On Leveraging Statistical and Relational Information for the Representation and Recognition of Complex Human Activities

Inauguraldissertation
zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften
der Universität Mannheim

vorgelegt von
Rim Helaoui
aus Tunis, Tunesien

Mannheim, 2016

Dekan: Professor Dr. Heinz Jürgen Müller, Universität Mannheim, Germany

Referent: Professor Dr. Heiner Stuckenschmidt, Universität Mannheim, Germany

Korreferent: Professor Dr. Daniele Riboni, University of Cagliari, Italy

Tag der mündlichen Prüfung: 19.02.2016

Abstract

Machine activity recognition aims to automatically predict human activities from a series of sensor signals. It is a key aspect to several emerging applications, especially in the pervasive computing field. However, this problem faces several challenges due to the complex, relational and ambiguous nature of human activities. These challenges still defy the majority of traditional pattern recognition approaches, whether they are knowledge-based or data-driven. Concretely, the current approaches to activity recognition in sensor environments fall short to represent, reason or learn under uncertainty, complex relational structure, rich temporal context and abundant common-sense knowledge. Motivated by these shortcomings, our work focuses on the combination of both data-driven and knowledge-based paradigms in order to address this problem. In particular, we propose two logic-based statistical relational activity recognition frameworks which we describe in two different parts.

The first part presents a Markov logic-based framework addressing the recognition of complex human activities under realistic settings. Markov logic [RD06] is a highly flexible statistical relational formalism combining the power of first-order logic with Markov networks by attaching real-valued weights to formulas in first-order logic. Thus, it unites both symbolic and probabilistic reasoning and allows to model the complex relational structure as well as the inherent uncertainty underlying human activities and sensor data. We focus on addressing the challenge of recognizing interleaved and concurrent activities while preserving the intuitiveness and flexibility of the modelling task. Using three different models we evaluate and prove the viability of using Markov logic networks for that problem statement. We also demonstrate the crucial impact of domain knowledge on the recognition outcome.

Implementing an exhaustive model including heterogeneous information sources comes, however, at considerable knowledge engineering efforts. Hence, employing a standard, widely used formalism can alleviate that by enhancing the portability, the re-usability and the extension of the model. In the second part of this document, we apply a hybrid approach that goes one step further than Markov logic network towards a formal, yet intuitive conceptualization of the domain of discourse. Concretely, we propose an activity recognition framework based on log-linear description logic [NNS11], a probabilistic variant of description logics. Log-linear description logic leverages the principles of Markov logic while allowing for a formal conceptualization of the domain of discourse, backed up with powerful reasoning and consistency check tools. Based on principles from the activity theory [KN12], we focus on addressing the challenge of representing and recognizing human activities at three levels of granularity: operations, actions and activities. Complying with real-life scenarios, we assess and discuss the viability of the proposed framework. In particular, we show the positive impact of augmenting the proposed multi-level activity ontology with weights compared to using its conventional weight-free variant.

Zusammenfassung

Die maschinelle Aktivitätserkennung hat das Ziel menschliche Aktivitäten mit Hilfe einer Reihe von Sensorsignalen vorauszusagen. Dies ist eine Schlüsseltechnologie aus derer sich zahlreiche Anwendungsfelder ergeben welche vor allem im Bereich des "Pervasive Computing" zu finden sind. Durch die komplexe, verflochtene und mehrdeutige Natur von menschlichen Aktivitäten gibt es viele Herausforderungen zu lösen. Aktuelle Lösungsansätze häufig versagen in der Modellierung, im Reasoning oder Lernen unter Unsicherheit, unter komplexen relationalen Strukturen, unter reichem zeitlichen Kontext und unter dem Vorhandensein von umfangreichem Domänenwissen Wissen. Unsere Arbeit überwindet diese Defizite in dem wir den Fokus auf die Kombination von datenbasierten und wissensbasierten Ansätzen legen. Diese Arbeit ist in zwei Teile unterteilt, welche zwei logikbasierte, statistik-relationale Aktivitätserkennungsframeworks vorstellen.

Der erste Teil ist ein Markov-Logik basiertes Framework welches die Erkennung von komplexen menschlichen Aktivitäten unter realistischen Bedingungen thematisiert. Markov-Logik [RD06] ist ein flexibler statistisch-relationaler Formalismus der die Stärken von Prädikatenlogik und Markov-Netzwerken vereint, in dem reelle Gewichte mit prädikatenlogischen Formeln verbunden werden. Folglich werden symbolisches und probabilstisches Reasoning kombiniert und erlauben es sowohl die komplexe relationale Struktur als auch die dazugehörige Unsicherheit, die menschlichen Aktivitäten und Sensordaten zugrunde liegt, zu modellieren. In dem Teil wenden wir uns der Herausforderung zu, verschachtelte und gleichzeitig stattfindende Aktivitäten zu erkennen. Unter der Verwendung von drei verschiedenen Modellen wird gezeigt, dass Markov-Logik Netzwerken für dieses Problem angewendet werden können. Ebenfalls wird der entscheidende Einfluss von Domänenwissen auf das Ergebnis des Erkennungsprozesses demonstriert.

Der zweite Teil dieses Dokuments behandelt einen hybriden Ansatz, der verglichen mit Markov-Logik Netzwerken einen Schritt weiter in Richtung einer formalen und dennoch intuitiven Konzeptualisierung der Domäne geht. Das erhöht die Portabilität, die Wiederverwendbarkeit und die Erweiterbarkeit des Modells. Das Aktivitätserkennungsframework basiert auf log-linearer Beschreibungslogik [NNS11], einer probalistischen Variante von Beschreibungslogiken. Die log-lineare Beschreibungslogik verwendet die Ansätze von Markov-Logik während gleichzeitig eine formale Konzeption der Domäne, ergänzt mit weitreichenden Ableitungs-und Konsistenzüberprüfungsmöglichkeiten, erlaubt wird. Basierend auf Prinzipien der Aktivitätstheorie [KN12] werden die Repräsentation und die Erkennung von menschlichen Aktivitäten auf drei Granularitätsebenen durchgeführt: Operationen, Aktionen und Aktivitäten. Die Realisierbarkeit des vorgeschlagenen Frameworks wird anhand lebensnaher Szenarien beurteilt und diskutiert. Wir werden insbesondere den positiven Einfluss der Erweiterung der vorgeschlagenen multi-level Aktivitätsontologie mit Gewichten mit der konventionellen Variante ohne Gewichte vergleichen.

Acknowledgement

I used to imagine this special moment quite often during the last few years. I used to close my eyes and see myself writing this section of *my* dissertation which I have finally completed. Quickly and sadly, though, I had to go back to the reality and retrieve my determination and courage to continue.

That strength wouldn't have existed without the valuable support of all those that helped me through this path. Thank you Heiner Stuckenschmidt for letting me join your chair, for your guidance and your exemplary understanding. You were more than "my professor", you were a mentor that coached me during the PhD process and inspired me beyond it. Thank you Mathias Niepert for patiently putting me on the right path and skillfully showing me how to move forward on my own, I owe you much. My sincere thanks to you, Daniele Riboni, for having offered me the chance to collaborate with you. Your initiative opened up new opportunities, and taught me extensively. You repeatedly played a role model to me. Finally, thanks to Claudio Bettini for his encouragement and feedback at my first PhD forum.

Day after day, my colleagues at the chair of artificial intelligence have been a great asset to me. I thank you all for the enjoyable coffee breaks and interesting discussions as well as for sharing with me my up and downs. Special thanks go to Christian Meilicke, my first office mate, for the sincere and funny moments we shared, thanks to Cäcilia Zirn for being my comforting friend. Thanks to Jan Noessner for your valuable help. Thanks to Daniel Fleichhacker for your generous support. I am also very thankful to Sebastian Skotthof for his continuous and benevolent help. Thanks to Johannes Knopp, my office mate and to Anne Schlicht for your help. Finally thanks to Laura Palmi for her valuable contribution during my collaboration with the University of Milan. A heartily and warm thank you to Stephanie Keil for always being there to share my joy and alleviate my anxiety. Your support was priceless.

Outside my daily office environment, I was also very lucky to be overwhelmed with love and support from my friends and family. Thank you all! Thanks to Marwa Dridi, Najoua Jouini and especially to my closest friend Maha Chourabi for always standing beside me, no matter what. Thanks to Sherine Rady for those special moments we shared and the mutual encouragement. Thanks to my friend and cousin Syrine Kerfai for the steady emotional support that helped me continue.

The deepest and biggest thank you goes to my parents and my sisters. I cannot find the words to express my gratitude for your unconditional love and phenomenal motivation; thank you my dad, Abderazzak Helaoui, my mom Samira Ben Mougou Helaoui and my sisters: Eya and Asma Helaoui. You are an unrivaled blessing.

A big dose of thank you goes to my loving husband, Amir Rmaile, for his precious support, understanding and care throughout the whole process and with all its ups and downs. Finally, thanks to my little treasure and most precious, my daughter Sarah Rmaile, for being there. Your smiles and innocence were my primary source of strength. I love you.

Contents

List of Figures

List of Tables

Introduction

"The most profound technologies are those that disappear.
They weave themselves into the fabric of everyday life
until they are indistinguishable from it."

—Mark Weiser

1 Introduction

The genesis of ubiquitous computing, also called pervasive computing, can be traced back to 1991 when Mark Weiser coined this term in his seminal paper "*The Computer for the 21st Century*" [Wei91]. There, he explains his vision of creating environments saturated with computing and communication capabilities, yet "calmly" integrating them with human users until they disappear into the background. Whereas this perception was ahead of time then, it has recently received growing attention due to the latest technology advances and application demand. Many sensor technologies that were out of reach in 1991 are now viable low-cost commercial products. Small, light-weight, low-power wired and wireless sensing technologies are making substantial progress. This enabled the creation of mobile, wearable sensing modalities as well as embedded sensing infrastructures for so called "smart spaces".

Wearable sensors together with smart environments have pushed the research contributions in ubiquitous computing from simple low-level sensor data processing to more sophisticated high-level data integration. A particularly relevant focus has been shifted to context reasoning and context-aware applications, a crucial aspect to fulfil Mark Weiser's vision. Usually, context is defined as "any information that can be used to characterize the situation of an entity" [Dey01]. A context-aware application is defined by the same authors as "an application that uses the context of an entity to modify its behaviour to best meet the context of the user". Thus, information that is directly acquired from sensor data can be seen as low-level context (e.g time, temperature, humidity, luminosity, etc.), while high-level context is information that is inferred from the low-level one.

Numerous solutions for a number of real-world problems have become increasingly reliant on one particular aspect of high-level context, namely human activity. The ability to automatically recognize what the user is doing and what they will do

next enables to reason about their current and upcoming needs. Inferring such context information augments the available services with *reactive* and even *proactive* assistance. Also referred to as "plan recognition", "goal recognition" or "intent recognition", the field of activity recognition plays a crucial role in a wide spectrum of applications. These applications range from the health-care domain, where user's health status and lifestyle are monitored and assessed contentiously (e.g. [HKAK10], [ACRV13]), to fall and anomaly detection (e.g. [LSH+09]), fitness tracking and promotion of healthy lifestyle (e.g. [CMT+08]), smart houses with context-aware services (e.g. [CCTK13]), robotics(e.g. [VV08]), security and surveillance (e.g. [LSPZ08]), pedestrian traffic (e.g. [MD14]), entertainment and video games (e.g. [KHL+13]), to task assistance and safety instructions in car manufacturing (e.g. [SRO+08]).

Sensor-based activity recognition: Activity recognition systems are generally either based on the use of visual sensing facilities or that of emerging light-weight sensors such as wearable and environmental dense sensors. Our work is classified into the second category to which we will refer as *sensor-based activity recognition* in contrast to *vision-based activity recognition*. An exhaustive review about vision-based activity recognition can be found in the work of Aggarwal and Ryoo [AR11].

Wearable sensors generally are used to capture the user's motions, location, vital signs, environmental data, and their interaction with surrounding objects. This latter also requires environmental sensors which are embedded in closed spaces such as "smart houses", "smart hospitals" and "smart meeting rooms". Besides object-interaction, these sensors usually capture the user's motion and their indoor location.

Among the most widely used wearable sensors we distinguish accelerometers and gyroscopes which measure proper acceleration and orientation respectively. These two sensors were successfully used to recognize physical movement of the user [LL13]. Other prominent wearable sensors are GPS receivers which are intensively used in outdoor activity recognition by tracking users' itinerary [FCRS13]. Vital signs sensors such as electrocardiogram (ECG), skin conductance sensors(SC) are emerging trends especially in health and fitness monitoring applications [ACRV13]. Finally, common to both groups of sensors are RFID tags and readers. Just like inertial sensors are indispensable for body locomotion and movement recognition, equipping surrounding items with RFID tags while wearing RFID reader is also compulsory for the recognition of specific activities such as "preparing a sandwich" and "cleaning" [MVC+10].

Complex human activities: In the activity recognition community, the term "activity" does not universally designate the same concept. Indeed, human activities can vary through a wide spectrum of granularity levels. It ranges from very short and simple gestures such as "move hand" to complex composite activities and situations such as "Shopping". Despite this irregularity, there is an implicit distinc-

tion between two big categories: low-level and high-level activities. Low-level activities are also called "actions", "atomic activities" or "simple activities". They generally denote simple ambulatory behaviour having a short and stable duration. These low-level activities are commonly atomic and can not be broken into finer grained components. Thus, they can not be interrupted and are always performed in sequential manners. Examples include "walking", "running", "sitting" and "open the door". Several statistical approaches and well established machine learning algorithms have proven to infer low-level activities with ease [LL13]. These focus on the problem of dealing directly with noisy sensor data to discover and extract interesting patterns that can be mapped into activities.

Convinced by the viability of the current approaches, more research efforts have recently appeared in order to extend the recognition from low-level to high-level activities. This extended recognition problem is best understood as a specific case of *abduction*, i.e. reasoning to the best explanation. The main idea states that "if the user is carrying out a high-level activity Y they would perform the sequence of actions X, and we if observe X, we may postulate that they are executing Y".

Whereas this formulation facilitates the understanding of the high-level activity recognition problem, the proposed approaches to solve are still facing several challenges due to the following aspects. High-level activities are typically composed of a sequence of low-level activities over an extended duration. These can be performed in different manners and in different sequences. The sequences are inherently variant in terms of their time span and temporal order of their components. For example, the activity "put the table" might be performed in one of the following simplified sequences (1) "open drawer", "fetch a plate", "fetch a spoon", "close the drawer", "walk", "put down plate", "put down spoon", or (2) "open drawer", "fetch a plate", "walk", "put down plate", "walk", "fetch a spoon", "close the drawer", "walk", "put down spoon". Besides such possible deviations, high-level activities can be interleaved, concurrent and even aborted. For instance, the subject could initiate the activity of "putting the table" then interrupts its sequence to "go to the bathroom" before resuming it.

Another crucial difference between low-level and high-level activities is the relevant impact of contextual information on the recognition performance. Context information can cover manifold aspects such as location, temporal features (e.g. weekdays versus weekends), environmental conditions (e.g. weather, luminosity, humidity, noise), and surrounding objects. Apart from wearable sensors, and especially accelerometers, Manzoor et al. [MVC+10] show, in their systematic study, that environmental sensors embedded in different objects are also mandatory for the recognition of specific high-level activities like "cleaning up".

In general, frameworks proposed to recognize high-level activities from lightweight sensor data can be classified in flat models, which attempt to recognize high-level activities directly from sensor data, and hierarchical ones which first infer low-level activities then use them to recognize high-level activities. The latter method has been shown to better address the recognition task [LC11].

Based on this basic two-levels conceptualization, this work focuses on the recognition of high-level activities. For a comprehensive review about the recognition of low-level activities the reader is invited to check the work of Preece et. al [PGK+09]. Throughout this thesis, we will employ the terms *activity*, *composite activity* and *complex activity* interchangeably to denote high-level activities. A more formal categorization of the different types of composite activities is detailed in the next Chapter. There, we present a multi-level structure of high-level activities founded on activity theory [KN12].

2

Trends in Human Activity Modelling and Recognition Approaches

The problem of automatically recognizing complex activities have been approached by various methods which can be generally classified as data-driven, knowledge-driven or hybrid. In this chapter we first identify the chief requirements of a realistic activity recognition system and associate them with the advantages and disadvantages of each paradigm. Then we provide a review of the existing approaches for sensor-based complex activity recognition following the same classification.

Using large amounts of collected sensor data, **data-driven approaches** employ mining and machine learning techniques to create *probabilistic activity models*. Based on these models, new sensor data can be classified into the corresponding human activities. Contrastively, **knowledge-driven approaches** basically rely on domain-related expertise to specify formal activity models using knowledge representation and engineering techniques. The activity models basically encode common-sense and domain knowledge about activities. Activity recognition is then realised by applying *logical reasoning* on the constructed models whenever sensor data is available.

These two paradigms have complementary strengths and weaknesses when applied to activity recognition. The major limitations are detailed in terms of requirements and desired aspects of a realistic activity recognition systems:

- **Addressing uncertainty**: uncertainty is a crucial aspect in activity modelling and recognition. Uncertain knowledge affects several levels of the activity recognition process: From noisy sensor data to the ambiguity of their interpretation. Whereas data-driven approaches provide great flexibility and allow to control different alternatives, knowledge-driven approaches

are static and can not handle uncertain data.

- **Ability to address complex activities**: modelling complex human activities and their underlying relational structure usually requires highly expressive description formalisms. Unlike knowledge-driven approaches, data-driven ones are not flexible enough to capture and model such complex relationships between different entities.

- **Ability to handle complex (temporal) relationships**: a special but essential type of complex relationships in activity models consists in temporal information. Temporal context is crucial for context-aware application in general and activity recognition in particular. It can significantly improve the recognition rates [Dav13]. Apart from few advanced statistical approaches such as Skip Chain Conditional Random Fields (SC-CRF) [HY08] and Emerging Patterns (EP) [GWT$^+$09], the majority of data-driven methods fail to recognize non-sequential activities such as concurrent and interleaved ones [KHC10]. Contrastingly, knowledge-driven approaches are very well-suited to model highly complex relationships between the model's entities. However, due to the lack of uncertainty support, they are inadequate for modelling and manipulating the required temporal information.

- **Portability and re-usability**: activity recognition applications are meant to be flexible in terms of their settings. They should support portability across environments and users. Not only the same activity is often carried out in different manners by different subjects, but it could also be performed in several ways by the same subject. Thus, it is difficult for data-driven approaches to collect adequate data sufficient to handle this variability. Thus, reusing the same model under different settings remains a challenge for these data-driven methods.

- **Ease of integration of background knowledge and rich context data**: the inherent structure and common-sense underlying our daily activities are a crucial aspect for their automatic recognition. For instance, certain activities are known to normally happen under a particular context, e.g. the activity of "brushing teeth" usually takes places in the bathroom in the morning after "waking up" and at the evening before "sleeping". Such trivial commonsense knowledge can easily be introduced in knowledge-driven approaches yet might not necessarily be captured by data-driven approaches, due to the lack of a sufficiently large training set.

- **Extensibility**: activity recognition applications require dynamic systems which support the easy extension of the activity models. Being strongly dependent on a given dataset, data-driven approaches would necessitate new data as well as the creation of a new model in order to support additional activities. Knowledge-driven approaches, however, are easily extensible through inserting new rules for instance.

- **Declarative and intuitive modelling**: comprehensible activity models strongly facilitate their interpretation, extension and application. While knowledge driven approaches are usually declarative and easy to understand, data-driven approaches, on the opposite, are less intuitive.
- **Cold-start problem**: activity recognition systems are supposed to perform well immediately. Since data-driven approaches require substantial dataset, they usually suffer from data sparsity and the "cold start" problem.

2.1 Data-driven approaches to activity recognition

Most state-of-the-art activity recognition systems rely on probabilistic models with supervised learning paradigms. Notable examples of these approaches employ Hidden Markov Models([PFKP05], [BPPW09], [LC11]), naïve Bayes [TIL04], dynamic Bayesian networks [vKK07] and conditional random fields([NDHC10], [VVL07]).

Hidden Markov models (HMM) are one of the most popular choices to address activity recognition. HMM are directed graphical models that aim at inferring the states a sequence of hidden variables $(a_1, ..., a_n)$ from an input of a sequence of observations $(o_1, ..., o_n)$. In activity recognition, the hidden states correspond to the human activities while the observation refer to the sensor data. For the sake of tractability, each sensor observation is assumed to be only dependent on the activity from the same time slice, and each activity is assumed to only depend on the previous one. HMM are **generative models**: in order to determine the most probable sequence of activities given the sensor observation, the joint probability $p(o, a)$ is maximized.

Due to their inflexible structure, HMM have serious limitations in presenting multiple interacting activities and modelling long-range dependencies [KHC10]. Representing relational information and arbitrary dependencies is, thus, difficult. In the best case, this would require to propositionalize the domain which results in a combinatorial increase in the number of variables and model's parameters [SK12]. To relax these strict independence assumptions, some researchers have applied dynamic Bayesian networks to recognize human activities [INK+09]. Despite their flexibility compared to HMM, both models perform only well under unrealistic settings where activities are performed in laboratory conditions and follow the same sequences [SZC13]. Addressing natural scenarios with interleaved and concurrent activities remains a challenge for these approaches [KHC10], [SZC13].

Besides generative probabilistic models, researchers have investigated **discriminative models** such as conditional random fields (CRF). CRF are undirected graphs allowing for arbitrary dependency relationships among the observed and hidden variables sequences. Unlike HMM, which rely on Bayes rule to estimate the distribution over hidden states from observations, CRFs directly represent the *conditional distribution* over hidden states *given* the observations. Thus, the most probable state sequence of the hidden variables is inferred by directly maximizing the

conditional probability $p(a|o)$ rather than maximizing the joint probability $p(a, o)$.

Even if linear-chain CRF outperform HMM in several activity recognition experiments ([NDHC10], [VVL07]), both models still share some weaknesses. Especially, they are unable to handle changes in activities [SZC13] and are inflexible in supporting arbitrary dependencies in the input space [GK06]. This is a particularly heavy limitation due to the relevance of the inherently rich and long-range inter- and intra-activity temporal relationships underlying human activity sequences.

Being more suitable for purely sequential data [KHC10], these techniques have been extended in different ways as a step towards supporting more sophisticated relational temporal and atemporal information. Examples include Skip-chain CRF (SCCRF) [HY08], interleaved HMM (IHMM) [MBK08], logical HMM (LoHMM) [NBT$^+$08] and CRF for logical sequence [GK06].

Hence, despite being well suited for simple activities, data-driven techniques, in general, have a number of shortcomings when applied to the recognition of complex high-level activities as summarised above. Especially in terms of portability, extensibility, and support for complex relational information and common-sense knowledge. Consequently, it is at best troublesome to add and acquire further contextual information to these models.

2.2 Knowledge-driven approaches to activity recognition

Most human activities generally involve a regular set of objects. For instance, the activity "dish-washing" implies opening the "dishwasher door", putting "washable dishes" into the "dishwasher", putting the "dishwasher detergent", closing the "dishwasher door" then selecting and starting the washing program. Another example is the activity "wash hands" which consists in interacting with the "water tab", the "soap" and finally the "towel". Nonetheless, depending on the subject's habits, available equipment and lifestyle, there might be some deviations even in such structured activities. "dish-washing", for example, might be performed manually by interacting with the water tab in the kitchen, the the dishes and the dish detergent.

Knowledge-driven approaches rely on such inherent common-sense and prior knowledge about human activities in order to create reusable formal activity models. Prior knowledge about activities of daily living (ADL) is especially rich and usually covers different contextual features such as specific objects, locations and times.

Compared to the data-driven paradigm, little research has been performed in knowledge-driven activity recognition [OCS12]. Among those, we note a domination of logic-based formalisms. Founded on existing knowledge representation and engineering techniques, the recognition task is solved by logical inference applied to axioms and rules specified in the activity model and the generated domain theory. The inference step is supported by a reasoning process which outputs the activities that semantically explain the given sensor observations.

In the context of plan recognition, the first attempts to develop an expressive logical framework could be traced back to the work of Schmidt [SSG78]. However, the proposed approach does not handle uncertain data and was never applied to real sensor data. Logic-based approaches have recently received more an more interest in the activity recognition community.

A major strength of this paradigm has been highlighted by Chen and Nugent [CNM+08] through their extended temporal reasoning framework based on **event calculus (EC)**. Event calculus (EC) was originally introduced by Kowalski and Sergot [KS86] as a logic programming formalism for representing events and their effects. The *events* are the origin of any state change in the domain. The states are referred to as *fluents* and can be seen as properties of the domain at a specific time point. The effect of events on fluent as well as the temporal relations between them are determined by *predicates*. In the context of activity recognition, events correspond to the sensor activations and deactivations, also called sensor events. The state of the domain's entities such as the objects the subject is interacting with or the states of the sensor themselves, are represented by the fluents. Simple and compound activities are inferred by deductive reasoning using a set of axioms stating how and when the truth holds based on causal relations of predicates. Despite its flexibility and its powerful temporal reasoning, the proposed framework was only validated with a simplistic scenario of "making tea". Also, the absence of a sound probabilistic reasoning seriously limits its applicability. A number of event calculus dialects have sprung up since Kowalski and Sergot's original paper. The majority use a subset of the full event calculus, proposed by Shanahan [Sha99]. Interestingly, some of the recently emerging dialects tried to combine EC with probabilistic reasoning [SPVA11].

Using manually designed rules to define human activities in terms of sensor observations and required temporal constraints, Cirillo et al. [CLPS09] propose a temporal reasoning framework (OMPS) to address activity recognition. The rules encode different temporal relationships based on the restricted Allen's interval Algebra [All83] such as "during", "started by" and "finished by". For instance, the activity "taking lunch" is defined as an activity taking place "during" the afternoon, is "started by" activity "cooking" and "finished by" the activity "eating". Given the sensor observations, the temporal constraints are synchronized. If the imposed requirements do not lead to a propagation failure, then the corresponding hypothesis holds. Thus, the framework offers significant support to address rich temporal context. However, due to the deterministic nature of the rules, the approach is too static to handle real-life scenario and their inherent uncertainty. In a recent extension of this work [PCD+13], the authors propose to relax their method by reasoning with multiple hypothesis instead of one.

Earlier attempts employing Allen's interval algebra to reason about complex temporal relationship in the context of smart houses can be found in the works of Augusto and Nugent [AN04] and that of Jakkula and Cook [JCC07]. The first combine active databases with temporal reasoning to define ECA (Event-Condition Action) rules. The authors define composite activities as a straightforward compo-

sition of primitive events. Their use of temporal reasoning is, thus, only for defining the precondition of the rules rather than recognizing complex activities. Similarly, Jakkula and Cook [JCC07] focus on mining temporal constraints between sensor events such as "Tv is on" and "light is on" and use them to detect abnormal event sequences violating the constraints without addressing the challenge of the recognition of complex activities.

Another distinguishable line of research explores the use of description logics (DLs) [BCM+03] to model human activities and ontological reasoning to recognize them. This trend has recently gained increasing attention due to its formal way to represent heterogeneous sensor and context data, as well as activities in a unified framework with well-structured terminology. This makes ontology-based systems understandable, shareable, and reusable by both humans and machines. Ontology-based approaches describe activities by linking them to constraints or sensor and context data through *properties*. They are especially well suited for elegantly modelling and reasoning with different abstraction levels of human activities. The recognition process matches the observed data to the required conditions defining each activity and infers the corresponding one. Motivated by the eminent role of context information and common-sense knowledge in modelling and recognizing human activities, several activity and context ontologies have been proposed. The majority of these ontologies, however, are used for data integration purposes, for learning unknown objects or for categorizing terms [CHN+12]. An extensive survey about ontologies for human activity representation is provided by Rodriguez et. al [RCLCF14]. Among the minority that explicitly conceptualize activities and their interrelationships in a unified framework, are the works of Chen et al. [CN09], [CNW12], that of Riboni et.al [RB11] and Springer et. al [ST09]. Despite the highly expressive DLs employed, the models include very simple activities such "whether a ringing person is authorized to enter or not" [ST09] and are hardly capable of addressing the temporal context of the activities, nor do they support uncertain knowledge, which impairs the system's performance. Overcoming the first two limitations is the focus of the work of Saguna et. al [SZC11] and that of Okeyo et. al ([OCS12], [OCW13]) which combines ontological and temporal knowledge modelling formalisms to create composite activity models.

In addition to the large modelling efforts required by knowledge-based approaches, this paradigm fails in addressing the imperative of handling uncertain knowledge. Recently, several researchers have endeavoured to explore extensions and combinations of both knowledge-based and data-based approaches in order to meet the requirements of realistic activity recognition systems. We refer to these methods as **hybrid approaches**.

2.3 Hybrid approaches to activity recognition

Different hybrid approaches have been recently applied to activity recognition. They usually attempt to extend knowledge driven approaches probabilistic aspects

([RB09], [FAT11], [HRN+12], [CNO14], [YSD14]) or to incorporate complex relational modelling techniques in data-driven approaches ([GK06], [KRR06], [MBK08], [HY08]), [NBT+08], [HNS11b], [MMvO+12], [SK12]). Being closely related to the proposed methods in this thesis, an accurate description of these works is provided in the related work sections.

3 Preliminaries

3.1 Machine recognition of human activities

As introduced above, machine activity recognition aims to automatically predict the activities of a human beings from a series of sensor signals.

In this chapter, we provide preliminary knowledge required for subsequent chapters. This includes a formal definition of the activity recognition problem preceded with a brief theoretical definition of human activities based on **activity theory**. The chapter also covers introductory background about graphical models as a basis for several statistical relational approaches. Finally, the last section formulates the research problem addressed in this thesis.

3.1.1 Human activities and activity theory

In a conceptual grounding originally developed by the Russian psychologist Aleksei Leontiev around 1930, human activities are defined as a relationship between an acting human being and an entity existing in the world called "object" [KN12]. This interaction is determined by a particular motive to meet certain need(s) of the subject, where objects are not necessarily physical entities as long as they exist in the world. This **object-orientedness** of activities is the first principle of activity theory.

The theoretical concept of activities presents their structure in a three-level hierarchy bridging the *Why*, the *What* and *How* respectively as depicted in Figure 3.1. The top level corresponds to the **activity** itself, which is driven by a *motive* in order to respond to a particular need such as "having a meal". Such an activity is realized through a series of conscious **actions**. These steps should lead to the goals required to achieve the *object motive*. A subject is typically aware of the goals they

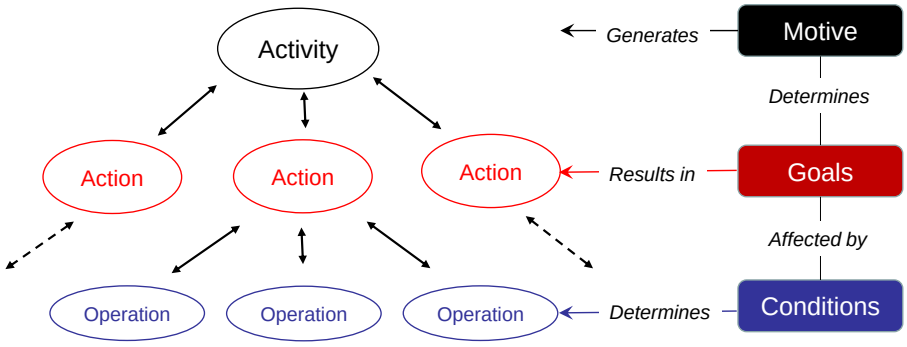


Figure 3.1: The hierarchical structure of human activities. Activities are composed of actions, which are, in turn, composed of operations. As indicated by the bi-directional bold arrows, the upper layer indicates the motive, the second is driven by goals and the lowest refers to conditions. [KN12]

want to attain. These are often decomposed into sub-goals and sub-sub-goals and so on. For instance, the activity of "*having a meal*" usually implicates the action of "*preparing meal*" which in turn might involve "*cutting vegetables*" and so forth. This decomposition goes on until it reaches the lowest layer, where actions turn into sub-conscious automatic **operations** such as "*opening the fridge*"(see Figure 3.1).

Nonetheless, activity theory does not provide a taxonomy of human activities. In fact, it is a descriptive and declarative framework to guide and support researchers ask the right questions and find out key aspects of their problem. Especially, the boundaries between the proposed layers are very vague. Revisiting the previous example, "*cutting vegetables*" can turn into an automatic routine if the subject practices it several times. Thus, it would belong to the operation layer rather than to the actions. This aspect makes activity theory difficult to apply in computational fields [YLC11].

Within the community of sensor-based activity recognition, some notions of activity theory have been successfully employed in computer systems. However, they have been designated with different terms leading to ambiguity in the utilized scientific discourse. Depending on the application and the available sensor data, different specifications have appeared at different levels of activity granularity. For instance, Saguna *et al.* differentiate between atomic activity and complex activity in their work [SZC13]. Chen and Nugent [CN09] create an ontology based on three categories: Sub-activity, activity and goals. Hong *et al.* also decompose activities in sub-activities and sub-sub-activities [HNM+09], while Hu and Yang [HY08] suggest a rich goal taxonomy to represent activities at different levels of complexity. Finally, several works, like the one of Singla *et al.* [SCSE09], selected a well-defined set of activities of daily living (ADLs) used by health professional to assess the functional status of a subject. ADLs are the necessary self-care activities carried out by human beings in their daily routines. They can be basic (transfers,

locomotion, dressing, personal hygiene, and feeding) [Kat83] or instrumental such as shopping and housekeeping [Gra08].

As mentioned earlier, we notice a common bold distinction between the lowest level of activities and the higher-level ones, despite these variations. The first typically coincide with physical gestures which require on-body sensors to capture the subject's movements such "*walking*" and "*running*". Higher levels generally refer to complex activities that usually require environmental sensing facilities to detect the interaction of the user with their surroundings [MVC+10] such as "*housekeeping*" or "*preparing a meal*". The activity categories in between remain ill-defined.

Sequential, interleaved and concurrent activities

Real life daily routines indicate that human beings tend to undertake multiple activities at a time rather than in a sequential manner (see Figure 3.2 (i)). These may be executed in parallel or even overlap.

A segment of activities can have different types of temporal structures. Typically, human activities are sequential, interleaved or concurrent. As illustrated in Figure 3.2 (iii), two activities A and B are **interleaved** if the actor starts carrying out activity A then interrupts it to move to activity B before coming back to A again.

Example 1. *The subject might start "cooking" (activity A), then goes to "answering the phone" (Activity B) before resuming "cooking" (activity A). In this case, A and B are **interleaved** (Figure 3.2 (iii))*

Now if we slightly alter this example, we could illustrate the case of concurrent activities:

Example 2. *Assuming that the actor is using a wireless phone, they can resume "cooking" while still "talking on the phone". In this case, A and B are **concurrent** (Figure 3.2 (ii)).*

Thus, a subject can be *actively* or *passively* engaged in more than one activity. Being *actively* engaged in more than one activity corresponds to the case of concurrent activities, whereas the second case corresponds to interleaved activities. In this context, we distinguish between *foreground* and *background activities*. Formally, an activity is a *background activity* at a given time step t if the subject initiates that activity at an anterior time step d ($d < t$) and interrupts it at some time step f ($d < f < t$) before resuming it at an ulterior time step g ($d < f < t < g$). A *foreground activity*, on the opposite, is an activity the user is actively carrying out. In Example 2 both "cooking" and "talking on the phone" are *foreground activities*. However, in Example 1 "cooking" is first considered as *foreground activity* until the user "answers the phone". At that moment, "cooking" becomes a *background activity*, while "talking on the phone" is then interpreted as *foreground activity*.

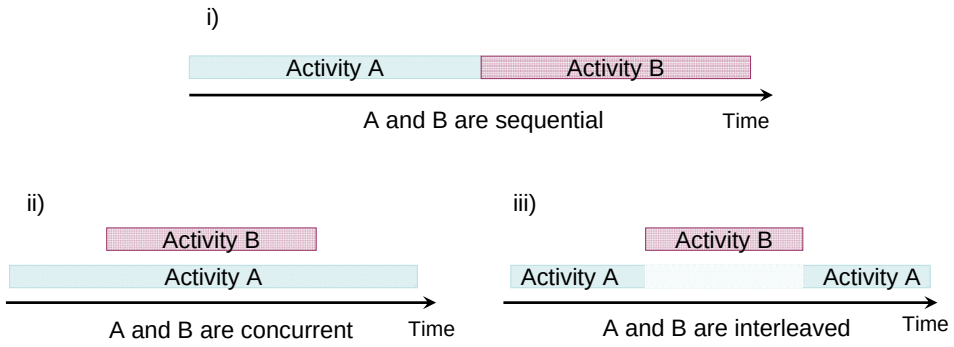


Figure 3.2: Possible temporal structures of two complex activities A and B

3.1.2 Recognizing human activities

In a simplistic scenario with strictly sequential activities, recognizing the activity of a particular subject can be formally defined as follows.

Definition 1. *Without loss of generality, let us assume a given set* $O = \{o_1, ..., o_n\}$ *of* n *vectors of sensors' observations collected at* n *time steps* t_i, $i = 0, ..., n$ *respectively. Let us also assume that at each time step* t_i, *the subject is engaged in exactly one activity* a_i *out of a set* $A = \{a_1, ..., a_m\}$ *of* m *predefined activities' labels, i.e. the activities to be recognized by the system. Activity recognition corresponds to finding a bijection* $f : O \mapsto A$ *such as:* $\forall\, o_i \in O\ f(o_i) = a_i$.

However, as mentioned above, real life situations usually implicate complex scenarios including interleaved and concurrent activities. Thus, the previous definition has to be relaxed depending on the application, the sensor data collected and the set A of activities to be recognized.

Determining the activity a_i corresponding to each observation o_i refers to an **event-based** recognition. In a more general **window-based** recognition approach, all the sensors' observations within a pre-defined time window w_i are collected as input vector o_{w_i}. The size of the time window can either be determined by a (a) maximum duration, (b) maximum number of events or (c) both.

Definition 2. *Under these settings we denote by* A_{w_i} *the set of activities carried out by the subject during* w_i. *Let* O_w=$\{o_{w_1}, ..., o_{w_n}\}$ *be the set of observation vectors collected during the time windows* w_i, $i = 0, ..., n$ *respectively. The objective of machine activity recognition is to find a mapping* $m : O_w \mapsto A$ *such as:*
$\forall o_{w_i} \in O_w,\ m(o_{w_i}) = A_{w_i}$

Thus, given the sensor observations, the activity recognition system returns the subset of the activities being carried out by the subject. In time windows where the user is not engaged in any predefined activity (such as wandering in the house without any purpose), it returns the empty set. Manifestly, event-based recognition is a special case of the window-based recognition approach, where the time windows w_i are reduced to one single event.

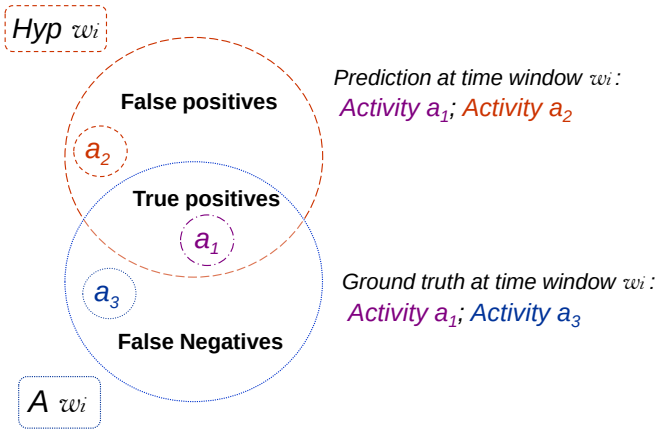


Figure 3.3: Evaluation Metrics: each predicted activity that is absent in the reference set is counted as a *false positive*. Each activity missing in the prediction set but present in the reference set is counted as *false negative*. Correctly predicted activities are *true positives*.

3.1.3 Evaluation of machine recognition of human activity

Typically, machine activity recognition is evaluated against a given ground truth based on well-known metrics from related fields such as information retrieval and pattern recognition. Among the existing works, the most popular metrics are *accuracy* ([CHN+12], [SCSE09], [HY08]) and the triple $precision$, $recall$ and F_1 $measure$ ([KCD10], [WPP+07], [GCTL10]).

$Accuracy$ is usually defined as the the portion of correctly recognized activities among the entire sequence.

$Precision$, $recall$ and F_1 $measure$ are calculated in terms of *true positives*, *false positives* and *false negatives*. These are explained below and illustrated in Figure 3.3.

Let Hyp_{w_i} be the set of activities $m(o_{w_i})$ predicted during the time window w_i. By definition, the corresponding ground truth is A_{w_i}; the set of activities carried out by the subject during the same time window. An activity a is counted as:

- $true\,positive, iff\ a \in \{A_{w_i} \cap Hyp_{w_i}\}$
- $false\,positive, iff\ a \in \{\neg A_{w_i} \cap Hyp_{w_i}\}$
- $false\,negative, iff\ a \in \{A_{w_i} \cap \neg Hyp_{w_i}\}$

Let TP_i, FP_i and FN_i respectively denote the sum of the *true positives*, *false positives* and *false negatives* within the time window w_i. Thus, the $precision$, $recall$ and F_1 $measure$ over a given sequence of time windows $w_0, ..., w_n$ can be obtained according to these formulae:

$$\text{Precision} = \frac{\sum_{i=1}^{n} TP_i}{(\sum_{i=1}^{n} TP_i + \sum_{i=1}^{n} FP_i)} \tag{3.1}$$

$$\text{Recall} = \frac{\sum_{i=1}^{n} TP_i}{(\sum_{i=1}^{n} TP_i + \sum_{i=1}^{n} FN_i)} \tag{3.2}$$

$$\text{F measure} = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{3.3}$$

The introduced expressions calculate global values, i.e. *micro-averages*, of a given activity sequence regardless of the activity specific recognition performance (equation (3.4)). Consequently, the weight of an activity is higher if it occurs more frequently within the given sequence. Alternatively, calculating the *macro-averages* corresponds to averaging the $precision$, $recall$ and $F_1 - measure$ of each activity (equation (3.5)).

The $precision\ P_a$ and $recall\ R_a$ of a specific activity a are obtained by replacing the total number of *true positives* ($\sum_{i=1}^{n} TP_i$), *false positives* ($\sum_{i=1}^{n} FP_i$) and *false negatives* ($\sum_{i=1}^{n} FN_i$) in equations (3.1), (3.2), (3.3) by the total number of those time windows where a appears as *true positive* (TP_a), *false positives* (FP_a) and *false negatives* (FN_a) respectively (equation (3.5)).

$$macro - average\ Precision = \frac{\sum_{a \in A} P_a}{|A|} \tag{3.4}$$

$$micro - average\ Precision = \frac{\sum_{a \in A} TP_a}{\sum_{a \in A} TP_a + \sum_{a \in A} FP_a} \tag{3.5}$$

Figure 3.4 explains this evaluation method through some examples. There, a simple routine is depicted, where a subject starts "*cooking*" then "*answers the phone*" during "*cooking*". At time window w_9 they start "*cleaning up*", while the "*cooking*" activities continues in the background. At time window w_{13} they resume "*cooking*" and finish it. Finally they start "*eating*" at time window w_{14}. Let us consider the predicted sequence of that routine also displayed in the same Figure. Comparing the predictions to the ground truth, we first observe an example of a time window, w_4 with two *true positives*: "*cooking*" and "*answer the phone*". Thus, in the evaluation process two *true positives* are added to the sum of *true positives*. Further, we see an example of a time window, w_{10}, with one *true positive* ("*cleaning up*") and one *false negative* ("*cooking*"). This results in increasing both the sum of *true positives* and that of *false negatives* by one. Finally, the time window w_{13} shows an example of a time window with both a *false negative* ("*cooking*") and *false positive* ("*eating*"). Following this explanation the total number of *true positives* in this mini example is 19. The total number of *false positives* is 1 (occurring in w_{13}) and the total number of *false negatives* is 5 (occurring in time windows w_9 to w_{13}). Thus, the resulting *precision*, *recall* and *F1-measure* for this mini-example are calculated as follows:

$$\text{Precision} = \frac{19}{(19+1)} = 0.95 \quad Recall = \frac{19}{(19+5)} = 0.79$$

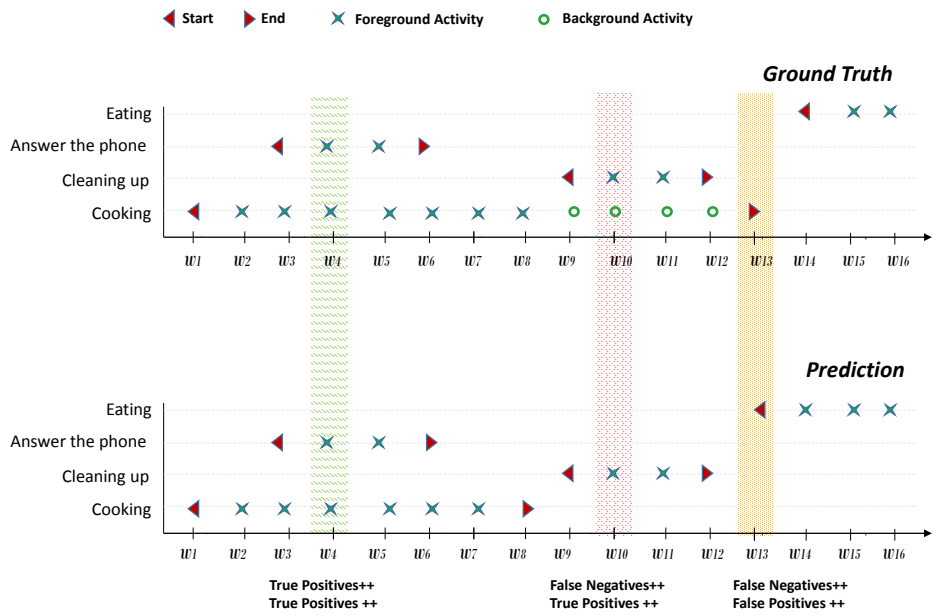


Figure 3.4: Evaluation method for predicting *foreground*, *background* and *concurrent* activities. At each time window, we compare the predicted with the actual activities. We increase the number of $true\,positives$ each time an activity is present in the ground truth and the predicted output, such as activities "*cooking*", "*answer the phone*" at time window w_4 and and "cleaning up" at time window w_{10}. If an activity is predicted, but is not in the ground truth, the number of $false\,positives$ is incremented by one, such as in time window w_{13}. Finally, for each activity that is actually carried out but not predicted, we increment the number of $false\,negatives$ by one, such as activity "*cooking*" in time windows w_{10} and w_{13}.

$$F1 = \frac{2 * 0.95 * 0.79}{0.95 + 0.79} = 0.86$$

Based on the given definitions and examples, the first intuition would be to approach activity recognition from a pure machine learning perspective. Activity recognition could then be seen as a sequential classification problem where the activity at a given time step t depends on that of previous and/or future time steps. The goal would be to optimize the number of time steps with correctly predicted activities. In real life routines, however, human activities often have strong dependencies and a a rich underlying structure. For example, we can not put the dishes in the dishwasher without opening the dishwasher first. To solve such problems with a significant background knowledge and entity relations, a paradigm that addresses statistical and relation features is required.

3.2 From graphical models to statistical relation models

Many, if not most, real world applications need to address two major challenges simultaneously: complex relational structure and uncertainty. Responding to these two pressing needs have been one of the recently emerging trends of both inductive logic programming (ILP) and statistical machine leaning communities [GT07a].

Purely statistical models usually abstract from the rich logical structure underlying data in complex systems. Generally, we are interested in representing and manipulating (even partially) structured data involving objects, events and their relationships. For instance, in a smart environment setting, we might want to detect not only the current action of a particular subject from sensor data but whether they are engaged in a high level situation of a "*cleaning the table after eating their breakfast in the kitchen*". However, dealing with real data, such as sensor readings, also requires addressing the uncertainty arising from noise, incomplete and ambiguous data. This uncertain aspect usually needs to be supported at each level of representation including the types of the involved objects, their identities and their quantitative and qualitative relationships.

As a step towards incorporating these complementary paradigms, both the ILP community and statistical machine learning community started developing novel methods. These are referred to as statistical relation learning (SRL) systems [GT07a]. Among their motives is the intuitive and compact representation of uncertain models including the underlying relational structure. They also aim at supporting efficient inference and learning algorithms for these models.

Among the proposed formalisms, the majority rely on the combination of graphical models, probabilistic grammars and logical formulae [GT07a]. These combinations are especially motivated by the rich expressiveness of logic and the ability of graphical models to capture uncertain knowledge and independence structure among entities.

3.2.1 Probabilistic graphical models in a nutshell

The true state of the world can rarely be determined with certainty by our observations. These are not only partial and incomplete but often noisy and erroneous. Consequently, real world applications require models which consider different possibilities as well as complex, non-deterministic entity relationships. This inescapably implies dealing with uncertainty and reasoning about what is probable, not just about what is possible [KF09].

Probabilistic graphical models leverage the principles of graph theory and probability theory to facilitate the construction of models which are effective in practice. Separating knowledge and reasoning, these declarative formalisms offer an appealing approach for a broad range of problems. They allow to represent and reason with the underlying probability distribution where the probabilistic parameters are usually learnt from cumulative data. The acquired parameters are fitted automatically to a given model supplied by human experts. The expert model exploits the

independence properties within a specific domain to achieve a compact distribution and alleviate the inference task.

There are different types of probabilistic graphical models. These can be classified into directed and undirected graphs. In this section, we give a brief description of each category and provide a summary of two fundamental aspects: representation and inference. Thereby, we use the two common classes Bayesian networks and Markov networks as examples.

Representation

The goal of probabilistic graphical models is to represent the relationships between different entities in order to provide an answer to any question about the modelled domain. The model should efficiently encode the probability distribution P over the set of random variables symbolizing the domain entities. Explicitly specifying the joint distribution P is computationally very expensive and often intractable even in very simple scenarios. To overcome this barrier, the distribution is often factored into modular components by exploiting the independence properties encoded in the domain. In Probability theory, two sets of random variables X and Y are **independent** in a distribution P, if and only if

$$\mathrm{P}(X = x, Y = y) = P(X = x)P(Y = y)$$

for all values $x \in Val(X)$ and $y \in Val(Y)$. Rewritten with condition probabilities, this corresponds to $P(X = x|Y = y) = P(X = x)$ for all values $x \in Val(X)$ and $y \in Val(Y)$. In other words, X and Y are independent if our guessing about the value of X does not change in the presence of any extra knowledge about the value of Y.

Conversely, X and Y are **conditionally independent** given a third set of random variables Z in a distribution P if and only if

$$\mathrm{P}(X = x, Y = y|Z = z) = P(X = x|Z = z)P(Y = y|Z = z)$$

or equivalently, $P(X|Y, Z) = P(X|Z)$ for all values $x \in Val(X)$, $y \in Val(Y)$ and $z \in Val(Z)$. In this case, X and Y are dependant as long as there is no evidence about Z. Once Z is observed, they become independent and any knowledge about X does not affect our guessing about Y any more and vice versa. To illustrate these notions, let us consider the following simple example.

Example 3. *We consider a world with four binary random variables $Daytime$, $HealthState$, $Activity$, and $BedSensor$. $Daytime$ represents the part of the day and can either be "day" or "night". $HealthState$ reflects the health state of a given person which can either be "sick" or "fit". $Activity$ encodes the activity of a given person and can take the value "sleeping" or "awake". Thereby we assume that the subject is usually "sleeping" during the "night" and "awake" during the "day". We also assume that the subject tends to be "sleeping" when*

"sick". $BedSensor$ detects the presence of the subject on the bed and usually takes the value "on" if the subject is "sleeping" and "off" if they are "awake".

*We consider a smart environment where the subject's activity and their health state can not be observed directly. Knowing that the current $Daytime$ equals " night" would favour our guessing that the subject is currently "sleeping" and consequently that $BedSensor$ is "on"(since they are most probably lying down on their bed) and vice versa. In this case, $Daytime$ and $BedSensor$ are **dependent**. However, observing the subject's activity makes them **independent**: Once we know for certain that the subject is "sleeping", any further information about the $Daytime$ can be ignored while inferring whether the user is lying on the bed. To summarize, we say that $Daytime$ and $BedSensor$ are **conditionally independent** given $Activity$.*

*Conversely, $DayTime$ and $HealthState$ are **independent** as long as the activity of the user is not known. Yet, they become **dependent** once we observe $Activity$. Indeed, knowing that the subject is "sleeping", the probability of them being "healthy" decreases when that of ($DayTime$ ="day") increases. This is called the "explaining away effect".*

Independence among the parameters of a given distribution P is a key concept for a compact representation. Instead of involving the entire set of parameters to calculate the conditional probability of a certain variable X, those variables that are (conditionally) independent on X can be ignored given the adequate evidence. This allows the factorization of the joint distribution.

Directed graph models: Bayesian network representation A Bayesian network is a data structure based on a directed acyclic graph G and capable of representing any full joint distribution. The nodes of the graph G represent the random variables of the domain and the edges represent the direct dependency between them. By exploiting conditional independence properties, it can usually provide a very *concise* representation of the distribution. Given the parents, the children and the parent's of the children of a given node X_i, the rest of the network can be ignored while computing the probability distribution of X_i. This set of nodes renders X_i independent of the rest of the network and is referred to as the **Markov blanket** of X_i. A Bayesian network associates a conditional probability distribution $P(X_i|Parents(X_i))$ with each node X_i. This is illustrated in Figure 3.5 where a simple Bayesian network of the scenario described in Example 3 is represented.

Undirected graph models: Markov network representation Unlike directed graphical models, undirected graphical models are useful when the direction of the variables influence is hardly discernible. Due to this abstraction, they usually offer a simpler alternative to the representation of the independence structure as well as to the inference task [KF09]. Markov networks are the second common class of probabilistic models. Like in Bayesian networks, each node of the graph

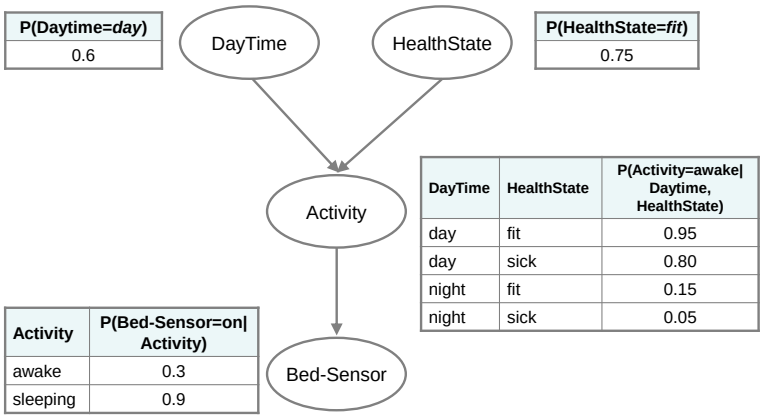
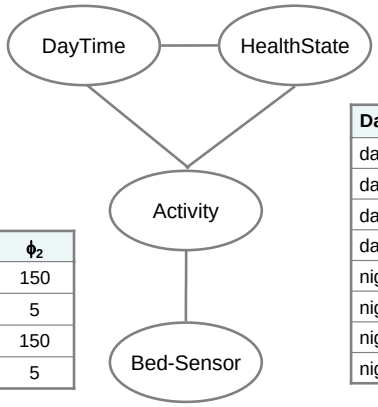


Figure 3.5: Baysian network representing the scenario introduced in **Example 3**. The graph and the associated conditional probability tables (CPT) represent one of the possible probability distributions described in Example 3

corresponds to a variable. An unmediated interaction between two nodes is represented with an edge linking them. Due to the absence of directed influence between the variables, a Markov network requires a symmetric parametrization instead of the conditional probability distributions. These parameters capture the affinity between a set of variables D in form of a potential function $\phi(D) : V(D) \mapsto \mathbb{R}^+$. A higher value $\phi(D)$ indicates more compatibility between the variables in D. The potential function $\phi(D)$ is a factor that can be seen as the contribution of the subset D to the overall joint distribution. Continuing Example 3, the corresponding Markov network can be represented as depicted in Figure 3.6.

Compared to the Bayesian network structure, understanding the dependencies in Markov networks is simpler. The dependency between two nodes is broken if every path between them is blocked by observing intervening nodes. Hence, a variable X_i is independent of the rest of the network given its immediate neighbours. As we can see in Figure 3.6, the network encodes the same set of independence assumption as in the previous Bayesian network (Figure 3.5). The extra edge linking the nodes $DayTime$ and $HealthState$ ensures that these remain dependent if $Activity$ is observed as explained in the previous section.

On another hand, the definition of the factors in Markov network allows to decide between creating a **discriminative** or a **generative** model. In the first the factor potentials are defined by some conditioned-on data where a clear distinction between observable(input) X and hidden(output) Y data is mandatory. Given the observed data, the parameter of discriminative graphical model define a *conditional* probability distribution $P(Y|X)$ over possible values of the hidden vari-



DayTime	HeathState	Activity	ϕ_1
day	fit	sleeping	10
day	fit	awake	100
day	sick	sleeping	80
day	sick	awake	20
night	fit	sleeping	90
night	fit	awake	10
night	sick	sleeping	150
night	sick	awake	5

Activity	Bed-Sensor	ϕ_2
sleeping	on	150
sleeping	off	5
awake	off	150
awake	on	5

Figure 3.6: Markov network representing the scenario introduced in **Example 3**. The network encodes the same set of independence assumptions as in the Bayesian network in Figure 3.5, but not exactly the same probability distribution.

ables. Generative models, on the opposite, represent the *joint* probability distribution $P(X, Y)$. This means that it requires $P(X)$, the probability distribution of the observable data. This makes them more general than the discriminative ones since they can address arbitrary prediction problems (such as erroneous input data). Nonetheless, conditional approaches have more freedom to fit the data because they do not have to estimate the same parameter O that represents both $P(X; O)$ and $P(Y|X; O)$ at the cost of ignoring P(X) [GT07a].

Parametrization and Log-linear models: the joint distribution $P(X)$ over a set of variables X encoded in a probabilistic graphical model G is determined by its structure and parameters. $P(X)$ is a **Gibbs distribution** parametrized by the set of k factors $\Phi = \{\phi_1(D_1), ..., \phi_k(D_k)\}$ as follows:

$$\text{For } X = \{X_1...., X_n\}, \quad P(X) = \frac{1}{Z} \prod_j \phi_j(D_j) \tag{3.6}$$

where

- Z is a normalisation constant given by $Z = \sum_{x \in X} \prod_j \phi_j(D_j)$.
- D_j is the subset of variables which participate in factor ϕ_j.
- Each subset D_j coincides with a clique if G is undirected.

For Bayesian networks, the factors simply correspond to the conditional probability distribution. Recall that a **clique** of an undirected graph G_u is a complete sub-graph of G_u. This is equivalent to a sub-graph where every pair of variables is connected by an edge.

In many graphical models, we can observe a context-specific structure. Such a structure presents distinguishable patterns for specific values of the model's variables. In order to make these patterns more apparent, an alternative parametrization of the factors converts them into log-space. More precisely, the factors are rewritten in $\phi(D) = \exp(-\epsilon(D))$ by introducing a function ϵ such as $\epsilon(D) = -\log \phi(D)$. The resulting probability distribution is, thus, guaranteed to be positive. For instance, by revisiting Example 3, we notice that $\phi(Activity, BedSensor)$ aspire to a high probability in instantiations where the values of $Activity$ and $BedSensor$ agree (by respectively acquiring the values "$sleeping$" and "on" and vice versa) and a low probability otherwise. This affinity pattern can be captured by the employing a **log-linear** framework introducing a function $f(D) : D \mapsto \mathbb{R}$ called a **feature**. Principally, $f(Activity, BedSensor)$ can be seen as an indicator function for the event "*Activity and BedSensor agree*"and would take the value 1 if the event holds, and 0 if it does not. Compared to the full factor representation, this allows more *compactness* through sparing the explicit specification of 2 extra values out of 2^2 originally. Based on this, the log-linear model can be generally defined as follows [KF09].

Definition 3. *A distribution P is a log-linear model of a Markov network over a graph G if it is associated with:*

- *a set of features $F = \{f_1(D_1), ..., f_k(D_k)\}$. where each D_i is a complete sub-graph in G.*
- *a set of weights $w_1, ...w_k$, such that*

$$P(X_1, ..., X_n) = \frac{1}{Z} \exp\left[-\sum_{i=1}^{k} w_i f_i(D_i) \right] \tag{3.7}$$

Inference

Inference in probabilistic graphical models is a mechanism to answer particular queries. We distinguish three common types of queries. The first computes the *conditional probability* of a subset of variables given some evidence and the second finds the *most probable assignment* to all non-evidence subset of variables (*Most probably explanation (MPE)*). The third is the so called *maximum a posteriori (MAP)* query. Its task is to determine the most likely assignment $\mathcal{X}^*$ to a selected subset of non-evidence variables X that forms the query. To do so, the following equation has to be solved. MAP queries combines, in a way, elements from the first two query types (summation as an element of conditional probability query and maximizations as a component of MPE query) [GT07a].

$$\mathcal{X}^* = \underset{\mathcal{X}}{\operatorname{argmax}} P(X = \mathcal{X}), P(X) = \frac{1}{Z} \prod_j \phi_j(D_j) \tag{3.8}$$

where

- Z is a normalisation constant given by $Z = \sum_{x \in X} \prod_j \phi_j(D_j)$.
- D_j is the domain of the factor ϕ_j (i.e. the subset of variables which participate in factor ϕ_j).

Since Z is a constant and the logarithmic function is monotone, maximizing the expression in Equation 3.8 is equivalent to maximizing the following sum.

$$\mathcal{X}^* = \underset{\mathcal{X}}{\operatorname{argmax}} \sum_j \log(\phi(D_j))$$

Theoretically, solving these inference tasks is possible by simply generating the joint distribution then deriving the required conditional probability or finding out the most likely variable assignment. This naive approach is called "*Enumeration-Ask*" algorithm. More efficient derivatives such as "Variable Elimination" have been proposed to reduce the number of required computational operations through caching intermediate results [RN10]. For example, referring to the Markov network in Figure 3.6, answering the query "what is the most probable activity of the user during the day knowing that they are fit and in a good health" would imply comparing the the probability of the subject $sleeping$ under this evidence as well as the probability that they are $awake$ under the same evidence. The activity with the higher probability would be the answer to the MAP query. In this simple example we have the following.

$$\underset{x \in \{awake, sleeping\}}{\operatorname{argmax}} \sum_j \log(\phi(D_j)) = \underset{x \in \{awake, sleeping\}}{\operatorname{argmax}} \big[\log(\phi_1) + \log(\phi_2)\big]$$

$$= \underset{x \in \{awake, sleeping\}}{\operatorname{argmax}} \big[\log(\phi(Activity = x, HealthState = fit, DayTime = day)) + \log(\phi(Activity = x, BedSensor = off)\big]$$

From the factors table in Figure 3.6 we have:

$$If\ Activity = sleeping,\ \sum_j \log(\phi(D_j)) = (\log(10) + \log(5)) = 3.9$$

$$If\ Activity = awake,\ \sum_j \log(\phi(D_j)) = (\log(100) + \log(150)) = 9.6$$

Hence, given this evidence, we obtain the most probable assignment

$$\mathcal{X}^* = \{day, fit, awake, on\}$$

The activity $awake$ yields the *maximum a posteriori estimation* of the user's activity. On the other hand, if the query is concerned with *conditional probability*

values then Equation 3.6 can be applied as follows. Here, Z refers to the partition function obtained by $Z = \sum_{x \in X} \prod_j \phi_j(D_j)$ and α is a normalization constant.

$$P(Activity = sleeping | HealthState = fit, DayTime = day, BedSensor = off)$$

$$= \alpha * P(Activity = sleeping, HealthState = fit, DayTime = day, BedSensor = off)$$

$$= \alpha * \frac{1}{Z}(10 * 5) = \frac{1}{Z} * \alpha * 50$$

$$P(Activity = awake | HealthState = fit, DayTime = day, BedSensor = off)$$

$$= \frac{1}{Z} * \alpha * P(Activity = awake, HealthState = fit, DayTime = day, BedSensor = off)$$

$$= \alpha \frac{1}{Z}(100 * 150) = \frac{1}{Z} * \alpha * 15 * 10^3$$

The complexity of exact inference highly depends on the structure of the network and its width. However, in the general case, it remains $NP - hard$ [GT07a]. This motivates formulating the inference task as an optimization problem where sampling-based inference techniques are usually employed.

Inference as Optimization Within the optimization framework, inference principally attempts to approximate the target function P_Φ with an easier distribution Q. This approximation usually encodes similar independence structure but allows simpler query answering than P_Φ. The main challenge is to find the best approximation Q^* out of a predefined class of "easy" distributions $\mathcal{Q}$. Based on a similarity function, computing marginals of the distribution can be formulated as an optimization problem minimizing the value of a distance function between P_Φ and Q^*, such as the relative entropy, subject to some constraint space [KF09]. Solving MAP inference can also be effectively approached under the optimization framework by applying **integer linear programming (ILP)** [RY05]. The problem is converted to maximizing a linear objective function over a finite number of integer variables, subject to a set of linear constraints over these variables [Sch98]. The linear objective function is obtained by introducing a vector μ of binary variables allowing to represent every possible assignment $\mathcal{X}$ to the graph's set of variables X and maximize the a-posteriori probability over them.

$$\forall x_i \in Val(X_k), \forall x_j \in Val(X_l), \forall (X_k, K_l) \in E :$$
$$\underset{\mu}{\text{maximize}} \left[\sum_{X_k} \sum_{x_i} \theta_i(x_i)\mu_i(x_i) + \sum_{(X_k,X_l)} \sum_{x_i,x_j} \theta_{ij}(x_i,x_j)\mu_i j(x_i,x_j) \right]$$

subject to :

$$\mu_i(x_i) \in \{0,1\} \text{ and } \mu_{ij}(x_i,x_j) \in \{0,1\}, \tag{3.9}$$

$$\sum\nolimits_{x_i} \mu_i(x_i) = 1 \text{ and } \sum\nolimits_{x_i,x_j} \mu_{ij}(x_i,x_j) = 1, \tag{3.10}$$

$$\mu_{x_i} = \sum\nolimits_{x_j} \mu_{ij}(x_i,x_j) \text{ and } \mu_{x_j} = \sum\nolimits_{x_i} \mu_{ij}(x_i,x_j) \tag{3.11}$$

More precisely, the given constraints can be expressed as follows. For each possible state $x_i \in Val(X_k)$ of a variable X_k, we define $\mu_i(x_i)$ such as $\mu_i(x_i) = 1$ if x_i belongs to a particular assignment $\mathcal{X}_u$ and $\mu_i(x_i) = 0$ otherwise (Equations 3.9 and 3.10). To encode the dependencies within the graph G, we need further variables $\mu_{ij}(x_i, x_j)$ for each instantiation $x_i \in Val(X_k)$ and $x_j \in Val(X_l)$ where X_k and X_l are linked with an edge the graph G. In particular, $\mu_i j(x_i, x_j) = 1$ if $x_i = 1$ and $x_j = 1$. Otherwise, $\mu_i j(x_i, x_j) = 0$ (Equations 3.10 and 3.11). In the ILP community, very powerful and fast solvers have been implemented such as Gurobi[1] and CLPEX[2] to calculate a (possibly) exact solution.

Typically, these methods are unlikely to scale and are not efficient for big and complex models. Except for particular classes of graphical models, they can not operate in polynomial time and are rather seen as fast alternative for small and middle-sized problems.

To deal with the worst-case combinatorial explosion of big and complex models, **sampling-based methods** are commonly used. These methods are also called particle-based approximate inference. A set of particles is a set of generated instantiations designed to present and estimate a good approximation to the joint distribution [KF09]. The existing sampling methods vary in the way they generate samples from the posterior distribution. The **Markov Chain Monte Carlo (MCMC)** class of sampling algorithms offers widely used techniques which apply equally well to direct and undirected graphs. The main idea is to generate a sequence of samples such as they progressively get closer and closer to the desired posterior distribution. The sampling process is simulated based on a Markov chain with a predefined stationary distribution, where the nodes correspond to the the set of possible instantiations of the distribution's variables. A stationary distribution presupposes that each possible instantiation is aperiodically reachable from any other instantiation. A new sample is obtained by randomly changing the preceding one. **Gibbs sampling** is a simple and effective representative of MCMC which

[1] http://www.gurobi.com/

[2] http://www-03.ibm.com/software/products/de/de/ibmilogcpleoptistud/

returns consistent estimates for posterior probabilities. It works by sampling each variable in turn given its Markov blanket in the network. This defines a specific transition probability between the states of the Markov chain. After a suitable burn-in period, the process settles into a dynamic equilibrium and reaches the desired stationary distribution. The posterior probability values are proportional to the fraction of time spent in each state [RN10].

Also belonging to the MCMC family [KF09], **MaxWalkSAT** [SKC96] is a local search algorithm for MAP inference in probabilistic graphical models. It is an optimization version of the local-search satisfiability solver **WalkSAT** [SK96]. The latter attempts to find an assignment satisfying a given set of propositional clauses in conjunctive normal form (CNF). To do so, the algorithm begins by randomly generating an assignment to the formula's variables. As long as the formula is not satisfied, it iteratively selects one of the unsatisfied clauses randomly. With a probability p, it flips the value assigned to one of its literals and with $1 - p$ it flips a literal that maximizes the number of satisfied clauses. In the MaxWalkSAT variant, the formula's clauses are *weighted*. Hence, the goal is not only to find an assignment that satisfies the formula but the one that maximizes the total weight of the satisfied clauses.

3.2.2 Logic-based statistical relational models

Whereas statistical relational systems can elegantly be introduced from the inductive logic perspective as extending logical formulae with probabilistic information, the bottom up view starting from the probabilistic graphical models is imperative to understanding them.

A particularly large number of these formalisms use variants of first order logic to compactly represent repetitive structures in graphical models. The key idea is to make abstraction of specific instances and allow to share information among groups of them. Hence, they model the meta-information sufficient to construct the probabilistic graphical model and obtain the corresponding probability distribution. This construction, called grounding, consists in substituting the variables of the higher level specification with concrete instances from the domain of discourse. The resulting instantiated graphical models are referred to as **ground models**.

This principle of **template model** has several benefits.

From a knowledge engineering perspective, the abstraction from specific instantiations allows *similar* elements to share the same parameters and properties. This trait is even more relevant for applications with *rich background knowledge* since this knowledge can easily be represented as a set of general regularities. Thanks to such relational and logical abstractions, knowledge acquired about one instance can generalize to other similar entities including unseen ones. For instance, let us consider a smart house equipped with RFID tags attached to different objects in order to recognize the inhabitant's activities. As an example, all the *washable dishes* in the kitchen can be grouped as similar items since they usually are involved in the activity "putting the dishes into the dishwasher". Thus,

describing this activity could be lifted to that abstract group of entities instead of specifying every possible *washable item* in the kitchen. Additionally, this insures that new elements of this group are automatically related to the same activity.

From a technical point of view, template models allow to avoid the full instantiation of graphical models during inference which improves the runtime and the accuracy [Rie08]. Decoupling the representation semantics from the underlying inference algorithms offers an attractive declarative aspect. Thus, application developers can improve domain-specific models independently of the reasoning algorithms. Conversely, machine learning researchers can focus on foundations and reasoning algorithms. Further details about first order probabilistic languages can be found in the exhaustive survey of Salvo Braz $et\,al.$ [dSBAR08].

A multitude of logic-based languages have been proposed by the statistical relational community. While the majority builds upon directed graphical models such as Bayesian networks [GT07a], undirected models have drawn increasing interest recently. These are especially convenient for models where the acyclicity requirement can not be easily met. Much of success is the language of **Markov logic networks (MLN)** [RD06] which combines first-order logic with Markov networks. First-order logic formulae easily and flexibly encode structural and relational information underlying both observed and hidden variables. The model's formulae can be seen as *soft* constraints on the set of possible instantiations of the graph's variables. These assignments are usually referred to as *possible worlds*. However, unlike in traditional logic, a possible world does not have to satisfy every logical formula of the model. Instead of becoming impossible, it simply becomes less and less probable by violating more and more formulae. The strength of these soft constraints is determined by an associated weight. Besides the soft constraints, MLN also supports hard constraints. Hard constraints are logic formulae that must always hold. These are distinguished by infinite weights. The probability distribution over the possible worlds is calculated as log-linear model over the resulting weighted ground formulae. An in-depth explanation of MLN and their processing steps is provided in Part II of this thesis.

Since MLN combine first-order logic with probabilistic graphical models, it can be seen as a generalization of many other SRL approaches based on special cases of first-order logic [GT07a]. For instance, to convert a relational Markov network [TAK02] in a MLN, it suffices to introduce a formulae with its corresponding weight for each possible state of each clique template in the relational Markov network [DR04].

Another research stream has opted for directed graphical models such as Bayesian networks to approach statistical relation learning. An important class of models which includes relational Bayesian networks [Jae97] and Bayesian logic programs [GT07b] is referred to as Knowledge Based Model Construction(KBMC) [Bac93]. The key idea is to use probabilistic logical knowledge bases to generate a specific propositional probabilistic model. This is specified by a set of horn clauses c_i along with the corresponding conditional probability distribution encoding $P(head(c_i)|body(c_i))$. Thus, the nodes of the generated Bayesian network represent the ground predi-

cates. The parents of a node n appearing in the $heads$ of a set S of Horn clauses are those predicates that appear in the $bodies$ of S. Whereas causal models, such as Bayesian networks, often allow a more intuitive representation of probabilistic influence thanks to conditional probabilities, they encounter important modelling issues. For instance, the set of parents having a direct influence on a given variable may vary as the number of domain elements may change among the possible instantiations. To address the problem, some works (e.g. [Jae97]) propose combination functions such as noisy-or to map several separately modelled conditional distributions to a single one. For a detailed overview of the existing SRL approaches, we refer the reader to the work of Getoor and Taskar [GT07a]).

The emergence of new statistical relational representation formalisms have also raised new challenges for the underlying inference algorithms. Reasoning with probabilistic *and* deterministic dependencies includes a constraint satisfaction problem (CSP). Consequently, applying approximate inference via sampling requires the samples to be a solution to the constraint satisfaction problem encoded in model. Under strong dependencies of a variable given its Markov blanket, state transition becomes very unlikely. Thus, the convergence of the sampling mechanism becomes extremely slow when the weights get larger and the required ergodicity breaks down in the limit of deterministic dependencies [PD06]. Combining MCMC with satisfiability testing is a possible way to approach this challenge for Markov logic networks. In the MC-SAT algorithm, Poon and Domingos [PD06] employ slice sampling instead of Gibbs sampling to adapt the uniform SampleSAT (i.e. WalkSAT plus Simulated annealing) uniform sampler to highly non-uniform distributions over possible worlds. Another interesting aspect about inference of template based statistical relational systems is exploiting the relational structure and resulting regularities. The main idea is to *lift the inference* problem to the first-order model in order to avoid explicit state enumeration and eliminate groups of ground atoms in a single step. This idea is called **lifted inference** and was first applied by Poole on the variable elimination algorithm [Poo03]. Further efforts to propose other lifted variants of inference algorithms have followed. Among those, we mention lifted MaxWalkSAT which was introduced by Singla and Domingos [SD08] in the context of MLN.

4
Problem Statement

So far we have introduced the fundamentals of sensor-based recognition of human activities. Based on these we can now precisely formulate our research problem.

As explained previously, sensor-based activity recognition is a key aspect to several emerging applications. However, this problem is very challenging due to numerous reasons. Principally, the complex and relational nature of human activities and their ambiguity defy the majority of traditional pattern recognition approaches. Multitasking is generally an inherent characteristic in real world daily routines as shown by Hao Hu *et al.* [HHPZ+08]. Indeed, human activities spread over a wide range of granularity levels and are often overlapping, alternating, and sometimes abandoned. On the other hand, they are associated with rich prior and common-sense knowledge, which is susceptible to support the recognition task. The goal of this work is to *propose*, *design*, *implement* and *evaluate* activity recognition frameworks that comply with the requirements identified in the introductory part. Motivated by the shortcomings of the two main recognition paradigms applied in the literature (i.e data-driven and knowledge-drive), this work focuses on the combination of both of them in order to address this problem statement. Concretely, we opt for two *logic-based statistical relational* approaches which we describe in two different parts.

Part I covers a *Markov logic-based approach* [RD06] to address sensor-based activity recognition. The overall aim is to assess the viability of this formalism for recognizing complex human activities under realistic settings. Since these settings inevitably include concurrent and interleaved activities, we are interested in inferring the current performed activity(ies) *as well as* any other activities currently in progress from real-world sensor data. Especially, we appraise the ability of Markov logic to represent and reason with certain and uncertain multi-relational data, including sophisticated temporal relationships. We propose three models in

order to analyze and review the effect of these major features on the recognition performance and establish the evaluation process.

Part II focuses on representing and recognizing human activities at different levels of granularity. Given the importance of rich background knowledge and contextual data for human activity, this part proposes a framework to assimilate atomic operations and context data to represent, reason and recognize increasingly complex activities in a unified ontology based framework. Leveraging *log-linear description logic* [NNS11], the proposed solution not only provides a formal and comprehensible conceptualization of the domain of discourse but also offers powerful reasoning services including both certain and uncertain knowledge. We use real-life multi-modal sensor data to evaluate the performance of our system under realistic settings such as user-independent evaluation and real-time recognition.

4.1 Research questions

We propose to respond to the research challenges delineated above by answering the following questions.

I.1 How can Markov logic be applied to represent and recognize complex human activities and which advantages does it have compared to state of the art approaches?

This question can be considered as a motivating introduction to Part I of this thesis. Its answer requires an in-depth comparison of Markov logic with other approaches applied to sensor-based activity recognition. This comparison justifies our choice for this formalism. The advantages of a Markov logic-based framework are driven and illustrated by concrete modelling examples.

I.2 How can temporal information be modelled and reasoned about in a Markov logic-based framework in order to recognize interleaved and concurrent activities?

Concretising the previous question, this one corresponds to a major contribution of our work. Whereas simple temporal context might be sufficient to predict sequential activities, more sophisticated models might be necessary to infer interleaved and concurrent activities. To answer this question, we need a formalism capable of representing and reasoning with long-range temporal relationships. This is a key aspect and a major challenge in this task.

I.3 What is the impact of incorporating prior knowledge such as commonsense information on the recognition quality?

This question implicitly involves the viability of the proposed approach to flexibly integrate prior knowledge into the suggested framework. This feature is second major contribution of this work, since prior knowledge is inherent in the domain of human activities as explained in the previous chapters. The question is thus

concerned with formulating examples of relevant prior knowledge such as common sense information and determining its effect on the recognition accuracy.

I.4 How does this approach perform when applied to real-life sensor data?
This question complements the previous ones. The models designed to answer Question I.2 have to be evaluated with real-life datasets in order to assess their viability for realistic applications. The answer for this question is, thus, delivered through the results of this evaluation process.

I.5 What are the limitations of this approach in the context of complex activity recognition?
Naturally, answering Questions I.1-I.4 raises interrogations about the limitations of the proposed Markov logic-based framework. The encountered problems and weaknesses provide the answer to this research question.

II.1 How can we build a log-linear description logic based ontology to represent and reason with the background knowledge and relational structure underlying human activities?
Guided by the previous research questions, which we address in Part I of this thesis, the first research question of Part II builds upon the lessons learnt from applying the proposed Markov-logic based approach. Essentially, we are interested in expanding and reasoning with the background knowledge using a formal and commonly shared conceptualization of the human activities and their hierarchical structure, as described in the activity theory. Based on these requirements, we propose to investigate the use of log-linear DL to represent and reason about human activities at different levels of granularity.

II.2 How can we use such a log-linear DL based ontology to not only represent but also to recognize multi-level human activities from heterogeneous sensing modalities in one unified framework?

Given an ontology about multi-level human activities, the immediate research question that arises is how to use that same framework to recognize the activities being carried out by a person from real-life sensor data. The answer to this question should also cover the challenge of the heterogeneity of the required sensing modality.

II.3 What are the benefits and limitations of using a highly expressive and probabilistic DL to represent and recognize human activities at different levels of granularity? And how viable is this approach under real-time, real-life and user-independent settings?
Investigating the use of the log-linear DL formalism for activity and recognition automatically involves evaluating its advantages and limitations. In particular, this evaluation should cover testing the approach under real-life and real-time settings in order to show its viability for real-life applications. Also, to assess its potential for re-usability, the validation process should also include user-independent experiments.

4.2 Dissertation outline

The previous chapters exhaustively presented the prerequisites for this work and outlined the state of the art of the problem statement. The remaining of this document consists of two parts.

Part I presents a Markov logic-based framework for recognizing complex activities. The first chapter addresses the first research Question I.1. It distinguishes between three categories of closely related work and compares them to the proposed approach. These categories can be designated as: (1) probabilistic graphical models extended with techniques for modelling relational data, (2) data-driven approaches combined with probabilistic modelling and reasoning techniques and (3) hybrid approaches.

The second chapter establishes the core of Part I by answering both Question I.2 and I.3. It first explains the theoretical background for Markov logic networks and illustrates its different aspects with examples from the activity recognition domain. Besides the overall idea of this formalism, its syntax, semantics and processing steps, we also cover the main aspects explaining the inference and the parameter estimation processes. Next, it reveals the three concrete Markov logic models proposed to address the problem statement. The models are disclosed following three aspects: knowledge representation, the concrete set of formulae and the employed application data.

The evaluation method as well as the obtained results of applying the introduced models are thoroughly depicted in the third chapter. This completely covers Question I.4.

Finally, the fourth chapter summarizes Part I. It compiles the concrete answers to the research questions defined in the problem statement chapter and concludes the Part with a brief report of our related current and future work. The discussion section of this chapter concisely answers Question I.5.

Part II presents a log-linear description logic-based framework for representing and recognizing multi-level activities. In the first chapter we elaborate on the related works, where we provide an overview of ontology-based frameworks for sensor-based activity recognition. The overview is organized in two categories based on whether the proposed approaches support uncertainty or not.

The second chapter addresses the principle research questions, i.e. Question II.1 and II.2. It first provides exemplified fundamentals of both description logics and log-linear description logic. Then it presents the main contributions by explaining the proposed ontology-based framework to model and recognize human activities at different levels of granularity.

To address the last research Question, the approach is evaluated in the third chapter. There, we describe different experiments and report the obtained results. The experiments include user-independent evaluation under real-life settings. We complete the answer of that question by discussing the advantages and the limita-

tions of the proposed method. We also outline our current and future work towards overcoming some of the reported shortcomings.

4.3 Citations to previously published work

This dissertation systematizes and extends the content of the previous publications. A major part of the work described in Part I was realised under the supervision and guidance of Dr. Mathias Niepert and Prof. Heiner Stuckenschmidt. The published content can be found in the following selected publications:

Rim Helaoui, Mathias Niepert, and Heiner Stuckenschmidt. Recognizing interleaved and concurrent activities using qualitative and quantitative temporal relationships. Pervasive and Mobile Computing, 7(6):660670, 2011.

Rim Helaoui, Mathias Niepert, and Heiner Stuckenschmidt. Recognizing interleaved and concurrent activities: A statistical-relational approach. In PerCom, pages 19, 2011.

Part II was accomplished in cooperation with Prof. Daniele Riboni under the guidance of Prof. Heiner Stuckenschmidt and Prof. Claudio Bettini from the university of Milan. The published content can be accessed in these selected papers:

Rim Helaoui, Daniele Riboni, Mathias Niepert, Claudio Bettini, and Heiner Stuckenschmidt. Towards activity recognition using probabilistic description logics. Activity Context Representation: Techniques and Languages, 12. Jg., S. 05., 2012

Rim Helaoui, Daniele Riboni, and Heiner Stuckenschmidt. A probabilistic ontological framework for the recognition of multilevel human activities. In UbiComp, pages 345354, 2013.

Throughout of this document, we will roughly designate the contents originating from our publications. The omission of this indication signifies the novelty of the material.

Part I

Markov Logic and Recognizing Complex Activities

"Reality is not a function of the event as event, but of the relationship of that event to past, and future, events."

—Robert Penn Warren

1 Related Work and Contributions

As stated in the introductory part of this dissertation, a majority of the approaches to activity recognition in sensor environments fall short to represent, reason or learn with four decisive aspects of the domain: *(i) uncertainty*, *(ii) complex relational structure*, *(iii) rich temporal context (including long-range temporal relationships)* and *(iv) abundant common-sense knowledge*.

Approaches combining complementary aspects from the data-driven and knowledge-driven paradigms have shown to be a promising direction towards developing realistic activity recognition system. We roughly distinguish three categories of hybrid approaches. The first category essentially encompasses probabilistic graphical models extended with techniques for modelling relational data. The second encloses knowledge-base approaches combined with probabilistic modelling and reasoning techniques such as probabilistic ontologies for instance. Finally, the third category consists in other hybrid approaches mostly based on activity signature. Following this classification, in this chapter we discuss these approaches and accurately compare them to ours.

1.1 Relational extensions of probabilistic graphical models

Increasing efforts to apply standard probabilistic graphical models with more structured state spaces can be distinguished in the literature. Several extensions have been applied to the activity recognition problem. These extensions range from simple variants of standard probabilistic sequence models, i.e. HMM and linear-chain CRF to more general logic-based extensions of probabilistic graphical models such Bayesian networks and Markov networks. In the following, we explain and discuss

the main ones.

1.1.1 Relaxations of standard probabilistic sequence models

A particularly widely-used approach to sensor-based activity recognition are hidden Markov models (HMM) and their discriminative counterpart linear-chain conditional random fields as introduced earlier in this document. Being very well suited for several sequence recognition tasks such as speech recognition, these methods have shown serious limitations when applied to activity recognition under realistic settings ([GK06], [KHC10], [SZC13]). These limitations are mainly due to their inflexible structure. We identify two principal extensions of these standard methods to address the challenges of activity recognition.

Interleaved hidden Markov models (IHMM)
To relax this inflexibility and adapt HMM to the challenge of recognizing interleaved activities, Modayil et al. [MBK08] have proposed interleaved hidden Markov models (IHMM) where each state consists of a current activity *and* a record of the last object observed while performing each activity. By keeping track of the last object used before the activity changes, the probabilistic model gains an additional indicator that helps identify interrupted activities once they are resumed by the user. This added flexibility comes at the cost of the size of the state space which increases significantly. This not only requires adequate optimization techniques to maintain the efficiency of HMM but also necessitates larger training sets to train all possible resulting state paths.

Skip-chain conditional random fields (SCCRF)
The discriminative analogue of HMM, i.e. linear-chain conditional random fields (CRF), have also shown comparable limitations when it comes to representing long-range dependencies. A very similar extension idea to the IHMM has been also employed to address the recognition of interleaved and concurrent activities in [HY08]. Linear-chain CRFs are *undirected* graphical models with the same topology as HMM. Thus, due to their strict independence assumptions, they can not represent dependencies between distant terms in the input. Skip-chain conditional random fields (SCCRF) extend linear-chain CRFs with additional long-distance edges between sets of observations. The resulting model is a general CRF with two clique templates one for the linear chain portion and one for the skip edges. In their work, Hu and Yang [HY08] create skip edges between sensor observations which most probably correspond to the same activity. On the other hand, the authors create a separate correlation graph between activities to obtain the probabilities of concurrent ones. Based on the combination of the output of the SCCRF model and that of the correlation graph, they infer the user's activities. Unfortunately, SCCRF potential functions pose a computationally expensive inference problem especially when a large numbers of skip edges is involved [MS06]. Furthermore, to prevent the recognition accuracy from deteriorating, every partial model of the interleaved

activities has to be observed during the training phase.

Whereas these two extensions of standard probabilistic sequence models enhance the recognition of interleaved activities, their inflexibility makes them inadequate to represent and reason with complex relational structure such as reasoning with time intervals and modelling activity duration for example. Moreover, they lack an intuitive modelling interface to flexibly complement and control the automatically estimated parameters through the integration of common-sense knowledge within a unified framework. A very simple example of such knowledge is the exclusion of specific activity transitions such as the transition from "taking shower" to "leaving home" without going through the activity "dressing" for instance.

1.1.2 Logic-based extensions of probabilistic graphical models

As opposed to their propositional counterparts, logic-based probabilistic graphical models are abstractions in form of a first-order representation of the symbols used to generate the original graphical model. Hence, an abstract variable consists of a predicate name and a set of parameters that can be instantiated with constant ground values. The main idea is to allow for a compacter representation of repetitive structures in the graphical model and allow specific instances to share the same parameters. These approaches belong to logic-based statistical relational models described in the preliminaries chapter of this thesis. In the context of the activity recognition task, four major logic-based statistical relational approaches can be identified as discussed below.

Logical hidden Markov models (LOHMM)
Introduced by Kersting et. al [KRR06], LOHMM are a generalization of standard HMM allowing a compact representation of probability distributions over sequences of logical atoms. This distribution is defined by the transition probabilities between abstract states together with the probabilities of their instantiations. Leveraging logic-based reasoning techniques and the logical structure of the model, LOHMM offer an elegant formalism that often outperforms standard HMM [KRR06].

LOHMM where applied to activity recognition by Natarajan et al. [NBT+08]. However, despite the proposed efficient particle filter-based inference technique, the authors employed synthetic data to validate their approach. The selected data consists in a simplistic kitchen scenario totally isolated from realistic settings and challenges such as interleaved and concurrent activities. The logical extension of HMM offers one step towards more intuitive and compact modelling techniques, nonetheless selecting a structure of a LOHMM is a significant problem [KR12]. Furthermore, it remains highly constrained with its strong independence assumptions. Flexibly modelling highly inter-related entities and complex temporal relationships remains a challenge for this formalism.

Markov logic networks
Recall from the introductory part of this document that Markov logic networks extend first-order logic to probabilistic setting by attaching weights to formulae. These weighted formulae collectively construct a template for the generation of a Markov random field. Thus, compared to other statistical relational approaches, Markov logic networks is probably the most expressive formalisms [DR04]. The adoption of first-order logic not only offers a superior expressiveness but also creates a *particularly intuitive modelling interface*. This interface together with the highly flexible structure of Markov networks, establish a declarative unified framework best suited for addressing the four activity recognition challenges introduced above, i.e. u*ncertainty*, *complex relational structure*, *rich temporal context (including long-range temporal relationships)* and *abundant common-sense knowledge*. Since conditional random fields have shown to outperform generative graphical models in several labelling tasks, we opt for Markov logic network that casts a conditional random field to capture the conditional probabilities between the observable and hidden predicates. Thus, our proposed model does not require the representation of the probability distribution of the input data as imposed by its generative, directed counterparts such as **Bayesian logic networks** [MMvO$^+$12].

Markov logic networks have originally been explored in the context of activity recognition by Biswas et, al. [BTF07] and Tran et, al. [TD08]. Nonetheless, their algorithms are designed with visual activity recognition in mind and take video data as main input. Unlike our approach, their works address a very limited temporal context and only very *atomic* sequential activities such as "shaking hands" [TD08]. To the best of our knowledge, our work ([HNS10], [Hel10], [HNS11a], [HNS11b]) was the first attempt to employ Markov logic in order to address sophisticated temporal relationships and recognize complex human activities in sensor environments. Not surprisingly, several works appeared afterwards to leverage this powerful formalism in the context of activity recognition [SK12]. Among these efforts, we notice a particular focus on extending the idea of temporal reasoning based on the well-established theory of event calculus [SPVA11], [FASP12], [SPAV12].

Conditional random fields for logical sequence (TildeCRF)
Gutmann et al. [GK06] considered the special case of MLN where the ground Markov model is represented as a CRF. In that case, the potential factors of the graph define a probability distribution over possible outputs *conditioned* on observed inputs. The authors proposed a framework named TildeCRF and applied it on a very elementary job scheduling usecase with four cities and 8 activities to demonstrate the validity of their system. Besides being subsumed by MLN, the application of their system is not really related to our problem statement.

Bayesian logic networks
Also following the same principle as lifting Markov networks to the relational

level, Bayesian logic networks (BLN) [JvGB11] are a meta-model formulated in weighted first-order logic formulae which constructs a probability distribution from a Bayesian network and global logical constraints. Compared to MLN, BLN can only create a generative model where the probability distribution of the observable variables can not be omitted unlike the discriminative version of MLN. Whereas BLN have been applied for robotic action recognition through representing multi-object interactions in a scene [MMvO+12], there are no relevant works applying this formalism to address sensor-based activity recognition.

1.2 Probabilistic extensions of knowledge-driven approaches

Rather than lifting statistical techniques to the relational level, some researchers have proposed hybrid systems by extending knowledge-driven methods. Their ultimate motivation is to preserve the multiple advantages of knowledge-driven paradigms while solving the problem of supporting uncertainty.

In this context, sensor based activity recognition was approached by Filippaki et al. [FAT11] through the combination of *obligatory* and *optional* constraints. In their rule-based system, they attempt to recognize a simple sequence of scenarios with hierarchically organized activities such as “watch TV” and “phone call”. The integration of confidence values with the optional entities enables the system to cope with uncertain and incomplete data. To infer the activities with the highest confidence without violating the obligatory rules, they employ weighted Partial MaxSAT problem (WP-Max-SAT). Despite the similarity of the overall concept of combining certain and uncertain rules within one framework, our MLN based method offers an incomparably more expressive framework with sound probabilistic semantics.

Another line of research opted for extending ontology based representation of the user’s activities, their environment and their context with some support of probabilistic reasoning or statistical methods [RB09], [HRS13], [CNO14], [YSD14]. For the sake of concision, we omit a detailed description of these approaches and refer the reader to Part II of this work which is dedicated to this trend.

1.3 Other hybrid approaches

Essentially based on mining techniques, two major works can be assigned to this third category of hybrid approaches. In the first [GWT+09], the authors investigate the use of **emerging patterns (EP)** with sliding windows to address the recognition of interleaved and concurrent activities. An emerging pattern is a feature vector of an activity that describes significant changes between that activity and other activities. In other words, emerging patterns represent the item sets which maximize the growth rate from a data set to another. The advantage of this techniques is that

emerging patterns can be extracted from sequential activities and then be used for concurrent and interleaved scenarios. However, this approach is prone to limitations since a sliding time window might exclude some of the distinguishing features. This imposes the use of a segmentation algorithm to improve the results. In addition, the approach does not support the integration of *background knowledge* and *complex inter-entities relations*.

The second work [SZC13] focuses on the relevance of context data to represent and reason about concurrent and interleaved activities. Inferring the activities first goes through mapping the context data to a high-level situation such as "office room" before using it to recognize the user's activity (e.g "eating lunch"). Similarly to emerging patterns, the authors define the activities in terms of a weighted list of components (atomic activities). In this kind of activity signature, the weights indicate the respective relevance levels of the components to the occurrence of the defined activities. Compared to our work, the proposed framework shares the same weaknesses with the EP-based work of Gu et al. [GWT+09] discussed above.

2

Modelling and Recognizing Complex Activities with Markov Logic Networks

This chapter builds the core of Part I. Based on the preliminaries introduced previously, we first explain the theoretical background of Markov logic and illustrate it with examples from the activity recognition domain. Besides the overall idea of this formalism, its syntax, semantics and processing steps, we also cover the main aspects explaining the inference and the parameter estimation processes. Next, we present our Markov logic models addressing the research questions of this part. Whereas the first two models: The **Basic Model** and the **Start-End Model** originate from anterior publications ([HNS11a], [HNS11b]), The **States-Based Model** is new. For a systematic and consistent presentation of the models, we have significantly altered the organization provided in the mentioned publications.

The introduction and explanation of the models go through two basic sections. The first is rather abstract and covers the knowledge representation adopted. The second details the concrete set of formulae, the application data and the experimental settings.

2.1 Markov logic networks

Markov logic is perhaps one of the most *flexible* and general languages in the realm of statistical relational knowledge representation. It is a widely used formalism due to its *declarative* nature and *ease of experimentation* as well as the availability of efficient learning and reasoning algorithms. Incorporating both first-order logic and probabilistic graphical models, Markov logic allows objects and their complex relationships to be expressed in an *intuitive* and flexible manner. It thus offers a

common language unifying several well-known statistical relational approaches. These include Markov random fields, logistic regression, hidden Markov models and conditional random fields [DR04].

In the following, we briefly provide the necessary background in first order logic before we describe the fundamentals of the Markov logic theory and its application to human activity recognition.

2.1.1 Background

Propositional logic: one simple computer tractable formalism to describe facts about a given domain and reason about them is *propositional logic*. Syntactically, it consists of boolean *atomic sentences* called *propositions* that can construct complex sentences, also called **formulae**, using logical connectives. The commonly used logical connectives of propositional logic are negation *($\neg$ A is true if and only if A is false)*, conjunction *($A \wedge B$ is true if and only if both A and B are true)*, disjunction *($A \vee B$, which is true if and only if A or B is true)* and implication *($A \Rightarrow B$, which is true if and only if B is true or A is false)*. Together, the propositional formulae build up a propositional knowledge base. Every propositional knowledge base can be converted to a *conjunctive normal form (CNF)*. A *conjunctive normal form (CNF)* is a conjunction of *clauses*. A clause is a disjunction of literals and a *literal* refers to an atom or its negation. Finally, a *Horn clause* is a disjunction of literals of which at most one is not negated.

Semantically, the meaning of a sentence is evaluated by its truth value (true or false) with respect to an interpretation I. An *interpretation* I is an assignment of truth values (true or false) to all atoms in a knowledge base. I satisfies a given formulae F if and only if F is evaluated to be "true" under I. In that case, I is called a *model* of F and is denoted by $I \models F$. A knowledge base KB entails a formula F (denoted by $KB \models F$) if F is true in all interpretations where the knowledge base KB is true. Determining whether a given formula F if is entailed by a knowledge base KB is called *inference*. Deciding whether there is at least one interpretation that satisfies a given formulae is referred to as the *satisfiability (SAT)* problem which is a prototypical NP-complete problem.

Example 4. *Let's consider the following sentences based on Example 3 introduced in the previous Part. Let*

- *D be: "DayTime = day"*
- *H be: "HealthState = fit'*
- *A be: "Activity = awake"*
- *B be: "BedSensor = off'*

Reducing our scenario to deterministic (i.e. certain) dependencies between the different variables in Figures 3.5 and 3.6, we could describe it using the following simple propositional knowledge base.

- $D \wedge H \rightarrow A$
- $\neg D \vee \neg H \rightarrow \neg A$

- $A \rightarrow B$
- $\neg A \rightarrow \neg B$

Consequently, one ***model*** *of this knowledge base would be* $\{D = true,\ H = false,\ A = false$ *and* $B = false\}$*. The model describes the world where "the subject is not fit, thus is sleeping during the daytime and consequently, the bed sensor is turned on".*

First-order logic: Whereas propositional logic is expressive enough for such simple scenarios, it is incapable of describing complexer information about objects and their relations. Bringing the previous example closer to real life would suggest, for example, to include a new variable determining the subject concerned by this model. This is especially important for cases where two or more subjects are living in the same house, for example. Given this new information, our knowledge base is required to determine new formulae such as " if a resident is fit then that same resident is awake during daytime". Furthermore, it would be also helpful to precise that "the activity of that person at a given day is related to their health state during that *same* day".

Based on the foundations of Propositional logic, **first-order logic (FOL)** borrows representational ideas from natural language to allow atoms to have internal structures. Especially, it generalizes it by abstracting away from entity-specific propositions. Thus, symbols are allowed to have arguments instead of strictly predefined truth values.

The basic elements of the first order logic syntax are constants, predicates, functions and variables. *Constants* refer to objects in the domain of interest. *Variables* range over the objects and can be seen as place-holder, as opposed to constants. In our previous example, a resident is an object that could be represented by the variable r. The names of the residents, such as "Bob" and "Mary", would correspond to the constants. Variables and constants might be *typed*. In that case, they only range over a specific type of objects such as "residents" in our example.

Functions map tuples of objects to objects. They usually serve to avoid giving a distinct name to each object. For instance, with $BedOf(Bob)$ we can designate the bed of Bob without naming that concrete bed. Finally, *predicates* represent a property of or a relation between objects that can be *true* or *false*. As an example, the predicate $HasActivity(Bob, Sleeping)$ would return "*true*" if "Bob" is having the activity "sleeping" and "*false*" otherwise.

Similarly to propositional logic, an interpretation determines exactly which constants refer to which objects, relations and functions. Any expression referring to an object is called a *term*. Putting terms together creates atomic sentences that state facts. Concretely, atoms are obtained by applying predicate symbols to a tuple of terms. For instance, the atom $BedSensorState(On, BedOf(Bob))$ states that the bed sensor of Bob's bed is "on". As in propositional logic, complex sentences, also called formulae, are constructed by applying logical connectives to atomic sentences. The following example formula states that "if sleeping is the

activity of Bob, then his "bed-sensor" is "on":

$$HasActivity(Bob, Sleeping) \rightarrow BedSensorState(On, BedOf(Bob))$$

Generalizing such a statement to all the residents of the house is fortunately possible in FOL. This is enabled by two standard *quantifiers*: universal ($\forall$) and existential ($\exists$). $\forall x\, F$ is true in a given model, if and only if the logical expression F is true for all possible objects x in the domain of F . $\exists x\, F$ is true if and only if F is true for at least one object in the domain of F. The formulae of a knowledge base are implicitly conjoint and can be seen as one big formula.

A *ground term* is a term without variables and a *ground formula* is a formula containing only ground terms as arguments. The function symbols and constant symbols of a set of clauses S represent the *Herband universe* of S. The assignment of truth values to each possible ground predicate creates a *possible world*, also called a *Herband interpretation*.

To determine whether a formula is entailed by a given first-order logic knowledge base, it is convenient to convert it to the conjunctive normal form (CNF) and check whether $KB \wedge \neg F$ is unsatisfiable. Inference in first-order logic can be reduced to propositional inference by inferring non-quantified sentences from quantified ones. However, while inference is decidable for propositional logic, it is only *semidecidable* in first order logic. Thus, there exist effective methods that always provide a correct and finite proof if a given formula is entailed by some knowledge base. Nonetheless, given the set S_F of all the formulae not entailed by a given knowledge base KB, there is no algorithm, that for every $F \in S_F$ is capable of deciding that F is not entailed by KB.

There are several extensions and restrictions applicable to first-order logic allowing to alleviate the inference problem and make it more efficient. For instance, limiting the domain of discourse to a finite set of entities (and hence imposing a function-free first-order logic) allows any first-order logic knowledge base to be converted to a propositional knowledge base with a complete decision procedure. Furthermore, the use of *typed* entities and predicates reduces the number of ground atoms.

Example 5. *Generalizing Example 4 to several residents, we could state following formulae:*

- $F1 : \forall x\, HasHealthState(x, Fit) \wedge DayTime(Day)$
  $\rightarrow HasActivity(x, Awake))$
- $F2 : \forall x\, DayTime(Night) \wedge HasHealthState(x, Sick)$
  $\rightarrow HasActivity(x, Sleeping))$
- $F3 : \forall x\, HasActivity(x, Sleeping)$
  $\rightarrow BedSensorState(On, BedOf(x))$

- $F4 : \forall x\, HasActivity(x, Awake)$
  $\rightarrow BedSensorState(Off, BedOf(x))$
- $F5 : \forall x\, HasHealthState(x, Fit) \leftrightarrow \neg HasHealthState(x, Sick)$
- $F6 : DayTime(Day) \leftrightarrow \neg DayTime(Night)$
- $F7 : BedSensorState(Off, x) \leftrightarrow \neg BedSensorState(On, x)$

While Formulae F 1 to F 4 model the dependencies between the subject's health state, their activity and their "bed-sensor", the last three formulae (F 5 to F 7) express simple common-sense knowledge such as "a person can not be sick and fit at once".

Within a domain of discourse with two constants "Bob" and "Mary", the propositionalization of this first-order logic knowledge base would simply require grounding the sentences using all possible ground-term substitutions: $\{x/Bob\}$ *and* $\{x/Mary\}$*. Assuming that each person has only one bed, we replace the ground terms "BedOf(Bob)" and "BedOf(Mary)" by a new proposition to avoid infinitely nested terms such as "BedOf(BedOf(Bob))". Grounding Formulae F 1 and F 3, for instance, would result in the following sentences:*

- $HasHealthState(Bob, Fit) \wedge DayTime(Day)$
  $\rightarrow HasActivity(Bob, Awake))$
- $HasHealthState(Mary, Fit) \wedge DayTime(Day)$
  $\rightarrow HasActivity(Mary, Awake))$
- $HasActivity(Bob, Sleeping) \rightarrow BedSensorState(On, BedOfBob)$
- $HasActivity(Mary, Sleeping) \rightarrow BedSensorState(On, BedOfMary)$

2.1.2 Markov logic: formalism and processing steps

The simple first-order knowledge base depicted in Example 5 is too inflexible to conciliate with real world scenarios. Manifestly, there are many situations where a subject might be "sleeping" during the "day" even if they are "fit". The main idea of Markov logic [DR04] is to *soften first-order formulae* and *tolerate their violation*. Accordingly, a world that does not satisfy a given "soft" formula would be "less probable" instead of being "impossible".

Usually, the knowledge base formulae convey a set of constraints with different degrees of strength. For instance, following the previous example, the state of the "bed sensor" is usually a stronger indicator for the activity of the subject then their health state and the daytime. This means that the situations (i.e. "worlds") where the bed sensor indicates the correct activity are more frequent than those where the subject's "health state" and the "daytime" do. Hence, formulae F 3 and F 4 have a *higher number of more probable models* then formulae F 1 and F 2. The former

formulae are, thus, *more probable* and express stronger constraints. Whereas these *soft constraints* can be violated, it is clearly not the case for the last three formulae F 5 to F 7. Apparently, these have to hold in every possible world.

The Markov logic formalism associates first-order logic formulae with a *weight* indicating the constraint's strength. The weight of a formula F scales the difference in log-probability between a world w_n that satisfies n groundings of F and a world w_m that results in m true groundings of F, all else being equal [SK12]. Thus, the higher weight, the greater the difference in the probability between the worlds w_n and w_m, other things being equal. Reciprocally, the more evidence there is that a formula is valid the higher its probability and hence its weight.

The probability of a ground formula is the sum of the probabilities of the worlds where that formula is $true$. Despite the monotonically increasing relationship between the weights and the probability of ground formulae, there is no generic complementarity of a weight as compared to the weight for its negation [Spi12].

Collectively, the weighted formulae define a template model for the construction of a probabilistic graphical model and are called a *Markov logic network*. The weights parametrize the probability distribution over possible worlds. The probability of a possible world is proportional to the exponentiated sum of weights of ground formulas that are satisfied in that world.

Example 6. *One possible extension of the first-order knowledge base of Example 5 to a Markov logic network would be as follows.*

- $(w_1 = 1.5)$ $\forall x\, HasHealthState(x, Fit) \wedge DayTime(Day) \rightarrow HasActivity(x, Awake))$
- $(w_2 = 2.5)$ $\forall x\, DayTime(Night) \wedge HasHealthState(x, Sick) \rightarrow HasActivity(x, Sleeping))$
- $(w_3 = 4.0)$ $\forall x\, HasActivity(x, Sleeping) \rightarrow BedSensorState(On, BedOf(x))$
- $(w_4 = 4.0)$ $\forall x\, HasActivity(x, Awake) \rightarrow BedSensorState(Off, BedOf(x))$
- $(w_5 = \infty)$ $\forall x\, HasHealthState(x, Fit) \leftrightarrow \neg HasHealthState(x, Sick)$
- $(w_6 = \infty)$ $DayTime(Day) \leftrightarrow \neg DayTime(Night)$
- $(w_7 = \infty)$ $BedSensorState(Off, x) \leftrightarrow \neg BedSensorState(On, x)$

Syntax

A signature is a triple $S = (O, H, C)$ with O a finite set of typed observable predicate symbols, H a finite set of typed hidden predicate symbols, and C a finite

set of typed constants. Formally, a Markov logic network (MLN) is a pair of two sets $(\mathcal{F}^h, \mathcal{F}^s)$. $\mathcal{F}^s$ is a set of n pairs $\{(F_i, w_i)\}, i = 1, ..., n$ with each F_i being a function-free first-order formula built using predicates from $O \cup H$ and each $w_i \in \mathbb{R}$ a real-valued weight associated with formula F_i. $\mathcal{F}^h$ is a is a set of l function-free first-order formulae $\{F_i\}, i = 1, ..., l$. The elements of $\mathcal{F}^h$ are referred to as *hard* formulae and those of $\mathcal{F}^s$ as *soft* formulae. In the following, we employ the terms formulae, axioms, rules and constraints interchangeably.

Semantics

Let $M = (\mathcal{F}^h, \mathcal{F}^s)$ be a Markov logic network with signature $S = (O, H, C)$.
A *grounding* of a first-order formula F is generated by substituting each occurrence of every variable in F with constants in C with the same type. Existentially quantified formulae are substituted by the *disjunctions* of their groundings over the finite set of constants. This definition of the semantics of Markov logic makes several assumptions:
(a) different constants refer to different objects (unique names assumption)
(b) the only objects in the domain are those representable using the constants (domain closure assumption [RD06]).
These assumptions ensure that the resulting ground Markov logic network has a finite number of nodes. A set of ground atoms is a *possible world.*

Let $\mathcal{G}_F^C$ be the set of all possible groundings of formula F with respect to the set of constants C. Let $\mathcal{W}$ be the set of all possible worlds with respect to S. Then, the probability of a possible world $W \in \mathcal{W}$ is given by

$$p(W) = \begin{cases} \frac{1}{Z} \exp\left(\sum_{(F_i, w_i)} \sum_{G \in \mathcal{G}_{F_i}^C :\, W \models G} w_i \right) & \text{if } \forall F \in \mathcal{F}^h : W \models \mathcal{G}_F^C \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

where Z is the partition function.

The score s_W of a possible world W is the sum of the weights of the ground formulae that are satisfied in W

$$s_W = \sum_{(F_i, w_i)} \sum_{G \in \mathcal{G}_{F_i}^C :\, W \models G} w_i. \tag{2.2}$$

As indicated by Equation 2.1 and 2.2, the probability of a possible world W is proportional to its exponentiated score.

Usually, hard formulae are assigned very large weights compared to soft ones. Thus, the score of a world satisfying all hard formulae will be significantly higher than that of a world violating one of them. From the graphical models perspective, a Markov logic network generates a *ground Markov network* with a node for every ground term. The set of nodes belonging to the same ground formula G_F are fully connected as a clique D. For each clique D we define a feature function $f(D)$ mapping the truth assignment of its ground terms to the truth value of the

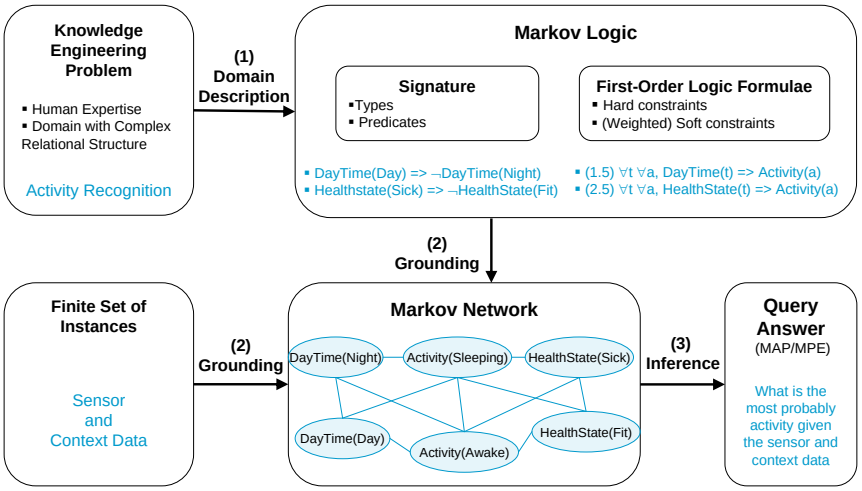


Figure 2.1: Illustration of the processing steps for a simple knowledge engineering problem with Markov logic. Each step is illustrated by a concrete example at the bottom. After building the template model based on the expertise knowledge in step (1), a ground Markov network network is constructed based on the domain's finite set of instances (step (2)). Step (3) executes the inference to answer the required query.

corresponding ground formula G_F. The feature function is an indicator whether the event "G_F is true" holds.

Together with the corresponding weights, the feature functions provide real-valued outputs that depend on the state of the corresponding cliques in the network. Thus, they define clique potentials of the Gibbs distribution that factorizes over the generated ground Markov network. Replacing each factor $\phi_i(D_i)$ of the Gibbs distribution by the exponentiated weight of the corresponding ground formula e^{w_i}, the probability distribution defined in Equation 2.1 can be written as a log-linear model as follows.

$$p(W) = \frac{1}{Z} \exp\left(\sum_{(F_i, w_i)} \sum_{G \in \mathcal{G}_{F_i}^C} w_i f(G) \right) \tag{2.3}$$

Setting $n_i(W)$ as the number of true groundings of the formula F_i in a world W, Equation 2.3 can be simplified as follows.

$$p(W) = \frac{1}{Z} \exp \sum_{(F_i, w_i)} \left(w_i n_i(W) \right) \tag{2.4}$$

Based on this syntax and these semantics, the key processing steps for applying Markov Logic to a given knowledge engineering problem are as depicted in Figure 2.1.

Step (1): consists in providing the input theory and the domain description in form of first order formulae adopting the Markov logic syntax. Following the

simple example provided, the model comprises four formulae: two *hard* constraints and two *soft* constraints with their respective weights. Along with the signature, the model simply states that both the "daytime" and the "health state" of a subject influence their "activity". The weights signify that the influence of "health state" of the subject on its "activity" is more important than that of the "daytime". The model also integrates straightforward but useful background knowledge in form of *hard* constraints. This indicates that the "daytime" can not be "Day" *and* "Night" at once, and that the subject can not be "Sick" *and* "Fit" at once neither.

Step (2): here a ground network is constructed based on both the created model and the corresponding finite set of instances. In the context of the illustrating example, there are only two possible interpretations for the subject's activity: "Awake" and "Sleeping". Looking at the generated ground Markov network we can see two cliques linking the ground atoms appearing in each of the formulae.

step (3): finally, the inference task takes place in this step. It answers the requested query. In the provided example, one possible inference task is to find the most probable activity of the user during the day given that they are sick. The main challenge of the inference step is to avoid as much as possible the exponential complexity emerging from taking all possible combinations of predicates and ground terms.

2.1.3 Inference and parameter estimation

Recall that ordinary queries in probabilistic graphical models are special cases of Markov logic network inference with zero-arity predicates. On another side, the logical query of whether a given formula F is entailed by a knowledge base KB can be answered by determining whether $P(F|KB) = 1$. This holds in finite domains and can be achieved by assigning infinite weights to all the formulae in the KB. Thus, addressing the inference problem in Markov logic networks implicates the need to handle both probabilistic and logical inference. Since the first is #P-complete and the second is NP-Complete, no better results can be expected [DR04]. Fortunately, inference in Markov logic network leverages advantages from both domains. Concretely, exploiting both approximate inference methods and context-specific independencies enhances the efficiency of the inference.
In the context of activity recognition, the main inference task is to determine the most probable state of a Markov logic network given some observations. Therefore, we need to compute the set of ground atoms of the hidden predicates that maximizes the probability of the world given both the ground atoms of observable predicates and all ground formulae. This is an instance of MAP (maximum a-posteriori) inference in the ground Markov logic network as introduced in the Preliminaries chapter.
Formally, this can be defined as follows. Given a Markov logic network with signature $S = (O, H, C)$, let $\mathbf{O}$ be the set of all ground atoms of observable predicates and $\mathbf{H}$ be the set of all ground atoms of hidden predicates both with respect to C. We make the closed world assumption with respect to the observable predicates.

Assume that we are given a set $\mathbf{O}' \subseteq \mathbf{O}$ of ground atoms of observable predicates. In order to find the most probable state of the Markov logic network we have to compute

$$\underset{\mathbf{H}' \subseteq \mathbf{H}}{\operatorname{argmax}} \sum_{(F_i, w_i)} \sum_{G \in \mathcal{G}_{F_i}^C :\, \mathbf{O}' \cup \mathbf{H}' \models G} w_i. \tag{2.5}$$

As denoted by Equation 2.5, the MAP inference problem reduces to finding the truth assignment that maximizes the sum of weights of satisfied clauses. Thus, a particularity suitable method for solving exact inference in Markov logic networks is Integer Linear Programming (ILP) [Rie08]. As introduced in the Preliminaries Chapter, ILP is concerned with optimizing a linear objective function over a finite number of integer variables, subject to a set of linear constraints over these variables [Sch98], [Ril02].

Alternatively, approximate inference approaches have been successfully applied to Markov logic networks. MaxWalkSAT [SKC96] is probably one of the most commonly used ones. Belonging to the MCMC family [KF09], MaxWalkSAT [SKC96] is an optimization version of the local-search satisfiability solver WalkSAT [SK96].

Since propositional inference schemes are quite expensive, recent methods have been introduced to avoid propositionalizing the entire domain. Referred to as lifted inference algorithms, their key idea is to answer queries and reason about them at the first-order level without grounding them thoroughly. This is achieved by treating indistinguishable groups of objects as a unit. A lifted version of the MaxWalkSat has been proposed and successfully applied to Markov logic networks [SG13]. Another recent extension of the MAP inference algorithms applied to Markov logic is the cutting plane approach [Rie08]. It is especially suitable for ILP since it maintains its exactness. The key idea is to progressively instantiate small parts of the Markov logic network and solve them. These small parts are iteratively selected based on the next most violated constraints.

From a knowledge engineering perspective, assigning adequate weights to Markov logic formulae is a particularly important and delicate task. In terms of probability, the effect of the weight of a single ground formula can be intuitively understood. Nonetheless, it might become too difficult to determine as soon as the same ground atoms appear in more than one formula. In the simplest case of a model with one single weighted ground formula (F_g, w_g) and according to Formula 2.4, the ratio between the probability of a world x_1 where F_g is true and a world x_2 where it is false is $\frac{P(x_1)}{P(x_2)} = \exp(w_g)$. Hence, the weight w_g of a formula corresponds to the log odds between the two words; $w_g = \log(\frac{P(x_1)}{P(x_2)})$.

However, even in the simplest applications, the same atoms usually appear in more then one single formula. In such a case, reversing a formula F would implicate changes in every other formula with common variables. The ratio between the probability of a world satisfying a formula F and one which does not is no longer uniquely dependent on the weight of F. The one-to-one correspondence between

the weights and the formulae would not hold. Instead, the formulae weights are **collectively** determined by the probabilities of all formulae. Since the interactions between the model's formulae are typically hard to predict, their weights should rather be seen as empirical probabilities. Instead of specifying them manually based on the logical form of the model, maximum likelihood weights should be estimated from data [RD06].

Like the majority of the training algorithms for log-linear models, learning formulae weights for Markov logic networks can be realised with standard learning methods based on the gradient of the conditional likelihood function. The model can be trained generatively by maximizing the (pseudo-) log-likelihood of the relational data base or discriminatively trained by maximizing the conditional log-likelihood (CLL) of the query predicates given the evidence ones [SD05]. Since pseudo-likelihood is consistently outperformed by discriminative training [LD07], we provide more information about the latter.

Given a collection of relational databases with the closed world assumption [DR04], the model's weights can be updated using the gradient **g** scaled by a learning rate η as indicated in Equation 2.6.

$$w_{t+1} = w_t - \eta \mathbf{g} \tag{2.6}$$

The derivative of the CLL with respect to weights can be calculated as follows.

$$\frac{\partial}{\partial w_j} \log p(y|x;w) = n_j(x,y) - \frac{\partial}{\partial w_j} \log Z \tag{2.7}$$

$$= n_j(x,y) - \frac{1}{Z} \sum_{y'} \frac{\partial}{\partial w_j} \exp \sum_{j'} n_{j'}(x,y') w_{j'} \tag{2.8}$$

$$= n_j(x,y) - \frac{1}{Z} \sum_{y'} [\exp \sum n_{j'}(y')] n_j(x,y') \tag{2.9}$$

$$= n_j(x,y) - \sum_{y'} n_j(x,y') \frac{\exp \sum_{j'} w_{j'} n_{j'}(x,y')}{\sum_{y''} \exp \sum_{j''} w_{j''} n_{j''}(x,y'')} \tag{2.10}$$

$$= n_j(x,y) - \sum_{y'} n_j(x,y') p(y'|x;w) \tag{2.11}$$

$$= n_j(x,y) - E_w[n_j(x,y')] \tag{2.12}$$

where we denote by $n_j(x,y)$ the number of true groundings of the formula F_j in a world with assignment x for the observed atoms (evidence) and y for the hidden ones. Thus, the partial derivative with respect to the jth weight w_j is the number of true groundings of the jth formula in the data minus its expectation according to the current model. The expected number of true groundings of F_j is calculated over the given training databases.

Algorithm 1 Voted perceptron algorithm for Markov logic with T epochs

$w_0 \leftarrow 0$
$epochWeight_0 = 0$
for $t \leftarrow 1...T$ **do**
  for $i = 1...N$ **do**
    $y_{MAP} \leftarrow ILP(x, y)$
    $w_i \leftarrow w_{i-1} + \eta[n(y_{CurrentModel}) - n(y_{MAP})]$
  end for
  $epochWeight_t = \frac{1}{N} \sum_{i=1...N} w_i$
end for
return $\frac{1}{T} \sum_{t=1...T} epochWeight_t$

Since counting the expectations $E_w[n_j(x, y')]$ is intractable [DR04], the **voted perceptron** algorithm [SD05] has been applied to train markov logic networks by approximating the expected number of true groundings with the most probable state of the non-evidence atoms given the evidence one [LD07]. Originally used to train Hidden Markov Models, voted perceptron was shown to deliver good results for Markov logic networks by replacing the orginal viterbi algorithm by a Markov logic MAP inference method [LD07]. For instance, the voted perceptron algorithm depicted below employs ILP as MAP inference method. Obviously, the algorithm is flexible in terms of the MAP inference method to be applied: ILP can be, for example, substituted by the MaxWalkSAT algorithm.

Algorithm 1 presents the pseudo-code of the voted perceptron for Markov logic. The models' weights are encoded in a vector w which is estimated by iterative updates using gradient ascent as explained above. The weights vector is updated by passing though the N training databases. At each training database i, the update rule introduced in Equation 2.6 is applied. The vector $n_i(y_{CurrentModel})$ represents the numbers of true groundings of the current model's formulae in the database i and $n_i(y_{MAP})$ the vector of the most probable state of world given the evidence(MAP) using the current weights vector w_i. The whole process (i.e. epoch) is repeated T times. As indicated by the last step of the algorithm, the final weight vector correspond to the average of the weights' vectors from all iterations and all databases. This reduces the risk of over-fitting.

To further optimize this learning algorithm, Lowd and Domingos [LD07] have proposed to address the problem of ill-conditioning of the data and its effect on the learning efficiency. Instead of using the same learning rate η for all weights, they assign a different one to each of them. The per weight learning rate η_j is defined as the ratio between a global learning rate η and the number n_j of true groundings of the related formula F_j. Thus, they reduces the effect of the variance of counts between formulae on the convergence speed of the gradient ascent algorithm. The new weight update rule is depicted in Equation 2.13.

$$w_{t+1} = w_t + \frac{\eta}{n_j}\mathbf{g} \tag{2.13}$$

Besides voted perceptron, other more sophisticated methods have been proposed to learn Markov logic weights. These includes multiplying the gradient **g** by the inverse Hessian for a faster convergence (diagonal Newton method) and the scaled conjugate gradient. The reader is invited to check the work of Lowd and Domingos [LD07] and that of Huynh and Mooney [HM11] for more details.

Given training databases, it is also possible to learn a model's formulae automatically or improve manually specified ones. This can be achieved with inductive logic programming (ILP) techniques. Learning Markov logic network structure goes beyond the scope of this thesis. For further information, we refer to the work of Kok and Domingos [KD05]

In general, Markov logic networks are especially appealing for applications with complex multi-relational data and significant apriori knowledge. Recognizing human activities integrates substantial common-sense knowledge which also includes complex temporal and non-temporal associations between the activities, the sensors and the user's context. The flexibility of Markov logic networks in representing such complex data with highly generic structure in a rigorous predicate approach makes particularly attractive.

2.2 Knowledge representation

We argued that an effective activity recognition framework should support easy, intuitive and flexible knowledge engineering. It should be robust to the different environments and settings by providing semantics rather than an appearance model [SKA+13]. It should also be expressive enough to model complex temporal and non-temporal relations and constraints between different entities. This is a crucial criteria to address interleaved and concurrent activities. Finally, an effective activity recognition framework should handle both certain and uncertain knowledge in a sound probabilistic manner. In the following, we describe our Markov logic based activity recognition framework which meets these requirements.

As introduced in the first part of this thesis, each human activity triggers particular **sensor events**. This sensor data is usually collected as sensor values with corresponding **timestamps**. In our framework we are interested in recognizing the activities which occurred at each of these timestamps. In the rest of this document, we use a simplified representation of these original timestamps and refer to the simplified values with "time steps".

Recall that we distinguish between *foreground activities* and *background activities*. Given a sensor event s at a time step t, a foreground activity at time step t is the activity that triggered s. At time step t, we designate by *background activity* each activity running in the background and is not necessarily involving the

user's interaction. As an example, let us assume that a subject, who is living in a smart-house, starts "preparing tea" at time step t. They first "use the electric water boiler" at t. Then, while the water is boiling, they "answer the phone" at time step $t + 1$. At time step $t + 2$ and $t + 3$ they resume "preparing tea" by "pouring the hot water" into the teacup. In this scenario, the triggered sensors would capture the interaction with the "electric water boiler" at t, the "phone" at $t + 1$, the "teacup" at $t + 2$ and the "electric water boiler" at $t + 3$. Following our terminology, we identify:

- Time step t: one *foreground activity*, "prepare tea" and no *background activities*.
- Time step $t + 1$: one *foreground activity* "answer the phone" and one *background activity* "prepare tea".
- Time step $t + 2$ and $t + 3$: one *foreground activity*, "prepare tea" and no *background activities*.

To show the viability of our approach to address the defined problem statement, we propose three models.

The **Basic Model** and the **Start-End Model** focus on intra-and inter-activity temporal relationships as well as the relevance of the integration of common-sense knowledge on the overall performance. While the **Basic Model** is designated to recognize the *foreground activity* of the subject at each time step, the **Start-End Model**, introduces implicit time intervals by detecting the start and ending points of each activity. This extension allows the recognition of both the *foreground activity* and the *background activities* of the user.

Leveraging the rich object-relational structure and related common-sense knowledge, the **States-Based Model** sheds some light on the viability of Markov logic networks to model and reason with uncertain relational data to recognize interleaved and concurrent fine-grained human activities. Unlike the two first models, the **States-Based Model** is designed to infer these activities from the user's low-level actions associated with their object interaction

We begin by introducing the models predicates. These represent binary variables defining the set of sensors and activity events as well as temporal features. An example of such a predicate would be $currentActivity(activity, timestep)$. Given the set of possible activities and time steps, the truth value of each grounding indicates whether a particular activity is being actively carried out at a specific time steps. The introduced predicates are used to determine a set of rules to infer the user's activities from the sensor observations.

2.2.1 Representing temporal events

Tables 2.1, 2.2, and 2.3 describe the core predicates used in each model respectively. The predicates are grouped into observable, i.e. can be observed from the

Table 2.1: Core Predicate used in the **Basic Model**.

Hidden Predicates		
currentActivity(a,t)	Activity a, Timestep t	Indicates whether activity a is being carried out at time step t
Observable Predicates		
sensor(s,t)	Sensor event a, Timestep t	Indicates whether sensor s is fired at time step t

input sensor data or hidden, i.e. have to be inferred. The first column depicts the predicate name, the second column determines the parameters accepted by the predicate and the last column provides a short description of the encoded semantics.

In our approach, we propose to explicitly model event instances with integer *time steps* as shown in the predicate tables. This offers a high flexibility in manipulating temporal information. Qualitative temporal relationships-such as "*after*" and "*before*"- can be detected by comparing time steps, whereas qualitative temporal information-such the *gap between* two events- can be treated by simple arithmetic operations applied to them.

As a step towards richer temporal relationships, the **Start-End Model** associates time steps with the *start/end* points of an activity by extending the predicate set with "startActivity(activity, timestep)" and "endActivity(activity, timestep)". The activity events are thus treated as *intervals*, which enables the recognition of both *foreground* and *background* activities as explained in the Recognition Framework Section.

Operating with intervals allows to capture the semantics of every possible temporal relationship between two event instances. These relationships and their algebra have been defined in Allen's interval algebra [All83] and can be easily modelled though adding further predicates and generic reasoning rules.

Since the state of the user's environment, and more specifically their surrounding objects, helps restrict the set of possible activities, the **States-Based Model** introduces this additional context. Thanks to the logical reasoning supported by our framework, it is easy to infer that an object remains in the same state *until* the user changes that state. We model this additional temporal information by adding the hidden predicate "currentObjectState(action, timestep)" as explained in the next sections.

2.2.2 Knowledge base

As noted above, a sequence of human activities usually exhibits significant structure and inherent common-sense knowledge. For example, our activities generally follow a typical routine in weekdays, while they might differ during the weekends and special occasions. Such background covers both *certain* and *uncertain* knowl-

Table 2.2: Core Predicate used in the **Start-End Model**.

Hidden Predicates		
currentActivity(a,t)	Activity a, Timestep t	Indicates whether activity a is being carried out at time step t
startActivity(a,t)	Activity a, Timestep t	Indicates whether activity a has started at time step t
endActivity(a,t)	Activity a, Timestep t	Indicates whether activity a has ended at time step t
Observable Predicates		
sensor(s,t)	Sensor event a, Timestep t	Indicates whether sensor s is fired at time step t

edge and can help differentiate between activities sharing a large set of similar sensor data. Certain knowledge could be, for example, that "eating a meal would not take place before preparing it" or that "going to work does not take place before dressing up". As uncertain knowledge we know, for instance, that "a person would usually take a shower after exercising and not before it" and that "they would brush their teeth before going to bed". In our models, we distinguish between three classes of formulae.

Abduction axioms are *soft* formulae that capture the dependency between the sensor events and the activities which triggered them. Since complex activities usually involve a sequence of sensor events rather than one single observation, an abduction formulae can, for example, link an activity at time step t to the sequence of sensor events at time steps $(t-1)$ and (t).

Temporal axioms are *soft* and *hard* formulae that express temporal relationships between the user's activities. Especially, the transition probabilities between activities play a crucial role, as explained in the previous paragraph. These transition probabilities can be easily learnt using soft rules manipulating successions of *end* and *start* points of activities. *Hard* temporal formulae in our model incorporate definite temporal constraints. As illustrated above, this can impose particular activity ordering such as stating that "if preparing a meal is occurring at time step t and eating that meal is taking place at time steps d then $d \geq t$".

General constraints are *hard* formulae about the model's predicate that assure its overall logical consistency. For instance, if an activity is being executed at timestamps t then it must have started at a timestamps d, where $d \leq t$.

Table 2.3: Core Predicate used in the **States-Based Model**.

Hidden Predicates		
currentActivity(a,t)	Activity a, Timestep t	Indicates whether activity a is being carried out at time step t
currentObjectState(c,t)	Action c Timestep t	Indicates whether the state of the object manipulated by action c is valid at time step t
Observable Predicates		
currentAction(c,t)	Action c, Timestep t	Indicates whether action c takes place at time step t
actionOpen(c)	Action c	Indicates whether action c corresponds to opening an object
actionClose(c)	Action c	Indicates whether action c corresponds to closing an object
hasWashableObject(c)	Action c	Indicates whether the object involved by action a is a washable object
hasFreshEntity(c)	Action c	Indicates whether the object involved by action c is a fresh entity
hasInDrawerEntity(c)	Action c	Indicates whether the object involved by action c belongs to the kitchen's drawers
hasObjectDrawer(c)	Action c	Indicates whether action c manipu- lates one of the the kitchen's drawers
haveSameObject(c, a)	Action c, Activity a	Indicates whether action c and activity a manipulate the same object
haveSameActionObject(c_1, c_2)	Action c_1, Action c_2	Indicates whether two actions c_1 and c_2 manipulate the same object

2.2.3 Activity recognition models based on Markov logic networks

Based on the set of predicates and the three formulae categories defined in the previous section, we present and explain the entire set of rules of our models in this section. For the sake of clarity, we present the model's axioms in pure Markov logic.
Note that the presented models are oriented towards activities from general daily routines, with a focus on kitchen-related activities. Given the adequate knowledge, the models can be adapted to other application domains.

The Basic Model: *inferring foreground activities*

Our first model aims at recognizing the *foreground activity* of the user at each time step. Especially, we focus on demonstrating the relevance of common-sense knowledge in improving the overall recognition accuracy. Recall from the previous section that we only have to incorporate one *hidden* predicate ($currentActivity(a, t)$) in this formulation. For the observable data we define one predicate, $sensor(s, t)$, modelling that sensor s has been triggered at time step t. Since the sensors bear the names of the objects they tag, such a predicate indicates that the user is using object s at time step t. **The Basic Model** consists of a set of 7 formulae depicted in Table 2.4 and illustrated in Figure 2.2.

The first two formulae are general constraints citing that (1) the user has to be carrying out at least one activity at each sensor event (2) unlike *background* activities, the user cannot be involved in more than one *foreground* activity at a time. Thus, these constraints together assure the condition of having *exactly* one *foreground activity* at each sensor observation.

Formulae (3) and (4) are weighted abduction axioms capturing the dependencies between the activities and the corresponding sensors. Whereas the first indicates the dependency level between the *foreground activity* and the sensor observation within the same time slice, the second extends this dependency with the previous sensor observation. As example consider the case where the user is currently interacting with a "spoon" during a morning routine. At that time point, many interpretations are possible such as "preparing oatmeal" or "making tea" for example. However, if the preceding observation indicates that the subject is interacting with "sugar" for instance, the activity "making tea" will be more probable and hence inferred as the current *foreground* activity.

Finally, Formulae (5), (6) and (7) incorporate some common-sense knowledge related to our application domain. These *hard* temporal axioms constraint the temporal order of an example of three activities sharing a particularly large set of common sensors in order to alleviate the ambiguity of their interpretation. Such ambiguity would usually lead to several recognition problems. Nonetheless, this sample common-sense knowledge helps to avoid most of these problems as shown by our experiments below. Formula (5) states that the activity of "eating breakfast" precedes the activity of "clearing the table". Formula (6) and (7) express that

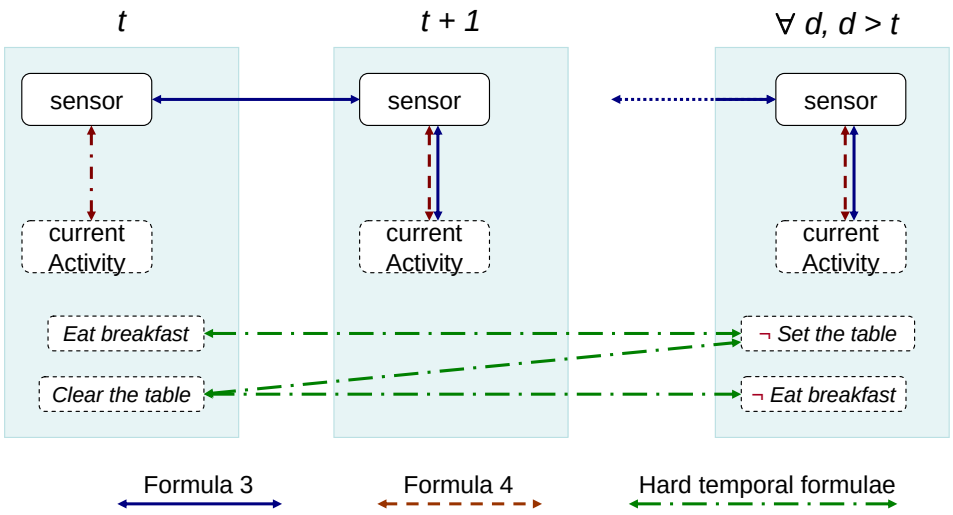


Figure 2.2: The figure illustrates *soft* and *hard* formulae modelling the abduction and temporal axioms of the **Basic Model**. Three time slices are depicted: $t-1$, t and d where $d > t$. In each time slice the corresponding predicates are represented by boxes labelled with their name. Boxes with dotted contour design hidden predicates while plain boxes correspond to observable predicates. Formulae (3) and (4) represent the model's abduction axioms capturing the dependencies between the activities and the corresponding sensors. The hard temporal formulae define qualitative temporal constraints on three activities: "$setting\ the\ table$", "$eating\ breakfast$" and "$clearing\ the\ table$".

the activity of "setting the table" precedes both activities of "eating breakfast" and "clearing the table".

Given these rules, predicting the *foreground activity* at a particular time step t is thus equivalent to computing the MAP state of the ground Markov logic network.

The Start-End Model: *inferring foreground and background activities*

We refer to our second model as the **Start-End Model**. The model aims at capturing long-range qualitative *and* quantitative temporal relationships between the sensor observations and the activities in order to recognize both *foreground* and *background activities* at each time step. More precisely, the model enables the recognition of the start and end points of an activity a using two additional predicates $startActivity(a,t)$ and $endActivity(a,t)$ (see Table 2.2). Thus, the states of three hidden predicates are inferred jointly, which potentially improves the overall recognition. The inference is based on the set of formulae depicted in Table 2.5. The formulae are illustrated in Figure 2.3 to reflect the intuitive and declarative aspect offered by the Markov Logic framework.

Additionally to the general constraints defined in the **Basic Model**, two further formulae: 3 and 4 are inserted. These ensure that an activity does not occur after it ends nor before it starts.

Table 2.4: Set of formulae for the **Basic Model**.

	General Constraints
1	$\forall\, Timestep\, t,\ \exists Activity\, a : currentActivity(a, t)$
2	$\forall\, Timestep\, t,\ Activity\, a_1,\ Activity\, a_2\ [a_1 \neq a_2] :$ $[currentActivity(a_1, t) \Rightarrow \neg\, currentActivity(a_2, t)]$
	Abduction Axioms
3	$\forall\, Sensor\, s,\ Timestep\, t,\ Activity\, a :$ $sensor(s, t) \Rightarrow [currentActivity(a, t)]$
4	$\forall\, Sensor\, s_1,\ s_2,\ Timestep\, t,\ Activity\, a :$ $[sensor(s_1, t) \wedge sensor(s_2, t + 1)] \Rightarrow currentActivity(a, t + 1)$
	***Hard* Temporal Axioms**
5	$\forall\, Timestep\, t_1,\ t_2 :$ $[t_2 \geq t_1 + 1] \Rightarrow [currentActivity(\text{“}Clear\, The\, Table\text{”}, t_1)$ $\Rightarrow \neg currentActivity(\text{“}Eat\, Breakfast\text{”}, t_2)]$
6	$\forall\, Timestep\, t_1,\ t_2 :$ $[t_2 \geq t_1 + 1] \Rightarrow [currentActivity(\text{“}Clear\, The\, Table\text{”}, t_1)$ $\Rightarrow \neg currentActivity(\text{“}Set\, The\, Table\text{”}, t_2)]$
7	$\forall\, Timestep\, t_1,\ t_2 :$ $[t_2 \geq t_1 + 1] \Rightarrow [currentActivity(\text{“}Set\, The\, Table\text{”}, t_2)$ $\Rightarrow \neg currentActivity(\text{“}Eat\, Breakfast\text{”}, t_1)]$

Formulae 6, 7 and 8 are *soft* formulae depicting the model's abduction axioms. The first one (formula 6) models the dependencies between sensor observations and activities within the same time slice. The two others formalize that the first and last pair of objects used during an activity are a good indicator for its start and end points respectively.

Finally, the **Start-End Model** expands the temporal common-sense background knowledge included in the **Basic Model** to also cover the start and end points of the activities (formulae 15, 16 and 17). This extension comprises three *hard* formulae stating that for the transition from activity "$eating\ breakfast$" to "$clearing\ the\ table$" to take place, the first one has to end and the second one has to start. This information helps reinforce the prediction of the correct transition between these activities.

Even though the last one among these three formulae would always be true given formula 2, we included it in the model to the sake of an intuitive comprehension of the common-sense knowledge incorporated in this model.

The model also encloses more general temporal relationship capturing the transition probabilities between activities as well as their temporal order. Concretely, the *soft* temporal formula 9 captures the likelihood that an activity a_1 is followed by a different activity a_2.

Besides the activities succession model, one further common aspect of daily routines is that some activities are usually carried out in a structured and almost the same manner. These activities implicate the same actions from their starting point to their end point. In a common daily living scenario there are many examples of such activities like taking a shower for instance. In the dataset under consideration, we depict such a structure in activities "$Making\ juice$" and "$Setting\ the\ table$" where the number of events separating their beginning and end time points hardly varies. We refer to this number as *duration*. The *soft* formulae 10 and 11 attribute a fixed *duration* to each of these two activities in order to highlight the benefit of **quantitative temporal features** on the recognition quality. The *duration* of these activities is hence fixed to its most probable value (7 and 15 respectively) and employed to reinforce the recognition of their start and end points. Even if assigning a weight to these formulae adds flexibility to this duration model, other more sophisticated formulation of the temporal quantitative features might be beneficial to improve the prediction of less structured activities of the set. This includes replacing fixed values with intervals for example.

The States-Based Model: *introducing entities' states to recognize fine-grained activities*

The focus of this model is to leverage domain-knowledge structure as well as entities states over time to recognize activities. The model is applied to the problem of recognizing fine grained activities such as "get cheese" and "put away cheese" from simpler actions such as "open fridge" and "fetch cheese". Thus, the state of objects the user interacts with is a significant indicator for the activity being

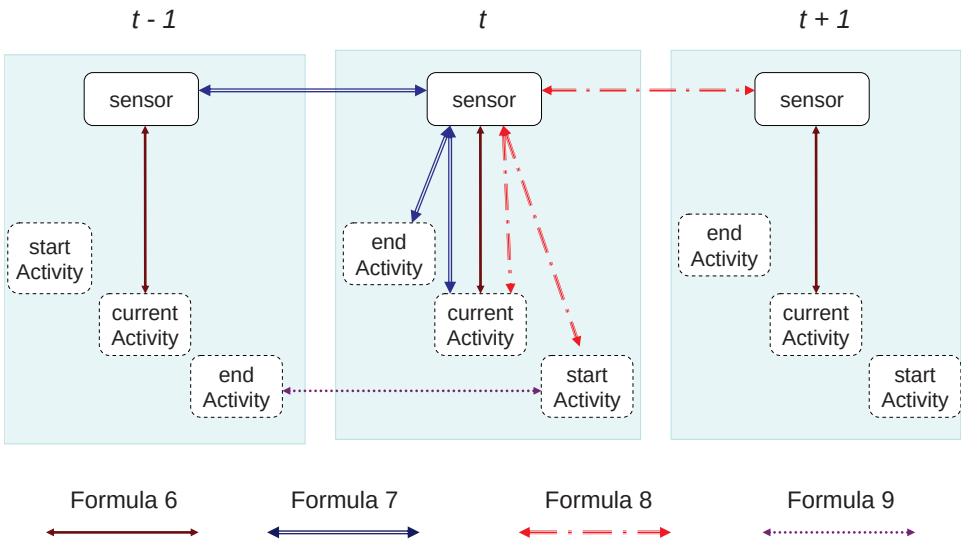


Figure 2.3: Illustration of the *soft* formulae of the **Start-End Model**. Predicates are represented by boxes labelled with their name. Boxes with dotted contour design hidden predicates while all other boxes correspond to observable predicates. Formula (6) models the dependency between the activities and the corresponding sensors. Formula (7) formalizes that the first pair of activated objects during an activity are a good indicator for its start time. Analogously, formula (8) models that the last pair of activated objects during an activity is a good indicator for the activities end time. Finally, formula (9) models the temporal relationship between activities, that is, the likelihood that an activity a_1 is followed by a different activity a_2 [HNS11b]

carried out. For instance, knowing that the "fridge" is "open" and that the user is grasping a fresh item (such as cheese), would highly favour the activity of "getting cheese" rather than "preparing sandwich" as being the current activity of the user. Inversely, omitting the state of the "fridge" increases the ambiguity of interpretation of the current action, and both activities "getting cheese" and "preparing sandwich" might become equally probable.

Deriving the state of an entity over an interval of time can be intuitively modelled in Markov logic. Indeed, an entity remains in the same state from timestep t to the following timestep $t+1$, as long as the action taking place at $t+1$ does not alter the state of the entity in questions. Such knowledge can be modelled by simple first order formulae as shown by the three ***hard* temporal axioms** 6, 7 and 8 in Table 2.6. Note that given the domain defined by the employed dataset, we are mainly interested in whether an object, such as a "drawer", "fridge" etc. is *open* at a given timestep t. This explains formula 6 which states that if the user is "opening" an object at timestep t, the state of that object corresponds to "open" at the same timestep. Formulae 7 and 8 extend the reasoning to the following timestep, i.e, $t+1$ as long as the user does not "close" the *same object*. Thus, the derived state spreads over a chain of consecutive timesteps until an action occurs to change

it.
In order to recognize the user's activities from simpler actions, we propose 5 **abduction axioms** as specified in Table 2.6. The first formula captures the conditional dependency between activities and actions belonging to the same time step. Unlike the previous two models, we relax the restriction of having exactly one activity at each time step. Manifestly, this renders the model more flexible and more general. However, the resulting recognition task is more ambiguous. Since the model aims at recognizing activities from simpler ones, namely actions, the sequence of these actions plays a crucial role in distinguishing between activities sharing a significant set of common actions such the activities "get milk" and "put away milk". Thus, an activity is composed of a sequence of actions. This knowledge is formalized by the second abduction axiom in Table 2.6.
The remaining three axioms (3, 4 and 5) are added to reinforce the recognition of specific activities, where the state of related objects is a strong indicator for them. More precisely, formula 3 is designed to boost the recognition of the activities of putting or getting an item from the fridge such as"milk" for instance. These activities are described as those where the user interacts with a "fresh" entity, while the "fridge is *open*". Thus, the weight of this formula captures the dependency between each action and activity which belong to the same time slice t and are consistent with the above description. Similarly, formula 4 addresses activities where the subject uses items which belong to one of the kitchen drawers while that "drawer is *open*". Finally, formula 5 handles the explicit case of the activity "put in dishwasher" which is linked to the occurrence of actions involving *washable* items while the "dishwasher is *open*". As apparent from these formulae, this model reflects a rich relational structure between the actions, activities and the related objects. These relations and attributes are intuitively represented with different predicates such as $hasSameObject(action, activity)$, $hasWashableObject(action)$ and $hasFreshEntity(action)$. Through these predicates, several actions and activities can be grouped to share predefined properties. This highlights the viability of MLN to handle the complexity of this domain of discourse.

Table 2.5: Set of formulae for the **Start-End Model**.

	General Constraints
1	$\forall\, Timestep\, t,\ \exists\ Activity\ a : currentActivity(a, t)$
2	$\forall\, Timestep\, t,\ Activity\, a_1,\ Activity\ a_2 :$ $[a_1 \neq a_2] \Rightarrow [currentActivity(a_1, t) \Rightarrow \neg\, currentActivity(a_2, t)]$
3	$\forall\, Timestep\, t_1,\ t_2,\ Activity\, a :$ $startActivity(a_1, t) \Rightarrow \neg\, endActivity(a_2, t)$
4	$\forall\, Timestep\, t_1,\ t_2,\ Activity\, a : [t_1 \geq t_2]$ $\Rightarrow [currentActivity(a, t_1) \Rightarrow \neg\, endActivity(a, t_2)]$
5	$\forall\, Timestep\, t_1,\ t_2,\ Activity\, a : [t_1 \leq t_2]$ $\Rightarrow [currentActivity(a, t_1) \Rightarrow \neg\, startActivity(a, t_2)]$
	Abduction Axioms
6	$\forall\, Sensor\, s,\ Timestep\, t,\ Activity\, a :$ $sensor(s, t) \Rightarrow [currentActivity(a, t)]$
7	$\forall\, Sensor\, s_1,\ s_2,\ Timestep\, t,\ Activity\, a :$ $[sensor(s_1, t) \wedge sensor(s_2, t+1)]$ $\Rightarrow [endActivity(a, t+1) \wedge currentActivity(a, t+1)]$
8	$\forall\, Sensor\, s_1,\ s_2,\ Timestep\, t,\ Activity\, a :$ $[sensor(s_1, t+1) \wedge sensor(s_2, t)]$ $\Rightarrow [startActivity(a, t) \wedge currentActivity(a, t)]$
	***Soft* Temporal Axioms**
9	$\forall\, Timestep\, t,\ Activity\, a_1,\ a_2 : endActivity(a_1, t) \wedge$ $startActivity(a_2, t+1) \wedge currentActivity(a_2, t+1)$
10	$\forall\, Timestep\, t_1 : startActivity(\text{“}MakeJuice\text{”}, t_1)$ $\Rightarrow [endActivity(\text{“}MakeJuice\text{”}, t_1 + 7)$
11	$\forall\, Timestep\, t_1 : startActivity(\text{“}Set\, Table\text{”}, t_1)$ $\Rightarrow [endActivity(\text{“}Set\, Table\text{”}, t_1 + 15)$
	***Hard* Temporal Axioms**
12	$\forall\, Timestep\, t_1,\ t_2 :$ $[t_2 \geq t_1 + 1] \Rightarrow [currentActivity(\text{“}Clear\, The\, Table\text{”}, t_1)$ $\Rightarrow \neg currentActivity(\text{“}Eat\, Breakfast\text{”}, t_2)]$
13	$\forall\, Timestep\, t_1,\ t_2 :$ $[t_2 \geq t_1 + 1] \Rightarrow [currentActivity(\text{“}Clear\, The\, Table\text{”}, t_1)$ $\Rightarrow \neg currentActivity(\text{“}Set\, The\, Table\text{”}, t_2)]$
14	$\forall\, Timestep\, t_1,\ t_2 :$ $[t_2 \geq t_1 + 1] \Rightarrow [currentActivity(\text{“}Set\, The\, Table\text{”}, t_2)$ $\Rightarrow \neg currentActivity(\text{“}Eat\, Breakfast\text{”}, t_1)]$
15	$\forall\, Timestep\, t_1 : currentActivity(\text{“}Eat\, Breakfast\text{”}, t_1) \wedge$ $currentActivity(\text{“}Clear\, The\, Table\text{”}, t_1 + 1)$ $\Rightarrow endActivity(\text{“}Eat\, Breakfast\text{”}, t_1)]$
16	$\forall\, Timestep\, t_1 : currentActivity(\text{“}Eat\, Breakfast\text{”}, t_1) \wedge$ $currentActivity(\text{“}Clear\, The\, Table\text{”}, t_1 + 1)$ $\Rightarrow startActivity(\text{“}Clear\, The\, Table\text{”}, t_1 + 1)]$
17	$\forall\, Timestep\, t_1 : currentActivity(\text{“}Eat\, Breakfast\text{”}, t_1) \wedge$ $currentActivity(\text{“}Clear\, The\, Table\text{”}, t_1 + 1)$ $\Rightarrow \neg currentActivity(\text{“}Eat\, Breakfast\text{”}, t_1 + 1)]$

Table 2.6: Set of formulae for the **States-Based Model**.

	Abduction Axioms
1	$\forall\, Action\, c,\, Timestep\, t,\, Activity\, a:$ $currentAction(c,t) \Rightarrow [currentActivity(a,t)]$
2	$\forall\, Action\, c_1,\, c_2,\, Timestep\, t,\, Activity\, a:$ $[currentAction(c_1,t) \wedge currentAction(c_2,t+1)]$ $\Rightarrow [currentActivity(a,t) \wedge currentActivity(a,t+1)]$
3	$\forall\, Action\, c,\, Timestep\, t,\, Activity\, a:$ $[currentAction(c,t) \wedge currentObjectState(\text{“}OpenFridge''{,}t)$ $\wedge hasFreshEntity(c) \wedge haveSameObject(c,a)]$ $\Rightarrow [currentActivity(a,t)]$
4	$\forall\, Action\, c_1, c_2\, Timestep\, t:$ $[currentAction(c_1,t) \wedge currentObjectState(c_2,t)$ $\wedge hasInDrawerEntity(c) \wedge actionOpen(c_2)$ $\wedge hasObjectDrawer(c_2)]$ $\Rightarrow [currentActivity(a,t)]$
5	$\forall\, Action\, c,\, Timestep\, t,\, Activity\, a:$ $[currentAction(c,t)$ $\wedge\, currentObjectState(\text{“}OpenDishwasher''{,}t)$ $\wedge\, hasWashableObject(c)]$ $\Rightarrow [currentActivity(\text{“}PutIntoDishwasher''{,}t)]$
	***Hard* Temporal Axioms**
6	$\forall\, Action\, c,\, Timestep\, t,: [currentAction(c,t) \wedge actionOpen(c)]$ $\Rightarrow [currentObjectSate(c,t)]$
7	$\forall\, Action\, c_1,\, c_1,\, Timestep\, t,:$ $[currentAction(c,t) \wedge \neg actionClose(c)]$ $\Rightarrow [currentObjectSate(c,t-1) \Rightarrow currentObjectSate(c,t)]$
8	$\forall\, Action\, c_1,\, c_1,\, Timestep\, t,:$ $[currentAction(c_1,t) \wedge actionClose(c_1)$ $\wedge \neg haveSameActionObject(c_1,c_2)]$ $\Rightarrow [currentObjectSate(c_2,t-1) \Rightarrow currentObjectSate(c_2,t)]$

3
Evaluation and Results

After presenting our models, we evaluate their performance under realistic settings and report the obtained results in this chapter. As already mentioned, the evaluation results of both the **Basic Model** and the **Start-End Model** were already released in [HNS11a] and [HNS11b]. Those of the **States-Based Model**, however, are unpublished. The evaluation scheme concerns the following tasks:

- Predict the *foreground* activity(ies) for each time step
- Infer start and end times of every activity
- Derive all *background* activities at each new sensor event
- Derive the states of surrounding objects and use these to recognize fine-grained activities

3.1 Recognition framework and experiments

We identify three major implementations for applying the introduced Markov logic networks: "*Alchemy*" [1], "*Tuffy*" [NRDS11], "*RockIt*" [NNS13] and "*Markov: TheBeast*" [Rie08]. We estimate the third framework to be most appropriate to our problem statement and defined models. This is justified by two main reasons.

The first is that, at the time of our experiments, only "*Markov: TheBeast*" allows for flawless arithmetic operations on integers, while the two others suffer from implementation bugs. This is a mandatory feature in our model in order to represent and reason with long-range qualitative and quantitative temporal relationships.

The second reason is that "*Markov: TheBeast*" framework provides a special "if a then b" syntax where a should contain only observable predicates while b

[1] http://alchemy.cs.washington.edu/

contains hidden ones. Unlike a weighted logical implication "$a \rightarrow b$", the weight of an expression in form of "if a then b" represents the degree to which a and b co-occur. Since a is always given as evidence, $P(a)$ is irrelevant to the model and by attaching a weight to $a \wedge b$ we represent the conditional probability of b given a. Based on this representation for probabilistic causal influence only groundings where a and b are $True$ are considered. Whereas all instances with $a = False$ would have been included in the case of a probabilistic logical implication.

This easily allows to create a discriminative model instead of a generative one. Just like the difference between conditional random fields and Markov random fields, the advantage of a discriminative Markov logic model is that it directly models the prediction problem $P(Activity|SensorObservations)$ instead of the full probability distribution $P(Activity, SensorObservations)$. As such, they are more accurate since they do not require the probability distribution over the input data (i.e. sensor observations).

Thus, the semantics of "*Markov: TheBeast*" [Rie08] require a clear distinction between hidden and observable predicates. The provided evidence should contain all the observable predicates and none of the hidden ones. If the truth value of an observable predicate is not explicitly specified, it is assumed to be false (closed-world assumption). Hence, specifying particular values of hidden variables is only possible in form a *hard* formulae.

3.1.1 Datasets

In our experiments, we have used three real-life datasets. The **Basic Model** and the **Start-End Model** were implemented using the *Intel dataset*, which consists of real data collected by Patterson $et\ al.$ [PFKP05]. The **States-Based Model** was applied to the "Opportunity dataset" [KHF+11], a part of the EU research project "Activity and Context Recognition with Opportunistic Sensor Configuration"[2]

The data provided by Patterson $et\ al.$ [PFKP05] was collected in a lab equipped with 60 RFID tags placed on different objects. The objects were involved in performing a set of eleven fine-grained activities depicted in Table 3.1. To detect the user's interaction with the objects, they wore two RFID gloves that triggered RFID tags within 2 inches, as shown in Figure 3.2. The data collection periods had a mean duration of 27 min per day on ten different days. The performed activities are highly interleaved in nature (see Figure 3.1) and are only performed once.

This dataset comprises two sets: "*standard data*" and "*full data*" The first is provided in form of timely ordered events relating, for each time step, the ID of active sensors and the activity being carried out. Unlike the standard data, the full data provides all concurrent activities for each time step. Given their purposes, we applied the **Basic Model** to the standard data and the **Start-End Model** to the full data.

The "*Opportunity dataset*" was collected in a highly rich networked smart

[2] http://www.opportunity-project.eu

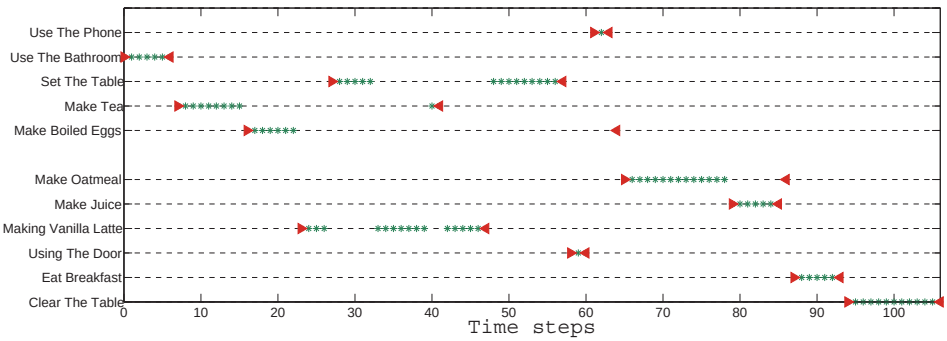


Figure 3.1: A sample from Patterson's assisted living dataset ([PFKP05]). The graph shows the highly interleaved nature of activities. The (red) triangles mark the begin and end times of each activity. The (green) stars indicate when an activity is in the foreground. For instance, up to four interleaved activities ("Making vanilla latte", "Make boiled eggs", "Make tea" and "Set the table") are in progress at time step 30.

room simulating a studio flat. A total of 72 sensors with 10 modalities were deployed in 15 wireless and wired networked sensor systems in the environment [KHF+11]. Several users participated in a naturalistic collection process of a *morning routine* [LPB+10]. As illustrated in Figure 3.3, the deployed sensors can be classified into wearable sensors such accelerometers and environmental sensors such as RFID tags and readers. The wearable sensors are used to infer body gestures like "reach" and "move" as well as modes of locomotion like "sit" and "lie". The environmental sensors detect the objects the user is interacting with. The dataset covers four levels of activity granularities. Our **States-Based Model** was applied to recognize the intermediate level referred to as "simple activities" from what is called "manipulative gestures" [HRS13]. As described by the **States-Based Model**, we refer to the input data ("manipulative gestures") as *actions* and the output ("simple activities") as *activities*, for the sake of simplicity. This recognition task is more complex than the two others (see Part II). Especially, the data was annotated by different persons and the resulting labels do not cover all the possible activities, leaving some sensor observations with no annotation. We use the data collected by three different participants $S10$, $S11$, and $S12$, with three different routines each ($ADL1$, $ADL2$, $ADL3$). The activities are interleaved and concurrent. A total of 21 different activities and 40 different actions are carried out in the dataset routines. Figure 3.3 depicts the sensing modalities used to collect the data: RFID tagged objects in the picture on the top and wearable sensors at the bottom picture. The collected activities are listed in Table 3.2.

3.1.2 Experimental settings

As mentioned above, we used "*Markov TheBeast*" [Rie08] to implement our recognition framework. We integrated the mixed integer programming solver Gurobi[3]

[3] http://www.gurobi.com/

Figure 3.2: RFID Glove worn by the used to detect their interaction with surrounding RFID-tagged objects in the **Intel dataset** [PFKP05]

1	Using the bathroom
2	Making soft-boiled eggs
3	Making vanilla latte
4	Setting the table
5	Using the door
6	Making a phone call
7	Preparing orange juice
8	Clearing the table
9	Making oatmeal
10	Making tea
11	Eating breakfast

Table 3.1: Activities collected in the **Intel dataset** [PFKP05]

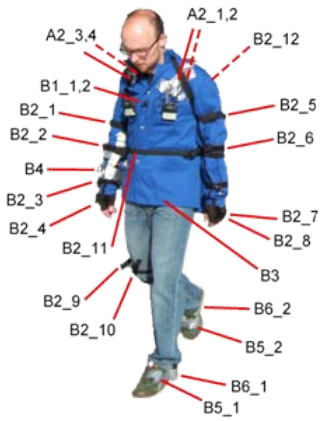


Figure 3.3: Sensing modalities in the **Opportunity dataset** [a]

[a] http://www.opportunity-project.eu

1	Put Sugar
2	Get Bottle
3	Put away Bottle
4	Get Milk
5	Put away Milk
6	Get Knife Salami
7	Get Knife Cheese
8	Get Cheese
9	Put away Cheese
10	Prepare Cheese Sandwich
11	Get Salami
12	Put away Salami
13	Get Bread
14	Put away Bread
15	Eat Bread
16	Get Plate
17	Drink from Glass
18	Drink from Cup
19	Put in Dishwasher
20	Lie on Lazychair
21	Prepare Salami

Table 3.2: Activities collected in the **Opportunity dataset** [LPB+10]

and applied it to the ILP instances for the MAP inference. The soft formulae weights were estimated via supervised on-line learning. In our experiments, we opted for discriminative learning since it was demonstrated to outperform generative learning for training Markov logic networks [SD05]. We applied 15 epochs of *perceptron rule*-based weight updates in the **Basic Model**, the **Start-End Model** and *margin infused relaxed algorithm (MIRA)* with 15 epochs in the **States-Based Model**. In the latter, the loss function is computed from the number of false positives and false negatives over the hidden atoms. *Markov: TheBeast* permits to learn the weight of first-order formulae as well as the weights of the individual groundings. This is especially relevant to capture the different dependencies between particular instances such as the activity $MakeTea$ and the object $Spoon$ for instance. To improve the learning of the weights, we omitted the input of identical successive sensor activity.

The models were tested using n-fold cross-validation. All experiments were conducted on a desktop PC with AMD Athlon Dual Core Processor 5400B with 2.6GHz and 1GB RAM.

3.2 Results and discussion

To evaluate the recognition performance of our system, we apply the $Precision$, $Recall$ and F_1 $score$ metrics explained in Preliminaries Chapter. Given that only one *foreground* activity can be carried out at each time step of the **Basic Model** and the **Start-End Model**, the resulting number of *false positives* and *false negatives* is identical in that case. This does not apply to the **States-Based Model** because the participants can be involved in two different activities simultaneously (due to wearing two RFID readers instead of one) or just in none.

In this section, we report and discuss the results of the described experiments. The results are organized per Model.

3.2.1 Inferring Foreground Activities: results of the *Basic Model*

Using the concise set of rules described in Table 2.4 we evaluate the **Basic Model** using leave-one out cross validation and the averaged recognition's precision and recall over the 10 morning routines confirm the viability of our approach by reaching an F1-measure of 93% with a standard deviation of $\sigma = 0.06$ (seeTable 3.3) . This slightly outperforms state-of-the-art approaches [HY08] applied on the same *standard data* of Patterson's dataset [PFKP05] where the authors report an accuracy of 92%. Note that, since exactly one activity is output at each time step, then each time step counts either as true positive or both a false positive and false negative. Thus, the total number false positive is equal to the total of false negatives and the sum of true positives is equal to the sum of true negatives. Thus, all of the precision, recall, F1-measure and accuracy have the same value.

To give an impression about the nature of the dataset, the results of our models

Table 3.3: Results for the recognition of *foreground* activities using the standard data for the **Basic Model** and the full data for the **Start-End Model**. The evaluation is computed from leave-one out cross validation.

	Basic Model	Start-End Model	Baseline I	Baseline II
Precision	$0.93(\sigma = 0.06)$	$0.92(\sigma = 0.03)$	0.31	0.16
Recall	$0.93(\sigma = 0.06)$	$0.92(\sigma = 0.03)$	0.31	0.16
F_1	$0.93(\sigma = 0.06)$	$0.92(\sigma = 0.03)$	0.31	0.16

are compared to those of the majority class baseline depicted in the third column (Baseline I) of the same Table 2.4. The majority class selects the activity with the highest number of occurrences from the training data and outputs it as the predicted current activity. The resulting performance is significantly lower.

An important aspect of our approach is to show the substantial effect of incorporating common-sense knowledge on the recognition accuracy. Figure 3.4 gives a detailed insight into this effect. Activities "$Set\,The\,Table$, $Eat\,Breakfast$", and "$Clear\,The\,Table$" share a particularly high number of common objects which makes their recognition ambiguous. Extending the model with the background knowledge, however, significantly improves their recognition. Especially, the average precision, recall and F1-score of the recognition of activities over 10 morning routines "$Set\,The\,Table$" (4) and "$Clear\,The\,Table$" (8) almost double. The overall F1-score raises from 0.88 to 0.93.

The potential of background knowledge to significantly improve the recognition of *foreground* activities is also confirmed by the results plotted in Figure 3.4. The figure unfolds the single precision values of each morning routine of the dataset and shows the robustness of the amelioration throughout the data.

3.2.2 Inferring Foreground and Background Activities: results of the *Start-End Model*

Recall that the **Start-End Model** uses the *full data* of Patterson's [PFKP05] dataset to jointly infer the foreground activities, the start point of the activities and their end points. Based on these, we first derive *Background* activities by assuming that an activity is still in progress from its predicted start point until its predicted end point. At each time step the set of predicted *background* activities is compared to that of the ground truth.

To reduce the complexity of the underlying ground model, we have omitted identical successive sensor data and used the resulting dataset for the evaluation of the **Start-End Model**. Note that it is straightforward to reconstruct the original sensor data including successive duplicate events by retaining the prediction results until the next new sensor event. Although we dropped these experiments here, we think that this evaluation most probably increases the precision and recall values

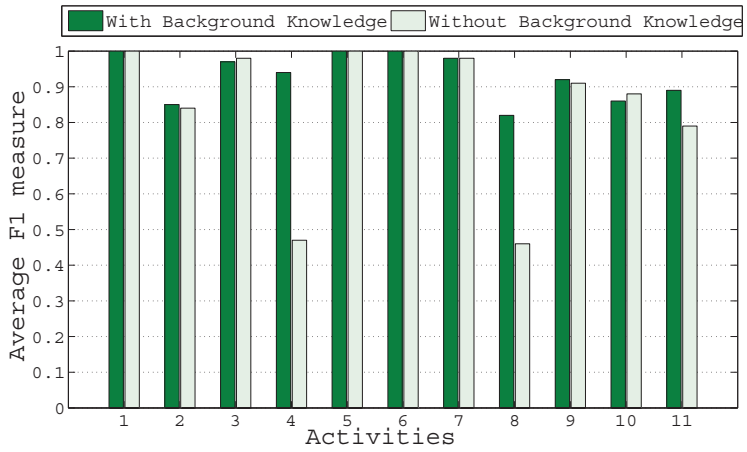


Figure 3.4: Average F_1 scores of **Basic Model** for the different activities with and without the common-sense constraints provided in Table 2.4. Table 3.1 shows the assignments of number to the individual activities. The recognition accuracy doubles for the activities "$Set\,The\,Table$" (4) and "$Clear\,The\,Table$" (8).

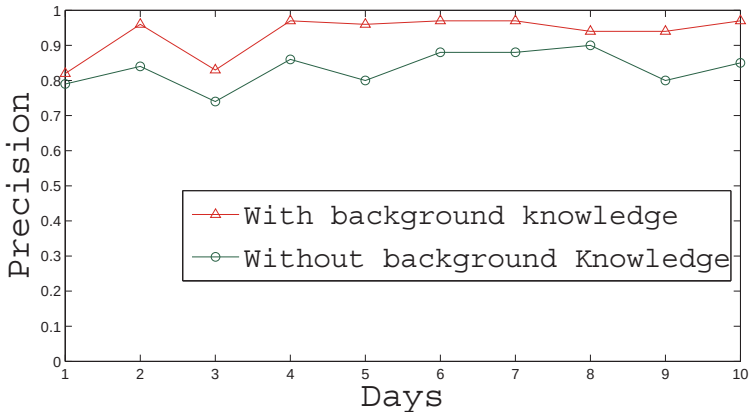


Figure 3.5: The average precision of **Basic Model** versus individual days with and without common-sense background knowledge.

compared to the reduced version of the *full data*.

Similarly to the **Basic Model**, our analysis of the evaluation results focuses on the importance of incorporating background knowledge for the overall recognition quality. First, we look into the precision, recall and F1-score of recognizing foreground activities with a subset of the formulae of the **Start-End Model** as depicted in the third row of Table 3.3. These values correspond the inference results of the *hidden* predicate $currentActivity(activity, timepoint)$ using the same common-sense knowledge as in the **Basic Model**. The values surpass 90% while applying the majority class baseline leads to an $F1$-score as low as 0.16. The particularly low standard deviation across the 10 routines constructing the dataset suggests a high robustness of the model.

Keeping the same subset of common-sense knowledge as in the *Basic Model*, we assess the recognition of *foreground* and *background* activities.

The inference precision, recall and F1-score of the three hidden predicates $currentActivity(activity, timepoint)$, $startActivity(activity, timepoint)$ and $endActivity(activity, timepoint)$ are summarised in Table 3.5. While the start points of the activities are recognized with a high recall and precision, recognizing the end points of the activities shows a clearly lower recall. This indicates that human activities might be initiated in a more consistent manner than they are terminated.

A crucial advantage of recognizing the start and end points of the activities is the ability to capture qualitative temporal relationships between the activities as stated by formula 9 in Table 2.5. The weights confirm the intuition that a daily "routine" preserves similar chronological order for some activities (the three first lines of the Table) and yet allows randomness for the others such as "*using the phone*".

Based on the inferred start and end points of the activities, we now evaluate the recognition of *background* activities. These are derived by being considered in progress during the interval separating their start and end time points. This derivation implicates the following cases.

Case I: false negatives of the predicate *endActivity(activity, timestep)* In case the end point of a particular activity is missing, the activity counts as a background activity for every subsequent timestep. For a more reliable evaluation method, we opt for this strict assumption despite the high number of false positives it induces.

Case II: false negatives of the predicate *startActivity(activity, timestep)* In the case the framework misses the starting point of an activity, we compare two different interpretation of this failure.

- **Assumption I** Each activity with a missing start point does not count among the *background* activities. Thus, if the activity is recognized as foreground activity at two distinct timesteps t and d ($t \leq d$), it still is not derived as

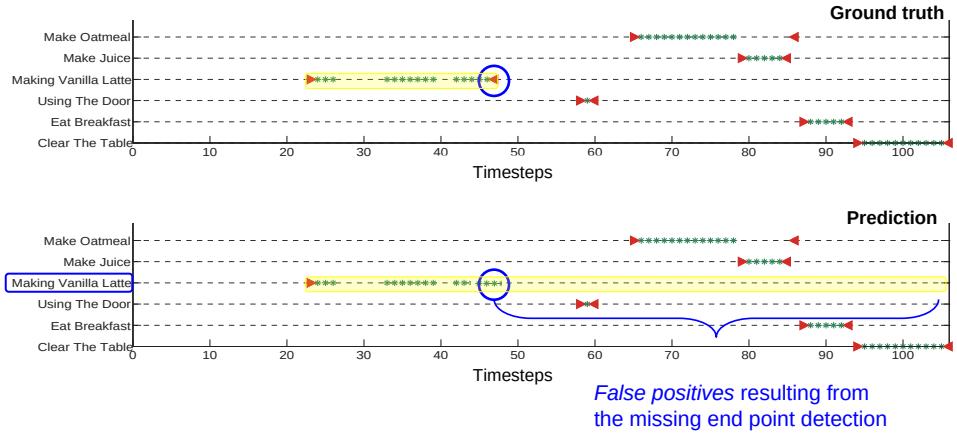


Figure 3.6: Illustration of a false negative for the predicate $endActivity$ and its impact on the overall recognition quality. Whereas the ground truth indicates that the activity $Making\ Vanilla\ Latte$ ends at time step 48, that event it not detected by the model. Thus, the activity is interpreted to continue as *background activity* along the remaining time steps till the end of routine. Obviously, this adds more than 50 false positives in the overall evaluation process

a background activity in during the intevall $[t, d]$. Thus, this strict assumption is expected to engenders several false negatives for the recognition of background activities.

- **Assumption II** Unlike the derivation of missing end points, it is easy to interpret the timestep of the first apparition of a particular activity in the inferred foreground activities as its starting point. In this assumption, if the model fails in detectin the starting point of an activity, we consider that activity "in progress" from its first apparition till its end.

Table 3.6 compares the recognition precision and recall for both assumptions. Independently of the assumption, the F1-score reached by the **Start-End Model** in recognizing both foreground and background activities jointly is higher than 80%. A closer look at Table 3.5 explains the lower values of precision compared to the very high recall. Indeed, the relatively low recall of inferring the predicate $endActivity(activity, timestep)$ indicates a higher number of activities with missing end points. As explained by our strict assumptions above, this results in a significant number of false positives of the corresponding activity. An example where the model fails in recognizing the end point of the activity "$MakingVanillaLatte$" is drawn in Figure 3.6.

Finally, we examine the evolution of the recognition performance by progressively enriching the a-priori domain knowledge within the model. As shown in Figure 3.7, incorporating further temporal axioms including new quantitative temporal relationships increases the F_1 score under both assumptions. The quantitative temporal relationships captured by the *soft* temporal axioms 10 and 11 in Table 2.5

Table 3.4: Some selected weights for successive activities. The activities can be interrupted by others. Higher weights are given for two activities a_1 and a_2 if a_2 starts when a_1 ends (see Table 2.5)

Activity A_1	Following Activity A_2	Average Weight
Eat Breakfast	Clear the Table	13.1
Make Oatmeal	Make Boiled Eggs	9.8
Make Tea	Eat Breakfast	1.6
Use The Phone	Set The Table	0.0

Table 3.5: Results for the recognition of the start and end points of the activities the full data for the **Start-End Model**. Both models rely on a reduced subset of background knowledge (rules 1-3). The evaluation is computed from leave-one out cross validation.

	Current	Start	End	Global
Precision	0.92	0.97	0.97	0.93
Recall	0.92	0.90	0.71	0.91
F_1	0.92	0.93	0.82	0.92

increase the recall values of the inference the hidden predicate $endActivity(activity, timepoint)$ from 0.71 to 0.76. It reinforces the inference of the start and end point for both "$Make\ juice$" and "$Set\ the\ table$" activities and thus explain the overall improvement in the recognition results. Since the dataset includes highly interleaved sequences of activities, we compared our results to a baseline which maximizes the recall value by always predicting all the activities as being in progress.

3.2.3 Inferring fine-grained interleaved and concurrent *foreground* activities: results of the *States-Based Model*

Whereas the previous two models mainly focus on representing and reasoning with inter-activities temporal axioms, this one features rich semantic information and highly-relational formulae including entity states over time. The overall performance of the model is evaluated using the same metrics explained above. Like in the **Start-End Model**, we address an event-based recognition task where we omit successive duplicates of input data as well as *null* events. Compared to the **Basic Model** and the **Start-End Model**, the **States-Based Model** does not require exactly one activity to be *actively* carried out at each time step. Instead, the participants can be "getting salami from the fridge" *and* "getting cheese from the fridge" at the same time point. They can even be performing a sequence of actions that does not correspond to any of the activities. Furthermore, the annotation of the available data was carried out offline by different persons [HRS13], which yields noteworthy deviations in how the actions' sequences are interpreted.

Table 3.6: Results for the recognition of both **background** and *foreground* activities using the full data for the **Start-End Model**. The evaluation is computed from leave-one out cross validation.

	Precision	Recall	$F1$-measure
Assumption I	0.73	0.99	0.84
Assumption II	0.71	0.99	0.82

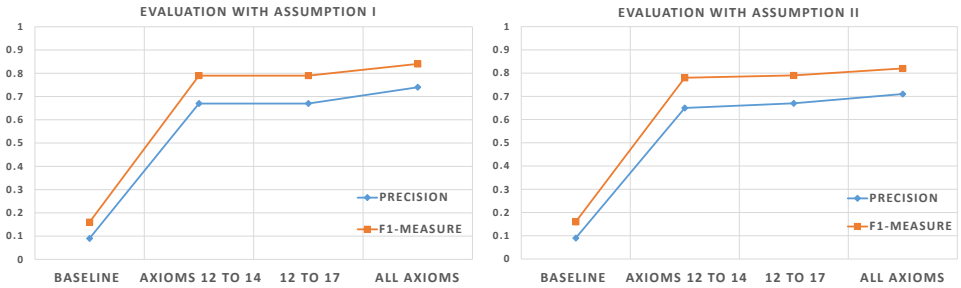


Figure 3.7: Results for the recognition of **concurrent activities** using the full data for the **Start-End Model**. The evaluation is computed from leave-one out cross validation. The concurrent activities have been derived under the **Assumption I** on the left and **Assumption II** on the right. The plot compares the recognition performance with different temporal relationships.

Two hidden predicates are defined in the **States-Based Model**: *currentObjectState(action, timestep)* and *currentActivity(activity, timestep)*. The first is an auxiliary predicate which is derived by three deterministic rules as depicted in Table 2.6. Consequently, it is always correctly inferred and will not be considered in the evaluation results. The second predicts the participant's foreground activity(ies) at a given time step. Since the user can be actively engaged in more than one *foreground* activity at a time, the calculation of the *true positives*, *false positives* and *false negatives* is realized following the same method described in the **Start-End Model**. Since our dataset was collected by three different participants $S10$, $S11$ and $S12$, with three routines each ($ADL1$, $ADL2$ and $ADL3$), we compute the results from user-independent leave-one cross validation and average the recognition precision and recall over the three routines. In other words, for each subject, we use their three routines as test data and train the system with the remaining six routines of the different participants. The final evaluation results correspond to the *average* of the evaluations of the three routines $ADL1$, $ADL2$ and $ADL3$.

To explore the dataset and give an impression on the discriminative degree of actions on activities, we first run a **baseline model** comparable to *logistic regression*. The model encodes only the first abduction axiom in Table 2.6, which then outputs the most probable activity given the current action. As summarized in Table 3.7, the model's precision is pretty high with an average of 0.72, while the

Table 3.7: Results for the recognition of *foreground* activities applying a baseline model to the opportunity dataset. The model uses only the first abduction axiom of the **States-Based Model** (Table 2.6) and thus outputs the most probably activity(ies) given the current action(s). The reported evaluation results are computed in a user-independent setting as explained above. The results of each user represent the averages and standard deviation (σ) over three routines $ADL1$, $ADL2$, and $ADL3$.

	Baseline		
	S10	S11	S12
Precision	$0.74\ (\sigma = 0.03)$	$0.61\ (\sigma = 0.06)$	$0.81\ (\sigma = 0.03)$
Recall	$0.32\ (\sigma = 0.03\ )$	$0.24\ (\sigma = 0.03)$	$0.41\ (\sigma = 0.07)$
F_1 measure	$0.44\ (\sigma = 0.02)$	$0.34\ (\sigma = 0.04)$	$0.54\ (\sigma = 0.07)$

Table 3.8: Results for the recognition of *foreground* activities applying the **States-Based Model** to the opportunity dataset. The model uses all axioms listed in Table 2.6. The reported evaluation results are computed in a user-independent setting as explained above. The results of each user represent the averages and standard deviation (σ) over three routines $ADL1$, $ADL2$, and $ADL3$.

	the **States-Based Model**		
	S10	S11	S12
Precision	$0.89\ (\sigma = 0.01)$	$0.85\ (\sigma = 0.03)$	$0.86\ (\sigma = 0.01)$
Recall	$0.8\ (\sigma = 0.03\ )$	$0.58\ (\sigma = 0.05)$	$0.83\ (\sigma = 0.07)$
F_1 measure	$0.84\ (\sigma = 0.01)$	$0.69\ (\sigma = 0.05)$	$0.84\ (\sigma = 0.03)$

Table 3.9: Global results of the **States-Based Model** with and without the three abduction axioms $3-5$ (see Table 2.6). The reported values are correspond to micro and macro averages over all activities using user-independent leave-one cross validation as explained above.

	With axioms $3-5$	Without axioms $3-5$
$F1$ measure micro-average	0.7	0.67
$F1$ measure macro-average	0.79	0.77

recall average value is very low (0.55). This is partially due to the annotation issues mentioned above and especially to unlabelled actions.Applying the entire set of axioms of the **States-Based Model** to the data significantly improves the quality of performance for each subject as shown in Table 3.8. While the average precision is only increased by 2%, the average recall value surpasses the *double* compared to the baseline model. Worthy of notice is the difference in performance between the subjects. The results of subject $S11$ is particularly low compared to $S10$ and $S12$. This underlines the effect of the annotation process used in this dataset. Indeed, as mentioned in [HRS13], the persons who annotated the data for $S10$ and $S12$ were able to communicate and agree about the same data interpretation, which did not apply for annotating $S11$. Additionally, taking a closer look into the per-activity recognition accuracy, we notice that the set of activities executed by $S10$ and $S12$ is smaller then those carried out by $S11$. Figure 3.8 illustrates per-activity $F1$ measures. Activities marked with an asterisk ($*$) are those that are absent in at least one of the three routines of $S10$ or $S12$ data. The figure indicates that these activities are those with the worst recognition results. Thus, their absence might boosts the overall performance of the model of the $S10$ and $S12$ data compared to that of $S11$.

To assess the effect of the semantic features encoded in the abduction axioms $3-5$ of Table 2.6, we compare the performance of the **States-Based Model** *without* and *with* these formulae. While the biggest jump in the recognition quality compared to the baseline is realised by the second abduction axiom (Table 2.6), the incorporation of abduction axioms $3-5$ raises the *micro-* and *macro-average* of F1-measures. Recall that the first is an average over activity instances while the second is an average over activity classes. The results of both averages are exposed in Table 3.9.

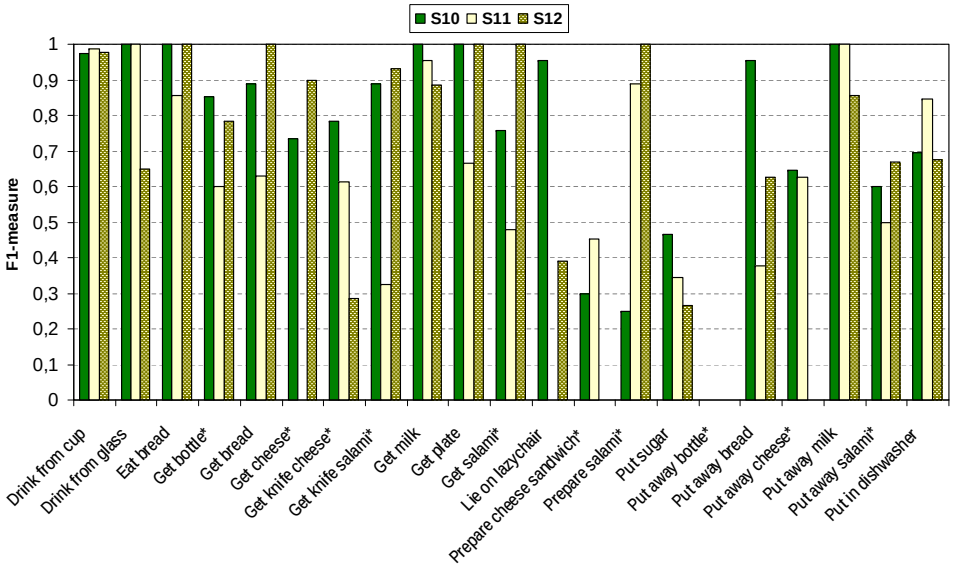


Figure 3.8: Per-activity recognition results for the **States-Based Model**. The reported F1-measure values of each subject ($S10$, $S11$ and $S12$) represent the averages over three routines $ADL1$, $ADL2$, and $ADL3$. Activities marked with an asterisk ($*$) are those that are absent in at least one of the three routines of the $S10$ or $S12$ data.

4 Conclusion

In this Part we have presented three Markov logic based frameworks to address the recognition of complex human activities under realistic settings. Our proposed approach benefits from a formal and declarative semantics, a variety of inference mechanisms, and methods for learning collections of sensor events and activities that satisfy some pattern. This facilitates efficient modelling and knowledge engineering, which are clearly separated from the generic inference mechanism.

After providing the theoretical background of Markov logic networks illustrated with simple examples, we drew the advantages of this approach compared to state of the art methods. Especially, we focus on addressing the challenge of recognizing interleaved and concurrent activities while preserving the intuitiveness and flexibility of the modelling task. Using three different models we have shown that Markov logic offers a simple but effective combination of statistical and relational features to accurately recognize interleaved and concurrent activities.

4.1 Summary

Throughout this part, we have answered the first five research questions defined in our problem statement. For **Question I.1**, we first exposed the weaknesses of pure data-driven approaches and knowledge-driven approaches. This highlighted the advantages of combining them to address the requirements of a realistic activity recognition system. The review was then refined through a detailed comparison of our logic-based statistical relational approach, Markov logic, with related methods proposed in the literature. The advantages of applying Markov logic for this recognition task were explained and illustrated with example models.

Moving to more realistic scenarios, we have presented three different models

to cover key knowledge representation features. These include representing temporal knowledge as well as the integration of background information. As with respect to **Question I.2**, the first proposed model focuses on point-wise temporal representation to model *qualitative* temporal relationship between the activities. The second model extends the first one with implicit interval-based temporal information. This is realised through the detection of the start and end points of the activities. Thus, additionally to the recognition of *foreground* activities, the model further allows the prediction of parallel activities running in the background. Inter and intra-activities temporal relationships are reinforced in this second model by the addition of *quantitative* temporal features such as the duration of an activity. Finally the third model accentuates the ability of Markov logic networks to elegantly represent and reason with highly relational data. The model leverages the inherent structure and common-sense knowledge to derive the states of surrounding artefacts over a particular period of time. This additional information has shown to boost the recognition of related activities. The ability to easily incorporate and reason with integer values and apply simple arithmetic operations on them greatly facilitates and supports modelling sophisticated temporal relationships.

Both uncertain and certain temporal information was included in the models and have shown to improve the recognition performance. Especially, temporal common-sense knowledge incorporated in form of *hard* constraints in the two first models almost doubled the recognition accuracy of particular activities without deteriorating the rest. This neatly positive impact of incorporating common-sense knowledge in the model solved **Question I.3**.

The three models were applied to two real-life datasets in order to answer **Question I.4**. The sensor data was collected in smart environments where the participants executed a set of complex activities in a naturalistic manner. Both datasets include highly interleaved activities. The second one also allows the participants to be actively involved in two activities at a time. The three models have successfully learned weights for *soft* formulae capturing temporal and non-temporal dependencies between individual activities. While the first two models were evaluated with the first dataset involving one single user, the data used for the third model was collected by three different subjects. This allowed a user-independent evaluation process. The reported results ascertain the viability of the proposed approach to the defined recognition task and have demonstrated a drastic increase compared to the baselines used.

Lastly we answer **Question I.5** in the discussion section of this chapter by summarizing the major limitations imposed by our Markov logic-based framework.

4.2 Impact

We discern a positive impact of our work on the activity recognition community. Convinced by the suitability of Markov logic to address complex event and activity recognition, several works have followed the same direction with a spe-

cial interest in Markov logic-based temporal reasoning. For instance, Skarlatidis et al. [SPVA11] focus on a very expressive model to represent and reason with interval-based temporal information based on Allen's algebra [All83]. Similarly, Song et al. [SKA+13] propose a Markov logic network for recognizing complex activities from gestures and objects interaction. They use both point-wise and interval-based temporal information to reason about the events and activities. Chahuara et al. also conclude in their work [CFPV12] that Markov logic based models are more adapted for activity recognition than traditional classifiers.

4.3 Discussion

While Markov Logic has been very successfully applied in several challenging applications such as activity recognition, Markov logic models are hard to interpret. Especially, a formula's weight is not an intuitive representation the probability that a grounding of that formula, independently of the rest of the model, holds [BMK+10]. Instead, the weight depends on the entire interactions between the model's atoms and their weights. Thus, it cannot be related to the probability of the formula without taking into account the weights of the other formulas. Since the outcome of the overall interactions between the model's formulae is almost impossible to foresee in the context of the knowledge engineering task, the automatic estimation of weights from annotated data is highly recommended. However, the quality of the learnt weights strongly depends on that of the annotation. As shown in our third model, in real-life datasets, annotations do not always correspond to the actual activities that are carried out by the user. Moreover, fully relying on automatic weight estimation often leads to less robustness due to the risk of over-fitting. A step towards controlling learnt weights and integrate prior knowledge at this level of modelling would be to combine both subjective and automatically learnt weights. A promising approach in this direction is the work of Papai et al. [PGK12] which presents a formalism for using prior expert knowledge for weight learning without requiring the consistency of that knowledge.

On the other hand, whereas using "*Markov: TheBeast*" [Rie08] is especially suitable for our discriminative models, it should be noticed that this framework is less appropriate for generative ones. Especially, it does not allow partial and uncertain observations and can not marginalize over missing input values since it does not model the full probability distribution. Thus, for applications with noisy sensor data for instance, this framework might not be the optimal choice.

4.4 Work in progress and future work

Inspired by the success of applying Markov logic to activity recognition, we are currently working on applying this formalism to activity forecasting as well as activity assessment. Both tasks belong to open challenges of the research field.

Activity forecasting is concerned with the prediction of future activities given a sequence of sensor data. Even if this sounds similar to the recognition problem addressed in this thesis, the task has fundamental differences. The major one is that the sensor data is not observed at the time steps where the activities have to be inferred. Thus additional background knowledge is crucial for this task. Another critical difference is that predicting the next activity might not be sufficient for several applications. Instead, it is usually of interest to infer the activities that would most probably take place within a predefined time slot (e.g. one hour). Preliminaries results have shown an improvement of the $F1$-measure over traditional machine learning techniques such as Naive Bayes and decision trees.

Activity assessment refers to attributing scores to the activities carried out in a smart environment indicating how well that activity was executed. The quality is determined by whether the activity was completed and how long its completion took. The assessment also distinguishes between critical and non-critical errors depending on their impact on the participants security and comfort. Interval-based temporal reasoning will be integrated in the Markov-model to flexibly reason with the different situations. This work is carried out in cooperation with our colleagues at the university of Milan [1] and is still at its early stages.

Generally, extending the Markov logic framework to real-time applications is also an appealing direction. The idea consists in applying the Markov model in a window-based setting with an iterative call of the learning and inference process after the input is updated with new data. Also, given the layered nature of human activities, our interests include the joint recognition of activities at different levels of granularities in a unified framework. Finally, moving towards automatic extraction of prior knowledge might help generate models with rich context data. This might support the portability and re-usability of the model by using widely spread knowledge description formalisms such description logic(DL) [BCM$^+$03]. The last three aspects played a major motivating role to the work described in Part II.

[1] http://homes.di.unimi.it/ riboni/

Part II

Representing and Recognizing Multilevel Activities with Log-Linear Description Logics

"To know an object is to lead to it through a context which the world provides."

—William James

1 Related Work and Contributions

Throughout the first Part of this document, the crucial impact of domain knowledge on the recognition of complex human activities has been demonstrated and particularly highlighted. Indeed, the integration of rich and expressive background knowledge enables further correlations between activities and other domain entities, which go beyond extracted patterns and attributes. However, implementing an exhaustive model including heterogeneous information sources comes at considerable knowledge engineering efforts. Hence, employing a standard, widely used formalism is highly recommended, in order to enhance the portability and re-usability of the model. To meet these additional requirements, we propose to employ a hybrid approach that goes one step further than Markov logic network towards a formal, yet intuitive *conceptualization* of the domain of discourse. Concretely, we propose to use a probabilistic variant of description logics as a common vocabularies for representing and reasoning about knowledge relevant to human activities and their semantic inter-connections, in order to automatically recognize them from sensor data. Complying with the general challenges of an activity recognition system depicted in the Introduction Chapter, this ontology-based approach addresses activities of different complexity levels and is capable of handling uncertainty related to any aspect within one unified framework.

Conventional ontology-based approaches have recently been gaining increasing popularity in the pervasive computing area in general and in the activity recognition community in particular. However, the existing solutions still face several shortcomings. Indeed, although it is usually assumed that knowledge-based activity recognition can create complete models, in reality it is very difficult, if not impossible, to manually cover all permutations of different users, activities, and performance styles [YDS+15]. Hence, the lack of native support for representing and reasoning over probabilistic knowledge within DL-based activity recognition

frameworks is a major open challenge. Unlike our system, which unites both symbolic and probabilistic reasoning, the majority of the proposed solutions need to decouple the recognition process from the semantic description of the activities in order to manage uncertainty.

In this Chapter, we provide an overview of the existing ontology-based approaches to recognize human activities from sensor data. We distinguish between two categories: frameworks that do not support uncertainty and those that do. We start by presenting the first category. There, the majority of the employed ontologies are usually only used for the representation step. They are combined with other techniques for the recognition step. Being closely related to our work, we put a special emphasize on approaches that use ontologies for both modeling and recognizing human activities. The second category is addressed in the second section of this Chapter. It presents recent attempts to empower ontology-based activity recognition frameworks with uncertainty support. Similarly to the first section, the overview pays particular attention to works that propose a unified framework for modeling *and* recognizing activities with a seamless integration of uncertain knowledge.

1.1 Ontology-based approaches to activity recognition: deterministic frameworks

Ontologies have been extensively used for context modeling in pervasive computing ([YDS+15], [RCLCF14]). They offer several advantages that make them particularly desirable for this field. These advantages include (1) their ability to effectively model and reason over *taxonomic knowledge*, (2) their support for consistency check and (3) their uniform and commonly-agreed vocabulary. The first feature meets the need of such systems for modelling contextual information at different levels of granularity and abstraction. Thanks to the supported subsumption reasoning, it also allows to derive further *implicit* and increasingly detailed contexts. The second helps deal with heterogeneous and imperfect context information coming from different sources. Indeed, information within a pervasive sensor-driven system usually include contextual data, domain knowledge and events. Contextual data mainly consists of abstracted sensor data describing properties of an environment or a user. This abstraction enhances their semantic meaning and allows their integration in consistent representation. Finally, the third is important for creating an understandable knowledge base, which can be easily shared and re-used by different platforms.

Such ontology-based models have been recently espoused to recognize and understand user's activities from sensor data. The majority of the works, however, do not use them for the recognition process. Instead, they exploit them as mapping mechanisms for multiple terms of an object, to categorize terms or to create a common conceptual template for data integration, interoperability, and reuse [CHN+12]. To the best of our knowledge, only very few works have been

approaching both activity representation *and* recognition in a unified ontological framework([YDS+15], [RCLCF14],[CHN+12]). Typically, ontological reasoning is used in these works to check the consistency of the aggregated set of contextual information and to infer higher level information such as the user's activity. One of the first works in this direction is that of Chen et al.([CNM+08], [CN09], [CNW12]). Very close to our work, the authors assume that there is an unknown activity corresponding to a given sensor input. Using ontological reasoning, the activity concept which contains as many perceived properties as possible is determined to be the predicted activity corresponding to the observed situation. Thus, the authors proceed to an incrementally specific recognition of the activities through the progressive activation of the sensors. However, this top down approach fails to recognize fine-grained activities unless the higher one is correctly recognized. Besides, the evaluation data used is collected in a partially predefined and strictly sequential manner including fixed time interval separating the complex activities. Finally, and most importantly, the proposed framework does not address the uncertainty aspect in human activities. Particularly, the model implicitly assumes a deterministic mapping from the context data to the activities' descriptions.

Similarly, the work of Springer et al. [ST09] leverages subsumption reasoning to infer activities at different levels of granularity based on the current contextual information. Their system is tested with simplistic cases such as "recognizing whether a ringing person is authorized to enter the house or not". The system's inability to address uncertainty imposes severe limitations towards applying it in real life scenarios.

The highly expressive ontological framework proposed by Riboni and Bettini [RB11] is very related to our work. Indeed, the authors use the same description logic language (OWL2) as in our proposed system. Combined with rules, they use their activity ontology to recognize activities for a smart home and a smart office scenario. The solution is proposed to overcome expressiveness limitations pointed out in their formal ontological framework COSAR [RB09], which combines ontological reasoning and multi-class logistic regression (MLR) for probabilistic activity recognition. Although their work includes statistical methods to recognize simple activities, the ontological reasoning about complex ones does not address uncertainty.

Another ontology-based approach that also supports data-driven learning capabilities has been recently proposed by Chen et al. [CNO14]. The approach uses semantic technologies as a conceptual backbone and technology enablers for modeling, classication, and learning of activities of daily living (ADL). Compared to their contributions, our framework supports finer grained activity levels. This emphasizes the need for a sound and integrated uncertainty support, which is missing in their system. Besides, the evaluation of the approach has been carried out under non-realistic assumptions exempt from interleaved and concurrent activities.

Finally, the MetaQ framework proposed by Meditskos *et al.* [MDK15] combines SPARQL queries and OWL 2 activity patterns to recognize activities in Ambient Assisted Living (AAL) environments. Whereas this approach has the appeal-

ing advantage of reasoning over intricate temporal dependencies between activities, the translation of the meta-knowledge OWL patterns into SPARQL queries is not able to handle uncertain and imperfect information.

Bridging the gap between ontology-based approaches and supporting uncertainty for activity recognition, has been the concern of some recent works as we explain in the next section.

1.2 Ontology-based approaches to activity recognition: frameworks with uncertainty support

Although much research has been devoted to extending DL-based models and reasoning services in order to handle uncertain information, only a limited number of works have explored their viability for activity recognition. Indeed the majority of the activity recognition approaches that leverage ontological modeling and uncertainty support only use ontologies to provide activity descriptors for activity definitions. Activity recognition is, then, performed based on probabilistic and/or statistical reasoning. For example, Knox *et al.* [KCD10] propose a lazy instance based approach where they use a vector of the sensors' values to define their cases. A semantically extended case base is created through extracting ontological relationships between sensors, locations and activities. This allows them to reduce the resulting number of cases.
Further efforts to exploit semantic information to improve the recognition system are detected in [YSK+07] and [WPP+07]. Relying on the subsumption hierarchy, the former involves ontology to handle unlearned objects and map them into learned classes. At the recognition step, parametric mixture models are applied. In the latter, the subsumption hierarchy helps automatically infer probability distributions over the current actions given the object in use. Thus, the integrated commonsense knowledge is used to learn a dynamic Bayesian network-based activity classifier. Other attempts to cope with uncertainty involve applying a hierarchy Bayesian networks based on the ontology's instances such as in [LLD07]. Including the challenge of recognizing concurrent activities in their work, Ye *et al.* [YSD14] have recently proposed the KCAR system which recognizes activities by matching segmented sensor sequences to ontological activity proles. They handle the ambiguity of interpretation of the sensor data by employing a hierarchy-based similarity measure to quantify the similarity on spatial, temporal, and thematic aspects in the corresponding ontologies.

All these works dissociate the inference step from the semantic model. This aspect limits the ability of incorporating rich background and common sense knowledge. It also strips the system from other advantages of symbolic reasoning such as consistency check. To the best of our knowledge, the works of Hong *et al.* [HNM+09], the one of Hoelz *et al.* [HKF13], and that of Rodriguez *et al.* [RCLCF14] are the only exceptions. In the first [HNM+09], the authors model the interrelationships between sensors, contexts and activities. They use the resulting hierarchical net-

work of ontologies to generate evidential networks. Following Dempster-Shafer theory of evidence, they calculate and define the heuristic relationships between the network's nodes in form of evidential mappings. These mappings are used through seven steps of evidential operations as inference process. Obviously, their evidential network discloses limited expressiveness compared to our DL language OWL2 [OWL09]. The second work [HKF13] is an extension of the one presented by Kurz *et al.* in [KHF$^+$11] focusing on the autonomous selection of the best set of available sensors to recognize a given goal. Using an ontological description of domain knowledge, the authors propose to use the semantic information between the different goals. This allows to reason with sub and super concepts in order to refine the recognition goal in case of missing sensing capabilities and exploit available data of related sensors. Thanks to the introduced "context predicates" and "Degree of Fulfilment (DoF)" notions, recognition goals can be modelled and inferred while enabling a weight distribution to sensor-goal mappings. Despite the promising aspect of this top-down approach, it was only evaluated with a simplistic low-level scenario involving one single recognition goal: "Locomotion". Moreover, the subsumption axioms do not support uncertainty, which might limit the applicability of the framework under realistic settings. Finally, the third work [RCLCF14] adopts fuzzy DL [BS08] to address uncertainty in activity recognition. The authors provide a proof of concept of their approach in work scenarios. Concretely, the approach consists in calculating the membership value of each attribute in the ontology based on membership functions and some predefined attributes values. However, unlike our log-linear ontology, in order to "fuzzify" a standard ontology, new attributes need to be created for each node in the ontology to represent uncertain information. Basically, the proposed formalism addresses fuzziness (i.e. vague knowledge), while we need to represent probable knowledge to recognize human activities from sensor data. This is because we are interested in reasoning about events (i.e. activities) which either happened or not, rather than facts with different degrees of truth. However, fuzzy knowledge can be a beneficial extension to our approach when reasoning about facts related to the activities. As an example, it would be interesting to be able to map continuous values, such as activity duration, into discrete concepts such "long activity" and "short activity".

2

Modelling and Recognizing Multi-level Activities with Log-linear Description Logics

Recall from the previous chapters that human activities are complex, ambiguous and have different levels of granularity. These characteristics remain valid even for restricted sets of activities such as basic activities of daily living (ADLs) [Org02]. Although ADLs can be performed within home environments with relatively clear semantics, providing a meaningful computational model remains a challenging task. In this chapter we approach this task using a hybrid framework based on formal knowledge representation mechanisms. Especially, we propose a log-linear description logic-based framework to model and reason about activities of daily living from the inhabitant gestures and their interaction with objects of interests. The framework shares the same fundamentals underlying Markov logic network while allowing for a formal conceptualization of the domain of discourse, backed up with powerful reasoning and consistency check tools.

The main part of this work has been realized and published in [HRN+12] and [HRS13] in cooperation with partners from the university of Milano [1].

In the following, we first provide an overview of the theoretic foundations underlying our proposed system. This covers fundamentals of description logics and that of log-linear description logic [NNS11]. Then, we present our log-linear ontology modeling the domain of discourse including multi-level activities. Finally, we propose a technique to leverage ontology reasoning to recognize the multi-level activities from sensor data.

[1] http://homes.di.unimi.it/riboni/

2.1 Description logics and log-linear description logic

The research in area of description logics (DLs) emerged from the idea of using first-order logic to represent network based systems [BCM+03] such as semantic networks and frames. This has been primary motivated by the need for precise semantic characterization unifying these different representation structures. Description logics have intended to adopt the intuitive and natural representation mechanisms of first-order logic yet with less complex reasoning techniques. Thus, they can be seen as a decidable fragment of first-order logic, which relies on unary predicates to denote sets of individuals and binary predicates to represent relationships between these individuals [BCM+03]. Depending on the required application, different levels of expressive power have appeared in form of various description logic formalisms.

Despite their advantages, DLs have crucial limitations when applied to several real life domains. Especially, they are lacking the ability to represent uncertain knowledge. Several extensions have been proposed to overcome this deficiency. Log-linear description logic [NNS11], is one of these. Based on the same principle as Markov logic, this formalism lies in the core of our proposed framework.

2.1.1 Foundations of description logics

Description logics refer to a family of knowledge representation formalisms established to allow a logic-based representation of the knowledge of a given application domain. The representation includes definitions of the domain's concepts as well as the specification of its objects and individuals. Central to these formalims is the reasoning about the created knowledge base content. The reasoning task essentially consists in inferring *implicit knowledge* from an explicit knowledge base in order to answer specific queries [BCM+03].

DL Syntax

The signature of a knowledge base (KB) system based on description logics consists of three components:

- Atomic **concepts** are designated by unary predicate symbols and denote types, categories or classes of entities.
- Atomic **roles** are designated by binary predicate symbols and are used to express relationships between concepts.
- **Individuals** stand for all the names used to represent the domain's entities. They correspond to constants in first order logic.

Similarly to first order logic, these atomic components can be used to build more complex expressions by applying several kinds of constructors such as standard first order logic boolean operators ($\vee$, $\wedge$, $\neg$), restricted form of quantifiers

Table 2.1: Major DLs concept constructors: Their syntax, semantics and symbol [Baa03]

Name	Syntax	Semantics	Symbol
Top	$\top$	$\Delta^{\mathcal{I}}$	$\mathcal{AL}$
Bottom	$\bot$	$\emptyset$	$\mathcal{AL}$
Intersec.	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$	$\mathcal{AL}$
Union	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$	$\mathcal{U}$
Negation	$\neg C$	$\Delta^{\mathcal{I}} - \{C^{\mathcal{I}}\}$	$\mathcal{C}$
Value Rest.	$\forall R.C$	$\{a \in \Delta^{\mathcal{I}} \vert \forall b.(a,b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\}$	$\mathcal{AL}$
Exist. Quant.	$\exists R.C$	$\{a \in \Delta^{\mathcal{I}} \vert \exists b.(a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}$	$\mathcal{E}$
Unqual.	$\geq n\, R$	$\{a \in \Delta^{\mathcal{I}} \vert\vert \{b \in \Delta^{\mathcal{I}} \vert (a,b) \in R^{\mathcal{I}}\}\vert \geq n\}$	
Numb.	$\leq n\, R$	$\{a \in \Delta^{\mathcal{I}} \vert\vert \{b \in \Delta^{\mathcal{I}} \vert (a,b) \in R^{\mathcal{I}}\}\vert \leq n\}$	$\mathcal{N}$
Rest.	$= n\, R$	$\{a \in \Delta^{\mathcal{I}} \vert\vert \{b \in \Delta^{\mathcal{I}} \vert (a,b) \in R^{\mathcal{I}}\}\vert = n\}$	
Qual.	$\geq n\, R$	$\{a \in \Delta^{\mathcal{I}} \vert\vert \{b \in \Delta^{\mathcal{I}} \vert (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}\vert \geq n\}$	
Numb.	$\leq n\, R$	$\{a \in \Delta^{\mathcal{I}} \vert\vert \{b \in \Delta^{\mathcal{I}} \vert (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}\vert \leq n\}$	$\mathcal{Q}$
Rest.	$= n\, R$	$\{a \in \Delta^{\mathcal{I}} \vert\vert \{b \in \Delta^{\mathcal{I}} \vert (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}\vert = n\}$	

($\forall$, $\exists$), and counting ($\leq$, $\geq$, $=$). However, concept expressions are variable-free, since they denote the set of all individuals satisfying the properties specified in the expression. Moreover, while some constructors are related to logical ones in first order logic, others have no matches. These include transitivity and functionality for instance. Like concepts, roles expression, such as role hierarchies, can also be built using role contractors.

Following this syntax, a DL knowledge bases comprise two components: the **TBox** and the **ABox**. The former introduces the *terminology* (concepts and roles) used to represent the application domain and builds a set of axioms modeling general knowledge about it. The second contains a set of *facts* expressing knowledge about specific situations through *assertions* about named individuals in terms of the introduced terminology.

The TBox: The axioms of a TBox can be divided into *definitions* and *subsumption* axioms. Definition axioms are called *concept equality* and state that a concept C is equivalent to concept D (denoted $C \equiv D$). This allows the introduction of symbolic names for complex expressions. The following example associates the description on the right hand to "Vacuuming". Thus, "Vacuuming" is *defined* as an activity whose actor is using a vacuum.

$$\begin{aligned} \textsc{Vacuuming} \equiv \textsc{Activity} \sqcap \exists\, \textsc{hasActor}. \\ \big(\textsc{Person} \sqcap \exists\, \textsc{usesObject}.\textsc{Vacuum}\big) \end{aligned} \tag{2.1}$$

In case a concept can not be defined precisely, subsumption axioms are used instead. These indicate the necessary conditions for a concept using *concept inclusion*. Thus, a concept C is subsumed by a concept D (denoted $C \sqsubseteq D$) if C is a subclass of D. This kind of axioms is also designated as a "*is-a*" relationship. For instance, a "social activity" requires at least two participants but not every activity engaging two participants is necessarily a "social activity". This can be expressed by the following subsumption axiom.

$$\textsc{SocialActivity} \sqsubseteq \textsc{Activity} \sqcap \geq 2\textsc{hasParticipant}.\textsc{Person} \tag{2.2}$$

The ABox: Introducing individuals by asserting names to the TBox concepts is the role of the ABox. Thus, specific states of the domain of discourse can be described. Using the subsumption axiom 2.2, an ABox can provide a specific social activity name, let's say "playing cards", which was carried out by some known participants, lets say "Mary" and "Bob". The corresponding concept and role assertions would look as follows.

Table 2.2: Terminological and assertional axioms in DL knowledge bases [Baa03]

Name	Syntax	Semantics
Concept inclusion	$C \sqsubseteq D$	$C^{\mathcal{I}} \sqsubseteq D^{\mathcal{I}}$
Role inclusion	$R \sqsubseteq S$	$R^{\mathcal{I}} \sqsubseteq S^{\mathcal{I}}$
Concept equality	$C \equiv D$	$C^{\mathcal{I}} = D^{\mathcal{I}}$
Concept assertion	$C(a)$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
Role assertion	$R(a, b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$

$$\begin{aligned} \text{SOCIALACTIVITY}(PlayingCards) \\ \text{HASPARTICIPANT}(PalyingCards, Mary) \\ \text{HASPARTICIPANT}(PalyingCards, Bob) \end{aligned} \tag{2.3}$$

DL Semantics

In description logics concepts are interpreted as a set of individuals and roles are interpreted as sets of individual pairs. Formally, this interpretation $\mathcal{I}$ is defined in terms a non-empty set $\Delta^{\mathcal{I}}$ representing the domain of discourse and an interpretation function assigning a set $\mathcal{A}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ to each atomic concept A and a binary relation $\mathcal{R}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ to each atomic role R. The interpretation of concept descriptions, subsumption and assertion axioms is obtained by extending the interpretation function $\mathcal{I}$ through inductive definitions as denoted in Table 2.2. Hence, a subsumption axiom $C \sqsubseteq D$ is satisfied by an interpretation $\mathcal{I}$ if and only if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. Similarly, an interpretation $\mathcal{I}$ satisfies an equivalence axiom $C \equiv D$ if and only if $C^{\mathcal{I}} = D^{\mathcal{I}}$. Finally, an interpretation $\mathcal{I}$ satisfies a disjointness axiom $C \sqcap D \sqsubseteq \bot$ if and only if $C^{\mathcal{I}} \cap D^{\mathcal{I}} = \emptyset$.

Assertion axioms(ABox) are given their semantics by extending an interpretation $\mathcal{I}$ to individual names. If the resulting interpretation maps distinct individuals names to distinct individuals, then it respects the so called *unique name assumption(UNA)*. We say that an interpretation $\mathcal{I}$ satisfies the concept assertion $C(a)$ if and only if $a^{\mathcal{I}} \in C^{\mathcal{I}}$. Also, it satisfies the role assertion $R(a, b)$ if and only if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$. Although assertion axioms can be compared to a relational database, it is important to note that its domain of interpretation can be infinite and obeys to the open-world assumption. Consequently, while absence of information in a database instance is interpreted as negative information, it only indicates incomplete knowledge in an ABox.

Based on this definition of the interpretation function $\mathcal{I}$, description logics can be identified as fragments of first-order predicate logic. Thus, atomic concepts can be considered as unary predicates, roles can be viewed as binary predicates and individuals as constants. Following this observation, any concept C can be translated into a predicate logic formula $\mathcal{F}_C(x)$ with one free variable x such that

for every interpretation $\mathcal{I}$, the set of elements of $\Delta^{\mathcal{I}}$ satisfying $\mathcal{F}_C(x)$ coincides with the interpretation set $C^{\mathcal{I}}$. Concretely, an atomic concept A is translated into the formula $A(x)$ and the basic constructors are translated into their counterparts.

The DLs Family:

Several description logic languages have been defined. They basically differ in the set of allowed operators and have, consequently, different expressiveness levels. The basic language is referred to as DL $\mathcal{AL}$, an abbreviation of "*attribute language*". Considering A as an atomic concept, C and D as concepts descriptions, and R as functional role, DL $\mathcal{AL}$ allows the universal concept ($\top$), the bottom concept ($\bot$), atomic negation ($\neg A$), concept intersection ($C \sqcap D$), complex concept negation ($\neg C$), universal restrictions ($\forall R.C$) and limited existential quantification ($\exists R.\top$) [SSS91]. These basic modeling features have been enriched and extended to allow more expressiveness for specifying and querying knowledge. The simplest extension consists in adding the negation of arbitrary concepts (e.g. $\neg(C \sqcap D) = \neg C \sqcup \neg D$). The resulting language is referred to as $\mathcal{ALC}$, an abbreviation for " *attribute language with complements*". The complete naming scheme for mainstream DLs comply with the following naming convention:

$$((\mathcal{ALC}|\mathcal{FL}|\mathcal{EL}|\mathcal{S})[\mathcal{H}]|\mathcal{SR})[\mathcal{O}][\mathcal{I}][\mathcal{F}|\mathcal{E}|\mathcal{U}|\mathcal{N}|\mathcal{Q}]^{(\mathcal{D})}$$

$\mathcal{FL}$ symbolizes a DL that allows *concept intersection*, *universal restriction*, *limited existential quantification* and *role restriction*. $\mathcal{EL}$ permits *subsumption* and *equivalence axioms* as well as *concept operators*, yet no role or axioms operators. $\mathcal{S}$ denotes an extension through *transitivity* axioms. The letter $\mathcal{O}$ introduces the support for *nominal* concepts, which are concepts that have exactly one instance used for their description. Role inverses (e.g. $hasParticipants \equiv isParticipantOf^-$) are allowed in DLs with names containing the letter $\mathcal{I}$. The letter $\mathcal{F}$ features $agreements$ (sometimes also called the $same-as$ constructor) and $disagreements$. $\mathcal{E}$ and $\mathcal{U}$ allow full existential quantification and concept union respectively. Finally, $\mathcal{Q}$ and $\mathcal{N}$ indicate the possibility to use qualified (e.g. $\geq 2 hasParticipants.Person$) and non-qualified (e.g. $\geq 2 hasParticipants$) *cardinality restrictions* respectively. The introduced constructors, their syntax, their semantics and their symbols are summarized in Table 2.2

Offering the logic basis of the web ontology languages OWL DL and OWL2 DL respectively, the $\mathcal{SHOIN}^{(D)}$ DL and $\mathcal{SROIQ}^{(D)}$ are key DLs representatives in our work.

Reasoning about knowledge in DL

Besides formal syntax and semantics, DLs offer powerful reasoning services which are based on efficient, sound and complete algorithms (e.g. Tableau algorithm [BS01]). Like in first-order predicate logic, a knowledge base comprising TBox and ABox

contains *implicit knowledge* that can be made explicit through inference. For instance, we can conclude that the activity PLAYINGSOCCER is not an IDLEACTIVITY from example 7 although this information does not figure explicitly.

Typical TBox reasoning tasks include *subsumption* and *satisfiability checking*. The first consists in determining whether a concept C subsumes a concept D such as determining whether the concept $Activity$ subsumes the concept $SocialActivity$ for instance. Subsumption checking allows the derivation of the implicit taxonomic relations and hierarchies among concepts. The second identifies whether a concept description has a model, i.e. whether there is an interpretation that satisfies it. A straightforward example of an unsatisfiable concept is $(C \wedge \neg C)$. Besides satisfiability and subsumption, two other concept relationships are particularly relevant for inference: Equivalence and disjointness. Formally, these four properties can be defined as follows [BCM$^+$03].

Definition 4. ***Satisfiability****: A concept C is satisfiable with respect to a TBox $\mathcal{T}$ if there exists a model $\mathcal{I}$ of $\mathcal{T}$ such that $C^{\mathcal{I}}$ is nonempty. In this case we say also that $\mathcal{I}$ is a model of C.*

Definition 5. ***Subsumption****: A concept C is subsumed by a concept D with respect to a TBox $\mathcal{T}$ if $C^{\mathcal{I}} \subseteq C^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{T}$. In this case we write $C \sqsubseteq_{\mathcal{T}} D$ or $T \models C \sqsubseteq D$.*

Definition 6. ***Equivalence****: Two concepts C and D are equivalent with respect to a TBox $\mathcal{T}$ if $C^{\mathcal{I}} = D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{T}$. In this case we write $C \equiv_{\mathcal{T}} D$ or $\mathcal{T} \models C \equiv D$.*

Definition 7. ***Disjointness****: Two concepts C and D are disjoint with respect to a TBox $\mathcal{T}$ if $C^{\mathcal{I}} \cap D^{\mathcal{I}} \equiv \emptyset$ for every model $\mathcal{I}$ of $\mathcal{T}$.*

Example 7. *Let's consider the following TBox:*

$$\begin{aligned} \text{SOCIALACTIVITY} &\sqsubseteq \text{ACTIVITY} \\ \text{SPORTACTIVITY} &\sqsubseteq \text{ACTIVITY} \\ \text{IDLEACTIVITY} &\sqsubseteq \text{ACTIVITY} \\ \text{IDLEACTIVITY} &\sqsubseteq \neg\text{SPORTACTIVITY} \\ \text{PLAYINGSOCCER} &\sqsubseteq \text{SOCIALACTIVITY} \sqcap \text{SPORTACTIVITY} \\ \text{SLEEPING} &\sqsubseteq \text{IDLEACTIVITY} \end{aligned} \tag{2.4}$$

With respect to this TBox, ACTIVITY *subsumes all of* SOCIALACTIVITY, SPORTACTIVITY *and* IDLEACTIVITY.

Since SPORTACTIVITY *and* SOCIALACTIVITY *are disjoint,* PLAYINGSOCCER *is also disjoint with* IDLEACTIVITY *and in particular with* SLEEPING. *Similarly,* SLEEPING *is disjoint with* SPORTACTIVITY. *These last disjointness relationships follow from the semantics of "$\sqsubseteq$".*

Central to the ABox reasoning tasks is *consistency checking*. An ABox A is said to be *consistent* with respect to a TBox T, if there is an interpretation $\mathcal{I}$ that satisfies the knowledge base $\mathcal{KB} = (T, A)$. For instance, if SOCIALACTIVITY and INDIVIDUALACTIVITY are defined as disjoint concepts in the TBox, then asserting both SOCIALACTIVITY$(playingCards)$ and INDIVIDUALACTIVITY$(playingCards)$ results in an *inconsistent* knowledge base. *Instance checking* is another important ABox reasoning problem which consists in deciding whether a particular assertion α is entailed by an ABox A ($A \models \alpha$). An ABox A entails an assertion α if every interpretation $\mathcal{I}$ that satisfies A also satisfies α. Thus, this task can be used to solve more complex problems such as answering queries. Concretely, it allows to retrieve all individuals a that belong to a given concept description C (i.e $A \models C(a)$). For example one might be interested in finding all social activities that involve at least 3 participants (SOCIALACTIVITY $\sqcap \geq 3 hasParticipants$). An interesting variant of this retrieval problem is the *realization problem*. It computes the most specific concept(s) that each individual is an instance of. For example, if a knowledge base consists of two concepts ACTIVITY and SOCIALACTIVITY such that SOCIALACTIVITY $\sqsubseteq$ ACTIVITY and two assertions ACTIVITY$(playingCards)$ and SOCIALACTIVITY$(playingCards)$ then the realization returns the concept SOCIALACTIVITY.

2.1.2 Log-linear description logic

Despite their advantages, description logics offer very restricted means of expressing real-life relationships between concepts. This deficiency is mainly due to the inability of these purely deterministic formalisms to handle imprecision and uncertainty. Imprecision and uncertainty are typical aspects of real life applications. They are particularly accentuated in the domain of sensor-based activity recognition which is characterized by complex, incomplete and erroneous data. Concretely, sensor readings could occasionally become unreliable or even absent and the same activity could be carried out in different manners. For instance, the activity "sleeping" could be defined as an "activity which has an actor that is in bed".

$$\text{SLEEPING} \sqsubseteq \text{ACTIVITY} \sqcap \exists \text{HASACTOR}. \big(\text{PERSON} \sqcap \exists \text{HASLOCATION}.\text{BED}\big) \tag{2.5}$$

Nonetheless, this description is not accurate enough for real life scenarios. In fact, sleeping might occasionally take place on the sofa, for instance, instead of the bed. Thus, to create a realistic model given specific sensing capabilities (here the location and temporal context only), a more flexible DL-based framework is urged. Such a framework should support the expression of knowledge with different degrees of confidence. Many attempts have recently appeared in this direction such as probabilistic description logics [Luk08], fuzzy OWL [Str05] and log-linear DL [NNS11]. In the following, we explain the principles of the latter, which we adopt in our proposed system.

Log-linear DL combines log-linear models [KF09] and DLs [BCM$^+$03] in order to represent and reason about uncertain knowledge. Borrowing the same idea underlying Markov logic [RD06], which is explained in Part I of this document, its syntax is that of description logics except that it is possible to assign real-valued weights to general concept inclusion axioms (GCIs), role inclusion axioms (RIs), and assertions. The semantics is defined by a log-linear probability distribution over *coherent* and *consistent* knowledge bases as explained below.

Syntax of log-linear DL

In the following, we use the terms *constraint box* (CBox) and *knowledge base* ($\mathcal{KB}$) interchangeably. Moreover, for ease of presentation, we will use the term *axiom* to denote general concept inclusions (GCIs), role inclusions (RIs), *and* concept and role assertions. Similarly to Markov logic, a log-linear knowledge base $\mathcal{C} = (\mathcal{C}^{\mathsf{D}}, \mathcal{C}^{\mathsf{U}})$ can be formally defined as a pair consisting of a *deterministic* knowledge base CBox $\mathcal{C}^{\mathsf{D}}$ and an *uncertain* knowledge base CBox $\mathcal{C}^{\mathsf{U}} = \{< c, w_c >\}$ with each c being an axiom and w_c a real-valued weight assigned to c.

The *uncertain* CBox $\mathcal{C}^{\mathsf{U}}$ contains weighted axioms, i.e axioms that might be violated. Hence, a log-linear $\mathcal{KB}$ may be inconsistent. The greater the weight of an uncertain axiom, the more confidence there is for it to hold. Revisiting the definition of the activity "sleeping", the axiom provided in 2.5 can be extended with a weight in order to express that sleeping *usually*, but not necessary, is a subclass of activities whose actors are in bed (let's say with weight 1.2). Thus, a CBox containing that axiom can, for example, also contain two weighted assertions like *"p is a person that is an actor of Sleeping"* and that *"p does not have location Bed"* with weights 0.6 and 0.9 respectively .

The *deterministic* CBox $\mathcal{C}^{\mathsf{D}}$ contains axioms that always hold and is assumed to be *coherent* and *consistent*. For instance, given the axiom defining a "social activity" as a subclass of "activities that have at least two actors", it is impossible to derive that a subject is having a "social activity" by themselves.

Semantics of log-linear DL

The semantics of log-linear DL is based on a probability distributions over *coherent* and *consistent* knowledge bases. Comparing the log-linear DL axioms to the Markov logic first order formulae, the definition of this distribution becomes straightforward; given a log-linear knowledge base $\mathcal{C} = (\mathcal{C}^{\mathsf{D}}, \mathcal{C}^{\mathsf{U}})$ and a CBox $\mathcal{C}'$ with $\mathcal{C}^{\mathsf{D}} \subseteq \mathcal{C}' \subseteq \mathcal{C}^{\mathsf{D}} \cup \{c : (c, w_c) \in \mathcal{C}^{\mathsf{U}}\}$, we have that

$$\Pr_{\mathcal{C}}(\mathcal{C}') = \begin{cases} \frac{1}{Z} \exp\left(\sum_{\{c \in \mathcal{C}' \setminus \mathcal{C}^{\mathsf{D}}\}} w_c\right) & \text{if } \mathcal{C}' \text{ is } \textit{coherent} \\ & \text{and } \textit{consistent}; \\ 0 & \text{otherwise} \end{cases}$$

where Z is the normalization constant of the log-linear distribution $\Pr_{\mathcal{C}}$. Notice that, based on these semantics, an axiom with weight 0 that is not in conflict with

any other axiom has the marginal probability of 0.5. This leads to a distribution compatible with the open-world assumption.

Let us consider the log-linear CBox $\mathcal{C} = (\mathcal{C}^{\mathsf{D}}, \mathcal{C}^{\mathsf{U}})$ introduced above and extend it as follows.

Deterministic CBox $\mathcal{C}^{\mathsf{D}}$	
c_1:	$\textsc{Person}(p)$
c_2:	$\textsc{Bed}(b)$
c_3:	$\textsc{Couch}(c)$
c_4:	$\textsc{Sleeping}(s)$
c_5:	$\textsc{Couch} \sqcap \textsc{Bed} \sqsubseteq \bot$
Uncertain CBox $\mathcal{C}^{\mathsf{U}}$	
c_6:	$< \textsc{Sleeping} \sqsubseteq \textsc{Activity} \sqcap \forall\, \textsc{hasActor}.$ $\Big(\textsc{Person} \sqcap \exists\, \textsc{hasLocation}.\textsc{Bed}\Big), 1.2 >$
c_7:	$< \textsc{HasActor}(s, p), 0.6 >$
c_8:	$< \textsc{HasLocation}(p, c), 0.8 >$

This CBox $\mathcal{C}$ comprises five concept descriptions: ACTIVITY, PERSON, BED, COUCH, and SPEELING. It contains also two role descriptions HASACTOR and HASLOCATION. The last four concepts are instantiated with the individuals p, b, s, c respectively. Moreover, the knowledge base states that "a Couch is not a Bed" (c_5), that "Sleeping is *usually* an activity of which the actor is in Bed" (c_6), that "the subject p is *probably* sleeping"(c_7) and, finally, that "the subject p is *probably* on the Couch" (c_8).

Considering this ontology in the traditional context, where all axioms are deterministic, would lead to inconsistency. In that case, the given axioms entail that the subject p must be sleeping on the couch, which contradicts the definition of the activity "sleeping" supposed to take place in bed. Relaxing Axioms $c_6 - c_8$ by adding weights to them allows to violate them and resolve the inconsistency. According to the semantics of log-linear DL, only coherent and consistent subsets of these axioms, which include the entire $\mathcal{C}^{\mathcal{D}}$, will be possible, i.e. have a probability greater than zero.

Concretely, the probability distribution over coherent and consistent knowledge bases is determined based on the following 8 cases.

- The first refers to the case where all uncertain axioms are considered not to hold, thus, the resulting ontology $\mathcal{C}'_1$ consists of four assertions ($c_1 - c_4$) and the disjointness axiom c_5.
- The second case results in a CBox $\mathcal{C}'_2$ where the activity "Sleeping" is defined to take place in "Bed" (axiom c_6) along with the deterministic CBox $\mathcal{C}^{\mathsf{D}}$.
- The third rejects c_6 and c_8. The resulting CBox $\mathcal{C}'_3$ refers to the case where the actor is sleeping but they don't have to be in "Bed", nor the "Couch".

- The fourth, $\mathcal{C}'_4$ corresponds to the case where the subject is on the couch but no information is given on whether they are sleeping or not.
- The fifth, $\mathcal{C}'_5$, refers to the case where sleeping is defined as *an activity whose actor is in bed* and that the subject p is sleeping. Thus, the fact that p is on the couch (hence not in bed) is omitted.
- The sixth, $\mathcal{C}'_6$, keeps the same definition of sleeping, yet does not state whether p is sleeping or not. Instead, it states that the subject p is on the couch.
- The seventh,$\mathcal{C}'_7$, excludes the definition of the activity sleeping while stating that p is sleeping on the couch.
- Finally, the eighth possibility, $\mathcal{C}'_8$. is nothing else than the entire CBox $\mathcal{C}$, which results in an inconsistent knowledge base, and thus, has a zero probability.

Following the log-linear model introduced above, the probabilities of these knowledge bases can be calculated as follows:

$$\begin{aligned}
&\Pr_{\mathcal{C}}(C'_1 = \{\mathcal{C}^{\mathsf{D}}\}) && = Z^{-1}\exp(0) && \approx 0.04\\
&\Pr_{\mathcal{C}}(C'_2 = \{\mathcal{C}^{\mathsf{D}}, c_6\}) && = Z^{-1}\exp(1.2) && \approx 0.14\\
&\Pr_{\mathcal{C}}(C'_3 = \{\mathcal{C}^{\mathsf{D}}, c_7\}) && = Z^{-1}\exp(0.6) && \approx 0.08\\
&\Pr_{\mathcal{C}}(C'_4 = \{\mathcal{C}^{\mathsf{D}}, c_8\}) && = Z^{-1}\exp(0.8) && \approx 0.1\\
&\Pr_{\mathcal{C}}(C'_5 = \{\mathcal{C}^{\mathsf{D}}, c_6, c_7\}) && = Z^{-1}\exp(1.8) && \approx 0.27\\
&\Pr_{\mathcal{C}}(C'_6 = \{\mathcal{C}^{\mathsf{D}}, c_6, c_8\}) && = Z^{-1}\exp(2.0) && \approx 0.33\\
&\Pr_{\mathcal{C}}(C'_7 = \{\mathcal{C}^{\mathsf{D}}, c_7, c_8\}) && = Z^{-1}\exp(1.4) && \approx 0.18\\
&\Pr_{\mathcal{C}}(C'_8 = \{\mathcal{C}^{\mathsf{D}}, c_6, c_7, c_8\}) && = 0
\end{aligned}$$

Where Z is a normalization constant calculated as: $Z = \exp(0) + \exp(1.2) + \exp(0.6) + \exp(0.8) + \exp(1.8) + \exp(2.0) + \exp(1.4) \approx 1 + 3.3 + 1.82 + 2.22 + 6.04 + 7.4 + 4 \approx 22.5$

Under the given syntax and semantics, the central inference task is the maximum a-posteriori (MAP) query, i.e.*"Given a log-linear ontology, what is a most probable coherent and consistent ontology over the same class and property names?"*

The application of the MAP query to the simple ontology shown in the example would return the CBox C^*, that is coherent, consistent and having the highest probability. According to the probability distribution presented in the same example, the output C^* of the MAP inference coincides with the CBox C'_6. Hence, the most probable coherent and consistent ontology is the one that keeps the fact about the user's location, i.e. the couch as well as the definition of the activity "sleeping" as an activity whose actor is in bed, while omitting whether the p is sleeping or not.

Solving the MAP inference in log-linear DL first goes through a transformation of the knowledge base into a Markov logic network [RD06]. This permits to use the well-established inference algorithms developed for the Markov logic formalism and explained in Part I of this thesis. For a detailed insight into this transformation procedure, the user is referred to the work of Niepert *et al.* [NNS11].

2.2 Representing multi-level activities with log-linear DL

Raw sensor data often originate from different modalities. It needs to be aggregated into more consistent conclusions and lifted to higher levels of abstractions, such as inferring body postures from wearable sensors for instance. Reasoning with the information collected from these sources necessitates modeling rich semantic relations. Also, it demands a uniform way of representing heterogeneous information, including different contextual aspects, e.g. location, body posture, objects, so that it can be re-used and shared regardless of the underlying sensing technologies.

In this section, we describe our approach for representing human activities at different levels of granularity using log-linear description logic [NNS11]. Concretely, our model espouses the basic hierarchy underlying activity theory [KN12] introduced in the preliminaries chapter of this thesis. *Activities*, *actions* and *operations* are defined as ontological concepts in terms of lower level components that are required in order to perform them. For example, the action "dishwashing" usually involves the operations "opening the dishwasher", "putting down the dishes in the dishwasher", "closing the dishwasher door" and "turning on the dishwasher". The action "dishwashing" is in its turn a component of the definition of the activity "cleaning up". Aside from activity, action and operation concepts, other major entities from the domain are also modeled. This includes the objects the user is interacting with, the user's body gestures and postures as well as related properties to establish temporal relationships between the operations.

To illustrate this structure, we adopt the framework proposed by the European project "*Opportunity*" [KHF+11] as introduced in Part I, Chapter 3. Its hierarchical scheme designates the highest level of abstraction (level 1) as ***Complex Activities*** level. This would correspond to "activities" in the activity theory model. "cleaning up" is an example of a *complex activity* defined by this framework.

One level lower (level 2), parallel to "actions" in activity theory, the notion of ***Simple Activities*** is introduced. A *Simple Activity* can refer to an action such as "get salami" for instance. At this level, the temporal sequence of the underlying components is especially relevant. Indeed, "getting salami" usually goes through three operations: first the fridge door is opened, then the salami is fetched before the fridge door is closed again.

Operations correspond to level 3 and are labeled ***Manipulative Gestures*** in the *Opportunity* framework. They, themselves, can still be represented in terms of even finer grained gestures translating body movements linked to the objects used by the subject. For instance, the *Manipulative Gesture* "fetch salami" can be fragmented into "reach salami" and "move salami". Such gestures are denoted as ***Atomic Gestures*** and constitute the finest grained level in the framework, i.e. level 4.

Using log-linear DL, we model the presented multilevel structure as follows. Central to our ontology, is the class PERSON representing the subject carrying out the different activities. A person interacts with their environment through their arms, represented by a class ARM and/or their body posture, referred to as

LOCOMOTION TYPE. The arms allow the subject to use objects, which are represented by a class OBJECT. The manner the user's arms manipulate the objects is described through the class FUNCTION. Thus, and as shown in Example 8, increasingly complex activities can be iteratively defined in terms of simpler ones based on the properties linking the ontology classes. These classes and the properties linking them are depicted in Figure 2.1.

Example 8. *The* Complex Activity CLEANUP *can be defined as a subclass of the concept* COMPLEX ACTVITY *whose actor is a person having* PUTAWAYMILK *as "Simple Activity".*

$$\text{CLEANUP} \sqsubseteq \text{COMPLEXACTIVITY} \sqcap \exists \text{HASACTOR}. \left(\text{PERSON} \sqcap \exists \text{DOESSIMPLEACTIVITY}.\text{PUTAWAYMILK}\right) \tag{2.6}$$

The "Simple Activity" PUTAWAYMILK *can be, in its turn, defined as a "Simple Activity" whose actor is a* PERSON *that has the "Manipulated Gesture"* PUTDOWNMILK.

$$\text{PUTAWAYMILK} \sqsubseteq \text{SIMPLEACTIVITY} \sqcap \exists \text{HASACTOR}. \left(\text{PERSON} \sqcap \exists \text{DOESMANIPULATIVEGESTURE}.\text{PUTDOWNMILK}\right) \tag{2.7}$$

The entity PUTDOWNMILK *can now be defined in terms of the "Atomic Gesture"* REACHMILK. *This latter class is described as an* ATOMICGESTURE *that has an actor a* PERSON *which has an* ARM *with function* REACH *and object* MILK.

$$\text{PUTDOWNMILK} \sqsubseteq \text{MANIPULATIVEGESTURE} \sqcap \exists \text{HASACTOR}. \left(\text{PERSON} \sqcap \exists \text{DOESATOMICGESTURE}.\text{REACHMILK}\right) \tag{2.8}$$

$$\text{REACHMILK} \sqsubseteq \text{ATOMICGESTURE} \sqcap \exists \text{HASACTOR}. \Big(\text{PERSON} \sqcap \exists \text{HASARM}. \big(\text{ARM} \sqcap \exists \text{HASFUNCTION}.\text{REACH} \sqcap \exists \text{USESOBJECT}.\text{MILK}\big)\Big) \tag{2.9}$$

The described structure presents the basic idea behind our multi-level model. Nonetheless, it is too simple to comply with real life human activities. First, activities can not be merely defined in terms of finer grained ones, since the same operation can be a component of two or more *distinct* activities. In our framework, and as illustrated in Figure 2.2 for example, being involved in the *Simple Activity* "put away milk" can either mean that the subject is "cleaning up" or that they are "having a coffee". Second, the temporal sequence of actions and operations is essential to the definition of several activities and actions. For example,

the *Simple Activity* "get milk" is usually distinguished by *first* "opening the fridge" *then* "fetching milk". The same operations carried out in the inverse order would refer to the opposite *Simple Activity* "putaway milk". Among the four granularity levels proposed in our model, *Simple Activities* are especially sensitive to the temporal sequence of their components. However, the description logic underlying log-linear DL [NNS11] is OWL2, which does not natively support temporal reasoning. Therefore, we adopt an ad-hoc method based on an ontology pattern [2] that allows the representation of *triadic properties*. Concretely, we introduce a class T-MANIPULATIVEGESTURE, having three properties to represent the actor, the performed *Manipulative Gesture*, and its order of execution. Thus, the *Simple Activity* GETMILK can be describes as in indicated in Example 9.

Example 9. *The "Simple Activity" "*GETMILK*" defined as a sequence of "*OPENFRIDGE*" then "*FETCHMILK*" would then be defined through the following axiom*

$$\begin{aligned}\text{GETMILK} \sqsubseteq \text{SIMPLEACTIVITY} \sqcap \\ \exists\,\text{HASACTOR}.\Big(\text{PERSON} \sqcap \exists\,\text{HAST-MANIPGESTURE}. \\ \big(\text{T-MANIPULATIVEGESTURE} \sqcap \\ \exists\text{HASMANIPGESTURE}.\text{OPENFRIDGE} \\ \sqcap \exists\text{HASORDER} = 1\big) \sqcap \\ \exists\,\text{HAST-MANIPGESTURE}. \\ \big(\text{T-MANIPULATIVEGESTURE} \sqcap \\ \exists\text{HASMANIPGESTURE}.\text{FETCHMILK} \\ \sqcap \exists\text{HASORDER} = 2)\big)\Big).\end{aligned} \tag{2.10}$$

Incorporating temporal sequences in the activity model draws it closer to real world scenarios. Yet, it is very common that the same activity is carried out in different manners. For example, while "getting milk" is often invariable, "putting away milk", on the opposite, might either go through "opening the fridge" first then "fetching the milk" or the other way around. In order to keep the fridge's door closed as long as possible, it is more probable that the subject first "fetches the milk" then "opens the fridge". Hence, these temporal sequences are not deterministic and require uncertainty support. The same holds for the ambiguity of interpretation of finer-grained activities in terms of coarser-grained one. Indeed the fact that the same operation can often contribute to conflicting actions (i.e., actions that cannot be executed at the same time) would lead to the contradictory conclusion that the subject is involved in both of them.

Log-linear DL allows to cope with these obstructions. By introducing weighted axioms as concept description. The weighted axioms compose our *uncertain* CBox $\mathcal{C}^{\mathsf{U}}$, while we express the incompatibility of certain activities as disjointness axioms in the *deterministic* CBox $\mathcal{C}^{\mathsf{D}}$. As shown in Example 10, the axioms weights can be

[2] http://www.w3.org/TR/2004/WD-swbp-n-aryRelations-20040721/

manually defined based on background knowledge, or can be automatically learned from a training set of performed activities (see next Chapter for more details).

Example 10. *Revisiting Level 3 (Manipulative Gesture) of the sample multi-level structure illustrated in Figure 2.2, the corresponding* uncertain *CBox* $\mathcal{C}^{\mathsf{U}}$ *includes of the following axioms:*

$$\text{FETCHMILK} \sqsubseteq \text{MANIPULATIVEGESTURE} \sqcap \exists \text{HASACTOR}.\big(\text{PERSON} \sqcap \exists \text{DOESATOMICGESTURE.MOVEMILK}\big), 0.9 \tag{2.11}$$

$$\text{FETCHMILK} \sqsubseteq \text{MANIPULATIVEGESTURE} \sqcap \exists \text{HASACTOR}.\big(\text{PERSON} \sqcap \exists \text{DOESATOMICGESTURE.REACHMILK}\big), 1.2 \tag{2.12}$$

$$\text{PUTDOWNMILK} \sqsubseteq \text{MANIPULATIVEGESTURE} \sqcap \exists \text{HASACTOR}.\big(\text{PERSON} \sqcap \exists \text{DOESATOMICGESTURE.MOVEMILK}\big), 0.9 \tag{2.13}$$

$$\text{PUTDOWNMILK} \sqsubseteq \text{MANIPULATIVEGESTURE} \sqcap \exists \text{HASACTOR}.\big(\text{PERSON} \sqcap \exists \text{DOESATOMICGESTURE.RELEASEMILK}\big), 1.2 \tag{2.14}$$

The corresponding deterministic *CBox* $\mathcal{C}^{\mathsf{D}}$ *consists of the following axiom:*

$$\text{PUTDOWNMILK} \sqcap \text{FETCHMILK} \sqsubseteq \bot \tag{2.15}$$

The main components of the introduced multi-level activity ontology are recapitulated in Figure 2.1 and the entire ontology is attached as appendix. Based on this model, we describe our ontology-based technique to recognize human activities at different levels of granularity in the next section.

2.3 Recognizing multi-level activities

Given the probabilistic ontology described above, we can reason about the input sensor data in order to infer the user's operations, actions and activities in *real time*. As imposed by real-life scenarios, our recognition method is not limited to sequential performance of activities but also covers *concurrent ones*.

Concretely, raw sensor data is segmented and classified into higher level information using statistical methods. The output is linked to properties of our ontology in order to map the user's situation to a computational model and automatically reason about it. Given a specific situation in which the user is performing particular movements and interacting with particular object(s), we can assume that there is one or more unknown operations that have generated the observed situation. This concept of *unknown operation* is defined as an operation whose actor

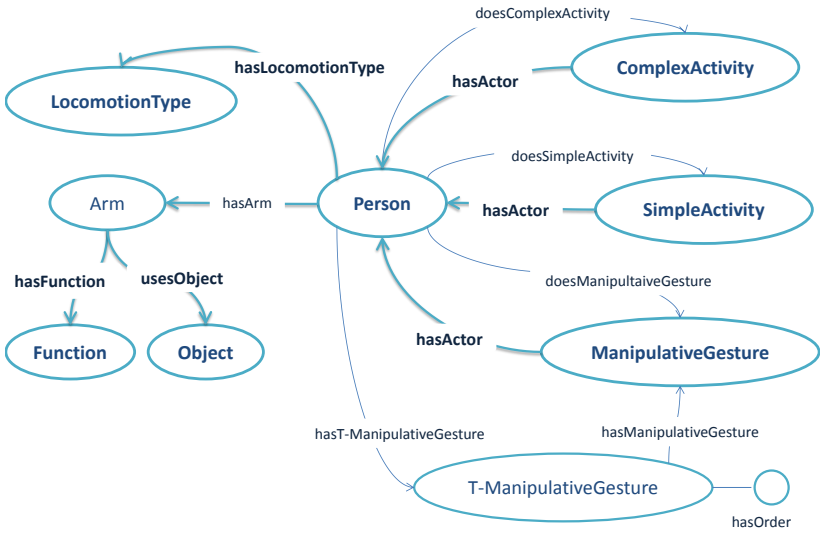


Figure 2.1: The core classes (represented as nodes) and properties (edges) of our multi-level activity ontology. It includes classes PERSON, ARM, FUNCTION and OBJECT to represent the user and their "arm functions" (e.g., push, pull, ...) as well as the used objects detected by the wearable and environmental sensors. The HASARM property relates each instance of class PERSON to their ARMS, and hence, to the sensor observations by properties HASFUNCTION and USESOBJECT. The ontology includes an extensive collection of *Atomic Gestures* (*Level* 4), *Manipulative Gestures*(*Level* 3), *Simple Activities* (*Level* 2) and *Complex Activities* (*Level* 1). As explained earlier, each of these classes is described in terms of finer grained ones, i.e. classes from the next level. *Simple Activities*, in particular, are defined in terms of the temporal sequences of *Manipulative Gestures* and modes of locomotion of the actor. We adopt an ontology pattern to keep track of those sequences, using the T-MANIPULATIVEGESTURE class and its HASORDER property.

is performing the observed sensor data. For instance, in the case of the "Opportunity" framework, specific gestures and postures of the user such as "reach", "move" and "lie" are inferred from wearable sensors like accelerometers and gyroscopes. The interaction with surrounding objects is detected via RFID tags and wearable RFID readers. Hence, the activation of sensors placed on a milk's bottle, for example, are fed into the ontology through the property usesObject(Milk), meaning that milk has been used. Similarly, the property hasFunction(Reach) links the observed movement "reach" to the ontology. The unknown operation is then defined as an "operation whose actor is reaching their hand and interacting with milk". If some operation concept(s) have been defined by this set of operation properties , e.g., FetchMilk, then that operation(s) can be deemed as the type of operation(s) for the perceived context, provided they are not contradictory. In the case of incompatibility (e.g. "open fridge" and "close fridge"), the operation(s) that lead to the most probable coherent ontology are selected. The same principle is employed

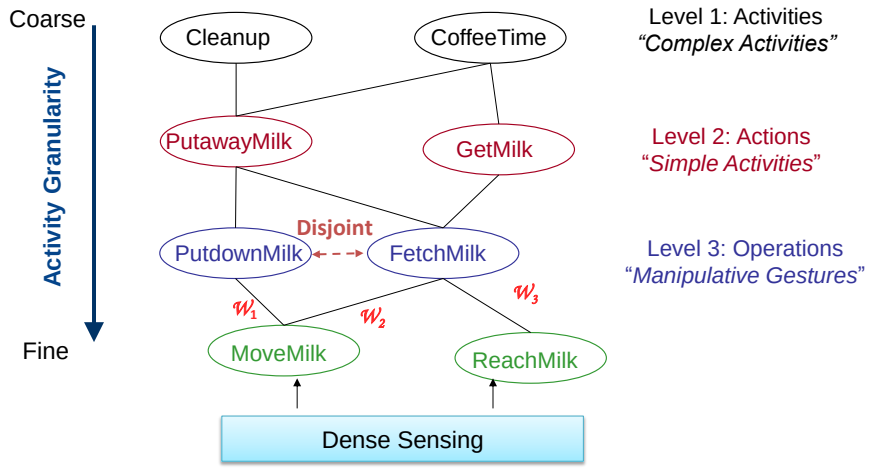


Figure 2.2: An example of the multi-level activities from the "*Opportunity*" framework [KHF+11]. The framework comprises four levels of granularity. While $level\,4$ corresponds to the sensor data, $level\,3$, $2\,and\,1$ depict the operation, action and activity levels respectively. In the Opportunity framework, these levels are referred to as *Manipulative Gesture*, *Simple Activity* and *Complex Activity*."move milk" and "reach milk" are two examples of *Manipulative Gesture*."Move milk" can either indicate that the user is "putting down milk" or that they are "fetching milk", but not both simultaneously due to the disjointness of the two actions. Adding weights (e.g. w_1) to the concept descriptions allows to model such inconsistent knowledge bases as explained below.

to recognize action(s) from the inferred operation(s) as well as activity (ies) from inferred action(s).

Thus, conceptually, the recognition problem can be mapped to the classification of the activity description using the multi-level activity ontology as classifier. Technically, it amounts to the task of subsumption reasoning with Description Logics (i.e., to decide whether a concept description created from sensor observations is is equivalent a concept definition within the activities model).

The following sections provide a detailed description of the recognition steps at the levels of operations (*Manipulative Gesture*), actions (*Simple Activity*) and activities (*Complex Activities*).

2.3.1 Recognizing operations (*Manipulative Gestures*)

Recall that operations, which are referred to as *Manipulative Gestures* in the "*Opportunity*" framework, are recognized based on the so called *Atomic Gestures* (e.g. "reach milk"), which are a straightforward combination of the used artifact (e.g., "milk") and the movement inferred from body worn sensors (e.g. "reach"). During a predefined time window τ_3, the performed body functions and used objects are first represented as ontological classes and assertions and added to the probabilistic ontology. Then, the resulting *Atomic Gestures* are linked to the actor via the properties hasFunction and usesObject. For each of these Atomic Gestures a *new axiom* is added to the deterministic CBox of the log-linear knowledge base. As explained above, this axiom describes the unknown *Manipulative Gesture(s)* (UNKNOWNMG) being carried out by the actor. The unknown *Manipulative Gesture* is defined in terms of the *Atomic Gestures* performed by the actor. For the example where the actor is observed to be engaged in "reaching milk", the following axiom is added:

$$\begin{aligned}\text{UNKNOWNMG} \equiv \text{MANIPULATIVEGESTURE} \sqcap \exists \text{HASACTOR}. \\ \big(\text{PERSON} \sqcap \exists \text{DOESATOMICGESTURE.REACHMILK}\big)\end{aligned} \tag{2.16}$$

After adding these observations, the resulting ontology may be inconsistent. For example, according to the multilevel activity structure, the set of *Atomic Gestures* performed during τ_3 may lead to the derivation of two (or more) disjoint *Manipulative Gestures*, i.e. *Manipulative Gestures* that cannot be executed at the same instant, like "open fridge" and "close fridge". Inconsistencies are resolved by computing the *most probable consistent and coherent ontology* $\mathcal{C}^*$ as explained in the previous section. The *Manipulative Gestures* performed during the time window τ_3 are then inferred through standard subsumption and equivalence reasoning on $\mathcal{C}^*$. These steps are summarized in Algorithm 2.

Note that collecting operations within a time window creates a semantic context that helps address the ambiguity of interpretation of these operations regardless of their temporal order. As example, let us consider the scenario where a person is performing the two operations "move milk" and "reach milk" during a time window of duration τ_3. As illustrated in Figure 2.2, "move milk" can either indicate that the user is "putting down milk" ($axiom_1$ with weight w_1) or that they are "fetching milk" ($axiom_2$ with weight w_2), but not both simultaneously due to the disjointness of the two actions. Given that the observed operation "reach milk" is only involved in action "fetch milk" ($axiom_3$ with weight w_3) but not in action "put down milk", the log-linear model will attribute a higher probability to the ontology where $axiom_2$ and $axiom_3$ hold, but not $axiom_1$. Thus, "move milk" is interpreted in the context of the operation "reach milk" since they both happened during the same time window.

Algorithm 2 Recognizing Operations (*Manipulative Gestures*)

$elapsedTime \leftarrow 0$;
$S \leftarrow \emptyset$;
for all $New\ Atomic\,Gesture\ AG$ **do**
  while $elapsedTime < \tau_3$ **do**
    $\mathcal{S} \leftarrow \mathcal{S} \cup AG$;
    $Update\,elapsedTime$;
  end while
  for all $AG \in \mathcal{S}$ **do**
    $newAxiom \leftarrow$ "UNKNOWNMG $\equiv$ MANIPULATIVEGESTURE $\sqcap$ $\exists$ HASACTOR.$\big($PERSON $\sqcap \exists$ DOESATOMICGESTURE.AG$\big)$";
    $\mathcal{C} \leftarrow \mathcal{C} \cup newAxiom$;
  end for
  $\mathcal{C}^* \leftarrow MAPQuery(\mathcal{C})$;
  ${\mathcal{C}^*}_s \leftarrow subsumptionChecking(\mathcal{C}^*)$
  for all $MG \sqsubseteq$ MANIPULATIVEGESTURE $in\ {\mathcal{C}^*}_s$ **do**
    if $MG \equiv UnknownManipulativeGesture$ **then**
      return MG
    end if
  end for
  $Reset\,elapsedTime$;
  $S \leftarrow \emptyset$;
end for

2.3.2 Recognizing actions (*Simple Activities*)

Similar to the recognition of operations (*Manipulative Gestures*) from sensor observations, the predicted operations are used to create a class description of the concept "*unknown Simple Activity*" UNKNOWNSA. However, instead of deleting the current collection of *Manipulative Gestures*, these are stored in a buffer as long as no *Simple Activity* has been recognized. In particular, from one time window to the next, the order of each buffered *Manipulative Gesture* is updated accordingly. Since in our ontology the longest sequence composing a *Simple Activity* consists of four *Manipulative Gestures* these can have a maximum order of four before they are deleted from the buffer. Once a *Simple Activity* concept is found to be equivalent to the UNKNOWNSA class description, the axiom explaining that equivalence is retrieved. If no equivalent classes are found, the buffer is updated by pushing the new *Manipulative Gestures* and deleting the oldest one(s). We refer to the time window covering these buffered *Manipulative Gestures* as τ_2. A detailed description of these recognition steps is depicted by Algorithm 3.

Algorithm 3 Recognizing Actions (*Simple Activities*)

$elapsedTime \leftarrow 0$;
$S \leftarrow \emptyset$;
$Buffer\ \mathcal{B}$;
for all $predicted\ ManipulativeGesture\ MG$ **do**
  while $Greatest - MG - Order < 5$ **do**
    $push\ MGs\ into\ \mathcal{B}$;
    $Increment\ maximumOrder$;
  end while
  for all $MG \in \mathcal{B}$ **do**
    $newAxiom \leftarrow description\ of\ concept$ UNKNOWNSA $using\ \mathcal{B}$
    $\mathcal{C} \leftarrow \mathcal{C} \cup newAxiom$;
  end for
  $\mathcal{C}^* \leftarrow MAPQuery(\mathcal{C})$;
  ${\mathcal{C}^*}_s \leftarrow subsumptionChecking(\mathcal{C}^*)$
  for all $SA \sqsubseteq$ SIMPLEACTIVITY $in\ {\mathcal{C}^*}_s$ **do**
    if $SA \equiv$ UNKNOWNSA **then**
      return SA
    end if
  end for
end for

2.3.3 Recognizing activities (*Complex Activities*)

As indicated by Algorithm 4, the process of recognizing activities from actions (i.e. *Complex Activities* from *Simple Activities*) undergoes similar steps as the recognition of operations (i.e. *Manipulative Gestures*). Here, the *Simple Activities* are collected over a longer time window τ_1. During that time, several *Simple Activities* might be performed. The resulting context helps discriminate between the corresponding *Complex Activities* and allows the omission of intra-activity temporal relationships while keeping a reasonable recognition performance (see next Chapter for results).

Algorithm 4 Recognizing Activities (*Complex Activities*)

$elapsedTime \leftarrow 0$;
$S \leftarrow \emptyset$;
for all $Recognized\,Simple\,Activity\,SA$ **do**
  while $elapsedTime < \tau_1$ **do**
    $\mathcal{S} \leftarrow \mathcal{S} \cup SA$;
    $Update\,elapsedTime$;
  end while
  for all $SA \in \mathcal{S}$ **do**
    $newAxiom \leftarrow$ "UNKNOWNCA $\equiv$ COMPLEXACTIVITY $\sqcap$ $\exists$HASACTOR.$\big($PERSON $\sqcap$ $\exists$DOESSIMPLEACTIVITY.SA$\big)$";
    $\mathcal{C} \leftarrow \mathcal{C} \cup newAxiom$;
  end for
  $\mathcal{C}^* \leftarrow MAPQuery(\mathcal{C})$;
  ${\mathcal{C}^*}_s \leftarrow subsumptionChecking(\mathcal{C}^*)$
  for all $CA \sqsubseteq$ COMPLEXACTIVITY $in\ {\mathcal{C}^*}_s$ **do**
    if $CA \equiv$ UNKNOWNCA **then**
      return CA
    end if
  end for
  $Reset\,elapsedTime$;
  $S \leftarrow \emptyset$;
end for

3
Evaluation and Results

In order to evaluate the approach described in the previous chapter, a prototype system has been implemented. The evaluation experiments have been carried out using a real-life dataset collected in the context of the EU research project "Activity and Context Recognition with Opportunistic Sensor Configuration" ("*Opportunity*") [KHF+11] [1]. In this chapter we describe our framework, present the evaluation experiments then report and discuss the obtained results. Additionally to the results published in [HRS13], this chapter leverages some new content such as the introduction of two baselines and an approach to learn log-linear axioms weights.

3.1 Evaluation dataset

Recall from Part I, Chapter 3 that a total of of 72 sensors with 10 modalities have been deployed in the context of the "*Opportunity*" project. The testbed simulates a studio flat where a naturalistic collection process of a *morning routine* has been carried out by several users. As visible in Figure 3.3, the deployed sensors can be classified into wearable sensors (e.g. accelerometers) and environmental sensors (e.g. RFID tags and readers). The worn sensors are used to detect the postures of the users and the movements of their hands (e.g. "lie", "reach"). The environmental sensors indicate which objects the user is manipulating (e.g. "Knife"). Whereas the dataset provides this inferred information, it only includes the annotation for the highest level of activities (i.e. *Complex Activities*). We have completed the annotation task for three different subjects $S10$, $S11$, and $S12$ with three different routines each ($ADL1 - 3$). The annotation task has been accomplished by three

[1] http://www.opportunity-project.eu

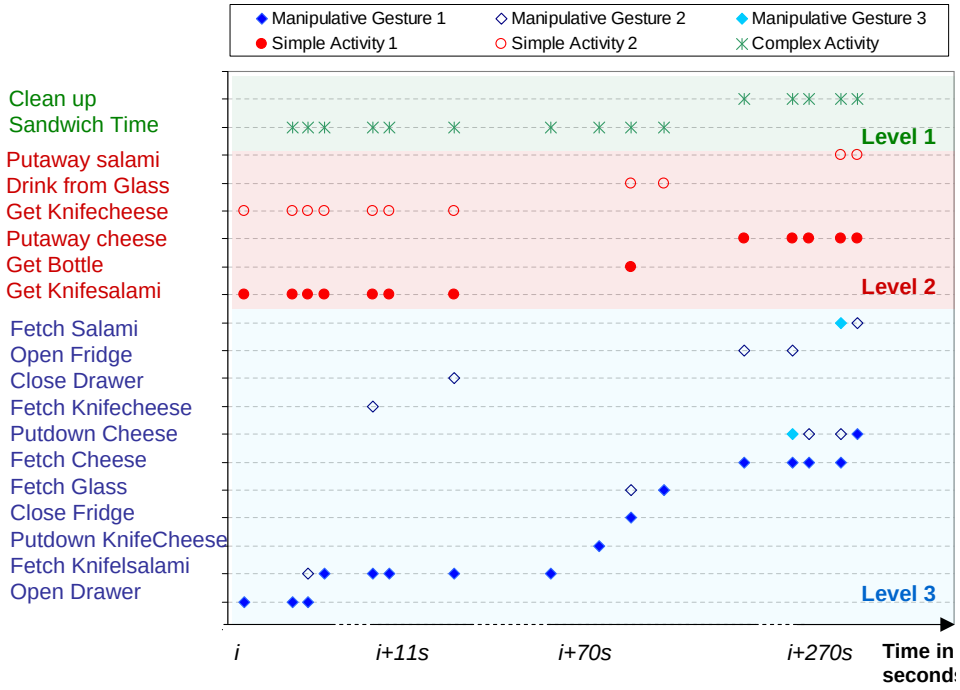


Figure 3.1: Illustration of a typical portion from the multilevel structured activities. This shows a high concurrency degree. As it can be depicted, the user can be simultaneously involved in "Fetch Glass", "Close Fridge" at level3, and "Drink from Glass", "Get Bottle" at level 2

other persons. The resulting diversity has inevitably impacted the consistency of data. While labels have been attributed to events at each level of granularity, they do not cover all entries, leaving some sensor observations with no annotation at one or more levels.

Figure 3.1 depicts a sample from the collected and annotated data. In the illustrated sequence, the user is first engaged in the *Complex Activity* "sandwich time" then they start "cleaning up". This is indicated by the asterisks plotted in the upper strip of the figure. In the following strip, the *Simple Activities* are represented by circles. According to these, the user mostly carries out two *Simple Activities* concurrently. For instance, around time point $i + 11$ they "get a cheese knife" and "get a salami knife" at the same time (each with one hand). Some time later, they "get a bottle" and start "drinking from glass" without "releasing the bottle". Finally, after few activities omitted in the figure, they "put down the cheese knife" as well as the "salami knife" simultaneously. Similarly, *Manipulative Gestures* are also highly concurrent. For instance, according to the plotted lozenges, the user is still "fetching the salami knife" with one hand while "closing the drawer" with the other.

Regardless of their granularity level, about 150 activities have been considered during the data collection. Among those, 40 belong to *Manipulative Gestures* level,

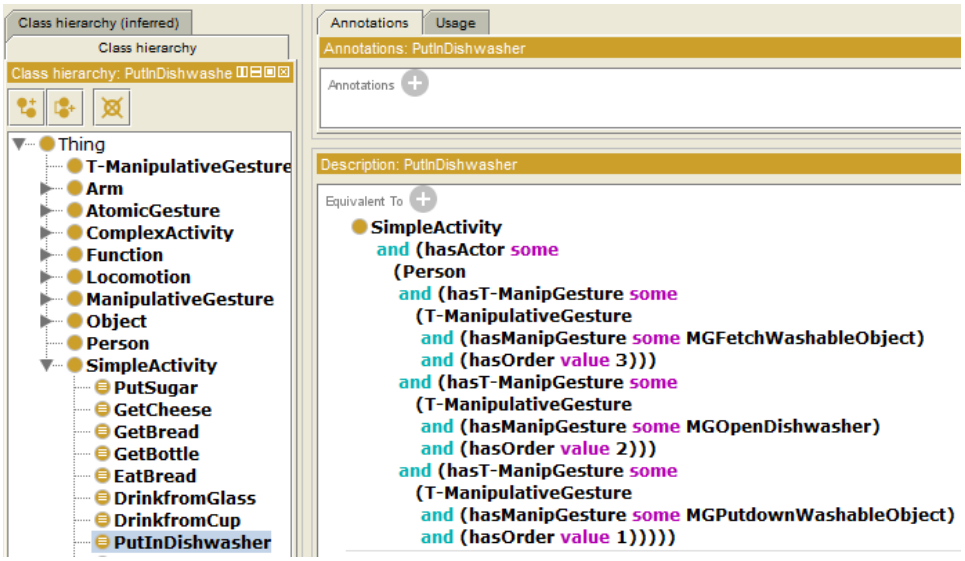


Figure 3.2: Axiom definition for *PutInDishwasher* in the Protégé editor [HRS13]

21 to the *Simple Activities* level. The *Complex Activities* set comprises "Clean up","Coffee time", "Relaxing" and "Sandwich time".

3.2 Implementation and experimental setup

Our proposed system presents two major components. The first consists of the log-linear ontology which models the domain of discourse. The second provides a java-based framework to parse the ontology and augment it with the input data, initiate the reasoning process and output the inferred activities. The following sections describe the proposed framework and the performed experiments.

3.2.1 Framework description

Following the "*Opportunity*" structure, our log-linear ontology has been developed using the Protégé OWL editor [KFNM04]. The activity classes and axiom's weights have been defined by observing the data of user $S10$. Concretely, the weights are encoded by attaching the annotation property confidence to the corresponding axioms. An example illustrating the definition of PUINDISHWASHER as shown in the Protégé editor is captured in the snapshot depicted in Figure 3.2.

For parsing the ontology, our prototype system relies on the OWL-API [2]. Following the algorithms delineated in the previous Chapter, the program starts by asserting an instance of PERSON representing the current individual to the ontology. Using the input data collected within a time window of $\tau_3 = 1s$ the classes ARM, FUNCION and OBJECT as well as the the corresponding properties are instantiated

[2] http://owlapi.sourceforge.net/

accordingly. Once the elapsed time at starting point exceeds τ_3, the reasoning process described below is triggered.
Based on the observed instances, the resulting *Atomic Gestures* are first added to the ontology and used to introduce the UNKNOWNMG class concept to the probabilistic ontology. We use the *Elog* reasoner [3] [NN11] to output the most probable coherent and consistent one. Concretely, the reasoner solves the MAP inference task by transforming the input log-linear knowledge base in a Markov logic knowledge base and inferring the MAP state using the Markov logic solver ROCKIT [NNS13]. After the execution of ROCKIT and, thus, solving the ILP problem, ELOG translates the retrieved MAP state back to ontology axioms and returns a materialized OWL ontology [Noe14]. To guarantee the consistency and coherency of the returned solution, Elog iteratively queries the Pellet reasoner [SPG$^+$07] to derive explanations for emerging incoherences or inconsistencies and adds those as new constraints to the ILP. The new problem is solved and the process is started over again until all inconsistencies and incoherences are resolved. The resulting coherent and consistent ontology serves again as input for the Pellet reasoner to infer the equivalent class(es) to the introduced "unknown class". The obtained classes represent the predicted *Manipulative Gestures* during the given time window. At the next sensor input, the collected data is deleted and the whole collection-reasoning process is triggered again. In order to reduce the complexity of this process, we dynamically discard the axioms that do not involve the currently observed input and reason with a subset of the axioms.
The obtained *Manipulative Gestures* are collected and introduced to the ontology through new axioms describing the UNKNOWNSIMPLEACTIVITY concept. As long as no *Simple Activity* is inferred, the whole process is repeated up to 4 times. Once a *Simple Activity* SA_i is found to be equivalent to the unknown simple activity concept, the system retrieves the axioms explaining that equivalence using the Pellet's explanation feature. This allows to identify the number n, $n \in \{1, ..., 4\}$ of *Manipulative Gestures* time windows τ_3 involved in recognizing SA_i. The recognition results are then completed retroactively by attributing SA_i as one of the predicted *Simple Activities* over the last n τ_3 time windows.
Finally, *Complex Activities* are recognized using the *Simple Activities* collected during a time window $\tau_1 = 30s$. The lengths of the respective time windows $\tau_i, i \in \{1, 2, 3\}$ have been estimated from the data used to create the ontology. Figure 3.3 recapitulates the rationale behind the recognition process by sketching the major steps implemented in our prototype .

3.2.2 Experiments and evaluation

We have conducted three sets of experiments in order to evaluate our approach. Each set corresponds to the data generated by one of the three available subjects, i.e. $S10$, $S11$ and $S12$. For each set, we have applied our recognition algorithm

[3] https://code.google.com/p/elog-reasoner/

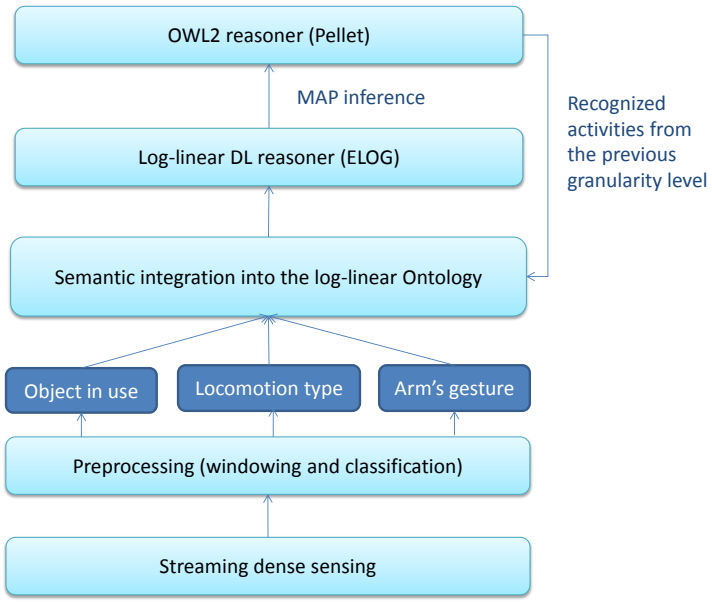


Figure 3.3: Schema of the proposed recognition framework: our prototype implements three phases in order to recognize multi-level activities. The first consists in integrating the pre-processed sensor data into the log-linear ontology along with a new unknown activity concept which models the activity(ies) to be predicted. The second runs the log-linear DL reasoner on the resulting ontology to obtain the most probable one. The third applies the DL reasoner Pellet to reason about the Elogs' output, in order to infer implicit knowledge about the activity concepts equivalent to the unknown activity concept. Those correspond to the recognition results for the finer grained level. Following the same steps explained above, these are in their turn integrated into the log-linear ontology in order to recognize coarser grained activities

to three daily routines, i.e. $ALD1$, $ADL2$ and $ADL3$. Additionally, we have carried out the same set of experiments using two **baseline** approaches. Their basic idea consists in reasoning with the classical non-probabilistic version of the activity ontology. Theoretically, in the absence of any support for uncertainty in our framework, one of the following two situations should appear at each time window: situation (1) either the input contains no confounding components and the introduced UNKNOWNACTIVITY class can be, thus, mapped to the corresponding activity(es) or situation (2) it does, and results in a set of disjoint activities, making the added UNKNOWNACTIVITY class unsatisfiable.

Nonetheless, simply using the weight-free version of the original log-linear ontology in the recognition process would still result in an unsatisfiable UNKNOWNACTIVITY concept even in the second situation. Indeed, as soon as the user is engaged in any activity that shares one or more of its definition axioms with its disjoint counterpart, the UNKNOWNACTIVITY class will be equivalent to those two disjoint classes and, hence, unsatifiable. Thus with such an ontology, the system

only returns a non-empty output for those time windows where the user in only engaged in an activity that has no disjoint classes. Given that only 10% of the *Manipulative Gestures* do not have a disjoint class and that the execution of the different activities is well balanced throughout the datasets, the recognition results (average F1-measure) are, in that case, as low as 0.15 for *Manipulative gestures*, 0.06 for *Simple Activities* and 0.27 for *Complex Activities*. To avoid this, we propose the following two alternatives, where the first addresses situation (1) and the second addresses situation (2) as detailed below.

Baseline 1: using subsumption axioms instead of equivalence In this baseline we change the definition (equivalence) axioms to subsumption axioms. Thus, and as illustrated in Figure 3.4, the *Manipulative Gesture* "fetch milk" for example, would be a superclass of the two anonymous classes "*Manipulative Gesture* which has an actor is performing the *Atomic Gesture* move milk" and "*Manipulative Gesture* which has an actor is performing the *Atomic Gesture* reach milk". Since the pellet reasoner would skip unnamed classes, these would be only considered in the reasoning when an UNKNOWNMG class in added as equivalent to those unnamed class expressions (i.e. "*Manipulative Gesture* which has an actor is performing the *Atomic Gesture* move milk" or "*Manipulative Gesture* which has an actor is performing the *Atomic Gesture* reach milk" in our example).

Thus, given a time window w, if the user is engaged in the *Atomic Gesture* "move milk", which would result in the two disjoint *Manipulative Gestures*, "fetch milk" and "put down milk", then the system's output would be *null*. However, if the user is engaged in the *Atomic Gesture* "reach milk", that is only part of the *Manipulative Gesture* "fetch milk", then the system still returns "fetch milk" as recognized *Manipulative Gesture*. Thus, the difference between this approach and the one with the equivalent axioms (i.e. the weights-free version of the original log-linear ontology) is that this approach models two disjoint activity classes in a way, that they are not always unsatisfiable, but only then, when the user is engaged in one common component, i.e. "move milk" in the above example.

After creating this static ontology variant, we alter the recognition process by skipping the log-linear reasoning (i.e. the Elog reasoner) and directly applying subsumption check to derive the **direct superclasses** of the unknown *Manipulative Gesture*, *Simple Activity* or *Complex Activity*.

Baseline 2: removing disjointness axioms In order to implement situation (2) explained above, this baseline approach creates another variant of the original log-linear ontology by removing the disjointness axioms. Thus, in each time window where the system can not decide whether the user is carrying out an activity or its opposite, this approach outputs both. In our "fetch milk" example, this method would recognize both "fetch milk" and "put down milk" in any time window where the user is "moving milk" and "reaching and/or releasing milk".

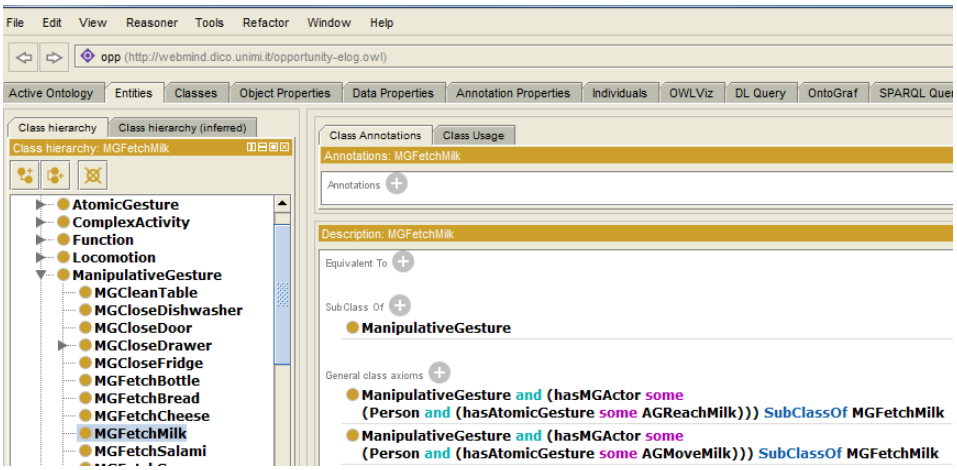


Figure 3.4: Example class description of the Baseline 1 ontology: In order to avoid the unsatisfiability of the unknown user classes, we alter the class descriptions by using subsumption axioms instead of equivalence ones. the Manipulative Gesture Fetch Milk would be then a superclass of two possible descriptions.

In order to evaluate the prediction output of the explained experiments, we first reproduce the same windowing technique on the ground truth and obtain a set of *Manipulative Gestures* for each τ_3 time window, a set of *Simple Activities* for each τ_2 time window and a set of *Complex Activities* for each τ_1 time window. Then we compare those to the set of predicted activities at the corresponding time windows. Based on this, we compute the precision, recall and F1-measure as explained in the Preliminaries Chapter.

3.3 Results and discussion

We depict the results of our multi-level recognition framework in Table 3.1. Given that the ontology has been created using data from the user $S10$, the first column represent user-dependent evaluation. User-independent evaluation is reported in the second and third columns using the data of user $S11$ and $S12$ respectively. The rows correspond to the three levels of abstraction adopted in our activity model. For each subject, we calculate the mean values of the precision, recall and F1-measure over the three routines (ADL1, ADL2 and ADL3) as well as the corresponding standard deviations σ.

For a compacter representation of the results, Table 3.2 portrays the overall average performance for each granularity level using the entire data. Despite the variability of the collected data due to involving different users and different annotators, our system delivers comparable performance levels, i.e. relatively small sigma values. Accordingly, the reported results validate the robustness of the approach under user-independent setting.

Generally, the system is highly precise independently of the granularity level. However, while the obtained recall values are acceptable for *Manipulative Gesture* and *Complex Activities*, the system seems to miss a relatively significant number of *Simple Activities*. This limitation has several reasons. On one side, the erroneous and missing predictions from the *Manipulative Gesture* level most probably violate an entire *Simple Activity* sequence. Now, since *Simple Activities* usually spread over two, three or four *Manipulative Gesture* time windows, each wrong prediction at the *Manipulative Gesture* level probably results in one, two, three or even four missing predictions at the *Simple Activity* level. For instance, if the ground truth contains the following *Manipulative Gesture* sequence: {("OpenFridge", t), ("FetchMilk", $t+1$), ("CloseFridge", $t+2$)}, which corresponds to the following sequence at the *Simple Activity* level: {("GetMilk', t'), ("GetMilk", $t+1$), ("GetMilk", $t+2$)}, then any error in the *Manipulative Gesture* sequence probably leads to a non-existent *Simple Activity* sequence and will consequently lead to three *Simple Activity* false negatives. Given that the user operates with two hands, such scenarios are very likely. In particular, the user might be still touching the "Fridge" while "fetching the milk", which alters the sequence into the following: "OpenFridge", "FetchMilk", "OpenFridge", "CloseFridge" and inhibits the recognition through the defined axiom. This shortcoming can be alleviated by adding further weighted axioms defining the same *Simple Activity* using all partial sequences of *Manipulative Gestures*. For the example given above, this corresponds to, for instance, adding further weighted axioms defining "GetMilk" in terms of subsequences such as {"OpenFridge", t, "FetchMilk", $t+1$}. Consequently, if the the system fails in recognizing "CloseFridge" at $t+2$, the system would still be able to output the correct *Simple Activity* for t and $t+1$. Our system already implements some of these axioms. However, extensively specifying all the different sequences of *Manipulative Gestures* that may characterize *Simple Activities* is infeasible due to the limited temporal reasoning support related to the adopted formalism.

Despite the low recall at *Simple Activities* level, *Complex Activities* have been recognized with a relatively high F1-measure (i.e. 0.75). This can be explained by the fact that the set of *Complex Activities* is limited to 4 disjoint classes, which can be discriminated by "*key*", highly weighted *Simple Activities*. For example, the *Complex Activity* "Clean up" can be strongly discriminated from the other *Complex Activities* by *key Simple Activities* such "put in dishwasher". Hence, the weighted axiom defining "Clean up" in terms of "put in dishwasher" will be highly weighted and will allow the correct recognition of that complex activity. Even if the system fails in recognizing several *Simple Activities* within a given τ_1 time window, it is highly probable that the correct *Complex Activity* is still recognized thanks to the high precision of the predicted *Simple Activities*.

Comparing these results to those of the baselines introduced in the previous section reinforces the viability of our framework. As visualized in Figure 3.5, Baseline 1 yields a high precision but very poor recall whereas Baseline 2 does

Table 3.1: Recognition results for the three subjects S10, S11 and S12. The values correspond to the average values over three morning routines of each subject. The standard deviation between these routines is symbolized by σ.

		Subject S10	Subject S11	Subject S12
Complex Activity	Precision	0.9(σ0.038)	0.92(σ0.025)	0.92(σ0.06)
	Recall	0.58(σ0.114)	0.71(σ0.05)	0.65(σ0.08)
	F1-measure	0.7(σ0.088)	0.8(σ0.041)	0.76(σ0.074)
Simple Activity	Precision	0.87(σ0.045)	0.82(σ0.075)	0.88(σ0.029)
	Recall	0.4(σ0.054)	0.37(σ0.008)	0.5(σ0.051)
	F1-measure	0.55(σ0.042)	0.51(σ0.021)	0.64(σ0.042)
Manipulative Gesture	Precision	0.87(σ0.017)	0.84(σ0.206)	0.84(σ0.021)
	Recall	0.82(σ0.193)	0.79(σ0.031)	0.82(σ0.21)
	F1-measure	0.85(σ0.017)	0.81(σ0.024)	0.83(σ0.016)

Table 3.2: Average recognition results over three routines for subjects $S10$, $S11$ and $S12$. Each subject was evaluated using three different routines.Only the data generated $S10$ was considered to build our ontology and define its axioms. The variation (σ) between the results of respective users is also reported.

All Users	*Manipulative Gestures*	*Simple Activities*	*Complex Activities*
Precision	$0.85(\sigma\, 0.01)$	$0.86(\sigma\, 0.02)$	$0.91(\sigma\, 0.01)$
Recall	$0.81(\sigma\, 0.01)$	$0.42(\sigma\, 0.05)$	$0.65(\sigma\, 0.04)$
F_1	$0.83(\sigma\, 0.01)$	$0.57(\sigma\, 0.05)$	$0.75(\sigma\, 0.03)$

the opposite for the recognition of Manipulative Gestures. This is an expected outcome given that Baseline 1 skips all time windows where both an activity and its disjoint opposite could be implied, hence the high precision and low recall. Reversely, Baseline 2 outputs all candidate activities even if those include opposite pairs, hence the high recall and low precision. These values give an insight into the challenges imposed by the *Opportunity* dataset. In particular, it indicates that only around 20% of the time intervals are unambiguous and can be directly correctly mapped to the correct Manipulative Gesture. It also reveals that almost half the time windows do have more than one interpretation and can not be straightforwardly mapped to a Manipulative Gesture. The resting 30% refer to time intervals that happen to contain two opposite Manipulative Gestures. Those explain the difference between the recall level of Baseline 1 and Precision level of Baseline 2. Our approach proposes a trade-off between both baselines and reaches a significantly higher F1-measure than those. It allows to address the ambiguous time intervals thanks to the probabilistic feature of the ontology and reasoning routine.

At the *Simple Activities* level, the recognition performance is remarkably low

for both baselines. They both perform poorer than our framework, as shown in Figure 3.6. Given the very low recall of the *Manipulative Gesture* recognition achieved by Baseline 1, that method would fail in recognizing almost all *Simple Activities* since these require a sequence of correctly predicted *Manipulative Gestures*. Whereas the recall value of Baseline 2 is comparable to the one of our framework, it is significantly less precise than our approach. This can be explained by the fact that several opposite *Simple Activities* share similar sequence patterns for opposite *Manipulative Gestures*. Thus, if two opposite *Manipulative Gestures* are recognized within one time window, these would probably result in the prediction of two opposite *Simple Activities* according to Baseline 2. Hence the high number of false positives compared to our approach.

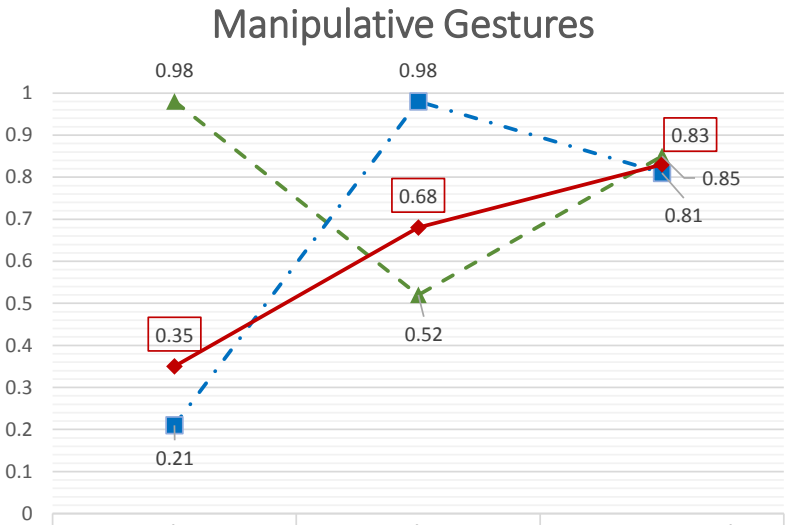
Finally, Figure 3.7 presents the recognition results at the *Complex Activities* level. At that level, our approach considerably outperforms both baselines in terms of recall and precision.

3.4 Work in progress and future work

We are currently considering further modelling techniques to address incomplete data. This can be realized by adding category classes such as "use fridge" as a superclass for all the *Atomic Gestures* involving the object "fridge" such as "open fridge" or "close fridge". Such an approach would allow to avail of the subsumption semantics and derive inferences of composite activities even if the correct "gesture" is missing from the sensor observations. Further improvement suggestions could also include adding weights to the disjointness axioms of our log-linear ontology. This might improve the results by possible addressing the occasional co-occurrence of opposite activities withing one time window.

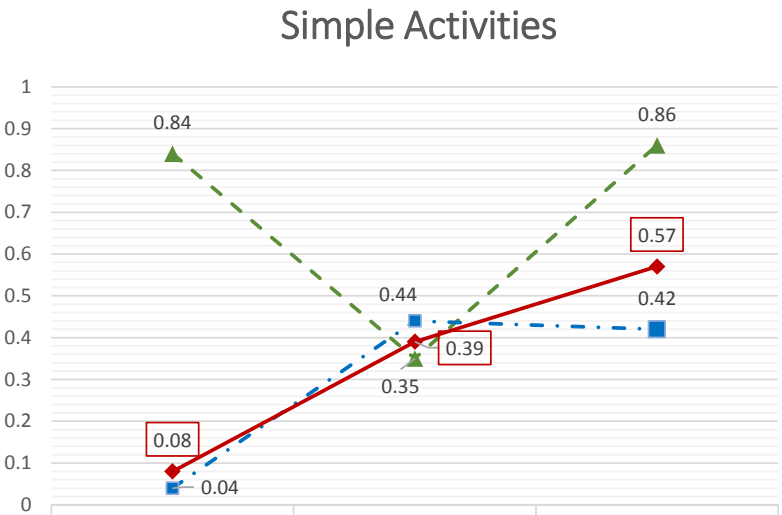
We additionally investigate two further directions for future work. The first is the automatic estimation of the axiom's weights from data and the second is a holistic approach to recognize the three levels of granularity instead of the current sequential approach.

Automatic estimation of the axiom's weights: In order to alleviate the modeling effort, we propose to automatically learn the axioms weights from the data. This can be done by creating the equivalent Markov logic network and using the existing learning algorithms such as voted perceptron [SD05]. We have applied this idea to the recognition of *Manipulative Gestures*. Concretely, we have defined the Markov network depicted in Table 3.3 where we model the operation carried out by the user as well as the object the user is interacting with, separately. We define two types of operations: AG_Operation , i.e. operations corresponding to the observed *Atomic Gestures* and MG_Operations, i.e. operations that correspond to the hidden *Manipulative Gesture*. The proposed Markov network lifts the ontology axioms defining the *Manipulative Gestures* by making statements about sets



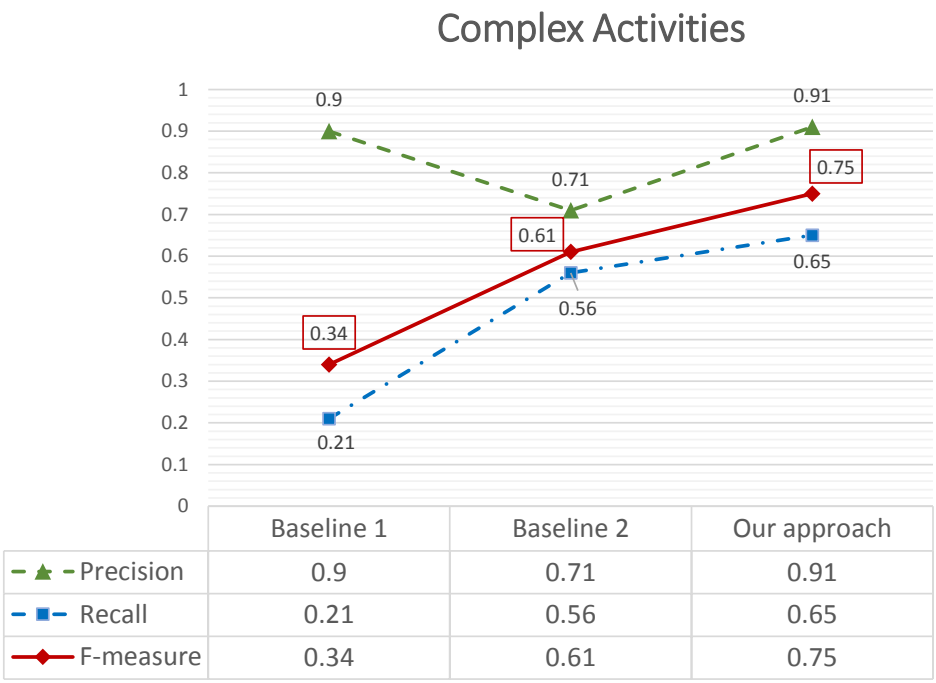
	Baseline 1	Baseline 2	Our approach
Precision	0.98	0.52	0.85
Recall	0.21	0.98	0.81
F-measure	0.35	0.68	0.83

Figure 3.5: *Manipulative Gestures* recognition performance compared to baseline results: As expected, Baseline 1 yields a high precision but very poor recall whereas Baseline 2 does the opposite. Our approach proposes a trade-off between both and reaches a significantly higher F1-measure



	Baseline 1	Baseline 2	Our approach
Precision	0.84	0.35	0.86
Recall	0.04	0.44	0.42
F-measure	0.08	0.39	0.57

Figure 3.6: *Simple Activities* recognition performance compared to baseline results: the effect of the limited temporal reasoning support in the adopted formalism can be seen in the low recognition performance for simple activities reached by our approach as well as the two baselines. The temporal sequences modeled in the ontologies are highly sensitive to the performance level of the recognition of the *Manipulative Gestures*. Also at this level our method outperforms both baselines



	Baseline 1	Baseline 2	Our approach
Precision	0.9	0.71	0.91
Recall	0.21	0.56	0.65
F-measure	0.34	0.61	0.75

Figure 3.7: *Complex Activities* recognition performance compared to baseline results: similarly to the *Manipulative Gesture* level, the expected performance of Baselines 1 and 2 is depicted in the respective low recall/high precision and high recall/low precision results. Given the better recognition results of the two other levels, our method outperforms the two baselines in both recall and precision.

of these. Hence, the ontology axioms for that level of granularity are factorized into one single soft rule and two hard formulae. The soft formula, i.e, formula 1, captures the weights between any "*related*" *Atomic Gesture* and *Manipulative Gesture* pair. An *Atomic Gesture* AG and a *Manipulative Gesture* MG are related if MG is defined in terms of AG in our ontology. The two hard formulae factorize the disjointness axioms by stating that the same object can not be "fetched" and "put down" at the same instant and the same for "opening" and "closing" the same object. The weights are learnt using the Markov logic engine TheBeast [Rie08]. The obtained weights are attributed to the corresponding ontology axioms and the recognition algorithm is applied. As depicted in Table 3.4, the obtained average results outperform those of the manual setting.

Motivated by these promising results, we intend to apply the same idea to the recognition of the two other levels of granularity. Modeling the set of template rules defining *Simple Activities* in terms of temporally ordered *Manipulative Gesture* is one of the main challenges.

Holistic recognition approach: Jointly recognizing the activities at different levels of granularity allows to reason with them as a whole rather than as a col-

Table 3.3: MLN formulae to automatically estimate the weights of our log-linear ontology axioms.

	Soft Constraints
1	$\forall Timestep\, t,\ AGOperation\, a,\ MGOperation\, m,\ Object\, o_1,\ o_2 :$ $[currentAG(a, o_1, t) \wedge areRelated(a, o_1, m, o_2)$ $\Rightarrow currentMG(m, o_2, t)]$
	Hard Constraints
2	$\forall Timestep\, t,\ Object, o :$ $currentMG(\text{"}Fetch\text{"}, o, t) \Rightarrow \neg currentMG(\text{"}Putdown\text{"}, o, t)]$
3	$\forall Timestep\, t,\ Object, o :$ $currentMG(\text{"}Open\text{"}, o, t) \Rightarrow \neg currentMG(\text{"}Close\text{"}, o, t)]$

Table 3.4: Results of recognizing *Manipulative Gestures* for the three subjects S10, S11 and S12 with automatically extracted weights versus manually designed weights. The F1-measures reported correspond to the average F1-measure over three morning routines of each subject. The standard deviation between these routines is symbolized by σ.

	$S10$	$S11$	$S12$
Automatically extracted weights	0.87(σ0.021)	0.83(σ0.035)	0.84(σ0.021)
Manually designed weights	0.85(σ0.017)	0.81(σ0.024)	0.83(σ0.016)

lection of levels. In particular, the recognition of finer grained activities can benefit from the context gained at a coarser grained level. For example, given that the user is carrying out the *Complex Activity* "cleaning up", the holistic approach would end up favoring the prediction of the *Manipulative Gestures* that would lead to that *Complex Activity*. Concretely, it would allow to choose the axioms that would lead to the overall greater sum of weights over the three levels rather than sequentially choosing the ones with the highest weights on each level separately. Figure 3.8 illustrates the difference between both approaches based on a simple example. In that Figure, the simple multi-level structure from Figure 2.2 is revised and extended with further weighted axioms defining the *Complex Activity* "clean up" in terms of the *Simple Activity* "put in dishwasher" which is in its turn defined in terms of the *Manipulative Gesture* "open the dishwasher". For the sake of simplicity, we omit the temporal order at the *Simple Activity* level. Let us assume that the user is "cleaning up". In particular, let us assume they are "having milk" in one hand to put it away and "opening the dishwasher" to "put the dirty dishes there". Hence, the system gets as input two *Atomic Gestures* "move milk" and "reach dishwasher" within a τ_3 time window. Based on the defined weights, the sequential approach would output "open dishwasher" and "fetch milk" as predictions for the *Manipulative Gestures* level. These would be input to the next recognition step, which would predict "get milk" and "put in dishwasher" as *Simple Activ-*

ities. Finally, the system would recognize "coffee time" as the user's *Complex Activity*. Allowing the system to reason with the different levels of activity granularity at the same time, however, would lead to choosing the paths that yield the highest *sum of weights*, i.e. those that go through the *Simple Activities* "put in dishwasher" and "put away milk" and the *Manipulative Gestures* "open dishwasher" and "put down milk". Thus, the holistic approach leverages an *indirect feedback* from coarser grained activity levels to finer grained ones to improve the prediction results. In order to realize this, we propose to update our ontology with three axioms instead of one. The axioms should define the concepts of UNKNOWMG, UNKNOWNSA and UNKNOWNCA respectively, where UNKNOWNSA is defined in terms of UNKNOWNMG and UNKNOWNCA is defined in terms of UNKNOWNSA. Obviously, a major challenge for the holistic approach remains in determining the appropriate weights. Unlike the sequential one, they are much less intuitive to define since they contribute to the whole network rather than from one level to the next. One solution to this problem is to automatically estimate the weights from the data.

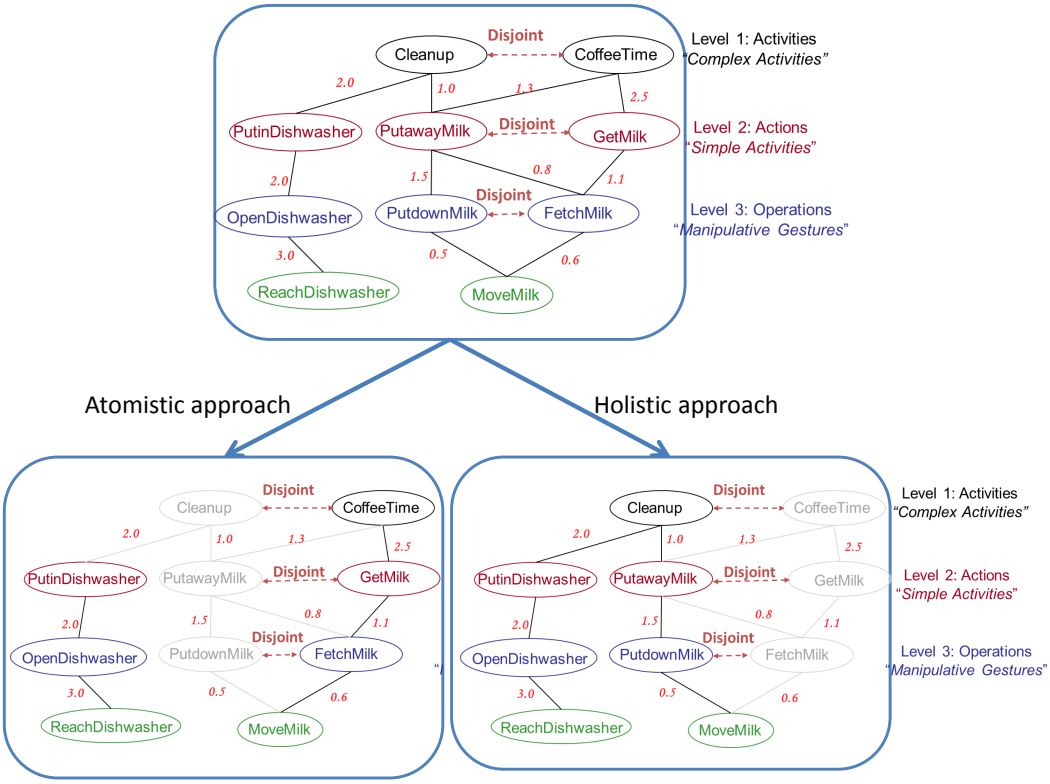


Figure 3.8: Different outputs of the holistic approach (right versus the atomistic approach (left), adopted in our recognition framework. The light nodes are inferred to be false and the dark ones are inferred as true. Allowing the system to reason with the different levels of activity granularity at the same time leads to choosing the paths that yield the highest *sum of weights* rather than sequentially choosing the ones with the highest weights on each level separately.

4
Conclusion

In this part we have presented a hybrid ontology-based framework to represent and recognize human activities from wearable and environmental sensor data. Based on highly expressive log-linear description logic [NNS11], the system unites both symbolic and probabilistic reasoning. This allows to model the complex relational structure as well as the inherent uncertainty underlying human activities and sensor data. Log-linear description logic leverages the same principles of Markov logic and those of ontology reasoning in a unified declarative and intuitive framework. After providing the theoretical background of log-linear description logic and explaining its application to the representation and recognition of human activities, we have drawn the advantages of this approach compared to existing ontology-based approaches to sensor-based activity recognition.

Unlike the majority of related works, it supports the inherent uncertain nature of human activities without sacrificing the advantages of ontological modeling and reasoning. These advantages include consistency checking, the ability of integrating rich background knowledge and the simultaneous recognition of coarse and fine-grained activities. The use of a standard description formalism enhances the portability and re-usability of the proposed system, and supports the representation of heterogeneous and uncertain context data.

Based on principles from the activity theory [KN12], we have focused on addressing the challenge of representing and recognizing human activities at three levels of granularity, while preserving the intuitiveness and flexibility of the modeling task. Complying with real-life scenarios, the proposed framework covers not only sequential activities but also concurrent ones. It is also viable for addressing state of the art challenges including user independent and real time activity recognition.

4.1 Summary

Throughout this part, we have answered the last three research questions defined in our problem statement. For **Question II.1**, we have proposed a log-linear ontology modeling human activities at three levels of granularity. The hierarchical structure is adopted from the well established activity theory [KN12] explained in the Preliminaries Chapter. It includes operations, actions and activities. We have espoused the Opportunity framework [KHF+11] to create an ontology containing a total of around 200 classes. Whereas the definitions of activities at the lowest and highest levels of granularity do not include temporal information, modeling those at the second level includes the temporal order of their components. The ontology axioms are either certain or uncertain. Uncertain axioms are annotated with weights that contribute to their probability distribution. Given the proposed ontology, we have applied a log-linear DL reasoner to recognize the three levels of activity granularity from real-life sensor data. The rationale of the recognition technique consists in the assumption that there is an *unknown activity* corresponding to a given sensor input. Using MAP inference, the *unknown activity* concept is added to the ontology then the activity concept which contains as many perceived properties as possible is determined to be the predicted *unknown activity* corresponding to the observed situation. This contribution covers the research **Question II.2**. Finally, in order to answer research **Question II.3**, we have evaluated our prototype against two different baselines. The main idea of the baselines consisted in using two weight-free variants of the proposed multi-level ontology in order to highlight the advantage of uncertainty support in our framework. The dataset [1] was collected using both wearable and environmental sensors which indicate the user's gestures as well as the objects in use. The validation scenario is compatible with real life settings since it includes concurrent and interleaved activities. Based on the obtained results we depict the main limitations of our approach in the next section.

4.2 Discussion

Whereas many of the features underpinning our log-linear description logic-based approach are well suited for formally capturing and reasoning over rich and uncertain semantic interconnection among entities, some issues remain unsolved.

In particular, our system does not efficiently support temporal reasoning, which is a key requirement for human activity recognition. There are some alternatives to address this limitation though. One of these consists in a Markov logic-based approach for Temporal reasoning using RDF(S). The formalism uses Allens interval algebra [All83] to express temporal relations between facts and reasons about these by transforming the RDFS statements and constraints to Markov Logic. The idea has been introduced by Huber, Meilicke and Stuckenschmidt [HMS14]. In

[1] http://www.opportunity-project.eu

a collaboration with the latter, it has been applied it to our multi-level activity framework. Generally, the obtained results have reached better recall but lower precision. Even though the overall performance has improved slightly, the approach still faces some shortcomings such as lower expressiveness and incapability to infer knew knowledge from the input intervals. A comparable alternative to addressing temporal reasoning in our framework is the use of temporal DL. For example, temporal DL is used for reasoning about actions and plans in [AF98]. Compared to our framework, actions would represent operations (like *Atomic Gestures* and *Manipulative Gestures* and plans correspond to activities, which are defined as temporally-constrained sequences of actions. Nonetheless, this upgrade comes at the cost of no support of uncertainty. A loosely-coupled technique where time is treated as *concrete domain* [LM07] can be also employed. Concretely, the ontology instances can be related to values of the temporal domain by functional properties. Reasoning about relationships among the time intervals corresponding to the different activities can be done using an external reasoner.

Additionally to the urge of supporting temporal reasoning in our system, alleviating the knowledge engineering task is highly desired. To do so, we propose to learn not only the axiom's weights from data but also the axioms themselves. Using ontology learning approaches [LV14] might be a promising direction towards this goal.

Appendices

Appendix:
Multi-level Activities Ontology

Classes

AGBiteBread

AGBiteBread $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Bite $\sqcap \exists$ hasObject Bread))
AGBiteBread $\sqsubseteq$ AtomicGesture

AGCleanTable

AGCleanTable $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Clean $\sqcap \exists$ hasObject Table))
AGCleanTable $\sqsubseteq$ AtomicGesture

AGCloseDishwasher

AGCloseDishwasher $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Close $\sqcap \exists$ hasObject Dishwasher))
AGCloseDishwasher $\sqsubseteq$ AtomicGesture

AGCloseDoor

AGCloseDoor $\sqsubseteq$ AtomicGesture

AGCloseDoor1

AGCloseDoor1 $\equiv$ AGCloseDoor $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Close $\sqcap \exists$ hasObject Door1))
AGCloseDoor1 $\sqsubseteq$ AGCloseDoor

AGCloseDoor2

AGCloseDoor2 $\equiv$ AGCloseDoor $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Close $\sqcap \exists$ hasObject Door2))
AGCloseDoor2 $\sqsubseteq$ AGCloseDoor

AGCloseDrawer

AGCloseDrawer $\sqsubseteq$ AtomicGesture

AGCloseDrawer1

AGCloseDrawer1 $\equiv$ AGCloseDrawer $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Close $\sqcap \exists$ hasObject Drawer1))
AGCloseDrawer1 $\sqsubseteq$ AGCloseDrawer

AGCloseDrawer2

AGCloseDrawer2 $\equiv$ AGCloseDrawer $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Close $\sqcap \exists$ hasObject Drawer2))
AGCloseDrawer2 $\sqsubseteq$ AGCloseDrawer

AGCloseDrawer3

AGCloseDrawer3 $\equiv$ AGCloseDrawer $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Close $\sqcap \exists$ hasObject Drawer3))
AGCloseDrawer3 $\sqsubseteq$ AGCloseDrawer

AGCloseFridge

AGCloseFridge $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Close $\sqcap \exists$ hasObject Fridge))
AGCloseFridge $\sqsubseteq$ AtomicGesture

AGCutBread

AGCutBread $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Cut $\sqcap \exists$ hasObject Bread))
AGCutBread $\sqsubseteq$ AtomicGesture

AGCutSalami

AGCutSalami $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Cut $\sqcap \exists$ hasObject Salami))
AGCutSalami $\sqsubseteq$ AtomicGesture

AGLockDoor

AGLockDoor $\sqsubseteq$ AtomicGesture

AGLockDoor1

AGLockDoor1 $\equiv$ AGLockDoor $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Lock $\sqcap \exists$ hasObject Door1))
AGLockDoor1 $\sqsubseteq$ AGLockDoor

AGLockDoor2

AGLockDoor2 $\equiv$ AGLockDoor $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Lock $\sqcap \exists$ hasObject Door2))
AGLockDoor2 $\sqsubseteq$ AGLockDoor

AGMoveBottle

AGMoveBottle $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Move $\sqcap \exists$ hasObject Bottle))
AGMoveBottle $\sqsubseteq$ AtomicGesture

AGMoveBread

AGMoveBread $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Move $\sqcap \exists$ hasObject Bread))
AGMoveBread $\sqsubseteq$ AtomicGesture

AGMoveChair

AGMoveChair $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Move $\sqcap \exists$ hasObject Chair))
AGMoveChair $\sqsubseteq$ AtomicGesture

AGMoveCheese

AGMoveCheese $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Move $\sqcap \exists$ hasObject Cheese))
AGMoveCheese $\sqsubseteq$ AtomicGesture

AGMoveCup

AGMoveCup $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Move $\sqcap \exists$ hasObject Cup))
AGMoveCup $\sqsubseteq$ AtomicGesture

AGMoveGlass

AGMoveGlass $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Move $\sqcap \exists$ hasObject Glass))

AGMoveGlass $\sqsubseteq$ AtomicGesture

AGMoveKnife

AGMoveKnife $\sqsubseteq$ AtomicGesture

AGMoveKnifeCheese

AGMoveKnifeCheese $\equiv$ AGMoveKnife $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Move $\sqcap \exists$ hasObject KnifeCheese))
AGMoveKnifeCheese $\sqsubseteq$ AGMoveKnife

AGMoveKnifeSalami

AGMoveKnifeSalami $\equiv$ AGMoveKnife $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Move $\sqcap \exists$ hasObject KnifeSalami))
AGMoveKnifeSalami $\sqsubseteq$ AGMoveKnife

AGMoveLazychair

AGMoveLazychair $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Move $\sqcap \exists$ hasObject Lazychair))
AGMoveLazychair $\sqsubseteq$ AtomicGesture

AGMoveMilk

AGMoveMilk $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Move $\sqcap \exists$ hasObject Milk))
AGMoveMilk $\sqsubseteq$ AtomicGesture

AGMovePlate

AGMovePlate $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Move $\sqcap \exists$ hasObject Plate))
AGMovePlate $\sqsubseteq$ AtomicGesture

AGMoveSalami

AGMoveSalami $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Move $\sqcap \exists$ hasObject Salami))
AGMoveSalami $\sqsubseteq$ AtomicGesture

AGMoveSpoon

AGMoveSpoon $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Move $\sqcap \exists$ hasObject Spoon))

AGMoveSpoon $\sqsubseteq$ AtomicGesture

AGMoveSugar

AGMoveSugar $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Move $\sqcap \exists$ hasObject Sugar))
AGMoveSugar $\sqsubseteq$ AtomicGesture

AGOpenDishwasher

AGOpenDishwasher $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Open $\sqcap \exists$ hasObject Dishwasher))
AGOpenDishwasher $\sqsubseteq$ AtomicGesture

AGOpenDoor

AGOpenDoor $\sqsubseteq$ AtomicGesture

AGOpenDoor1

AGOpenDoor1 $\equiv$ AGOpenDoor $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Open $\sqcap \exists$ hasObject Door1))
AGOpenDoor1 $\sqsubseteq$ AGOpenDoor

AGOpenDoor2

AGOpenDoor2 $\equiv$ AGOpenDoor $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Open $\sqcap \exists$ hasObject Door2))
AGOpenDoor2 $\sqsubseteq$ AGOpenDoor

AGOpenDrawer

AGOpenDrawer $\sqsubseteq$ AtomicGesture

AGOpenDrawer1

AGOpenDrawer1 $\equiv$ AGOpenDrawer $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Open $\sqcap \exists$ hasObject Drawer1))
AGOpenDrawer1 $\sqsubseteq$ AGOpenDrawer

AGOpenDrawer2

AGOpenDrawer2 $\equiv$ AGOpenDrawer $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Open $\sqcap \exists$ hasObject Drawer2))
AGOpenDrawer2 $\sqsubseteq$ AGOpenDrawer

AGOpenDrawer3

AGOpenDrawer3 $\equiv$ AGOpenDrawer $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Open $\sqcap \exists$ hasObject Drawer3))
AGOpenDrawer3 $\sqsubseteq$ AGOpenDrawer

AGOpenFridge

AGOpenFridge $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Open $\sqcap \exists$ hasObject Fridge))
AGOpenFridge $\sqsubseteq$ AtomicGesture

AGReachBottle

AGReachBottle $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Reach $\sqcap \exists$ hasObject Bottle))
AGReachBottle $\sqsubseteq$ AtomicGesture

AGReachBread

AGReachBread $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Reach $\sqcap \exists$ hasObject Bread))
AGReachBread $\sqsubseteq$ AtomicGesture

AGReachChair

AGReachChair $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Reach $\sqcap \exists$ hasObject Chair))
AGReachChair $\sqsubseteq$ AtomicGesture

AGReachCheese

AGReachCheese $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Reach $\sqcap \exists$ hasObject Cheese))
AGReachCheese $\sqsubseteq$ AtomicGesture

AGReachCup

AGReachCup $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Reach $\sqcap \exists$ hasObject Cup))
AGReachCup $\sqsubseteq$ AtomicGesture

AGReachDishwasher

AGReachDishwasher $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Reach $\sqcap \exists$ hasObject Dishwasher))

AGReachDishwasher $\sqsubseteq$ AtomicGesture

AGReachDoor

AGReachDoor $\sqsubseteq$ AtomicGesture

AGReachDoor1

AGReachDoor1 $\equiv$ AGReachDoor $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Reach $\sqcap \exists$ hasObject Door1))
AGReachDoor1 $\sqsubseteq$ AGReachDoor

AGReachDoor2

AGReachDoor2 $\equiv$ AGReachDoor $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Reach $\sqcap \exists$ hasObject Door2))
AGReachDoor2 $\sqsubseteq$ AGReachDoor

AGReachDrawer

AGReachDrawer $\sqsubseteq$ AtomicGesture

AGReachDrawer1

AGReachDrawer1 $\equiv$ AGReachDrawer $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Reach $\sqcap \exists$ hasObject Drawer1))
AGReachDrawer1 $\sqsubseteq$ AGReachDrawer

AGReachDrawer2

AGReachDrawer2 $\equiv$ AGReachDrawer $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Reach $\sqcap \exists$ hasObject Drawer2))
AGReachDrawer2 $\sqsubseteq$ AGReachDrawer

AGReachDrawer3

AGReachDrawer3 $\equiv$ AGReachDrawer $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Reach $\sqcap \exists$ hasObject Drawer3))
AGReachDrawer3 $\sqsubseteq$ AGReachDrawer

AGReachFridge

AGReachFridge $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Reach $\sqcap \exists$ hasObject Fridge))
AGReachFridge $\sqsubseteq$ AtomicGesture

AGReachGlass

AGReachGlass $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Reach $\sqcap \exists$ hasObject Glass))
AGReachGlass $\sqsubseteq$ AtomicGesture

AGReachKnife

AGReachKnife $\sqsubseteq$ AtomicGesture

AGReachKnifeCheese

AGReachKnifeCheese $\equiv$ AGReachKnife $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Reach $\sqcap \exists$ hasObject KnifeCheese))
AGReachKnifeCheese $\sqsubseteq$ AGReachKnife

AGReachKnifeSalami

AGReachKnifeSalami $\equiv$ AGReachKnife $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Reach $\sqcap \exists$ hasObject KnifeSalami))
AGReachKnifeSalami $\sqsubseteq$ AGReachKnife

AGReachLazychair

AGReachLazychair $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Reach $\sqcap \exists$ hasObject Lazychair))
AGReachLazychair $\sqsubseteq$ AtomicGesture

AGReachMilk

AGReachMilk $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Reach $\sqcap \exists$ hasObject Milk))
AGReachMilk $\sqsubseteq$ AtomicGesture

AGReachPlate

AGReachPlate $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Reach $\sqcap \exists$ hasObject Plate))
AGReachPlate $\sqsubseteq$ AtomicGesture

AGReachSalami

AGReachSalami $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Reach $\sqcap \exists$ hasObject Salami))
AGReachSalami $\sqsubseteq$ AtomicGesture

AGReachSpoon

AGReachSpoon $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Reach $\sqcap \exists$ hasObject Spoon))
AGReachSpoon $\sqsubseteq$ AtomicGesture

AGReachSugar

AGReachSugar $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Reach $\sqcap \exists$ hasObject Sugar))
AGReachSugar $\sqsubseteq$ AtomicGesture

AGReachSwitch

AGReachSwitch $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Reach $\sqcap \exists$ hasObject Switch))
AGReachSwitch $\sqsubseteq$ AtomicGesture

AGReachTable

AGReachTable $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Reach $\sqcap \exists$ hasObject Table))
AGReachTable $\sqsubseteq$ AtomicGesture

AGReleaseBottle

AGReleaseBottle $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Release $\sqcap \exists$ hasObject Bottle))
AGReleaseBottle $\sqsubseteq$ AtomicGesture

AGReleaseBread

AGReleaseBread $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Release $\sqcap \exists$ hasObject Bread))
AGReleaseBread $\sqsubseteq$ AtomicGesture

AGReleaseChair

AGReleaseChair $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Release $\sqcap \exists$ hasObject Chair))
AGReleaseChair $\sqsubseteq$ AtomicGesture

AGReleaseCheese

AGReleaseCheese $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Release $\sqcap \exists$ hasObject Cheese))

AGReleaseCheese $\sqsubseteq$ AtomicGesture

AGReleaseCup

AGReleaseCup $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Release $\sqcap \exists$ hasObject Cup))
AGReleaseCup $\sqsubseteq$ AtomicGesture

AGReleaseDishwasher

AGReleaseDishwasher $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Release $\sqcap \exists$ hasObject Dishwasher))
AGReleaseDishwasher $\sqsubseteq$ AtomicGesture

AGReleaseDoor

AGReleaseDoor $\sqsubseteq$ AtomicGesture

AGReleaseDoor1

AGReleaseDoor1 $\equiv$ AGReleaseDoor $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Release $\sqcap \exists$ hasObject Door1))
AGReleaseDoor1 $\sqsubseteq$ AGReleaseDoor

AGReleaseDoor2

AGReleaseDoor2 $\equiv$ AGReleaseDoor $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Release $\sqcap \exists$ hasObject Door2))
AGReleaseDoor2 $\sqsubseteq$ AGReleaseDoor

AGReleaseDrawer

AGReleaseDrawer $\sqsubseteq$ AtomicGesture

AGReleaseDrawer1

AGReleaseDrawer1 $\equiv$ AGReleaseDrawer $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Release $\sqcap \exists$ hasObject Drawer1))
AGReleaseDrawer1 $\sqsubseteq$ AGReleaseDrawer

AGReleaseDrawer2

AGReleaseDrawer2 $\equiv$ AGReleaseDrawer $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Release $\sqcap \exists$ hasObject Drawer2))
AGReleaseDrawer2 $\sqsubseteq$ AGReleaseDrawer

AGReleaseDrawer3

AGReleaseDrawer3 $\equiv$ AGReleaseDrawer $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Release $\sqcap \exists$ hasObject Drawer3))
AGReleaseDrawer3 $\sqsubseteq$ AGReleaseDrawer

AGReleaseFridge

AGReleaseFridge $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Release $\sqcap \exists$ hasObject Fridge))
AGReleaseFridge $\sqsubseteq$ AtomicGesture

AGReleaseGlass

AGReleaseGlass $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Release $\sqcap \exists$ hasObject Glass))
AGReleaseGlass $\sqsubseteq$ AtomicGesture

AGReleaseKnife

AGReleaseKnife $\sqsubseteq$ AtomicGesture

AGReleaseKnifeCheese

AGReleaseKnifeCheese $\equiv$ AGReleaseKnife $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Release $\sqcap \exists$ hasObject KnifeCheese))
AGReleaseKnifeCheese $\sqsubseteq$ AGReleaseKnife

AGReleaseKnifeSalami

AGReleaseKnifeSalami $\equiv$ AGReleaseKnife $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Release $\sqcap \exists$ hasObject KnifeSalami))
AGReleaseKnifeSalami $\sqsubseteq$ AGReleaseKnife

AGReleaseLazychair

AGReleaseLazychair $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Release $\sqcap \exists$ hasObject Lazychair))
AGReleaseLazychair $\sqsubseteq$ AtomicGesture

AGReleaseMilk

AGReleaseMilk $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Release $\sqcap \exists$ hasObject Milk))
AGReleaseMilk $\sqsubseteq$ AtomicGesture

AGReleasePlate

AGReleasePlate $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Release $\sqcap \exists$ hasObject Plate))
AGReleasePlate $\sqsubseteq$ AtomicGesture

AGReleaseSalami

AGReleaseSalami $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Release $\sqcap \exists$ hasObject Salami))
AGReleaseSalami $\sqsubseteq$ AtomicGesture

AGReleaseSpoon

AGReleaseSpoon $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Release $\sqcap \exists$ hasObject Spoon))
AGReleaseSpoon $\sqsubseteq$ AtomicGesture

AGReleaseSugar

AGReleaseSugar $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Release $\sqcap \exists$ hasObject Sugar))
AGReleaseSugar $\sqsubseteq$ AtomicGesture

AGReleaseSwitch

AGReleaseSwitch $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Release $\sqcap \exists$ hasObject Switch))
AGReleaseSwitch $\sqsubseteq$ AtomicGesture

AGReleaseTable

AGReleaseTable $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Release $\sqcap \exists$ hasObject Table))
AGReleaseTable $\sqsubseteq$ AtomicGesture

AGSipCup

AGSipCup $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Sip $\sqcap \exists$ hasObject Cup))
AGSipCup $\sqsubseteq$ AtomicGesture

AGSipGlass

AGSipGlass $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Sip $\sqcap \exists$ hasObject Glass))

AGSipGlass $\sqsubseteq$ AtomicGesture

AGSpreadCheese

AGSpreadCheese $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Spread $\sqcap \exists$ hasObject Cheese))
AGSpreadCheese $\sqsubseteq$ AtomicGesture

AGStirSpoon

AGStirSpoon $\equiv$ AtomicGesture $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Stir $\sqcap \exists$ hasObject Spoon))
AGStirSpoon $\sqsubseteq$ AtomicGesture

AGUnlockDoor

AGUnlockDoor $\sqsubseteq$ AtomicGesture

AGUnlockDoor1

AGUnlockDoor1 $\equiv$ AGUnlockDoor $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Unlock $\sqcap \exists$ hasObject Door1))
AGUnlockDoor1 $\sqsubseteq$ AGUnlockDoor

AGUnlockDoor2

AGUnlockDoor2 $\equiv$ AGUnlockDoor $\sqcap \exists$ hasAGActor (Person $\sqcap \exists$ hasArm (Arm $\sqcap \exists$ has-Function Unlock $\sqcap \exists$ hasObject Door2))
AGUnlockDoor2 $\sqsubseteq$ AGUnlockDoor

Arm

Arm $\sqsubseteq$ Thing

AtomicGesture

AtomicGesture $\sqsubseteq$ Thing

Bite

Bite $\sqsubseteq$ Function

Bottle

Bottle $\sqsubseteq$ Object

Bread

Bread $\sqsubseteq$ Object

CACleanup

CACleanup $\equiv$ ComplexActivity $\sqcap \exists$ hasCAActor (Person $\sqcap \exists$ hasSimpleActivity SAPutawayBread)
CACleanup $\equiv$ ComplexActivity $\sqcap \exists$ hasCAActor (Person $\sqcap \exists$ hasManipulativeGesture MGCleanTable)
CACleanup $\equiv$ ComplexActivity $\sqcap \exists$ hasCAActor (Person $\sqcap \exists$ hasSimpleActivity SAPutinDishwasher)
CACleanup $\equiv$ ComplexActivity $\sqcap \exists$ hasCAActor (Person $\sqcap \exists$ hasSimpleActivity SAPutawayMilk)
CACleanup $\equiv$ ComplexActivity $\sqcap \exists$ hasCAActor (Person $\sqcap \exists$ hasSimpleActivity SAPutawayBottle)
CACleanup $\equiv$ ComplexActivity $\sqcap \exists$ hasCAActor (Person $\sqcap \exists$ hasSimpleActivity SAPutawayCheese)
CACleanup $\equiv$ ComplexActivity $\sqcap \exists$ hasCAActor (Person $\sqcap \exists$ hasSimpleActivity SAPutawaySalami)
CACleanup $\sqsubseteq$ ComplexActivity
CACleanup $\sqsubseteq \neg$ CACoffeeTime
CACleanup $\sqsubseteq \neg$ CASandwichTime
CACleanup $\sqsubseteq \neg$ CARelaxing

CACoffeeTime

CACoffeeTime $\equiv$ ComplexActivity $\sqcap \exists$ hasCAActor (Person $\sqcap \exists$ hasSimpleActivity SAPutSugar)
CACoffeeTime $\equiv$ ComplexActivity $\sqcap \exists$ hasCAActor (Person $\sqcap \exists$ hasSimpleActivity SAGetMilk)
CACoffeeTime $\equiv$ ComplexActivity $\sqcap \exists$ hasCAActor (Person $\sqcap \exists$ hasSimpleActivity SADrinkfromCup)
CACoffeeTime $\sqsubseteq$ ComplexActivity
CACoffeeTime $\sqsubseteq \neg$ CARelaxing
CACoffeeTime $\sqsubseteq \neg$ CASandwichTime
CACoffeeTime $\sqsubseteq \neg$ CACleanup

CAIdle

CAIdle $\sqsubseteq$ ComplexActivity

CARelaxing

CARelaxing $\equiv$ ComplexActivity $\sqcap$ $\exists$ hasCAActor (Person $\sqcap$ $\exists$ hasSimpleActivity SALieonLazychair)
CARelaxing $\sqsubseteq$ ComplexActivity
CARelaxing $\sqsubseteq$ $\neg$ CASandwichTime
CARelaxing $\sqsubseteq$ $\neg$ CACoffeeTime
CARelaxing $\sqsubseteq$ $\neg$ CACleanup

CASandwichTime

CASandwichTime $\equiv$ ComplexActivity $\sqcap$ $\exists$ hasCAActor (Person $\sqcap$ $\exists$ hasSimpleActivity SAPrepareSalami)
CASandwichTime $\equiv$ ComplexActivity $\sqcap$ $\exists$ hasCAActor (Person $\sqcap$ $\exists$ hasSimpleActivity SAGetPlate)
CASandwichTime $\equiv$ ComplexActivity $\sqcap$ $\exists$ hasCAActor (Person $\sqcap$ $\exists$ hasSimpleActivity SAGetCheese)
CASandwichTime $\equiv$ ComplexActivity $\sqcap$ $\exists$ hasCAActor (Person $\sqcap$ $\exists$ hasSimpleActivity SAGetBread)
CASandwichTime $\equiv$ ComplexActivity $\sqcap$ $\exists$ hasCAActor (Person $\sqcap$ $\exists$ hasSimpleActivity SAGetBottle)
CASandwichTime $\equiv$ ComplexActivity $\sqcap$ $\exists$ hasCAActor (Person $\sqcap$ $\exists$ hasSimpleActivity SAEatBread)
CASandwichTime $\equiv$ ComplexActivity $\sqcap$ $\exists$ hasCAActor (Person $\sqcap$ $\exists$ hasSimpleActivity SAGetKnifeCheese)
CASandwichTime $\equiv$ ComplexActivity $\sqcap$ $\exists$ hasCAActor (Person $\sqcap$ $\exists$ hasSimpleActivity SAPrepareCheeseSandwich)
CASandwichTime $\equiv$ ComplexActivity $\sqcap$ $\exists$ hasCAActor (Person $\sqcap$ $\exists$ hasAtomicGesture AGCutBread)
CASandwichTime $\equiv$ ComplexActivity $\sqcap$ $\exists$ hasCAActor (Person $\sqcap$ $\exists$ hasSimpleActivity SADrinkfromGlass)
CASandwichTime $\equiv$ ComplexActivity $\sqcap$ $\exists$ hasCAActor (Person $\sqcap$ $\exists$ hasSimpleActivity SAGetKnifeSalami)
CASandwichTime $\equiv$ ComplexActivity $\sqcap$ $\exists$ hasCAActor (Person $\sqcap$ $\exists$ hasSimpleActivity SAGetSalami)
CASandwichTime $\sqsubseteq$ ComplexActivity
CASandwichTime $\sqsubseteq$ $\neg$ CARelaxing
CASandwichTime $\sqsubseteq$ $\neg$ CACleanup
CASandwichTime $\sqsubseteq$ $\neg$ CACoffeeTime

Chair

Chair $\sqsubseteq$ Object

Cheese

Cheese $\sqsubseteq$ Object

Clean

Clean $\sqsubseteq$ Function

Close

Close $\sqsubseteq$ Function

ComplexActivity

Cup

Cup $\sqsubseteq$ WashableObject

Cut

Cut $\sqsubseteq$ Function

Dishwasher

Dishwasher $\sqsubseteq$ Object

Door

Door $\sqsubseteq$ Object

Door1

Door1 $\sqsubseteq$ Door

Door2

Door2 $\sqsubseteq$ Door

Drawer

Drawer $\sqsubseteq$ Object

Drawer1

Drawer1 $\sqsubseteq$ Drawer

Drawer2

Drawer2 $\sqsubseteq$ Drawer

Drawer3

Drawer3 $\sqsubseteq$ Drawer

Fridge

Fridge $\sqsubseteq$ Object

Function

Function $\sqsubseteq$ Thing

Glass

Glass $\sqsubseteq$ WashableObject

Knife

Knife $\sqsubseteq$ WashableObject

KnifeCheese

KnifeCheese $\sqsubseteq$ Knife

KnifeSalami

KnifeSalami $\sqsubseteq$ Knife

Lazychair

Lazychair $\sqsubseteq$ Object

LeftArm

LeftArm $\sqsubseteq$ Arm

Lie

Lie $\sqsubseteq$ Locomotion

Lock

Lock $\sqsubseteq$ Function

Locomotion

Locomotion $\sqsubseteq$ Thing

MGCleanTable

MGCleanTable $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGCleanTable $\sqcap \exists$ hasAtomicGesture AGReleaseTable)
MGCleanTable $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGCleanTable $\sqcap \exists$ hasAtomicGesture AGReachTable)
MGCleanTable $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGCleanTable)
MGCleanTable $\sqsubseteq$ ManipulativeGesture

MGCloseDishwasher

MGCloseDishwasher $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGCloseDishwasher)
MGCloseDishwasher $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReachDishwasher)
MGCloseDishwasher $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReleaseDishwasher)
MGCloseDishwasher $\sqsubseteq$ ManipulativeGesture
MGCloseDishwasher $\sqsubseteq \neg$ MGOpenDishwasher

MGCloseDoor

MGCloseDoor $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGLockDoor1)
MGCloseDoor $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReachDoor2)
MGCloseDoor $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReleaseDoor2)
MGCloseDoor $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGCloseDoor1)
MGCloseDoor $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReleaseDoor1)
MGCloseDoor $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGCloseDoor2)
MGCloseDoor $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGLockDoor2)
MGCloseDoor $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReachDoor1)
MGCloseDoor $\sqsubseteq$ ManipulativeGesture
MGCloseDoor $\sqsubseteq \neg$ MGOpenDoor

MGCloseDrawer

MGCloseDrawer $\sqsubseteq$ ManipulativeGesture

MGCloseDrawer1

MGCloseDrawer1 $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReleaseDrawer1)
MGCloseDrawer1 $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGCloseDrawer1)
MGCloseDrawer1 $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReachDrawer1)
MGCloseDrawer1 $\sqsubseteq$ MGCloseDrawer
MGCloseDrawer1 $\sqsubseteq \neg$ MGOpenDrawer1

MGCloseDrawer2

MGCloseDrawer2 $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReleaseDrawer2)
MGCloseDrawer2 $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReachDrawer2)
MGCloseDrawer2 $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGCloseDrawer2)
MGCloseDrawer2 $\sqsubseteq$ MGCloseDrawer
MGCloseDrawer2 $\sqsubseteq \neg$ MGOpenDrawer2

MGCloseDrawer3

MGCloseDrawer3 $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGCloseDrawer3)
MGCloseDrawer3 $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReachDrawer3)
MGCloseDrawer3 $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReleaseDrawer3)
MGCloseDrawer3 $\sqsubseteq$ MGCloseDrawer
MGCloseDrawer3 $\sqsubseteq \neg$ MGOpenDrawer3

MGCloseFridge

MGCloseFridge $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReleaseFridge)
MGCloseFridge $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReachFridge)
MGCloseFridge $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGCloseFridge)

MGCloseFridge $\sqsubseteq$ ManipulativeGesture
MGCloseFridge $\sqsubseteq \neg$ MGOpenFridge

MGFetchBottle

MGFetchBottle $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGMoveBottle)
MGFetchBottle $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReachBottle)
MGFetchBottle $\sqsubseteq$ ManipulativeGesture
MGFetchBottle $\sqsubseteq \neg$ MGPutdownBottle

MGFetchBread

MGFetchBread $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGMoveBread)
MGFetchBread $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReachBread)
MGFetchBread $\sqsubseteq$ ManipulativeGesture
MGFetchBread $\sqsubseteq \neg$ MGPutdownBread

MGFetchCheese

MGFetchCheese $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGMoveCheese)
MGFetchCheese $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReachCheese)
MGFetchCheese $\sqsubseteq$ ManipulativeGesture
MGFetchCheese $\sqsubseteq \neg$ MGPutdownCheese

MGFetchCup

MGFetchCup $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReachCup)
MGFetchCup $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGMoveCup)
MGFetchCup $\sqsubseteq$ MGFetchWashableObject
MGFetchCup $\sqsubseteq \neg$ MGPutdownCup

MGFetchGlass

MGFetchGlass $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReachGlass)
MGFetchGlass $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGMoveGlass)

MGFetchGlass $\sqsubseteq$ MGFetchWashableObject
MGFetchGlass $\sqsubseteq \neg$ MGPutdownGlass

MGFetchKnifeCheese

MGFetchKnifeCheese $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReachKnifeCheese)
MGFetchKnifeCheese $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGMoveKnifeCheese)
MGFetchKnifeCheese $\sqsubseteq$ MGFetchWashableObject
MGFetchKnifeCheese $\sqsubseteq \neg$ MGPutdownKnifeCheese

MGFetchKnifeSalami

MGFetchKnifeSalami $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReachKnifeSalami)
MGFetchKnifeSalami $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGMoveKnifeSalami)
MGFetchKnifeSalami $\sqsubseteq$ MGFetchWashableObject
MGFetchKnifeSalami $\sqsubseteq \neg$ MGPutdownKnifeSalami

MGFetchMilk

MGFetchMilk $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReachMilk)
MGFetchMilk $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGMoveMilk)
MGFetchMilk $\sqsubseteq$ ManipulativeGesture
MGFetchMilk $\sqsubseteq \neg$ MGPutdownMilk

MGFetchPlate

MGFetchPlate $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReachPlate)
MGFetchPlate $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGMovePlate)
MGFetchPlate $\sqsubseteq$ MGFetchWashableObject
MGFetchPlate $\sqsubseteq \neg$ MGPutdownPlate

MGFetchSalami

MGFetchSalami $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGMoveSalami)
MGFetchSalami $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReachSalami)

MGFetchSalami $\sqsubseteq$ ManipulativeGesture
MGFetchSalami $\sqsubseteq \neg$ MGPutdownSalami

MGFetchSpoon

MGFetchSpoon $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReachSpoon)
MGFetchSpoon $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGMoveSpoon)
MGFetchSpoon $\sqsubseteq$ MGFetchWashableObject
MGFetchSpoon $\sqsubseteq \neg$ MGPutdownSpoon

MGFetchSugar

MGFetchSugar $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGMoveSugar)
MGFetchSugar $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReachSugar)
MGFetchSugar $\sqsubseteq$ ManipulativeGesture
MGFetchSugar $\sqsubseteq \neg$ MGPutdownSugar

MGFetchWashableObject

MGFetchWashableObject $\sqsubseteq$ ManipulativeGesture

MGInteractwithChair

MGInteractwithChair $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGMoveChair)
MGInteractwithChair $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReachChair)
MGInteractwithChair $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReleaseChair)
MGInteractwithChair $\sqsubseteq$ ManipulativeGesture

MGInteractwithLazychair

MGInteractwithLazychair $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGMoveLazychair)
MGInteractwithLazychair $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReachLazychair)
MGInteractwithLazychair $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReleaseLazychair)
MGInteractwithLazychair $\sqsubseteq$ ManipulativeGesture

MGOpenDishwasher

MGOpenDishwasher $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReleaseDishwasher)
MGOpenDishwasher $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGOpenDishwasher)
MGOpenDishwasher $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReachDishwasher)
MGOpenDishwasher $\sqsubseteq$ ManipulativeGesture
MGOpenDishwasher $\sqsubseteq \neg$ MGCloseDishwasher

MGOpenDoor

MGOpenDoor $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGUnlockDoor2)
MGOpenDoor $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReachDoor1)
MGOpenDoor $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReachDoor2)
MGOpenDoor $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReleaseDoor1)
MGOpenDoor $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReleaseDoor2)
MGOpenDoor $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGOpenDoor1)
MGOpenDoor $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGOpenDoor2)
MGOpenDoor $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGUnlockDoor1)
MGOpenDoor $\sqsubseteq$ ManipulativeGesture
MGOpenDoor $\sqsubseteq \neg$ MGCloseDoor

MGOpenDrawer

MGOpenDrawer $\sqsubseteq$ ManipulativeGesture

MGOpenDrawer1

MGOpenDrawer1 $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReachDrawer1)
MGOpenDrawer1 $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGOpenDrawer1)
MGOpenDrawer1 $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReleaseDrawer1)
MGOpenDrawer1 $\sqsubseteq$ MGOpenDrawer

MGOpenDrawer1 $\sqsubseteq \neg$ MGCloseDrawer1

MGOpenDrawer2

MGOpenDrawer2 $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReleaseDrawer2)
MGOpenDrawer2 $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGOpenDrawer2)
MGOpenDrawer2 $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReachDrawer2)
MGOpenDrawer2 $\sqsubseteq$ MGOpenDrawer
MGOpenDrawer2 $\sqsubseteq \neg$ MGCloseDrawer2

MGOpenDrawer3

MGOpenDrawer3 $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReachDrawer3)
MGOpenDrawer3 $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGOpenDrawer3)
MGOpenDrawer3 $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReleaseDrawer3)
MGOpenDrawer3 $\sqsubseteq$ MGOpenDrawer
MGOpenDrawer3 $\sqsubseteq \neg$ MGCloseDrawer3

MGOpenFridge

MGOpenFridge $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReleaseFridge)
MGOpenFridge $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReachFridge)
MGOpenFridge $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGOpenFridge)
MGOpenFridge $\sqsubseteq$ ManipulativeGesture
MGOpenFridge $\sqsubseteq \neg$ MGCloseFridge

MGPutdownBottle

MGPutdownBottle $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGMoveBottle)
MGPutdownBottle $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReleaseBottle)
MGPutdownBottle $\sqsubseteq$ ManipulativeGesture
MGPutdownBottle $\sqsubseteq \neg$ MGFetchBottle

MGPutdownBread

MGPutdownBread $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReleaseBread)
MGPutdownBread $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGMoveBread)
MGPutdownBread $\sqsubseteq$ ManipulativeGesture
MGPutdownBread $\sqsubseteq \neg$ MGFetchBread

MGPutdownCheese

MGPutdownCheese $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGMoveCheese)
MGPutdownCheese $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReleaseCheese)
MGPutdownCheese $\sqsubseteq$ ManipulativeGesture
MGPutdownCheese $\sqsubseteq \neg$ MGFetchCheese

MGPutdownCup

MGPutdownCup $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReleaseCup)
MGPutdownCup $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGMoveCup)
MGPutdownCup $\sqsubseteq$ MGPutdownWashableObject
MGPutdownCup $\sqsubseteq \neg$ MGFetchCup

MGPutdownGlass

MGPutdownGlass $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReleaseGlass)
MGPutdownGlass $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGMoveGlass)
MGPutdownGlass $\sqsubseteq$ MGPutdownWashableObject
MGPutdownGlass $\sqsubseteq \neg$ MGFetchGlass

MGPutdownKnifeCheese

MGPutdownKnifeCheese $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReleaseKnifeCheese)
MGPutdownKnifeCheese $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGMoveKnifeCheese)
MGPutdownKnifeCheese $\sqsubseteq$ MGPutdownWashableObject
MGPutdownKnifeCheese $\sqsubseteq \neg$ MGFetchKnifeCheese

MGPutdownKnifeSalami

MGPutdownKnifeSalami ≡ ManipulativeGesture ⊓ ∃ hasMGActor (Person ⊓ ∃ hasAtomicGesture AGMoveKnifeSalami)
MGPutdownKnifeSalami ≡ ManipulativeGesture ⊓ ∃ hasMGActor (Person ⊓ ∃ hasAtomicGesture AGReleaseKnifeSalami)
MGPutdownKnifeSalami ⊑ MGPutdownWashableObject
MGPutdownKnifeSalami ⊑ ¬ MGFetchKnifeSalami

MGPutdownMilk

MGPutdownMilk ≡ ManipulativeGesture ⊓ ∃ hasMGActor (Person ⊓ ∃ hasAtomicGesture AGMoveMilk)
MGPutdownMilk ≡ ManipulativeGesture ⊓ ∃ hasMGActor (Person ⊓ ∃ hasAtomicGesture AGReleaseMilk)
MGPutdownMilk ⊑ ManipulativeGesture
MGPutdownMilk ⊑ ¬ MGFetchMilk

MGPutdownPlate

MGPutdownPlate ≡ ManipulativeGesture ⊓ ∃ hasMGActor (Person ⊓ ∃ hasAtomicGesture AGReleasePlate)
MGPutdownPlate ≡ ManipulativeGesture ⊓ ∃ hasMGActor (Person ⊓ ∃ hasAtomicGesture AGMovePlate)
MGPutdownPlate ⊑ MGPutdownWashableObject
MGPutdownPlate ⊑ ¬ MGFetchPlate

MGPutdownSalami

MGPutdownSalami ≡ ManipulativeGesture ⊓ ∃ hasMGActor (Person ⊓ ∃ hasAtomicGesture AGMoveSalami)
MGPutdownSalami ≡ ManipulativeGesture ⊓ ∃ hasMGActor (Person ⊓ ∃ hasAtomicGesture AGReleaseSalami)
MGPutdownSalami ⊑ ManipulativeGesture
MGPutdownSalami ⊑ ¬ MGFetchSalami

MGPutdownSpoon

MGPutdownSpoon ≡ ManipulativeGesture ⊓ ∃ hasMGActor (Person ⊓ ∃ hasAtomicGesture AGMoveSpoon)
MGPutdownSpoon ≡ ManipulativeGesture ⊓ ∃ hasMGActor (Person ⊓ ∃ hasAtomicGesture AGReleaseSpoon)
MGPutdownSpoon ⊑ MGPutdownWashableObject
MGPutdownSpoon ⊑ ¬ MGFetchSpoon

MGPutdownSugar

MGPutdownSugar $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGMoveSugar)
MGPutdownSugar $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReleaseSugar)
MGPutdownSugar $\sqsubseteq$ ManipulativeGesture
MGPutdownSugar $\sqsubseteq \neg$ MGFetchSugar

MGPutdownWashableObject

MGPutdownWashableObject $\sqsubseteq$ ManipulativeGesture

MGSwitchSwitch

MGSwitchSwitch $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReleaseSwitch)
MGSwitchSwitch $\equiv$ ManipulativeGesture $\sqcap \exists$ hasMGActor (Person $\sqcap \exists$ hasAtomicGesture AGReachSwitch)
MGSwitchSwitch $\sqsubseteq$ ManipulativeGesture

ManipulativeGesture

ManipulativeGesture $\sqsubseteq$ Thing

Memory

Milk

Milk $\sqsubseteq$ Object

Move

Move $\sqsubseteq$ Function

Object

Object $\sqsubseteq$ Thing

Open

Open $\sqsubseteq$ Function

Person

Person $\sqsubseteq$ Thing

Plate

Plate $\sqsubseteq$ WashableObject

Reach

Reach $\sqsubseteq$ Function

Release

Release $\sqsubseteq$ Function

RightArm

RightArm $\sqsubseteq$ Arm

SADrinkfromCup

SADrinkfromCup $\equiv$ SimpleActivity $\sqcap \exists$ hasMemory (Memory $\sqcap \exists$ hasAG AGSip-Cup $\sqcap$ $hasValue$ hasOrder "1" $\sqcap \exists$ hasMemory (Memory $\sqcap \exists$ hasMG MGFetch-Cup $\sqcap$ $hasValue$ hasOrder "2"
SADrinkfromCup $\equiv$ SimpleActivity $\sqcap \exists$ hasMemory (Memory $\sqcap \exists$ hasAG AGSip-Cup $\sqcap$ $hasValue$ hasOrder "2" $\sqcap \exists$ hasMemory (Memory $\sqcap \exists$ hasMG MGFetch-Cup $\sqcap$ $hasValue$ hasOrder "3" $\sqcap \exists$ hasMemory (Memory $\sqcap \exists$ hasMG MGPutdown-Cup $\sqcap$ $hasValue$ hasOrder "1"
SADrinkfromCup $\sqsubseteq$ SimpleActivity

SADrinkfromGlass

SADrinkfromGlass $\equiv$ SimpleActivity $\sqcap \exists$ hasMemory (Memory $\sqcap \exists$ hasAG AGSip-Glass $\sqcap$ $hasValue$ hasOrder "2" $\sqcap \exists$ hasMemory (Memory $\sqcap \exists$ hasMG MGFetch-Glass $\sqcap$ $hasValue$ hasOrder "3" $\sqcap \exists$ hasMemory (Memory $\sqcap \exists$ hasMG MGPut-downGlass $\sqcap$ $hasValue$ hasOrder "1"
SADrinkfromGlass $\equiv$ SimpleActivity $\sqcap \exists$ hasMemory (Memory $\sqcap \exists$ hasAG AGSip-Glass $\sqcap$ $hasValue$ hasOrder "1" $\sqcap \exists$ hasMemory (Memory $\sqcap \exists$ hasMG MGFetch-Glass $\sqcap$ $hasValue$ hasOrder "2"
SADrinkfromGlass $\sqsubseteq$ SimpleActivity

SAEatBread

SAEatBread $\equiv$ SimpleActivity $\sqcap \exists$ hasMemory (Memory $\sqcap \exists$ hasAG AGBite-Bread $\sqcap$ $hasValue$ hasOrder "1" $\sqcap \exists$ hasMemory (Memory $\sqcap \exists$ hasMG MGFetch-Bread $\sqcap$ $hasValue$ hasOrder "2"
SAEatBread $\equiv$ SimpleActivity $\sqcap \exists$ hasMemory (Memory $\sqcap \exists$ hasAG AGBite-Bread $\sqcap$ $hasValue$ hasOrder "2" $\sqcap \exists$ hasMemory (Memory $\sqcap \exists$ hasMG MGFetch-

Bread ⊓ $hasValue$ hasOrder "3" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGPut-
downBread ⊓ $hasValue$ hasOrder "1"
SAEatBread ⊑ SimpleActivity

SAGetBottle

SAGetBottle ≡ SimpleActivity ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGFetch-
Bottle ⊓$hasValue$ hasOrder "1" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGOpen-
Fridge ⊓ $hasValue$ hasOrder "2"
SAGetBottle ≡ SimpleActivity ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGClose-
Fridge ⊓ $hasValue$ hasOrder "1" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGFetch-
Bottle ⊓ $hasValue$ hasOrder "2" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGOpen-
Fridge ⊓ $hasValue$ hasOrder "3"
SAGetBottle ≡ SimpleActivity ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGClose-
Fridge ⊓ $hasValue$ hasOrder "1" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGFetch-
Bottle ⊓ $hasValue$ hasOrder "2"
SAGetBottle ⊑ SimpleActivity
SAGetBottle ⊑ ¬ SAPutawayBottle

SAGetBread

SAGetBread ≡ SimpleActivity ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGClose-
Drawer3 ⊓ $hasValue$ hasOrder "1" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGFetch-
Bread ⊓ $hasValue$ hasOrder "2"
SAGetBread ≡ SimpleActivity ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGFetch-
Bread ⊓ $hasValue$ hasOrder "1" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGOpen-
Drawer3 ⊓ $hasValue$ hasOrder "2"
SAGetBread ≡ SimpleActivity ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGClose-
Drawer3 ⊓ $hasValue$ hasOrder "1" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGFetch-
Bread ⊓ $hasValue$ hasOrder "2" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGOpen-
Drawer3 ⊓ $hasValue$ hasOrder "3"
SAGetBread ⊑ SimpleActivity
SAGetBread ⊑ ¬ SAPutawayBread

SAGetCheese

SAGetCheese ≡ SimpleActivity ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGFetchCheese ⊓ $hasValue$ ha-
sOrder "1" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGOpenFridge ⊓ $hasValue$ ha-
sOrder "2"
SAGetCheese ≡ SimpleActivity ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGClose-
Fridge ⊓ $hasValue$ hasOrder "1" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGFetchCheese ⊓ $hasValue$ ha-
sOrder "2"
SAGetCheese ≡ SimpleActivity ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGClose-
Fridge ⊓ $hasValue$ hasOrder "1" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGFetchCheese ⊓ $hasValue$ ha-

sOrder "2" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGOpenFridge ⊓ $hasValue$ hasOrder "3"
SAGetCheese ⊑ SimpleActivity
SAGetCheese ⊑ ¬ SAPutawayCheese

SAGetKnifeCheese

SAGetKnifeCheese ≡ SimpleActivity ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MG-CloseDrawer1 ⊓ $hasValue$ hasOrder "1" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MG-FetchKnifeCheese ⊓ $hasValue$ hasOrder "2" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGOpen-Drawer1 ⊓ $hasValue$ hasOrder "3"
SAGetKnifeCheese ≡ SimpleActivity ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MG-FetchKnifeCheese ⊓ $hasValue$ hasOrder "1" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGOpen-Drawer1 ⊓ $hasValue$ hasOrder "2"
SAGetKnifeCheese ≡ SimpleActivity ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MG-CloseDrawer1 ⊓ $hasValue$ hasOrder "1" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MG-FetchKnifeCheese ⊓ $hasValue$ hasOrder "2"
SAGetKnifeCheese ⊑ SimpleActivity

SAGetKnifeSalami

SAGetKnifeSalami ≡ SimpleActivity ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGFetchKnife-Salami ⊓ $hasValue$ hasOrder "1" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGOpen-Drawer1 ⊓ $hasValue$ hasOrder "2"
SAGetKnifeSalami ≡ SimpleActivity ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MG-CloseDrawer1 ⊓ $hasValue$ hasOrder "1" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGFetchKnife-Salami ⊓ $hasValue$ hasOrder "2" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGOpen-Drawer1 ⊓ $hasValue$ hasOrder "3"
SAGetKnifeSalami ≡ SimpleActivity ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MG-CloseDrawer1 ⊓ $hasValue$ hasOrder "1" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGFetchKnife-Salami ⊓ $hasValue$ hasOrder "2"
SAGetKnifeSalami ⊑ SimpleActivity

SAGetMilk

SAGetMilk ≡ SimpleActivity ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGClose-Fridge ⊓ $hasValue$ hasOrder "1" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGFetch-Milk ⊓ $hasValue$ hasOrder "2" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGOpen-Fridge ⊓ $hasValue$ hasOrder "3"
SAGetMilk ≡ SimpleActivity ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGFetch-Milk ⊓ $hasValue$ hasOrder "1" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGOpen-Fridge ⊓ $hasValue$ hasOrder "2"
SAGetMilk ≡ SimpleActivity ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGClose-Fridge ⊓ $hasValue$ hasOrder "1" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGFetch-Milk ⊓ $hasValue$ hasOrder "2"

SAGetMilk $\sqsubseteq$ SimpleActivity
SAGetMilk $\sqsubseteq \neg$ SAPutawayMilk

SAGetPlate

SAGetPlate $\equiv$ SimpleActivity $\sqcap \exists$ hasMemory (Memory $\sqcap \exists$ hasMG MGClose-Drawer2 $\sqcap$ $hasValue$ hasOrder "1" $\sqcap \exists$ hasMemory (Memory $\sqcap \exists$ hasMG MGFetch-Plate $\sqcap$ $hasValue$ hasOrder "2" $\sqcap \exists$ hasMemory (Memory $\sqcap \exists$ hasMG MGOpen-Drawer2 $\sqcap$ $hasValue$ hasOrder "3"
SAGetPlate $\equiv$ SimpleActivity $\sqcap \exists$ hasMemory (Memory $\sqcap \exists$ hasMG MGFetch-Plate $\sqcap$ $hasValue$ hasOrder "1" $\sqcap \exists$ hasMemory (Memory $\sqcap \exists$ hasMG MGOpen-Drawer2 $\sqcap$ $hasValue$ hasOrder "2"
SAGetPlate $\equiv$ SimpleActivity $\sqcap \exists$ hasMemory (Memory $\sqcap \exists$ hasMG MGClose-Drawer2 $\sqcap$ $hasValue$ hasOrder "1" $\sqcap \exists$ hasMemory (Memory $\sqcap \exists$ hasMG MGFetch-Plate $\sqcap$ $hasValue$ hasOrder "2"
SAGetPlate $\sqsubseteq$ SimpleActivity

SAGetSalami

SAGetSalami $\equiv$ SimpleActivity $\sqcap \exists$ hasMemory (Memory $\sqcap \exists$ hasMG MGClose-Fridge $\sqcap$ $hasValue$ hasOrder "1" $\sqcap \exists$ hasMemory (Memory $\sqcap \exists$ hasMG MGFetch-Salami $\sqcap$ $hasValue$ hasOrder "2" $\sqcap \exists$ hasMemory (Memory $\sqcap \exists$ hasMG MGOpen-Fridge $\sqcap$ $hasValue$ hasOrder "3"
SAGetSalami $\equiv$ SimpleActivity $\sqcap \exists$ hasMemory (Memory $\sqcap \exists$ hasMG MGFetch-Salami $\sqcap$ $hasValue$ hasOrder "1" $\sqcap \exists$ hasMemory (Memory $\sqcap \exists$ hasMG MGOpen-Fridge $\sqcap$ $hasValue$ hasOrder "2"
SAGetSalami $\equiv$ SimpleActivity $\sqcap \exists$ hasMemory (Memory $\sqcap \exists$ hasMG MGClose-Fridge $\sqcap$ $hasValue$ hasOrder "1" $\sqcap \exists$ hasMemory (Memory $\sqcap \exists$ hasMG MGFetch-Salami $\sqcap$ $hasValue$ hasOrder "2"
SAGetSalami $\sqsubseteq$ SimpleActivity
SAGetSalami $\sqsubseteq \neg$ SAPutawaySalami

SALieonLazychair

SALieonLazychair $\equiv$ SimpleActivity $\sqcap \exists$ hasMemory (Memory $\sqcap \exists$ hasLocomotion Lie $\sqcap$ $hasValue$ hasOrder "1" $\sqcap \exists$ hasMemory (Memory $\sqcap \exists$ hasLocomotion Sit $\sqcap$ $hasValue$ hasOrder "2" $\sqcap \exists$ hasMemory (Memory $\sqcap \exists$ hasMG MGInteractwithLazychair $\sqcap$ $hasValue$ hasOrder "2"
SALieonLazychair $\equiv$ SimpleActivity $\sqcap \exists$ hasMemory (Memory $\sqcap \exists$ hasLocomotion Lie $\sqcap$ $hasValue$ hasOrder "1"
SALieonLazychair $\sqsubseteq$ SimpleActivity

SAPrepareCheeseSandwich

SAPrepareCheeseSandwich ≡ SimpleActivity ⊓ ∃ hasMemory (Memory ⊓ ∃ hasAG AGSpread-Cheese ⊓ $hasValue$ hasOrder "1" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MG-FetchKnifeCheese ⊓ $hasValue$ hasOrder "2"
SAPrepareCheeseSandwich ≡ SimpleActivity ⊓ ∃ hasMemory (Memory ⊓ ∃ hasAG AGSpread-Cheese ⊓ $hasValue$ hasOrder "2" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MG-FetchCheese ⊓ $hasValue$ hasOrder "4" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MG-FetchKnifeCheese ⊓ $hasValue$ hasOrder "3" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MG-PutdownKnifeCheese ⊓ $hasValue$ hasOrder "1"
SAPrepareCheeseSandwich ≡ SimpleActivity ⊓ ∃ hasMemory (Memory ⊓ ∃ hasAG AGSpread-Cheese ⊓ $hasValue$ hasOrder "1" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MG-FetchCheese ⊓ $hasValue$ hasOrder "3" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MG-FetchKnifeCheese ⊓ $hasValue$ hasOrder "2"
SAPrepareCheeseSandwich ≡ SimpleActivity ⊓ ∃ hasMemory (Memory ⊓ ∃ hasAG AGSpread-Cheese ⊓ $hasValue$ hasOrder "2" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MG-FetchKnifeCheese ⊓ $hasValue$ hasOrder "3" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MG-PutdownKnifeCheese ⊓ $hasValue$ hasOrder "1"
SAPrepareCheeseSandwich ⊑ SimpleActivity

SAPrepareSalami

SAPrepareSalami ≡ SimpleActivity ⊓ ∃ hasMemory (Memory ⊓ ∃ hasAG AGCut-Salami ⊓ $hasValue$ hasOrder "1" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGFetchKnife-Salami ⊓ $hasValue$ hasOrder "2" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGFetch-Salami ⊓ $hasValue$ hasOrder "3"
SAPrepareSalami ≡ SimpleActivity ⊓ ∃ hasMemory (Memory ⊓ ∃ hasAG AGCut-Salami ⊓ $hasValue$ hasOrder "2" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGFetchKnife-Salami ⊓ $hasValue$ hasOrder "3" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGFetch-Salami ⊓ $hasValue$ hasOrder "4" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGPut-downKnifeSalami ⊓ $hasValue$ hasOrder "1"
SAPrepareSalami ≡ SimpleActivity ⊓ ∃ hasMemory (Memory ⊓ ∃ hasAG AGCut-Salami ⊓ $hasValue$ hasOrder "2" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGFetchKnife-Salami ⊓ $hasValue$ hasOrder "3" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGPut-downKnifeSalami ⊓ $hasValue$ hasOrder "1"
SAPrepareSalami ≡ SimpleActivity ⊓ ∃ hasMemory (Memory ⊓ ∃ hasAG AGCut-Salami ⊓ $hasValue$ hasOrder "1" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGFetchKnife-Salami ⊓ $hasValue$ hasOrder "2"
SAPrepareSalami ⊑ SimpleActivity

SAPutSugar

SAPutSugar ≡ SimpleActivity ⊓ ∃ hasMemory (Memory ⊓ ∃ hasAG AGStir-Spoon ⊓ $hasValue$ hasOrder "1" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGFetch-

Spoon $\sqcap$ $hasValue$ hasOrder "2" $\sqcap$ $\exists$ hasMemory (Memory $\sqcap$ $\exists$ hasMG MGFetch-Sugar $\sqcap$ $hasValue$ hasOrder "3"
SAPutSugar $\equiv$ SimpleActivity $\sqcap$ $\exists$ hasMemory (Memory $\sqcap$ $\exists$ hasAG AGStir-Spoon $\sqcap$ $hasValue$ hasOrder "2" $\sqcap$ $\exists$ hasMemory (Memory $\sqcap$ $\exists$ hasMG MGFetch-Spoon $\sqcap$ $hasValue$ hasOrder "3" $\sqcap$ $\exists$ hasMemory (Memory $\sqcap$ $\exists$ hasMG MGFetch-Sugar $\sqcap$ $hasValue$ hasOrder "4" $\sqcap$ $\exists$ hasMemory (Memory $\sqcap$ $\exists$ hasMG MGPut-downSpoon $\sqcap$ $hasValue$ hasOrder "1"
SAPutSugar $\sqsubseteq$ SimpleActivity

SAPutawayBottle

SAPutawayBottle $\equiv$ SimpleActivity $\sqcap$ $\exists$ hasMemory (Memory $\sqcap$ $\exists$ hasMG MG-CloseFridge $\sqcap$ $hasValue$ hasOrder "1" $\sqcap$ $\exists$ hasMemory (Memory $\sqcap$ $\exists$ hasMG MGFetch-Bottle $\sqcap$ $hasValue$ hasOrder "3" $\sqcap$ $\exists$ hasMemory (Memory $\sqcap$ $\exists$ hasMG MGPut-downBottle $\sqcap$ $hasValue$ hasOrder "2"
SAPutawayBottle $\equiv$ SimpleActivity $\sqcap$ $\exists$ hasMemory (Memory $\sqcap$ $\exists$ hasMG MG-CloseFridge $\sqcap$ $hasValue$ hasOrder "1" $\sqcap$ $\exists$ hasMemory (Memory $\sqcap$ $\exists$ hasMG MGFetch-Bottle $\sqcap$ $hasValue$ hasOrder "4" $\sqcap$ $\exists$ hasMemory (Memory $\sqcap$ $\exists$ hasMG MGOpen-Fridge $\sqcap$ $hasValue$ hasOrder "3" $\sqcap$ $\exists$ hasMemory (Memory $\sqcap$ $\exists$ hasMG MGPut-downBottle $\sqcap$ $hasValue$ hasOrder "2"
SAPutawayBottle $\equiv$ SimpleActivity $\sqcap$ $\exists$ hasMemory (Memory $\sqcap$ $\exists$ hasMG MGFetch-Bottle $\sqcap$ $hasValue$ hasOrder "3" $\sqcap$ $\exists$ hasMemory (Memory $\sqcap$ $\exists$ hasMG MGOpen-Fridge $\sqcap$ $hasValue$ hasOrder "2" $\sqcap$ $\exists$ hasMemory (Memory $\sqcap$ $\exists$ hasMG MGPut-downBottle $\sqcap$ $hasValue$ hasOrder "1"
SAPutawayBottle $\sqsubseteq$ SimpleActivity
SAPutawayBottle $\sqsubseteq$ $\neg$ SAGetBottle

SAPutawayBread

SAPutawayBread $\equiv$ SimpleActivity $\sqcap$ $\exists$ hasMemory (Memory $\sqcap$ $\exists$ hasMG MG-CloseDrawer3 $\sqcap$ $hasValue$ hasOrder "1" $\sqcap$ $\exists$ hasMemory (Memory $\sqcap$ $\exists$ hasMG MGFetch-Bread $\sqcap$ $hasValue$ hasOrder "3" $\sqcap$ $\exists$ hasMemory (Memory $\sqcap$ $\exists$ hasMG MGPut-downBread $\sqcap$ $hasValue$ hasOrder "2"
SAPutawayBread $\equiv$ SimpleActivity $\sqcap$ $\exists$ hasMemory (Memory $\sqcap$ $\exists$ hasMG MG-CloseDrawer3 $\sqcap$ $hasValue$ hasOrder "1" $\sqcap$ $\exists$ hasMemory (Memory $\sqcap$ $\exists$ hasMG MGFetch-Bread $\sqcap$ $hasValue$ hasOrder "4" $\sqcap$ $\exists$ hasMemory (Memory $\sqcap$ $\exists$ hasMG MGOpen-Drawer3 $\sqcap$ $hasValue$ hasOrder "3" $\sqcap$ $\exists$ hasMemory (Memory $\sqcap$ $\exists$ hasMG MG-PutdownBread $\sqcap$ $hasValue$ hasOrder "2"
SAPutawayBread $\equiv$ SimpleActivity $\sqcap$ $\exists$ hasMemory (Memory $\sqcap$ $\exists$ hasMG MGFetch-Bread $\sqcap$ $hasValue$ hasOrder "3" $\sqcap$ $\exists$ hasMemory (Memory $\sqcap$ $\exists$ hasMG MGOpen-Drawer3 $\sqcap$ $hasValue$ hasOrder "2" $\sqcap$ $\exists$ hasMemory (Memory $\sqcap$ $\exists$ hasMG MG-PutdownBread $\sqcap$ $hasValue$ hasOrder "1"
SAPutawayBread $\sqsubseteq$ SimpleActivity
SAPutawayBread $\sqsubseteq$ $\neg$ SAGetBread

SAPutawayCheese

SAPutawayCheese ≡ SimpleActivity ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MG-FetchCheese ⊓ $hasValue$ hasOrder "3" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGOpen-Fridge ⊓ $hasValue$ hasOrder "2" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGPut-downCheese ⊓ $hasValue$ hasOrder "1"
SAPutawayCheese ≡ SimpleActivity ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MG-CloseFridge ⊓ $hasValue$ hasOrder "1" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MG-FetchCheese ⊓ $hasValue$ hasOrder "3" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MG-PutdownCheese ⊓ $hasValue$ hasOrder "2"
SAPutawayCheese ≡ SimpleActivity ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MG-CloseFridge ⊓ $hasValue$ hasOrder "1" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MG-FetchCheese ⊓ $hasValue$ hasOrder "4" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGOpen-Fridge ⊓ $hasValue$ hasOrder "3" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGPut-downCheese ⊓ $hasValue$ hasOrder "2"
SAPutawayCheese ⊑ SimpleActivity
SAPutawayCheese ⊑ ¬ SAGetCheese

SAPutawayMilk

SAPutawayMilk ≡ SimpleActivity ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGFetch-Milk ⊓ $hasValue$ hasOrder "3" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGOpen-Fridge ⊓ $hasValue$ hasOrder "2" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGPut-downMilk ⊓ $hasValue$ hasOrder "1"
SAPutawayMilk ≡ SimpleActivity ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MG-CloseFridge ⊓ $hasValue$ hasOrder "1" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGFetch-Milk ⊓ $hasValue$ hasOrder "4" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGOpen-Fridge ⊓ $hasValue$ hasOrder "3" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGPut-downMilk ⊓ $hasValue$ hasOrder "2"
SAPutawayMilk ≡ SimpleActivity ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MG-CloseFridge ⊓ $hasValue$ hasOrder "1" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGFetch-Milk ⊓ $hasValue$ hasOrder "3" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGPut-downMilk ⊓ $hasValue$ hasOrder "2"
SAPutawayMilk ⊑ SimpleActivity
SAPutawayMilk ⊑ ¬ SAGetMilk

SAPutawaySalami

SAPutawaySalami ≡ SimpleActivity ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGFetch-Salami ⊓ $hasValue$ hasOrder "3" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGOpen-Fridge ⊓ $hasValue$ hasOrder "2" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGPut-downSalami ⊓ $hasValue$ hasOrder "1"
SAPutawaySalami ≡ SimpleActivity ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MG-CloseFridge ⊓ $hasValue$ hasOrder "1" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGFetch-Salami ⊓ $hasValue$ hasOrder "4" ⊓ ∃ hasMemory (Memory ⊓ ∃ hasMG MGOpen-

Fridge $\sqcap$ $hasValue$ hasOrder "3" $\sqcap$ $\exists$ hasMemory (Memory $\sqcap$ $\exists$ hasMG MGPutdownSalami $\sqcap$ $hasValue$ hasOrder "2"
SAPutawaySalami $\equiv$ SimpleActivity $\sqcap$ $\exists$ hasMemory (Memory $\sqcap$ $\exists$ hasMG MGCloseFridge $\sqcap$ $hasValue$ hasOrder "1" $\sqcap$ $\exists$ hasMemory (Memory $\sqcap$ $\exists$ hasMG MGFetchSalami $\sqcap$ $hasValue$ hasOrder "3" $\sqcap$ $\exists$ hasMemory (Memory $\sqcap$ $\exists$ hasMG MGPutdownSalami $\sqcap$ $hasValue$ hasOrder "2"
SAPutawaySalami $\sqsubseteq$ SimpleActivity
SAPutawaySalami $\sqsubseteq$ $\neg$ SAGetSalami

SAPutinDishwasher

SAPutinDishwasher $\equiv$ SimpleActivity $\sqcap$ $\exists$ hasMemory (Memory $\sqcap$ $\exists$ hasMG MGCloseDishwasher $\sqcap$ $hasValue$ hasOrder "1" $\sqcap$ $\exists$ hasMemory (Memory $\sqcap$ $\exists$ hasMG MGFetchWashableObject $\sqcap$ $hasValue$ hasOrder "3" $\sqcap$ $\exists$ hasMemory (Memory $\sqcap$ $\exists$ hasMG MGPutdownWashableObject $\sqcap$ $hasValue$ hasOrder "2"
SAPutinDishwasher $\equiv$ SimpleActivity $\sqcap$ $\exists$ hasMemory (Memory $\sqcap$ $\exists$ hasMG MGFetchWashableObject $\sqcap$ $hasValue$ hasOrder "3" $\sqcap$ $\exists$ hasMemory (Memory $\sqcap$ $\exists$ hasMG MGOpenDishwasher $\sqcap$ $hasValue$ hasOrder "2" $\sqcap$ $\exists$ hasMemory (Memory $\sqcap$ $\exists$ hasMG MGPutdownWashableObject $\sqcap$ $hasValue$ hasOrder "1"
SAPutinDishwasher $\equiv$ SimpleActivity $\sqcap$ $\exists$ hasMemory (Memory $\sqcap$ $\exists$ hasMG MGCloseDishwasher $\sqcap$ $hasValue$ hasOrder "1" $\sqcap$ $\exists$ hasMemory (Memory $\sqcap$ $\exists$ hasMG MGFetchWashableObject $\sqcap$ $hasValue$ hasOrder "4" $\sqcap$ $\exists$ hasMemory (Memory $\sqcap$ $\exists$ hasMG MGOpenDishwasher $\sqcap$ $hasValue$ hasOrder "3" $\sqcap$ $\exists$ hasMemory (Memory $\sqcap$ $\exists$ hasMG MGPutdownWashableObject $\sqcap$ $hasValue$ hasOrder "2"
SAPutinDishwasher $\sqsubseteq$ SimpleActivity

Salami

Salami $\sqsubseteq$ Object

SimpleActivity

Sip

Sip $\sqsubseteq$ Function

Sit

Sit $\sqsubseteq$ Locomotion

Spoon

Spoon $\sqsubseteq$ WashableObject

Spread

Spread $\sqsubseteq$ Function

Stand

Stand $\sqsubseteq$ Locomotion

Stir

Stir $\sqsubseteq$ Function

Sugar

Sugar $\sqsubseteq$ Object

Switch

Switch $\sqsubseteq$ Object

Table

Table $\sqsubseteq$ Object

Thing

Unlock

Unlock $\sqsubseteq$ Function

UserCA

UserCA $\sqsubseteq$ ComplexActivity

UserMG

UserMG $\sqsubseteq$ ManipulativeGesture

UserSA

UserSA $\sqsubseteq$ SimpleActivity

Walk

Walk $\sqsubseteq$ Locomotion

WashableObject

WashableObject $\sqsubseteq$ Object

Object properties

hasAG

$\exists$ hasAG Thing $\sqsubseteq$ Memory
$\top \sqsubseteq \forall$ hasAG AtomicGesture

hasAGActor

$\exists$ hasAGActor Thing $\sqsubseteq$ AtomicGesture
$\top \sqsubseteq \forall$ hasAGActor Person

hasArm

$\exists$ hasArm Thing $\sqsubseteq$ Person
$\top \sqsubseteq \forall$ hasArm Arm

hasAtomicGesture

$\exists$ hasAtomicGesture Thing $\sqsubseteq$ Person
$\top \sqsubseteq \forall$ hasAtomicGesture AtomicGesture

hasCAActor

$\exists$ hasCAActor Thing $\sqsubseteq$ ComplexActivity
$\top \sqsubseteq \forall$ hasCAActor Person

hasFunction

$\exists$ hasFunction Thing $\sqsubseteq$ Arm
$\top \sqsubseteq \forall$ hasFunction Function

hasLocomotion

$\exists$ hasLocomotion Thing $\sqsubseteq$ Person
$\top \sqsubseteq \forall$ hasLocomotion Locomotion

hasMG

$\exists$ hasMG Thing $\sqsubseteq$ Memory
$\top \sqsubseteq \forall$ hasMG ManipulativeGesture

hasMGActor

$\exists$ hasMGActor Thing $\sqsubseteq$ ManipulativeGesture
$\top \sqsubseteq \forall$ hasMGActor Person

hasManipulativeGesture

$\exists$ hasManipulativeGesture Thing $\sqsubseteq$ Person
$\top \sqsubseteq \forall$ hasManipulativeGesture ManipulativeGesture

hasMemory

$\exists$ hasMemory Thing $\sqsubseteq$ SimpleActivity
$\top \sqsubseteq \forall$ hasMemory Memory

hasObject

$\exists$ hasObject Thing $\sqsubseteq$ Arm
$\top \sqsubseteq \forall$ hasObject Object

hasSimpleActivity

$\exists$ hasSimpleActivity Thing $\sqsubseteq$ Person
$\top \sqsubseteq \forall$ hasSimpleActivity SimpleActivity

Data properties

hasOrder

$\exists$ hasOrder Datatype Literal $\sqsubseteq$ Memory
$\top \sqsubseteq \forall$ hasOrder Datatype integer

Bibliography

[ACRV13] G. Acampora, D.J. Cook, P. Rashidi, and A.V. Vasilakos. A survey on ambient intelligence in healthcare. *Proceedings of the IEEE*, 101(12):2470–2494, Dec 2013.

[AF98] Alessandro Artale and Enrico Franconi. A temporal description logic for reasoning about actions and plans. *Journal of Artificial Intelligence Research (JAIR)*, 9:463–506, 1998.

[All83] James F. Allen. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843, November 1983.

[AN04] Juan Carlos Augusto and Chris D. Nugent. The use of temporal reasoning and management of complex events in smart homes. In *ECAI*, pages 778–782, 2004.

[AR11] J.K. Aggarwal and M.S. Ryoo. Human activity analysis: A review. *ACM Comput. Surv.*, 43(3):16:1–16:43, April 2011.

[Baa03] Franz Baader. Description logic terminology. In *The Description Logic Handbook: Theory, Implementation, and Applications*, pages 485–495, 2003.

[Bac93] Fahiem Bacchus. Using first-order probability logic for the construction of bayesian networks. In *Proceedings of the Ninth international conference on Uncertainty in artificial intelligence*, UAI'93, pages 219–219, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.

[BCM+03] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[BMK+10] Maurice Bruynooghe, Theofrastos Mantadelis, Angelika Kimmig, Bernd Gutmann, Joost Vennekens, Gerda Janssens, and Luc De Raedt. Problog technology for inference in a probabilistic first order logic. In *ECAI*, pages 719–724, 2010.

[BPPW09] M. Buettner, R. Prasad, M. Philipose, and D. Wetherall. Recognizing daily activities with rfid-based sensors. In *Proceedings of the 11th international conference on Ubiquitous computing*, pages 51–60. ACM, 2009.

[BS01] Franz Baader and Ulrike Sattler. An overview of tableau algorithms for description logics. *Studia Logica*, 69(1):5–40, October 2001.

[BS08] Fernando Bobillo and Umberto Straccia. fuzzydl: An expressive fuzzy description logic reasoner. In *FUZZ-IEEE*, pages 923–930, 2008.

[BTF07] R. Biswas, S. Thrun, and K. Fujimura. Recognizing activities with multiple cues. In *Workshop on Human Motion*, pages 255–270, 2007.

[CCTK13] Diane J. Cook, Aaron S. Crandall, Brian L. Thomas, and Narayanan Chatapuram Krishnan. Casas: A smart home in a box. *IEEE Computer*, 46(7):62–69, 2013.

[CFPV12] Pedro Chahuara, Anthony Fleury, Franois Portet, and Michel Vacher. Using markov logic network for on-line activity recognition from non-visual home automation sensors. In Fabio Patern, Boris Ruyter, Panos Markopoulos, Carmen Santoro, Evert Loenen, and Kris Luyten, editors, *Ambient Intelligence*, volume 7683 of *Lecture Notes in Computer Science*, pages 177–192. Springer Berlin Heidelberg, 2012.

[CHN+12] Liming Chen, Jesse Hoey, Chris D. Nugent, Diane J. Cook, and Zhiwen Yu. Sensor-based activity recognition. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 42(6):790–808, 2012.

[CLPS09] M. Cirillo, F. Lazellotto, F. Pecora, and A. Saffiotti. Monitoring domestic activities with temporal constraints and components. In *Proc of the 5th Int Conf on Intelligent Environments*, pages 117–124, 2009.

[CMT+08] Sunny Consolvo, David W. McDonald, Tammy Toscos, Mike Y. Chen, Jon Froehlich, Beverly Harrison, Predrag Klasnja, Anthony LaMarca, Louis LeGrand, Ryan Libby, Ian Smith, and James A. Landay. Activity sensing in the wild: A field trial of ubifit garden. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 1797–1806, New York, NY, USA, 2008. ACM.

[CN09] Liming Chen and Chris D. Nugent. Ontology-based activity recognition in intelligent pervasive environments. *International Journal of Web Information Systems*, 5(4):410–430, 2009.

[CNM+08] Liming Chen, Chris Nugent, Maurice Mulvenna, Dewar Finlay, Xin Hong, and Micheal Poland. A logical framework for behaviour reasoning and assistance in a smart home. 2008. International Journal of ARM,VOL 9, NO.4 December 2008.

[CNO14] Liming Chen, Chris D. Nugent, and George Okeyo. An ontology-based hybrid approach to activity modeling for smart homes. *IEEE T. Human-Machine Systems*, 44(1):92–105, 2014.

[CNW12] Liming Chen, Chris D. Nugent, and Hui Wang. A knowledge-driven approach to activity recognition in smart homes. *IEEE Trans. Knowl. Data Eng.*, 24(6):961–974, 2012.

[Dav13] Nigel Davies. The essence of time. *IEEE Pervasive Computing*, 12(1):2–4, 2013.

[Dey01] Anind K. Dey. Understanding and using context. *Personal Ubiquitous Comput.*, 5(1):4–7, January 2001.

[DR04] Pedro Domingos and Matthew Richardson. Markov logic: A unifying framework for statistical relational learning. In *Proceedings of the ICML-2004 workshop on statistical relational learning and its connections to other fields*, pages 49–54, 2004.

[dSBAR08] Rodrigo de Salvo Braz, Eyal Amir, and Dan Roth. A survey of first-order probabilistic models. In *Innovations in Bayesian Networks*, volume 156, pages 289–317. Springer, 2008.

[FASP12] Jason Filipou, Alexander Artikis, Anastasios Skarlatidis, and Georgios Paliouras. A probabilistic logic programming event calculus. *CoRR*, abs/1204.1851, 2012.

[FAT11] Chrysi Filippaki, Grigoris Antoniou, and Ioannis Tsamardinos. Using constraint optimization for conflict resolution and detail control in activity recognition. In David V. Keyson, Mary Lou Maher, Norbert Streitz, Adrian David Cheok, Juan Carlos Augusto, Reiner Wichert, Gwenn Englebienne, Hamid K. Aghajan, and Ben J. A. Krse, editors, *AmI*, volume 7040 of *Lecture Notes in Computer Science*, pages 51–60. Springer, 2011.

[FCRS13] Barbara Furletti, Paolo Cintia, Chiara Renso, and Laura Spinsanti. Inferring human activities from gps tracks. In *Proceedings of the 2Nd ACM SIGKDD International Workshop on Urban Computing*, UrbComp '13, pages 5:1–5:8, 2013.

[GCTL10] Tao Gu, Shaxun Chen, XianPing Tao, and Jian Lu. An unsupervised approach to activity recognition and segmentation based on

object-use fingerprints. *Data Knowledge Engineering*, 69(6):533–544, 2010.

[GK06] B. Gutmann and K. Kersting. Tildecrf: Conditional random fields for logical sequences. In J. Fürnkranz, T. Scheffer, and M. Spiliopoulou, editors, *Proceedings of the 15th European Conference on Machine Learning (ECML-2006)*, LNAI (Lecture Notes in Artificial Intelligence), pages 174–185. Springer, 2006.

[Gra08] C. Graf. The lawton instrumental activities of daily living scale. *Am J Nurs*, 108(4):52–62; quiz 62–3, 2008.

[GT07a] L. Getoor and B. Taskar. *Introduction to Statistical Relational Learning*. MIT Press, 2007.

[GT07b] Lise Getoor and Ben Taskar, editors. *Bayesian Logic Programming: Theory and Tool*, chapter 10. MIT Press, 2007.

[GWT+09] Tao Gu, Zhanqing Wu, Xianping Tao, Hung Keng Pung, and Jian Lu. epsicar: An emerging patterns based approach to sequential, interleaved and concurrent activity recognition. *Pervasive Computing and Communications, IEEE International Conference on*, pages 1–9, 2009.

[Hel10] Rim Helaoui. Towards a proactive system based on activity recognition. In *PerCom Workshops*, pages 849–850, 2010.

[HHPZ+08] Derek Hao Hu, Sinno Jialin Pan, Vincent Wenchen Zheng, Nathan Nan Liu, and Qiang Yang. Real world activity recognition with multiple goals. In *UbiComp '08*, pages 30–39, New York, NY, USA, 2008. ACM.

[HKAK10] Yu-Jin Hong, Ig-Jae Kim, Sang Chul Ahn, and Hyoung-Gon Kim. Mobile health monitoring system based on activity recognition using accelerometer. *Simulation Modelling Practice and Theory*, 18(4):446 – 455, 2010. Modeling and Simulation Techniques for Future Generation Communication Networks.

[HKF13] Gerold Hlzl, Marc Kurz, and Alois Ferscha. Goal processing and semantic matchmaking in opportunistic activity and context recognition systems. In *The 9th International Conference on Autonomic and Autonomous Systems (ICAS2013)*, page 7, March 2013.

[HM11] Tuyen N. Huynh and Raymond J. Mooney. Online max-margin weight learning for markov logic networks. In *SDM*, pages 642–651. SIAM / Omnipress, 2011.

[HMS14] Jakob Huber, Christian Meilicke, and Heiner Stuckenschmidt. Applying markov logic for debugging probabilistic temporal knowledge bases. 2014.

[HNM+09] Xin Hong, Chris Nugent, Maurice Mulvenna, Sally McClean, Bryan Scotney, and Steven Devlin. Evidential fusion of sensor data for activity recognition in smart homes. *Pervasive Mob. Comput.*, 5(3):236–252, 2009.

[HNS10] R. Helaoui, M. Niepert, and H. Stuckenschmidt. A statistical-relational activity recognition framework for ambient assisted living systems. In *Ambient Intelligence and Future Trends-International Symposium on Ambient Intelligence*, 2010.

[HNS11a] Rim Helaoui, Mathias Niepert, and Heiner Stuckenschmidt. Recognizing interleaved and concurrent activities: A statistical-relational approach. In *PerCom*, pages 1–9, 2011.

[HNS11b] Rim Helaoui, Mathias Niepert, and Heiner Stuckenschmidt. Recognizing interleaved and concurrent activities using qualitative and quantitative temporal relationships. *Pervasive and Mobile Computing*, 7(6):660–670, 2011.

[HRN+12] Rim Helaoui, Daniele Riboni, Mathias Niepert, Claudio Bettini, and Heiner Stuckenschmidt. Towards activity recognition using probabilistic description logics. *Activity Context Representation: Techniques and Languages*, 12:05, 2012.

[HRS13] Rim Helaoui, Daniele Riboni, and Heiner Stuckenschmidt. A probabilistic ontological framework for the recognition of multilevel human activities. In *UbiComp*, pages 345–354, 2013.

[HY08] D. Hao Hu and Q. Yang. Cigar: Concurrent and interleaving goal and activity recognition. In *AAAI'08:Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, pages 1363–1368, 2008.

[INK+09] Tomohito Inomata, Futoshi Naya, Noriaki Kuwahara, Fumio Hattori, and Kiyoshi Kogure. Activity recognition from interactions with objects using dynamic bayesian network. In *Proceedings of the 3rd ACM International Workshop on Context-Awareness for Self-Managing Systems*, Casemans '09, pages 39–42, New York, NY, USA, 2009. ACM.

[Jae97] Manfred Jaeger. Relational bayesian networks. In *Proceedings of the 13th Annual Conference on Uncertainty in AI (UAI)*, pages 266–273. Morgan Kaufmann, 1997.

[JCC07] Vikramaditya Jakkula, Diane J. Cook, and Aaron S. Crandall. Temporal pattern discovery for anomaly detection in a smart home. In *The IET International Conference on Intelligent Environments*, pages 339–345, 2007.

[JvGB11] Dominik Jain, Klaus von Gleissenthall, and Michael Beetz. Bayesian Logic Networks and the Search for Samples with Backward Simulation and Abstract Constraint Learning. In *KI 2011: Advances in Artificial Intelligence, 34th Annual German Conference on AI*, volume 7006 of *Lecture Notes in Computer Science*, pages 144–156. Springer, 2011.

[Kat83] S. Katz. Assessing self-maintenance: activities of daily living, mobility, and instrumental activities of daily living. *J Am Geriatr Soc*, 31(12):721–7, 1983.

[KCD10] Stephen Knox, Lorcan Coyle, and Simon Dobson. Using ontologies in case-based activity recognition. In *FLAIRS Conference*, 2010.

[KD05] Stanley Kok and Pedro Domingos. Learning the structure of markov logic networks. In Luc De Raedt and Stefan Wrobel, editors, *ICML*, volume 119 of *ACM International Conference Proceeding Series*, pages 441–448. ACM, 2005.

[KF09] D. Koller and N. Friedman. *Probabilistic Graphical Models*. MIT Press, 2009.

[KFNM04] Holger Knublauch, Ray W. Fergerson, Natalya Fridman Noy, and Mark A. Musen. The protégé owl plugin: An open development environment for semantic web applications. In *Proceedings of the Third International Semantic Web Conference*, volume 3298 of *Lecture Notes in Computer Science*, pages 229–243. Springer, 2004.

[KHC10] E. Kim, S. Helal, and D. Cook. Human activity recognition and pattern discovery. *IEEE Pervasive Computing*, 9:48–53, 2010.

[KHF+11] Marc Kurz, Gerold Hlzl, Alois Ferscha, Alberto Calatroni, Daniel Roggen, Gerhard Tröster, Hesam Sagha, Ricardo Chavarriaga, José del R. Millán, David Bannach, Kai Kunze, and Paul Lukowicz. The opportunity framework and data processing ecosystem for opportunistic activity and context recognition. *International Journal of Sensors, Wireless Communications and Control, Special Issue on Autonomic and Opportunistic Communications*, pages 102–125, 2011.

[KHL+13] Shian-Ru Ke, Le Uyen Thuc Hoang, Yong-Jin Lee, Jenq-Neng Hwang, Jang-Hee Yoo, and Kyoung-Ho Choi. A review on video-based human activity recognition. *Computers*, 2(2):88–131, 2013.

[KN12] Victor Kaptelinin and Bonnie Nardi. Activity theory in hci: Fundamentals and reflections. *Synthesis Lectures Human-Centered Informatics*, 5(1):1–105, 2012.

[KR12] Kristian Kersting and Tapani Raiko. 'say em'for selecting probabilistic models for logical sequences. *arXiv preprint arXiv:1207.1353*, 2012.

[KRR06] Kristian Kersting, Luc De Raedt, and Tapani Raiko. Logical hidden markov models. *Journal of Artificial Intelligence Research*, 25:2006, 2006.

[KS86] R Kowalski and M Sergot. A logic-based calculus of events. *New Gen. Comput.*, 4(1):67–95, January 1986.

[LC11] Young-Seol Lee and Sung-Bae Cho. Activity recognition using hierarchical hidden markov models on a smartphone with 3d accelerometer. In *Proceedings of the 6th International Conference on Hybrid Artificial Intelligent Systems - Volume Part I*, HAIS'11, pages 460–467. Springer-Verlag, 2011.

[LD07] Daniel Lowd and Pedro Domingos. Efficient weight learning for markov logic networks. In *In Proceedings of the Eleventh European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 200–211, 2007.

[LL13] Oscar D. Lara and Miguel A. Labrador. A Survey on Human Activity Recognition using Wearable Sensors. *Communications Surveys & Tutorials, IEEE*, 15(3):1192–1209, 2013.

[LLD07] Fatiha Latfi, Bernard Lefebvre, and Céline Descheneaux. Ontology-based management of the telehealth smart home, dedicated to elderly in loss of cognitive autonomy. In *OWLED*, 2007.

[LM07] Carsten Lutz and Maja Milicic. A tableau algorithm for description logics with concrete domains and general tboxes. *J. Autom. Reasoning*, 38(1-3):227–259, 2007.

[LPB+10] Paul Lukowicz, Gerald Pirkl, David Bannach, Florian Wagner, Alberto Calatroni, Kilian Förster, Thomas Holleczek, Mirco Rossi, Daniel Roggen, Gerhard Tröster, Jakob Doppler, Clemens Holzmann, Andreas Riener, Alois Ferscha, and Ricardo Chavarriaga. Recording a complex, multi modal activity data set for context recognition. In *ARCS Workshops*, pages 161–166, 2010.

[LSH+09] Qiang Li, John A. Stankovic, Mark A. Hanson, Adam T. Barth, John Lach, and Gang Zhou. Accurate, fast fall detection using gyro-

scopes and accelerometer-derived posture information. In *Proceedings of the 2009 Sixth International Workshop on Wearable and Implantable Body Sensor Networks*, BSN '09, pages 138–143, Washington, DC, USA, 2009. IEEE Computer Society.

[LSPZ08] Weiyao Lin, Ming-Ting Sun, Radha Poovendran, and Zhengyou Zhang. Human activity recognition for video surveillance. In *ISCAS*, pages 2737–2740. IEEE, 2008.

[Luk08] Thomas Lukasiewicz. Expressive probabilistic description logics. *Artificial Intelligence*, 172(6-7):852 – 883, 2008.

[LV14] Jens Lehmann and Johanna Völker. *Perspectives on Ontology Learning*, volume 18. IOS Press, 2014.

[MBK08] J. Modayil, T. Bai, and H. Kautz. Improving the recognition of interleaved activities. In *UbiComp '08: Proceedings of the 10th international conference on Ubiquitous computing*, pages 40–43, 2008.

[MD14] Marco Maier and Florian Dorfmeister. Fine-grained activity recognition of pedestrians travelling by subway. In Grard Memmi and Ulf Blanke, editors, *Mobile Computing, Applications, and Services*, volume 130 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 122–139. Springer International Publishing, 2014.

[MDK15] Georgios Meditskos, Stamatia Dasiopoulou, and Ioannis Kompatsiaris. Metaq: A knowledge-driven framework for context-aware activity recognition combining sparql and owl 2 activity patterns. *Pervasive and Mobile Computing*, 2015.

[MMvO+12] Bogdan Moldovan, Plinio Moreno, Martijn van Otterlo, Jos Santos-Victor, and Luc De Raedt. Learning relational affordance models for robots in multi-object manipulation tasks. In *ICRA*, pages 4373–4378. IEEE, 2012.

[MS06] A. McCallum and C. Sutton. An introduction to conditional random fields for relational learning. In Lise Getoor and Ben Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, 2006.

[MVC+10] Atif Manzoor, Claudia Villalonga, Alberto Calatroni, Hong Linh Truong, Daniel Roggen, Schahram Dustdar, and Gerhard Tröster. Identifying important action primitives for high level activity recognition. In *EuroSSC*, pages 149–162, 2010.

[NBT+08] Sriraam Natarajan, Hung H. Bui, Prasad Tadepalli, Kristian Kersting, and Weng keen Wong. Logical hierarchical hidden markov models for modeling user activities. In *In Proc. of ILP-08*, 2008.

[NDHC10] Ehsan Nazerfard, Barnan Das, Lawrence B. Holder, and Diane J. Cook. Conditional random fields for activity recognition in smart environments. In *Proceedings of the 1st ACM International Health Informatics Symposium*, IHI '10, pages 282–286. ACM, 2010.

[NN11] Jan Noessner and Mathias Niepert. Elog: a probabilistic reasoner for owl el. In *Proceedings of the 5th international conference on Web reasoning and rule systems*, pages 281–286. Springer, 2011.

[NNS11] Mathias Niepert, Jan Noessner, and Heiner Stuckenschmidt. Log-linear description logics. In Toby Walsh, editor, *IJCAI*, pages 2153–2158. IJCAI/AAAI, 2011.

[NNS13] Jan Noessner, Mathias Niepert, and Heiner Stuckenschmidt. Rockit: Exploiting parallelism and symmetry for map inference in statistical relational models. *arXiv preprint arXiv:1304.4379*, 2013.

[Noe14] Jan Noessner. *Efficient Maximum A-Posteriori Inference in Markov Logic and Application in Description Logics*. PhD thesis, Universitätsbibliothek Mannheim, 2014.

[NRDS11] Feng Niu, Christopher Ré, AnHai Doan, and Jude Shavlik. Tuffy: Scaling up statistical inference in markov logic networks using an rdbms. *Proc. VLDB Endow.*, 4(6):373–384, March 2011.

[OCS12] George Okeyo, Liming Chen, Hui Wang 0001, and Roy Sterritt. A hybrid ontological and temporal approach for composite activity modelling. In Geyong Min, Yulei Wu, Lei (Chris) Liu, Xiaolong Jin, Stephen A. Jarvis, and Ahmed Yassin Al-Dubai, editors, *TrustCom*, pages 1763–1770. IEEE Computer Society, 2012.

[OCW13] George Okeyo, Liming Chen, and Hui Wang. An agent-mediated ontology-based approach for composite activity recognition in smart homes. *J. UCS*, 19(17):2577–2597, 2013.

[Org02] World Health Organization. International classification of functioning, disability and health (icf), 2002. `http://www.who.int/classifications/icf/en/`.

[OWL09] W3C OWL Working Group. *OWL 2 Web Ontology Language: Document Overview*. W3C Recommendation, 27 October 2009. Available at `http://www.w3.org/TR/owl2-overview/`.

[PCD+13] Federico Pecora, Marcello Cirillo, Francesca Dell'Osa, Jonas Ullberg, and Alessandro Saffiotti. A constraint-based approach for proactive, context-aware human support. AAAI, 2013.

[PD06] Hoifung Poon and Pedro Domingos. Sound and efficient inference with probabilistic and deterministic dependencies. In *AAAI*, volume 6, pages 458–463, 2006.

[PFKP05] Donald J. Patterson, Dieter Fox, Henry Kautz, and Matthai Philipose. Fine-grained activity recognition by aggregating abstract object usage. In *Proceedings of the Ninth IEEE International Symposium on Wearable Computers*. IEEE Computer Society, 2005.

[PGK+09] S Preece, JY Goulermas, LPJ Kenney, D Howard, K Meijer, and R Crompton. Activity identification using body-mounted sensors a review of classification techniques. *Physiological Measurement*, 30:R1–R33, 2009.

[PGK12] Tivadar Papai, Shalini Ghosh, and Henry A. Kautz. Combining subjective probabilities and data in training markov logic networks. In *ECML/PKDD (1)*, pages 90–105, 2012.

[Poo03] David Poole. First-order probabilistic inference. In *Proceedings of the 18th international joint conference on Artificial intelligence*, IJCAI'03, pages 985–991, San Francisco, CA, USA, 2003. Morgan Kaufmann Publishers Inc.

[RB09] D. Riboni and C. Bettini. Context-aware activity recognition through a combination of ontological and statistical reasoning. In *UIC '09*, pages 39–53, 2009.

[RB11] Daniele Riboni and Claudio Bettini. Owl 2 modeling and reasoning with complex human activities. *Pervasive and Mobile Computing*, 7(3):379–395, 2011.

[RCLCF14] Natalia Díaz Rodríguez, Manuel P Cuéllar, Johan Lilius, and Miguel Delgado Calvo-Flores. A survey on ontologies for human behavior recognition. *ACM Computing Surveys (CSUR)*, 46(4):43, 2014.

[RD06] M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62:107–136, 2006.

[Rie08] S. Riedel. Improving the accuracy and efficiency of map inference for markov logic. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 468–475, 2008.

[Ril02] Kyle Riley. Introduction to mathematical programming, applications and algorithms. *Mathematics and Computer Education*, 36(1):85, 2002.

[RN10] Stuart Russell and Peter Norvig. *Artificial intelligence : a modern approach*. Prentice Hall, 3 edition, 2010.

[RY05] Dan Roth and Wen-tau Yih. Integer linear programming inference for conditional random fields. In *Proceedings of the International Conference on Machine Learning*, pages 736–743, 2005.

[Sch98] Alexander Schrijver. *Theory of Linear and Integer Programming*. Wiley & Sons, 1998.

[SCSE09] G. Singla, D. Cook, and M. Schmitter-Edgecombe. Tracking activities in complex settings using smart environment technologies. *International Journal of BioSciences, Psychiatry and Technology*, 1:25–35, 2009.

[SD05] Parag Singla and Pedro Domingos. Discriminative training of markov logic networks. In *AAAI*, volume 5, pages 868–873, 2005.

[SD08] Parag Singla and Pedro Domingos. Lifted first-order belief propagation. In *AAAI*, volume 8, pages 1094–1099, 2008.

[SG13] Somdeb Sarkhel and Vibhav Gogate. Lifting walksat-based local search algorithms for map inference. In *AAAI Workshop: Statistical Relational Artificial Intelligence*. Citeseer, 2013.

[Sha99] Murray Shanahan. The event calculus explained. In *Artificial Intelligence Today*, pages 409–430. 1999.

[SK96] Bart Selman and Henry A. Kautz. Knowledge compilation and theory approximation. *J. ACM*, 43(2):193–224, 1996.

[SK12] Adam Sadilek and Henry Kautz. Modeling success, failure, and intent of multi-agent activities under severe noise. *Mobile Context Awareness*, pages 9–63, 2012.

[SKA+13] Young Chol Song, Henry A. Kautz, James F. Allen, Mary D. Swift, Yuncheng Li, Jiebo Luo, and Ce Zhang. A markov logic framework for recognizing complex events from multimodal data. In Julien Epps, Fang Chen, Sharon Oviatt, Kenji Mase, Andrew Sears, Kristiina Jokinen, and Bjrn Schuller, editors, *ICMI*, pages 141–148. ACM, 2013.

[SKC96] Bart Selman, Henry A. Kautz, and Bram Cohen. Local search strategies for satisfiability testing. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 521–532, 1996.

[SPAV12] Anastasios Skarlatidis, Georgios Paliouras, Alexander Artikis, and George A. Vouros. Probabilistic event calculus for event recognition. *CoRR*, abs/1207.3270, 2012.

[SPG+07] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A practical owl-dl reasoner. *Web Semantics: science, services and agents on the World Wide Web*, 5(2):51–53, 2007.

[Spi12] Marcus Spies. Knowledge discovery from constrained relational data: A tutorial on markov logic networks. volume 138 of *Lecture Notes in Business Information Processing*, pages 78–102. Springer, 2012.

[SPVA11] Anastasios Skarlatidis, Georgios Paliouras, George A. Vouros, and Alexander Artikis. Probabilistic event calculus based on markov logic networks. In Frank Olken, Monica Palmirani, and Davide Sottara, editors, *RuleML America*, volume 7018 of *Lecture Notes in Computer Science*, pages 155–170. Springer, 2011.

[SRO+08] Thomas Stiefmeier, Daniel Roggen, Georg Ogris, Paul Lukowicz, and Gerhard Tröster. Wearable activity tracking in car manufacturing. *IEEE Pervasive Computing*, 7(2):42–50, 2008.

[SSG78] Charles F. Schmidt, N. S. Sridharan, and John L. Goodson. The plan recognition problem: An intersection of psychology and artificial intelligence. *Artif. Intell.*, 11(1-2):45–83, 1978.

[SSS91] Manfred Schmidt-Schaubßand Gert Smolka. Attributive concept descriptions with complements. *Artif. Intell.*, 48(1):1–26, February 1991.

[ST09] Thomas Springer and Anni-Yasmin Turhan. Employing description logics in ambient intelligence for modeling and reasoning about complex situations. *J. Ambient Intell. Smart Environ.*, 1(3):235–259, August 2009.

[Str05] Umberto Straccia. A fuzzy description logic for the semantic web. In *Fuzzy logic and the semantic web*, pages 167–181. Elsevier, 2005.

[SZC11] Saguna, Arkady B. Zaslavsky, and Dipanjan Chakraborty. Recognizing concurrent and interleaved activities in social interactions. In *DASC*, pages 230–237. IEEE, 2011.

[SZC13] Saguna Saguna, Arkady Zaslavsky, and Dipanjan Chakraborty. Complex activity recognition using context-driven activity theory and activity signatures. *ACM Trans. Comput.-Hum. Interact.*, 20(6):32:1–32:34, 2013.

[TAK02] Ben Taskar, Pieter Abbeel, and Daphne Koller. Discriminative probabilistic models for relational data. In *Proceedings of the Eighteenth*

conference on Uncertainty in artificial intelligence, UAI'02, pages 485–492, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[TD08] S. D. Tran and L. S. Davis. Event modeling and recognition using markov logic networks. In *Proceedings of the 10th European Conference on Compute Vision*, pages 610–623, 2008.

[TIL04] Emmanuel Munguia Tapia, Stephen S. Intille, and Kent Larson. Activity recognition in the home using simple and ubiquitous sensors. In *In Pervasive*, pages 158–175, 2004.

[vKK07] T. van Kasteren and B. Krose. Bayesian activity recognition in residence for elders. In *Intelligent Environments, 2007. IE 07. 3rd IET International Conference on*, pages 209–212, Sept 2007.

[VV08] Douglas L. Vail and Manuela M. Veloso. Feature selection for activity recognition in multi-robot domains. In *AAAI*, pages 1415–1420, 2008.

[VVL07] Douglas L. Vail, Manuela M. Veloso, and John D. Lafferty. Conditional random fields for activity recognition. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '07, pages 235:1–235:8. ACM, 2007.

[Wei91] Mark Weiser. The computer for the 21st century. *Scientific American*, 265(3):66–75, January 1991.

[WPP+07] Shiaokai Wang, William Pentney, Ana M. Popescu, Tanzeem Choudhury, and Matthai Philipose. Common sense based joint training of human activity recognizers. In *IJCAI'07: Proceedings of the 20th international joint conference on Artifical intelligence*, pages 2237–2242, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[YDS+15] Juan Ye, Stamatia Dasiopoulou, Graeme Stevenson, Georgios Meditskos, Efstratios Kontopoulos, Ioannis Kompatsiaris, and Simon Dobson. Semantic web technologies in pervasive computing: A survey and research roadmap. *Pervasive and Mobile Computing*, 2015.

[YLC11] Jaeyoung Yang, Joonwhan Lee, and Joongmin Choi. Activity recognition based on rfid object usage for smart mobile devices. *J. Comput. Sci. Technol.*, 26(2):239–246, 2011.

[YSD14] Juan Ye, Graeme Stevenson, and Simon Dobson. KCAR: A knowledge-driven approach for concurrent activity recognition. *Pervasive and Mobile Computing*, 2014.

[YSK+07] Naoharu Yamada, Kenji Sakamoto, Goro Kunito, Yoshinori Isoda, Kenichi Yamazaki, and Satoshi Tanaka. Applying ontology and probabilistic model to human activity recognition from surrounding things. *IPSJ Digital Courier*, 3:506–517, 2007.

Ehrenwörtliche Erklärung

Ich versichere, dass ich die vorliegende Dissertation ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen. Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.