 0921-8890. <ce:title>Ubiquitous Robotics</ce:title>. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S0921889013000596>.

PREUVENEERS, D. et al. Towards an extensible context ontology for ambient intelligence. In: **Ambient intelligence**. [S.l.]: Springer, 2004. p. 148–159.

PROTéGé. <http://protege.stanford.edu/>. Accessed: 2015-12-10.

PRUD'HOMMEAUX, E.; SEABORNE, A. et al. Sparql query language for rdf. **W3C recommendation**, v. 15, 2008.

QUINLAN, J. R. **C4. 5: programs for machine learning**. [S.l.]: Elsevier, 2014.

RIBONI, D. et al. Is ontology-based activity recognition really effective? In: INTERNATIONAL CONFERENCE ON PERVASIVE COMPUTING AND COMMUNICATIONS WORKSHOPS (PERCOM WORKSHOPS). **Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference on**. [S.l.]: IEEE, 2011. p. 427–431.

RODRÍGUEZ, N. D. et al. A survey on ontologies for human behavior recognition. **ACM Computing Surveys (CSUR)**, ACM, v. 46, n. 4, p. 43, 2014.

ROGGEN, D. et al. Opportunistic human activity and context recognition. **Computer**, v. 46, n. 2, p. 36–45, Feb 2013. ISSN 0018-9162.

ROY, P. C. et al. A possibilistic approach for activity recognition in smart homes for cognitive assistance to alzheimer's patients. In: **Activity Recognition in Pervasive Intelligent Environments**. [S.l.]: Springer, 2011. p. 33–58.

SEBBAK, F. et al. Dempster–shafer theory-based human activity recognition in smart home environments. **annals of telecommunications - annales des télécommunications**, Springer Paris, v. 69, n. 3-4, p. 171–184, 2014. ISSN 0003-4347. Available from Internet: <http://dx.doi.org/10.1007/s12243-013-0407-2>.

SHAFER, G. Constructive probability. **Synthese**, Springer, v. 48, n. 1, p. 1–60, 1981.

SHAFER, G. Belief functions and parametric models. **Journal of the Royal Statistical Society. Series B (Methodological)**, JSTOR, p. 322–352, 1982.

SHAFER, G. et al. **A mathematical theory of evidence**. [S.l.]: Princeton university press Princeton, 1976.

SHELKEY, M.; WALLACE, M. Katz index of independence in activities of daily living (adl). **The Gerontologist**, v. 10, n. 1, p. 20–30, 1998.

SMETS, P. Analyzing the combination of conflicting belief functions. **Information Fusion**, Elsevier, v. 8, n. 4, p. 387–412, 2007.

SMETS, P.; KENNES, R. The transferable belief model. **Artificial intelligence**, Elsevier, v. 66, n. 2, p. 191–234, 1994.

STUDER, R.; BENJAMINS, V.; FENSEL, D. Knowledge engineering: Principles and methods. **Data & Knowledge Engineering**, v. 25, n. 1–2, p. 161 – 197, 1998. ISSN 0169-023X. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S0169023X97000566>.

TAPIA, E. M. **Sensor installation examples**. <http://courses.media.mit.edu/2004fall/mas622j/04.projects/home/>. Accessed: 2015-12-01.

THE Ullman Triangle: conceptualizations are abstractions of reality, represented in a language. <http://www.pilod.nl/wiki/Boek/BrandtDaniele>. Accessed: 2015-12-01.

TOLSTIKOV, A. et al. Comparison of fusion methods based on dst and dbn in human activity recognition. **Journal of Control Theory and Applications**, South China University of Technology and Academy of Mathematics and Systems Science, CAS, v. 9, n. 1, p. 18–27, 2011. ISSN 1672-6340. Available from Internet: <http://dx.doi.org/10.1007/s11768-011-0260-7>.

VILLALON, M. P. et al. A context ontology for mobile environments. CEUR-WS, 2010.

WANG, X. H. et al. Ontology based context modeling and reasoning using owl. In: ANNUAL CONFERENCE ON PERVASIVE COMPUTING AND COMMUNICATIONS WORKSHOPS. **Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on**. [S.l.]: IEE, 2004. p. 18–22.

WITTEN, I. H.; FRANK, E. **Data Mining: Practical machine learning tools and techniques**. [S.l.]: Morgan Kaufmann, 2005.

YAGER, R. R. Entropy and specificity in a mathematical theory of evidence. **International Journal of General System**, Taylor & Francis, v. 9, n. 4, p. 249–260, 1983.

YAU, S. S.; LIU, J. Hierarchical situation modeling and reasoning for pervasive computing. In: INTERNATIONAL WORKSHOP ON COLLABORATIVE COMPUTING, INTEGRATION, AND ASSURANCE. **Software Technologies for Future Embedded and Ubiquitous Systems, 2006 and the 2006 Second International Workshop on Collaborative Computing, Integration, and Assurance. SEUS 2006/WCCIA 2006. The Fourth IEEE Workshop on**. [S.l.]: IEEE, 2006. p. 6–pp.

YE, J.; DOBSON, S.; MCKEEVER, S. Situation identification techniques in pervasive computing: A review. **Pervasive and mobile computing**, Elsevier, v. 8, n. 1, p. 36–66, 2012.