



UNIVERSITY OF BREMEN
INSTITUTE FOR
ARTIFICIAL INTELLIGENCE



Interpretation of Natural-language Robot Instructions

*Probabilistic Knowledge Representation,
Learning, and Reasoning*

Daniel Nyga

Vollständiger Abdruck der vom Fachbereich 3 (Mathematik und Informatik) der Universität Bremen zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:	Prof. Dr. Carsten Lutz <i>Universität Bremen</i>
1. Prüfer:	Prof. Michael Beetz, PhD <i>Universität Bremen</i>
2. Prüfer:	Prof. Anthony G. Cohn, PhD <i>University of Leeds</i>
Beisitzerin:	Prof. Dr. Tanja Schultz <i>Universität Bremen</i>

Die Dissertation wurde am 10.04.2017 bei der Universität Bremen eingereicht und durch den Prüfungsausschuss am 02.05.2017 angenommen.

Abstract

A robot that can be simply told in natural language what to do – this has been one of the ultimate long-standing goals in both Artificial Intelligence and Robotics research. In near-future applications, robotic assistants and companions will have to understand and perform commands such as “set the table for dinner”, “make pancakes for breakfast”, or “cut the pizza into 8 pieces.” Although such instructions are only vaguely formulated, complex sequences of sophisticated and accurate manipulation activities need to be carried out in order to accomplish the respective tasks. The acquisition of knowledge about how to perform these activities from huge collections of natural-language instructions from the Internet has garnered a lot of attention within the last decade. However, natural language is typically massively unspecific, incomplete, ambiguous and vague and thus requires powerful means for interpretation.

This work presents PRAC – Probabilistic Action Cores – an interpreter for natural-language instructions which is able to resolve vagueness and ambiguity in natural language and infer missing information pieces that are required to render an instruction executable by a robot. To this end, PRAC formulates the problem of instruction interpretation as a reasoning problem in first-order probabilistic knowledge bases. In particular, the system uses Markov logic networks as a carrier formalism for encoding uncertain knowledge. A novel framework for reasoning about unmodeled symbolic concepts is introduced, which incorporates ontological knowledge from taxonomies and exploits semantically similar relational structures in a domain of discourse. The resulting reasoning framework thus enables more compact representations of knowledge and exhibits strong generalization performance when being learnt from very sparse data. Furthermore, a novel approach for completing directives is presented, which applies semantic analogical reasoning to transfer knowledge collected from thousands of natural-language instruction sheets to new situations. In addition, a cohesive processing pipeline is described that transforms vague and incomplete task formulations into sequences of formally specified robot plans. The system is connected to a plan executive that is able to execute the computed plans

in a simulator. Experiments conducted in a publicly accessible, browser-based web interface showcase that PRAC is capable of closing the loop from natural-language instructions to their execution by a robot.

Zusammenfassung

Ein Roboter, dem man einfach sagt, was zu tun ist – das ist seit jeher eines der höchsten Ziele der Forschung sowohl in Künstlicher Intelligenz als auch in Robotik. In naher Zukunft schon werden Roboter als unsere Assistenten und Begleiter Anwendung finden, und Befehle wie „Decke den Tisch zum Abendessen“, „Mache Pfannkuchen zum Frühstück“ oder „Schneide die Pizza in 8 Stücke“ verstehen und ausführen müssen. Obwohl solche Instruktionen nur sehr vage formuliert sind, bedarf es der Ausführung komplizierter Folgen anspruchsvoller und filigraner Manipulationssaktionen, um die entsprechenden Aufgaben zu bewältigen. Innerhalb der letzten 10 Jahre hat der Erwerb von Wissen darüber, wie solche Aktivitäten auszuführen sind, mit Hilfe von natürlichsprachlichen Instruktionen aus dem Internet stark an Aufmerksamkeit gewonnen. Allerdings ist natürliche Sprache extrem vage, unpräzise, unvollständig und mehrdeutig und erfordert daher leistungsstarke Methoden für ihre Interpretation.

Diese Dissertation beschreibt PRAC – Probabilistic Action Cores – einen Interpreter für natürlichsprachliche Instruktionen, der in der Lage ist, sprachliche Vagheit und Mehrdeutigkeit aufzulösen und Wissen zu erschließen, das zur Ausführbarkeit der Instruktion durch einen Roboter fehlt. Hierfür wird die Interpretation einer Instruktion als Inferenzaufgabe in probabilistischen relationalen Wissensbasen formuliert. Im Besonderen werden Markov-Logik-Netzwerke als Formalismus zur Repräsentation von unsicherem Wissen betrachtet. Es wird ein neues Verfahren zum Schließen aus unmodellierten Konzepten vorgestellt, welches ontologisches Wissen über Konzept-taxonomien berücksichtigt und semantisch ähnliche relationale Strukturen von Konzepten der Anwendungsdomäne nutzt. Das damit entwickelte Grundgerüst für probabilistisches Schließen ermöglicht kompakte Repräsentationen von Wissen und zeichnet sich durch starke Abstraktions- und Generalisierungsfähigkeit aus, insbesondere, wenn nur sehr wenige Daten zum Lernen verfügbar sind. Desweiteren wird ein neuer Ansatz zur Vervollständigung von Instruktionen vorgestellt, der auf dem Prinzip des semantischen Analogie-Schließens beruht und den Wissenstransfer in neue Situationen ermöglicht. Außerdem wird ein in Algorithmus vorgestellt, der unvollständige

Aufgabenbeschreibungen in Folgen von formal vollständig spezifizierten Roboterplänen abbildet. PRAC ist an ein Planausführungssystem angebunden, welches in der Lage ist, die erstellten Roboterpläne in Simulation auszuführen. Experimente, die in einer öffentlich zugänglichen Web-Anwendung durchgeführt werden, zeigen, dass das PRAC System den vollständigen Inferenzprozess von der natürlichen Sprache bis hin zur Ausführung von Roboterplänen ermöglicht.

Acknowledgments

During the six years of research for my PhD, I greatly enjoyed meeting a number of amazing people whom I was very lucky to work with. They have helped and inspired me in exciting collaborations and lively discussions. These people have effectively contributed directly or indirectly to the contents of this thesis and thus deserve my highest acknowledgments, which I want to express at this point.

First and foremost, I want to thank my supervisor, Michael Beetz, who has continuously supported me already since my undergraduate studies at TUM and who has given me the opportunity to do my PhD in his lab. His outstanding expertise, visionary ideas and widespread support have motivated and helped me a lot in writing this thesis, but also in becoming a computer scientist, programmer, researcher, and lecturer. Not least, his unlimited optimism has had me never give up despite miscellaneous setbacks. I also want to thank Tony Cohn, Tanja Schultz, and Carsten Lutz, who kindly consented to serve in my thesis committee.

I would like to thank all of my colleagues from the former Intelligent Autonomous Systems Group at the Technical University of Munich and from the Institute for Artificial Intelligence at the University of Bremen, who have been forming a very inspiring environment it is a great honor to be a part of. My special thanks go to Moritz Tenorth, Dominik Jain, Lorenz Mösenlechner and Uli Klank from TUM, and to Mareike Picklum, Ferenc Bálint-Benczédi, Gheorghe Lisca, Mihai Pomarlan and Andrei Haidu, whom it was not only a lot of fun to collaborate with, but who have also spent with me a couple of enjoyable hours outside the lab.

Considerable parts of the work presented in this thesis would not have been possible without the help of my students, who contributed a lot in programming, data acquisition, model design and conducting experiments. Many thanks to Mareike Picklum, Sebastian Koralewski, Marc Niehaus, Florian Meyer, Susanne Knoop, Stephan Epping,

Valentine Chiwome, Nicholas Kirk, Ilija Dianov, and Dominik Vollmer.

My parents, Annette and Gerd Nyga, have provided me with their unconditional support and encouragement throughout my life, exciting my curiosity and thirst of knowledge, and nurturing my aptitude, for which I am very grateful. Not least I want to thank my girlfriend Mareike for her great support and patience while I was writing this thesis.

April 2017,
Daniel Nyga

This work has received funding from the CoTeSys (Cognition for Technical Systems) cluster of excellence of the German Research Foundation (DFG) and from the European Union Seventh Framework Programme FP7 projects ROBOHOW (grant number 288533) and ACAT (grant number 600578).

Contents

1	Introduction	1
1.1	Interpretation of Natural-language Instructions	5
1.1.1	Key Concepts: Uncertainty and Similarity	7
1.1.2	Exemplary Scenarios	9
1.1.3	Challenges & Opportunities	11
1.2	Contributions	14
1.3	Outline	16
1.4	Notation	17
2	Probabilistic Knowledge Representation	19
2.1	Logic	20
2.1.1	Propositional Logic	20
2.1.2	First-order Logic	23
2.1.3	Prolog & Datalog	24
2.1.4	Description Logic	25
2.1.5	Relational Algebra	28
2.2	Probabilistic Graphical Models	29
2.2.1	Probability Theory	30
2.2.2	Bayesian Networks	34
2.2.3	Markov Random Fields	37
2.3	Probabilistic Relational Models	41
2.3.1	Markov Logic Networks	42

2.4	Uncertainty versus Vagueness	49
2.4.1	Summary	51
3	Probabilistic Knowledge Bases for Instruction Interpretation	53
3.1	Probabilistic Action Cores	54
3.1.1	Definition	54
3.1.2	Alternative Approaches	55
3.2	Everyday Rationality	58
3.3	How much Knowledge does a Robot Need?	61
3.3.1	How Many Actions Are There?	63
3.3.2	How Much Variation Is There?	65
3.4	Conceptual Framework	67
3.4.1	Reasoning Tasks	70
3.4.2	System Architecture	72
3.4.3	PRAC Instructions	75
3.4.4	PRAC Dictionary	77
3.4.5	PRAC Knowledge Base	78
3.4.6	PRAC Howto Library	81
3.4.7	The PRAC Plan Library	84
3.5	Learning and Reasoning in PRAC	85
3.5.1	Reasoning Pipeline	87
3.6	Knowledge Acquisition	92
3.6.1	One-shot Learning from Natural Language	92
3.6.2	Data Acquisition with Mechanical Turk	94
3.7	Coreference Resolution	97
3.7.1	Probabilistic Coreference Resolution	99
3.7.2	Experiments	102
3.8	Instruction Completion	105
3.8.1	Probabilistic Completion & Refinement	106
3.8.2	Reasoning by Analogy	108

3.8.3	Analogical Completion & Refinement	112
3.8.4	Similarity of Frames	113
3.8.5	Action Role Completion	114
3.8.6	Plan Expansion	116
3.8.7	Plan Adaptation	117
3.9	Execution in Simulation	120
3.10	Related Work	122
4	Reasoning in Large Taxonomies	127
4.1	Motivation	128
4.1.1	Running Example	129
4.1.2	Semantic Similarity	130
4.1.3	Fuzzy Logic	132
4.2	FUZZY Markov Logic Networks	133
4.2.1	Definition	134
4.2.2	Semantics	134
4.2.3	Knowledge Representation	135
4.2.4	Example	136
4.3	Evaluation	142
4.4	Related Work	145
4.5	Discussion	147
5	Probabilistic Knowledge Bases for Robot Perception	153
5.1	Ensemble Learning	154
5.1.1	Overview	154
5.1.2	Processing Pipeline	156
5.1.3	Ensemble Learning	159
5.1.4	Experiments and Results	162
5.1.5	Discussion & Related Work	168
5.2	Interpretation of NL Object Descriptions	171
5.2.1	Semantic Models for Robot Perception	173

5.2.2	The ROBOSHERLOCK Perception Framework	175
5.2.3	Identifying Objects from Descriptions	176
5.2.4	Semantic Similarity Measures	179
5.2.5	Experiments	184
5.2.6	Related Work	185
5.3	Conclusions	187
6	Evaluation	189
6.1	Open-source Software	190
6.2	Comparison to Related Systems	192
6.3	Robot Demonstrators	195
6.3.1	Chemical Laboratory	195
6.3.2	Assistive Household	198
6.3.3	Statistics	201
7	Conclusions	205
A	Prior Publications	209
B	WordNet Concepts	213
	Glossary	220
	Bibliography	222

List of Figures

1.1	PR2 Reasoning About and Performing a Neutralization Task	5
1.2	Semantic Network Representation of a Neutralization Task	6
1.3	TUM-Rosie and PR2 Performing Everyday Activities	9
2.1	Exemplary Kitchen Ontology	27
2.2	Exemplary Bayesian Network Structures	35
2.3	Exemplary Markov Random Field	38
2.4	Exemplary Fuzzy Set	50
3.1	PR2 Performing Two Different Filling Activities	56
3.2	Screenshot of the wikihow.com Web Page	62
3.3	Taxonomy of Different Flipping Actions	65
3.4	Refinement and Parameterization of a Neutralization Instruction	68
3.5	Posterior Distribution over a Class Taxonomy Conditioned on a Neutralization Action	72
3.6	Architecture of the PRAC Framework	74
3.7	Excerpt of the WordNet Upper Ontology	77
3.8	Relational Data Model of the PRAC Howto Library	82
3.9	Posterior Distributions over a Class Taxonomy Conditioned on a Seasoning Action	86
3.10	Inference Tree of an Italian Dinner Example	87
3.11	Mechanical Turk: Example of a Generated HIT	97
3.12	Exemplary Coreferences in PRAC	99
3.13	Illustration of Probabilistic Role Completion	105

List of Figures

3.14 Analogical Reasoning using Large Amounts of Documents	109
3.15 Exemplary Action Role Completion Using Analogical Reasoning	111
3.16 Two Worlds in the Gazebo-based Robot Simulator.	121
3.17 Architecture of the Execution in Simulation	122
4.1 Excerpt of the WordNet Taxonomy	129
4.2 Visualization of the WUP Similarity	132
4.3 Posterior Distributions for Semantic Role Labeling	140
4.4 Posterior Distributions for Word-sense Disambiguation	141
4.5 Cross-validation Results for Word-sense Disambiguation	143
4.6 Interpolation in the Probability Density Function based on Semantic Distance	150
5.1 PR2 Looking at a Breakfast Table	154
5.2 Probabilistic Processing Pipeline for Object Classification	156
5.3 Object Classification Tasks in a Household Scenario	169
5.4 Pipeline for Detecting Objects from Natural Language	171
5.5 Taxonomy of Perceptual Attribute Types	173
5.6 Exemplary Annotations by Perception Experts	174
5.7 High-level Architecture of the Object Recognition System	176
5.8 Taxonomy of Custom Similarities	180
5.9 Symbolic Colors in HSV Color Space	182
5.10 Taxonomy of Shape Concepts	183
5.11 Results for Object Detection from Natural-language Descriptions	185
5.12 Table-top Scene with Objects Detected from Natural-language	186
6.1 Screenshots of the PRAC Web Application	190
6.2 Screenshot of the PRACMLN Web Application	191
6.3 Completion of a <i>Neutralizing</i> Instruction	195
6.4 Simulated Execution of Plans generated by PRAC from different natural- language instructions	197
6.5 Completion of a Flipping Instruction	198

List of Tables

3.1	Most Frequent Action Verbs in the wikihow.com Dataset and their Number of Occurrences.	64
3.2	Cluster Sizes of Prepositional Relations	66
3.3	Selection of Syntactic Relations	75
3.4	Selection of Different Word Meanings in WordNet	78
3.5	FrameNet Role Definitions for the Action Concept <i>MovingInPlace</i> . . .	80
3.6	Experimental Results of the Coreference Resolution Experiments. . .	103
3.7	Probability Distribution over the <i>AchievedBy</i> Predicate	107
3.8	Exemplary Queries for Semantically Most Similar Frames	120
4.1	Semantic Representation of Two Natural-language Instructions . . .	130
4.2	Cross-validation Results for Word-sense Disambiguation	144
5.1	Annotators Available in ROBOSHERLOCK	157
5.2	Class-specific Error Measures	163
5.3	Confusion Matrix for Object Detection with the Whole Ensemble of Annotators	164
5.4	Confusion Matrices for Object Detection with Isolated Annotators . .	165
5.5	Evaluation of Annotators in Isolation	166
5.6	Distribution over Perceptual Cues per Object Class	167
6.1	Overview of the Sizes of the PRAC Knowledge Bases	202
6.2	Runtime Analysis of Five Natural-language Instructions	203

List of Algorithms

1	MCMC	45
2	PRAC-QUERY	89
3	NEXT-MODULE	90
4	PRAC-TELL	93

Introduction

Roadmaps for research and technology identify robotic (co-)workers, assistants, and companions as promising targets for near-future robotic applications. Their forecasts anticipate robotics technology becoming dominant in the coming decade and significantly influencing every aspect of work and home (SPARC, 2014b). Until now, the mainstay of robotics applications has been restricted to factories and production lines, where mainly stationary robots perform the same preprogrammed motions repeatedly over long periods of time, unaware of their environment and of what they are doing. However, economists and technologists consent that the advent of a fourth disruptive transformation of manufacturing technologies will be largely driven by autonomization of manufacturing machinery, and a pervasive communication infrastructure of interconnected physical devices. The development of mobile robots with increased sensory capabilities and situation-, context- and environment-aware control routines allows robots to be applicable not only within safety cages in factories but safely, reconcilably and cooperatively interacting with humans in our everyday lives. Potential applications are being envisioned in numerous domains, such as the manufacturing, health care, agriculture, civil engineering, logistics and transport as well as consumer and entertainment sectors, among many others.

But not only will manufacturing and physically demanding jobs undergo a disruptive change by information and communication technologies, also knowledge intensive work will be increasingly taken over by computer and information systems, performing activities that so far have been deemed impossible to achieve without human intelligence. Such tasks include complex analyses of huge amounts of data, subtle judgments and interpretations thereof and creative problem solving and decision

making (Manyika et al., 2013). Both most prominent and most impressive is perhaps the IBM Watson system (Ferrucci et al., 2010b), which has set new standards in knowledge acquisition and reasoning from big, unstructured data and already today assists in medical diagnosing and financial forecasting. We are also seeing more and more products with Artificial Intelligence (AI) technology entering the consumer market, for example in the form of personal assistants in mobile phones, smart home appliances or self-driving vehicles.

Indeed, in recent years, we have seen tremendous progress in the mechatronics, sensing, and computational infrastructure, enabling robots to act faster, more strongly and more accurately than humans can ever do. Researchers have implemented robots that autonomously perform challenging manipulation tasks, such as making pancakes (Beetz et al., 2011) and pizza (Beetz et al., 2016), folding clothes (Srivastava et al., 2015), baking cookies (Bollini et al., 2011) and conducting chemical experiments (Lisca et al., 2015). Furthermore, the robotics community has become increasingly active in pushing robots' capabilities in perception, planning and manipulation towards more and more challenging scenarios (Wurman and Romano, 2015; Pratt and Manzo, 2013). However, albeit such impressive advances in manipulation skills, these robots are still far away from the desired generality and adaptability to bring about robust, autonomous, intelligent behavior.

It is the declared goal of the emerging field of *artificial cognitive robotics* to

“build systems that can act on their own to achieve goals: perceiving their environment, anticipating the need to act, learning from experience, and adapting to changing circumstances.”
— (Vernon, 2014)

As a research direction intersecting computer science, AI, robotics, developmental psychology, and cognitive neuroscience,

“cognition is the system-wide process that provides an agent with the ability to understand, given only partial knowledge, how things might possibly be, not just now but at some point in the future, and to use this understanding to influence action”
— (SPARC, 2014a)

Cognition brings together the advances in computational intelligence and physically embodied systems to allow a robot to act reliably and safely, to learn, to adapt, and to improve.

The above-mentioned examples and application domains have in common that autonomous robots have to act in *open worlds*: They must perform complete jobs

including a variety of human-scale activities, allow human interaction as naturally as possible, operate with high accuracy over extended periods of time without intervention, and autonomously extend their repertoire of high-level skills in unknown environments. A household robot, for example, will be tasked with instructions stated in natural language (NL) as vaguely as “clean up the kitchen,” “prepare pancakes for breakfast,” or “serve pizza and wine for dinner.” They have to perform what is commonly referred to as *everyday activity*. While humans are able to perform such activities with great ease, they still remain challenging for most of our today’s robots.



Everyday
Activity

One of the biggest challenges in implementing artificial cognitive systems is undoubtedly the ubiquity of severe incompleteness, ambiguity, vagueness, and uncertainty. These phenomena come in different manifestations. Consider, for example, a robotic assistant in a chemical laboratory, which is to “extract the DNA” from a given sample or to “determine the pH-value” of a substance, for instance. Such instructions are characterized by extreme vagueness and high abstraction on the one hand, but on the other hand performing them involves the execution of (sequences of) complex actions requiring sophisticated manipulation of objects. As part of a DNA extraction procedure, for instance, a robot may be instructed to “neutralize 75 ml of hydrochloric acid.” The robot has to know that it needs an appropriate alkaline counterpart for the neutralization, namely sodium hydroxide (NaOH). It must decide that the appropriate amount of NaOH needs to be added to the acid. And, depending on the amount, it also needs to choose an action to take. If the quantity of the substance is small and precisely specified, a pipette might be a suitable utensil, but when the amount is larger, a pouring action using a measuring cup might be more appropriate. Endowing robots with the ability to infer detailed and plausible formal specifications of executable action plans from vague and incomplete data is therefore a necessity for pushing the manipulation skills and hence the autonomy and universal applicability of today’s service robots to more advanced levels. In other words, incomplete and vague instructions need to be made executable.

A promising direction for tackling this is to equip robots with comprehensive general domain knowledge they can use to answer queries like the aforementioned (Tenorth, 2011). The human brain is a remarkably performant knowledge base (KB) that allows reasoning of this kind. Researchers from the cognitive sciences view the brain as an information processing unit where reasoning typically involves the inference of new information from information that has been put in from the senses, prior knowledge and other sources (Chater et al., 2006). Researchers in *Bayesian* cognition (Griffiths et al., 2008) consider probability theory a powerful mathematical apparatus for explaining and implementing theories of cognition:

“How does abstract knowledge guide inference from incomplete data? Abstract knowledge is encoded in a probabilistic generative model, a kind of mental model that describes the causal processes in the world giving rise to the learner’s observations as well as unobserved or latent variables that support effective prediction and action if the learner can infer their hidden state. Generative models must be probabilistic to handle the learner’s uncertainty about the true states of latent variables and the true causal processes at work. A generative model is abstract in two senses: It describes not only the specific situation at hand, but also a broader class of situations over which learning should generalize, and it captures in parsimonious form the essential world structure that causes learners’ observations and makes generalization possible.”
— (Tenenbaum et al., 2011)

Probabilistic methods have their greatest appeal perhaps in their generality and universality. However, more general formalisms incur higher computational costs for learning and reasoning. As most of the computational problems related to probabilistic methods require exponential time or space, these formalisms suffer from the high dimensionality that universality implies. As a consequence, the *practical* applicability of universal generative models still remains challenging in many cases, which is why probability theory has for a long time been deemed too limited in scope and scalability to be of practical use in cognitive science and has only lately moved into its research focus. These restrictions in probabilistic models have been significantly reduced by substantial technical progress in the mathematics and computer science of probabilistic models that enable the modeling of knowledge and beliefs of cognitive agents on the one hand. On the other hand researchers have moved away from the target of building rational agents that act optimally on a global scope, but agents that are aware of their limited computational resources and therefore have to balance deliberation effort and utility (Gershman et al., 2015). In addition, recent implementations of outstandingly successful AI systems appear to gain their performance from *specialization* rather than *generalization*. Examples of such systems are IBM Watson (Ferrucci et al., 2010a) and Google’s AlphaGo (Silver et al., 2016), which, for the first time in history, have significantly outperformed the world’s human champions in the quiz show ‘Jeopardy!’ and the board game ‘Go’, respectively. As a consequence, Bayesian cognition and the ‘Bayesian brain’ (Doya et al., 2007; Knill and Pouget, 2004) have become a recently emerging and promising target in cognitive science that is increasingly garnering attention.



Computational
Rationality

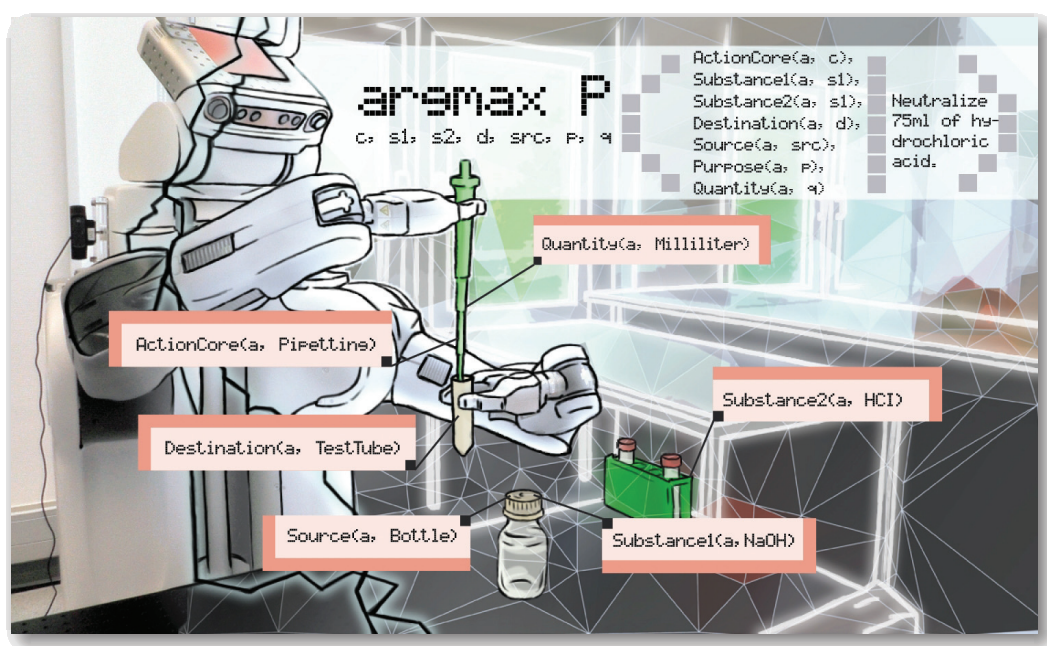


Figure 1.1: The PR2 robot reasoning about and performing a neutralization action by operating a pipette. The task of interpreting the natural-language instruction is phrased as a probabilistic maximum a-posteriori inference. (By courtesy of Nadine Freye)

1.1 Interpretation of Natural-language Instructions

This work largely follows the argumentation of Tenenbaum et al. (2011) and advocates the acquisition and use of probabilistic generative KBs to implement models of human cognition. More specifically, it investigates the representation, learning and reasoning in first-order probabilistic KBs for the interpretation and completion of vague, ambiguous, and under-determined instructions stated in natural language. The primary working hypothesis for this work is that the understanding and interpretation of natural-language (NL) instructions for robots can be formulated as and solved through reasoning problems in probabilistic relational models. Figure 1.1 illustrates a maximum a-posteriori (MAP) query to a probabilistic KB in form of a conditional probability, which, in its query part, retrieves a formal specification of an action a given the unstructured NL string “Neutralize 75 ml of hydrochloric acid” as evidence. Ultimately, the goal of the query is to determine the most probable type of action c to conduct as well as its formal, symbolic arguments, such as what kind of substances s_1 and s_2 to use, what the source and destination containers src and d are, and how much q of the substances to use. A possible response to this arg-max query is visualized by the red annotations of the objects in the image, which indicate that the robot believes the neutralization of the hydrochloric acid (HCl) can be achieved



Working
Hypothesis

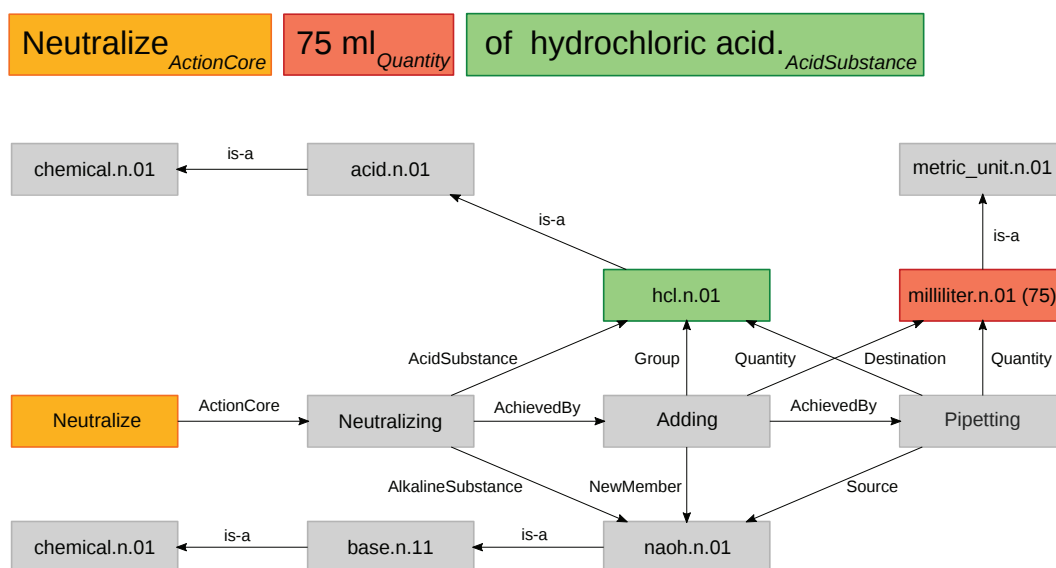


Figure 1.2: Exemplary instantiation of action cores and their action roles for the neutralization example. The colored nodes are given as evidence from the NL instruction, whereas the gray nodes and the role assignments need to be inferred.

by pipetting from a bottle holding the sodium hydroxide (NaOH) into the test tube with the HCl.

A more technical account to the rationale behind this kind of reasoning is depicted in Figure 1.2. It shows a fragment of a first-order representation of knowledge that explicates the reasoning process in a graph of entities and relations in between. There are symbols referring to actions, such as *Neutralizing*, *Adding* and *Pipetting*, which we call *action cores*, as well as symbols denoting object types like *hcl.n.01*, *naoh.n.01* or *milliliter.n.01*. Attached to the action cores there are edges assigning *action roles* to objects, such as the *AcidSubstance* and the *AlkalineSubstance* of the *Neutralizing* action core. The action roles thus can be considered abstract, semantic features of an action core. In addition, all object-related symbols are grounded in a class hierarchy of ontological concepts by the *is-a* relation. The *hcl.n.01* concept, for example, *is-a* kind of *acid.n.01*, which in turn *is-a* *chemical.n.01*. Moreover, the action cores in this example are sequentially connected via the *AchievedBy* relation, indicating that one action specification can be substituted by the respective other. Accordingly, the neutralization of 75 milliliters of HCl can be *AchievedBy* adding 75 milliliters of NaOH, which in turn can be *AchievedBy* pipetting of the respective substances. Contrasting the information pieces that have been given by the original NL instruction, highlighted in colors, against the whole network, it becomes obvious that the unstructured bits and pieces from the instruction merely serve as little evidence in a more complex semantic network of actions, objects, and relations. In

order to successfully accomplish the task of neutralization, the network as a whole needs to be fully instantiated, i.e. the gray nodes as well as the relational propositions need to be inferred from what has been given by an instruction.

1.1.1 Key Concepts: Uncertainty and Similarity

In order to account for the inherent uncertainty, ambiguity and underspecification in the reasoning from vague instructions to formal, unambiguous semantic representations like the one in the previous example, probabilistic first-order representations lend themselves to learn joint probability distributions over semantic networks of the kind shown in Figure 1.2. Joint probability distributions are of particular importance in the context of language interpretation as they allow to query a KB for any arbitrary aspect Q given any arbitrary aspect E as evidence. This is an important feature since one cannot make any commitment in advance which of the variables will be given in an NL statement and which ones need to be inferred. Most notably, the ultimate strength of probabilistic relational models (PRMs) is their capability to reason about *all* aspects simultaneously taking into account also the interactions among entities and thus to compute a posterior belief $P(Q | E)$ that is guaranteed to be probabilistically sound and globally consistent.

Following these considerations, an *action core* can be informally defined as a conceptualization of one or more *action verbs* and represents an abstract event type defined over the set of inter- and intra-conceptual semantic relations. The relations assign to every entity that is affected by the respective action a role from a set of *action roles*, which is attached to and specific to an action core. Action roles thus can be regarded as abstract symbolic action parameters assigning action-related semantics to objects. Knowledge about all roles of a particular action in turn is required to fully specify the action under consideration. The primary idea of Probabilistic Action Cores (PRAC) is to represent a joint probability distribution over the action roles such that evidence given by an NL instruction can be used to infer the missing roles that are required for fully specifying the action (cf. Nyga and Beetz, 2012).



Action Cores
and Roles



Probabilistic
Action Cores

Besides uncertainty, the notion of *semantic similarity* is another key concept used in this work. In the PRAC framework, symbols referring to objects are grounded in the *WordNet* (Fellbaum, 1998) lexical database. WordNet is a digital dictionary of the English language, which associates words to so-called *synsets* that are hierarchically structured and thus form an upper ontology of terms. The incorporation of a deep taxonomy of concepts allows to represent knowledge about action cores at an ap-

appropriate level of abstraction, such that it can be transferred to a large number of scenarios of similar types. Being able to abstract away from single instances of events to more general patterns is crucial in understanding NL. As an example, consider the two specific instructions “Fill_{ActionCore} a pot_{Destination} with water_{Theme}” and “Fill_{ActionCore} a glass_{Destination} with milk_{Theme},” where *Theme* and *Destination* denote the roles of the action core *Filling*. Employing a taxonomy of the action roles arguments allows to generalize the two examples of *Filling* activities to a more generic pattern like “Fill_{ActionCore} a container_{Destination} with a liquid_{Theme}.” Being queried for an unknown object, e.g. ‘juice,’ the knowledge about *Filling* acquired from the two examples can be transferred to the new concept by exploiting its similarity in the taxonomy.

PRACS therefore implement a mechanism that learns and represents events in everyday activity as stereotyped, generic event patterns from very few examples and thereby is an attempt to reproduce the remarkably efficient and effective ability to acquire language in humans (cf. Bailey, 1997). At its core, PRAC is also inspired by and closely related to Minsky’s frame structures:

“When one encounters a new situation (or makes a substantial change in one’s view of the present problem) one selects from memory a structure called a Frame. This is a remembered framework to be adapted to fit reality by changing details as necessary. A frame is a data-structure for representing a stereotyped situation, like being in a certain kind of living room, or going to a child’s birthday party. Attached to each frame are several kinds of information. Some of this information is about how to use the frame. Some is about what one can expect to happen next.”
— (Minsky, 1974)

Although the frame theory incorporates a notion of *expectations* about the slots of a frame, it does not explicitly commit to the use of probabilistic models. It is a rather informal, deterministic representation of everyday knowledge with correspondences in *semantic networks* (Simmons, 1963) and *default logic* (Reiter, 1980). The *FrameNet* project (Baker et al., 1998) is an initiative to implement the idea of frame semantics and provides a comprehensive set of frame definitions and also annotated data, but it offers neither computational models nor algorithms to learn or reason about the frames. However, the frame definitions from FrameNet are typically taken as a basis for the design of action cores.



(a) The robot ‘TUM-Rosie’ preparing pancakes in a kitchen environment. (b) The robot ‘PR2’ performing a pipetting action in a chemical laboratory.

Figure 1.3: TUM-Rosie and the PR2 performing everyday activities in human-scale environments.

1.1.2 Exemplary Scenarios

In recent experiments, the feasibility of robotic assistants performing sophisticated manipulation of objects has been shown in principle by executing plans for making pancakes in a kitchen environment (Beetz et al., 2011) and performing selected steps of a DNA extraction procedure in a chemical laboratory setup (Lisca et al., 2015) on the robotic platforms TUM-Rosie and a PR2 robot, which are illustrated in Figure 1.3. Throughout this thesis, I will refer to selected activities from these two different scenarios, in context of which most of the work presented has been conducted. For several reasons I consider a kitchen and a chemical laboratory very well suited as demonstrators for near-future robotic applications, which I will briefly depict in the following.

Assistive Household Leading technology scouting agencies and think tanks anticipate that, “within a decade, our living spaces will be enhanced by a host of new devices and technologies, performing a range of household functions and redefining what it means to feel at home.” (Coumau et al., 2017) Such ‘homebots’ may, for instance, assist elderly people at home, take over household chores and thereby help them prolonging their independence of elder care and enabling them to stay at home for a longer time (Schaal, 2007). Imagine a robotic assistant that is placed in a human household. Robots acting in human environments must be able to proficiently perform a wide range of complete jobs in unfamiliar environments that they have not been preprogrammed for. Such activities range from very simple actions to complex procedures involving multiple steps to be conducted in a sequence or even

in parallel. On the one hand, for example, a single action might be to just transport one object from one place and to put it down somewhere else. A severely complex task, on the other hand, would be to prepare a whole meal, such as making a pancake or preparing dinner. The kitchen in a human household is, though very common for most of us, a highly complex environment for a robot. A lot of sophisticated manipulation skills and knowledge about objects and actions involved is required to perform the tasks typically done in a kitchen.

Chemical Laboratory Consider a robot, which is placed in a chemical laboratory and is supposed to conduct a simple chemical experiment involving different containers, instruments and substances (e.g. liquids). The experiment is written in form of an instruction sheet comprising the single steps of the experiment. Also in this case, the robot needs powerful means for inferring missing information. For instance, an instruction such as “Add 5 drops of substance X to substance Y,” implies that a pipette has to be used as an instrument, or “Add a pinch of Y” implies that a scoopula has to be used, whereas an action “Add 500ml of water” implies using a measuring cup, a scale or the tap. The robot then at each step has to infer information it needs in order to perform the particular action successfully, but that is left out in the NL instruction. The example of “neutralize the hydrochloric acid” has already been discussed above. The robot might have to search for an alternative action parameterization (than the original specified in the instructions) because one particular object is not available in the lab, for instance. An example of a more complex task is conducting a pH-value test for a given substance or performing a DNA extraction procedure. In a biological and medical laboratory, there are many activities that are highly repetitive and monotonous. For example, the precisely same amount of substance needs to be pipetted into many small test tubes. At the same time, the activities must be performed repeatedly with the same high accuracy so results are not being biased. Hence, activities of this kind are salient candidates for being taken over by robots. In addition, lab automation and the ‘robot scientist’ is already today on the agenda of researchers and technology companies (Williams et al., 2015; King et al., 2004).

1.1.3 Challenges & Opportunities

Reasoning about vaguely and incompletely specified task descriptions has been identified as an essential capability of robotic agents that has great potential to become a powerful complement to action planning, and the retrieval of high-level action sequences from natural-language recipes has increasingly gained attention in the research community (Beetz et al., 2015b). These so-called *howtos* can be found in abundance on the web, for example on pages like [wikihow.com](http://www.wikihow.com). Howtos provide rough and sketchy sequences of actions that need to be executed in order to accomplish a task. Three minimalistic (and made up, but still representative) examples of such howtos are

How to make an Italian dinner

1. Set the table.
2. Prepare a pizza.
3. Cut the pizza and serve.
4. Serve some wine.

How to make a pizza

1. Spread tomato sauce over the rolled dough.
2. Sprinkle with salami.
3. Sprinkle with cheese.
4. Put the pizza in the oven and bake at 250°.
5. Season the pizza with oregano.

How to serve a drink

1. Take a glass from the cupboard.
2. Fill it with the drink.
3. Put the glass on the table.

The biggest challenge is that these instructions are written by humans and are intended for human use. Humans tend to omit important information that is necessary for performing a particular action, such that understanding and executing them requires appropriate interpretation. In the following, I will provide an exemplary, informal and incomplete list of reasoning tasks that need to be considered in order to enable automatic interpretation of NL instructions.

Ambiguity & Underspecification Ambiguity and incompleteness have already been mentioned as characteristics of NL instructions. In classical machine reading and translation, ambiguity can often be retained since queries and the respective answers typically have similar levels of abstraction. Instruction *execution*, however, requires to resolve those ambiguities in every detail and to fill in missing information pieces (Nyga and Beetz, 2015b). An example of ambiguous meanings of words is given by the two instructions “add a cup of salt” and “fill a cup with salt” and the meanings of the word ‘cup’ therein. In the first instruction, cup refers to an abstract

unit of measure, whereas in the second example, it denotes a physical container. Distinguishing between and correctly recognizing the two is crucial for the successful interpretation.

Multi-step Instructions Most of the high-level instructions must be resolved by and refined to finer-grained instructions that typically consist of multiple instruction steps, which also can cross-reference one another. In the above examples, the “Italian dinner” howto contains, for instance, a reference to the “pizza preparation” howto and to the “serve drink” howto. The vagueness in such instructions can be resolved by looking up the single instruction steps in other howtos that contain more specific commands for how to perform a certain action.

Coreference Tightly coupled to instructions consisting of multiple steps is the co-referencing of words across multiple sentences. Instead of repeating a term referenced in multiple subsequent instructions, the respective word is just replaced and referred to by means of a pronoun as in the instruction “Fill *it* with the drink.” The most extreme case of coreference is on hand in Step 3 of the “prepare dinner” example. Here, just the action verb ‘serve’ is given without any further information about what should be served.

Action Refinement Rarely are instructions stated in NL directly executable. They rather refer to actions that need to be performed in very different ways depending on the current context. An example is the instruction “cut the pizza and serve.” Depending on what kind of object needs to be cut, different cutting implements need to be used and different motions are required. Cutting a pizza, for instance, can be done with a cutting wheel, whereas cutting bread should be done with a knife or a slicing machine.

Object Substitution In the case that an object required to perform an action is not available or an agent lacks the physical capabilities to manipulate it, an appropriate substitute should be determined. This can be done by choosing an alternative that has either already worked before in the same context, or is likely to work because it has worked in a similar context.

Knowledge Transfer Related to the ability of substituting the parameterization of an object is the ability to transfer whole howtos to new situations. A howto written in NL should therefore be adaptable to new scenarios that an agent has not necessarily encountered before. An example is given by the “serve drink” howto that needs to be adapted to be applicable to the “serve some wine” instruction.

One-shot Learning Data-driven learning techniques play a predominant role for automatic knowledge acquisition in various areas in AI. However, for ‘open-world’ scenarios, comprehensive data sets for training are often unavailable or, if so, only for selected, very specific subproblems. In addition, learning is often critical with respect to the number of instances required to induce a sufficient bias to a model. It is, for example, unacceptable for a robot to let milk perish multiple times before it gains the knowledge that the refrigerator is a more appropriate storage place for milk than a normal cupboard. The knowledge must instead be acquired and permanently available after telling the robot *once* to “always store perishable items in the fridge.” Consequently, autonomous service robots will have to “acquire new knowledge quickly, on the fly, during task performance. Hence, we need to augment data-driven methods with other methods that allow for online learning from possibly only a few exemplars.” (Krause et al., 2014)

Object Properties Besides action-specific information, it is also crucial for a robotic agent to have an account to the semantics of NL descriptions of objects and their visual attributes. Instructions like “add drops of NaOH until the indicator turns *purple*” or “form the dough into *small, elongated* rolls” contains important information about how to perform the respective actions.

Opportunities The above-mentioned characteristics of vague and under-determined instructions are often considered difficulties that impede easy and flexible knowledge acquisition from NL. Vague descriptions of tasks and activities are, however, not only demanding key challenges for robotic agents but they are also an opportunity for more flexibility, generality, and robustness of plans. The massive lack of information thus can be considered free parameters in action specifications, which need to be adjusted to fit the needs of the agent’s current situation, environment and knowledge and therefore needs to be regarded helpful and even necessary in automated decision making.

1.2 Contributions

Technical Contributions In this thesis, I will present PRAC (Probabilistic Action Cores), an interpreter for natural-language, which phrases the problem of instruction understanding as a reasoning problem in probabilistic first-order knowledge bases in the fashion laid out in Section 1.1 and which addresses the challenges described in Section 1.1.3. More specifically, the main contributions of this work can be roughly dedicated to the areas of probabilistic knowledge *representation*, *reasoning* and *learning*:



Representation


- I will introduce the concept of *Probabilistic Action Cores* (PRAC) for representing action-specific knowledge for everyday activity, formalize PRAC and the computational problem of inferring the most probable executable action specification from vague instructions, and show how PRACs can be realized as Markov logic knowledge bases and learned from very few examples. An analysis of the kitchen and household domain with respect to the NL instructions that can be found on the wikihow.com web page. Based on this study I will hypothesize and give estimations for the amount of knowledge that robots might need for the proficient performance of envisioned tasks, how this knowledge can be represented, acquired and used. PRAC is able to solve the challenging and ubiquitous problems of vagueness, ambiguity, and incompleteness in natural language (NL) in a unifying probabilistic representational and inferential framework.
- I will introduce a representation and reasoning method for semantically storing and querying a large body of natural-language instruction sheets in a document-based database, which provides associative access to many similar instructions, which can be used to effectively and efficiently transfer and adapt existing knowledge about instructions to new situations and to infer missing information pieces.



Reasoning

- I will present a reasoning algorithm that comprehensively transforms vaguely stated natural-language commands into a sequence of robot plans, and demonstrate their executability by a proof-of-concept connection to the CRAM (Mösenlechner, 2016) plan executive.
- I will present a novel approach for reasoning about unknown concepts in a taxonomy by exploiting semantic similarity to known concepts in Markov logic, which typically impedes a compact representation of classes that are hierarchically organized in a taxonomy, which I call fuzzy Markov logic network

(FUZZY-MLN). FUZZY-MLN enables inference in presence of vague evidence, which allows the very compact representation of knowledge in MLNs and learning from very sparse data.

- I will present an approach for learning Markov logic KBs for combining arbitrary, complementary perception algorithms to form an ensemble of experts, which combines the strengths of all algorithms while compensating for their weaknesses. The probabilistic KBs also take into account co-occurrences of objects of daily use, whose correlations can be exploited to significantly boost the performance of state-of-the-art perception systems.  Learning
- I will introduce the notion of *knowledge-based* attributes in object perception – attributes that are symbols grounded in a rich taxonomy of concepts assigning them symbolic meaning, which can be used by a robot to identify objects just by NL descriptions of their appearances.

Publications The work conducted in context of this thesis has led to several publications in international conferences. The most relevant are

Daniel Nyga, Michael Beetz, “Cloud-based Probabilistic Knowledge Services for Instruction Interpretation,” *In International Symposium of Robotics Research (ISRR), Sestri Levante (Genoa), Italy, 2015.*

Daniel Nyga, Mareike Picklum, Sebastian Koralewski, Michael Beetz, “Instruction Completion through Instance-based Learning and Semantic Analogical Reasoning,” *In International Conference on Robotics and Automation (ICRA), Singapore, 2017. Accepted for publication*

Daniel Nyga, Michael Beetz, “Everything Robots Always Wanted to Know about Housework (But were afraid to ask),” *In 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vilamoura, Portugal, 2012.*

Daniel Nyga, Michael Beetz, “Reasoning about Unmodelled Concepts – Incorporating Class Taxonomies in Probabilistic Relational Models,” *In Arxiv.org, 2015. Preprint*

Daniel Nyga, Mareike Picklum, Michael Beetz, “What No Robot Has Seen Before – Probabilistic Interpretation of Natural-language Object Descriptions,” *In International Conference on Robotics and Automation (ICRA), Singapore, 2017. Accepted for publication*

Daniel Nyga, Ferenc Balint-Benczedi, Michael Beetz, “PR2 Looking at Things: Ensemble Learning for Unstructured Information Processing with Markov Logic Networks,” *In IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 2014.*

A complete list of my prior publications can be found in the Annex “Prior Publications” of this work.

Open-source Software Tools Large parts of the work have been released as open-source software under the BSD license (Open Source Initiative, 2017) and are actively used by partners from academia and industry.

Besides the PRAC system¹, PRACMLN² has been released as the learning and reasoning engine in PRAC, which is also available as a standalone toolbox for statistical relational learning and reasoning in Markov logic networks, implemented in the Python programming language. PRACMLN has started as a fork of the PROBCOG toolbox by Dominik Jain and has been extended by the developments in learning and reasoning presented in this thesis, such as the FUZZY-MLN reasoning mechanism.

Both PRAC and PRACMLN have their own project web pages and come with browser-based web applications for showcasing their performances, which have been developed in collaboration with Mareike Picklum.

1.3 Outline

Although this thesis is intended to be read in a linear fashion, I took care that the single chapters are as self-contained as possible such that they can also be read individually. The remainder of this thesis is organized as follows.

Chapter 2 briefly revisits the fundamental formalisms in (probabilistic) knowledge representation, learning and reasoning. It starts with a review of representation languages based on propositional and first-order logic, and explicates their limitations in settings that are subject to uncertainty. Some fundamentals in probability theory and probabilistic graphical models, in particular Bayesian and Markov networks are discussed. Finally, probabilistic relational models, specifically Markov logic networks, are introduced.

¹<http://www.actioncores.org>

²<http://www.pracmln.org>

Chapter 3 presents the PRAC learning and reasoning framework for the interpretation of NL robot instructions. It starts with an analysis of the human household domain and a discussion of the role of knowledge in intelligent autonomous decision making. It presents the main architectural components of the PRAC system and its probabilistic knowledge representation, and proposes a novel approach for the completion of under-determined NL instructions from large corpora of text found on the Web. The prior publications related to this chapter are Nyga and Beetz (2012, 2015b); Nyga et al. (2017b); Beetz et al. (2015b); Lisca et al. (2015) and Pomarlan et al. (2017).

Chapter 4 addresses the task of deep transfer of knowledge learnt from very sparse data. It introduces the concept of FUZZY-MLNs, a novel framework of performing reasoning in large taxonomies like WordNet by exploiting the taxonomic structure and semantic similarity of reasoning problems. Related prior publications are Nyga and Beetz (2015a).

Chapter 5 proposes an approach to use Markov logic networks for learning and combining ensembles of experts consisting of multiple diverse perception algorithms into one consistent and sound probabilistic model. In addition, a novel approach for perceiving objects only from textual descriptions is presented, which is implemented in PRAC. The major results presented in this chapter have been published in Nyga et al. (2014, 2017a).

Chapter 6 gives a synopsis of the latest state of the presented PRAC system and reports on experiments and public demonstrations, in which the methods developed in this work have been successfully applied.

1.4 Notation

It is likely that I was not able to consequently stick to a consistent nomenclature of mathematical symbols throughout this thesis. However, I still tried to comply with some common notational conventions to keep confusion as little as possible.

Following the convention in logic programming and probability theory, variable names are typically written uppercase, such as X . In some cases, X refers to a single variable and to a vector of variables in others. Which one applies should be clear from the context in most cases. Constant symbols or values of a variables are written lowercase, such as x .

The symbols \top and \perp refer to the Boolean values *true* and *false*. In context of upper ontologies, \top also denotes the most abstract concept or the set of all concepts.

Probability distributions are denoted by $P(\cdot)$, conditional distributions by $P(Q|E)$, where the Q part is referred to as the *query* and the E part is referred to as the *evidence* or *observation*.

Symbols of the form $X.Y.Z$ refer to word senses in the WordNet ontology, where X is a lemmatized, human-readable concept designator, $Y \in \{n, v, a, r\}$ is an identifier referring to a part of speech (*noun*, *verb*, *adjective*, *adverb*), and Z is a two-digit number denoting the index of the sense. An example is *cup.n.03*, which refers to the third sense of the noun ‘cup’. Concept names of this form follow the naming conventions of the NLTK toolbox (Loper and Bird, 2002), which is used in the implementation. A selection of word senses from the WordNet ontology can be found in Appendix B.

Probabilistic Knowledge Representation and -Acquisition

As described in Chapter 1, the meaning of a natural-language instruction encompasses many more constituents than the original instruction refers to. Consequently, representation languages are needed, which allow to represent what is said and what is meant, and to infer what is meant from what is said. Formalisms based on logic have a long tradition in the fields of natural-language understanding and Artificial Intelligence (AI) as they provide intuitive representation languages and formally sound and complete calculi. However, purely logical rules do only apply to deterministic worlds. Thus the practical applicability to applications subject to strong ambiguity and uncertainty, such as interpretation of human language, is limited.

The field of statistical relational learning (SRL), more specifically probabilistic relational models (PRMs), is a subfield of AI and statistics that intersects knowledge representation, logical reasoning, and machine learning. The field of SRL investigates combinations of probability theory, inductive statistical reasoning and first-order relational models, such as first-order logic (FOL). The ultimate goal of SRL is to bring together the benefits of two worlds: On the one hand, FOL provides a formalism that allows the compact representation of very complex and comprehensive knowledge. However, knowledge bases (KBs) in FOL collapse in case inconsistency. They are therefore largely inapplicable to domains subject to uncertainty. On the other hand, probabilistic (graphical) models can very well deal with inconsistency and uncertainty, but are typically defined over a fixed number of random variables.

Their propositional nature thus constrains their applicability to open domains due to limited expressiveness.

In this chapter, I give a brief introduction to different techniques in the field of (probabilistic) knowledge processing and PRMs. In particular, I motivate the use of Markov logic networks (MLNs) for encoding uncertain knowledge in technical cognitive systems, one of the perhaps best-known and most-widely used formalisms in the domain of PRMs. I begin with a brief review of classical logic-based representations and basic probability theory. Then, I motivate and introduce the combination of FOL and probabilistic graphical models (PGMs) in PRMs. The explications in this chapter are far away from being complete. Excellent and more comprehensive treatments of probabilistic relational models (PRMs) and PGMs can be found in Jain (2012); Getoor (2007a) and Koller and Friedman (2009).

2.1 Knowledge Representation with Logic

Logical languages have been intensively studied as a tool for describing the syntax and semantics of natural languages. As a mathematical apparatus providing expressive representation languages and sound and complete calculi, logic has become the predominant tool to represent common human knowledge in consistent formal theories, which allow the deduction of new provably correct knowledge from existing knowledge by means of systematic, syntactic manipulations of logical propositions. However, as I will review in this section, in case of inconsistency, their practical applicability is limited. Two of the most prominent variants of logics are propositional logic (PL) and first-order logic (FOL), which I will introduce in the following.

2.1.1 Propositional Logic

Propositional logic (PL) is one of the most basic formalisms in logic-based knowledge representation (cf. Russell and Norvig, 2003). In PL, logical statements can be made in the form of atomic propositions from an alphabet of symbols Σ , whose truth values are being determined by a function $\mathcal{I} : \Sigma \mapsto \mathbb{B}$, where \mathbb{B} is the set of Boolean truth values $\mathbb{B} = \{\top, \perp\}$ (where $\top \hat{=} \text{'true'}$ and $\perp \hat{=} \text{'false'}$). \mathcal{I} is called an *interpretation* or *possible world*, which assigns a truth value to every atomic proposition. More complex logical propositions can be constructed by combining sentences using logical junctors

as follows. If ϕ and ψ are sentences in propositional logic, then new sentences can be constructed by

- *negation*: $\neg\phi$
- *conjunction*: $\phi \wedge \psi$
- *disjunction*: $\phi \vee \psi$
- *implication*: $\phi \rightarrow \psi \equiv \neg\phi \vee \psi$
- *bi-implication* (also known as *equivalence*): $\phi \leftrightarrow \psi \equiv (\phi \rightarrow \psi) \wedge (\psi \rightarrow \phi)$,

which are again sentences themselves. Every atom in Σ is also a sentence. A *literal* denotes an atom or its negation, disjunctions of literals are called *clauses*. Atomic and complex sentences are also called logical formulae. In this work I use the terms formula, sentence and proposition interchangeably.

An interpretation \mathcal{I} of a sentence ϕ is called a *model* of ϕ , if the sentence is true under the interpretation \mathcal{I} , in symbols $\mathcal{I} \models \phi$. A conjunction is true under \mathcal{I} iff¹ all its constituents are true under \mathcal{I} , a negation inverts the truth value of its argument sentence in \mathcal{I} and a disjunction is true under \mathcal{I} iff at least one of its constituents is true under \mathcal{I} . A conjunction of sentences that is not a constituent of any other sentence under consideration is called a knowledge base (KB), i.e. a set of formulae all of which must be true. A KB thus can be regarded as a conjunction of formulae constituting the knowledge about the particular domain of discourse.

There are a few fundamental properties of KBs in PL I will briefly revisit in the following. All of these properties can be defined in terms of the models of a sentence. Let therefore $\mathcal{M}(\phi)$ denote the set of models of a sentence ϕ . Obviously, the number of models of a sentence is upper-bounded by the number of possible worlds, i.e. $|\mathcal{M}(\phi)| \leq 2^{|\Sigma|}$.

Satisfiability A sentence ϕ in PL is called *satisfiable*, iff there is at least one interpretation \mathcal{I} under which ϕ is true, i.e. $|\mathcal{M}(\phi)| > 0$. Otherwise, the sentence is called *unsatisfiable*.

Falsifiability A sentence ϕ is called *falsifiable*, iff there is at least one interpretation \mathcal{I} under which ϕ is false, i.e. $|\mathcal{M}(\phi)| < 2^{|\Sigma|}$.

Validity A sentence ϕ is called *valid*, iff ϕ is true in all possible worlds, i.e. under all interpretations, i.e. $|\mathcal{M}(\phi)| = 2^{|\Sigma|}$. Valid sentences are also called *tautologies*.

¹“if and only if”

Entailment A sentence ϕ entails another sentence ψ , in symbols $\phi \models \psi$, if every model of ϕ is also a model of ψ , i.e. $\mathcal{M}(\phi) \subseteq \mathcal{M}(\psi)$. Intuitively, entailment refers to the deducibility of one sentence from another, so if $\phi \models \psi$, one can be sure to recover ψ whenever ϕ is observed. ψ thus logically ‘follows’ from ϕ .

Some of the properties are mutually exclusive, some others imply one another. Obviously, any valid sentence is also satisfiable. More interestingly, if a sentence ϕ is valid, then its negation $\neg\phi$ is unsatisfiable and vice versa. There is, however, a more



Deduction
Theorem

fundamental property of sentences in PL exposed by the *deduction theorem*, which states that a sentence ϕ entails another sentence ψ if and only if the implication $\phi \rightarrow \psi$ is valid, i.e.

$$\phi \models \psi \iff \phi \rightarrow \psi \text{ is valid.}$$

Considering the propositions ψ that are entailed by a knowledge base KB , we find that

$$KB \models \psi \iff KB \rightarrow \psi \text{ is valid.} \tag{2.1}$$

$$\iff \neg(KB \rightarrow \psi) \text{ is unsatisfiable.}$$

$$\iff \neg(\neg KB \vee \psi) \text{ is unsatisfiable.}$$

$$\iff KB \wedge \neg\psi \text{ is unsatisfiable.} \tag{2.2}$$



Contradiction
Theorem

Equation (2.2) is also known as the *contradiction theorem*. Note that in (2.2), the sentence $KB \wedge \neg\psi$ can be regarded as a new knowledge base KB' , which originates from KB by adding a negated variant of ψ to it. These considerations yield two fundamental insights of KBs in propositional logic: First, in order to prove that a proposition ψ logically follows from a KB , $\neg\psi$ can simply be appended to KB . The resulting new KB' in turn must be proven unsatisfiable, as is done by, for instance, the resolution algorithm. Second, if a KB is unsatisfiable since the very beginning, i.e. $KB \models \perp$, then the right-hand side of (2.1) is constantly true. Practically speaking, this means that if a KB is inconsistent (unsatisfiable), any arbitrary proposition can be deduced, i.e. $\forall\psi. KB \models \psi$, which is, obviously undesirable for any KB. One can see that the deduction theorem imposes a heavy constraint on any logical KB: To be of practical use, a KB must be *consistent*.

2.1.2 First-order Logic

A more powerful logical formalism is first-order logic (FOL), which allows more general statements about the application domain under consideration and thus gets closer to the expressiveness of natural languages.

In PL, all atomic propositions are represented in the alphabet Σ . Every atom $A \in \Sigma$ hence corresponds to a fact in the real world that can be either true or false. For practical applications, knowledge representation with PL, however, can be quite cumbersome because of its limited ability to express knowledge about many entities at a time. Suppose one is to create a KB entailing that all dairy products are perishable. Given the fact that *Milk* is a dairy product, the KB is to entail that *Milk* is also perishable. In PL, this can be expressed by two atomic propositions “Milk is a dairy product” and “Milk is perishable” and the sentence “Milk is a dairy product” \rightarrow “Milk is perishable”. If the fact “Milk is a dairy product” now is true, it follows that “Milk is perishable”. However, this does not reflect the original intention to model the fact that *all* dairy items are perishable. If the domain of discourse now gets extended by another product, for instance, the KB does not generalize the knowledge to the new item.

FOL extends propositional logic by the following aspects. According to Schöning (1987), the basic elements of the FOL syntax are *constant symbols*, *predicate symbols*, *function symbols* and *variable symbols*. Constant symbols refer to entities or attributes in the world. Predicates refer to relations between these entities and are used to form atomic propositions. A *term* denotes a syntactic expression that refers to an entity. Every constant symbol and every variable is a term. A function can be applied to a tuple of terms, which again forms a term. A term that does not contain any variable is called a *ground term*. An atomic sentence that does not contain any variable is called a *ground atom* and a sentence/formula consisting only of ground atoms is called a *ground formula*.

Atomic propositions can be made in FOL by applying the predicate symbol to a tuple of terms, e.g. *dairy(milk)*. The most powerful element of FOL is perhaps the possibility to quantify propositions to hold for either *all* entities or for *at least one* entity in the domain.

The universal quantifier \forall followed by variable symbol and a sentence is used to extend the validity of the sentence to hold for all substitutions of the respective variable in the sentence. For example, $\forall x. (\text{dairy}(x) \rightarrow \text{perishable}(x))$ expresses the knowledge that all dairy items are perishable.

The existential quantifier \exists is used to ascertain the existence of at least one entity

for which a specific sentence holds. The sentence $\exists x. (\text{perishable}(x))$ states that there must be at least one substitution of x from the constant symbols for which the relation *perishable* holds.

The so-called *Herbrandt expansion* H_S of a set of clauses S is the set of all ground terms that can be constructed from all function symbols (if any) and the constant symbols (if none, the constant symbol A) by substituting all variables in S by elements from H_S . It is easy to see that, in principle, inferences about entailment of sentences in FOL can be done by finding substitutions of variables from the Herbrandt universe that generate grounded constituents of formulae that make logical expressions look identical. For example, applying the substitution of x by *milk* in the above formula, in symbols $(\text{dairy}(x) \rightarrow \text{perishable}(x))_{[x/milk]}$ yields the propositionalized sentence $\text{dairy}(\text{milk}) \rightarrow \text{perishable}(\text{milk})$, which can be used to deduce $\text{perishable}(\text{milk})$ given $\text{dairy}(\text{milk})$. The *unification* algorithm is a method that computes substitutions for any pair of sentences in FOL to make them look identical such that inference rules such as the resolution rule can be applied. However, since there are infinitely many substitutions induced by the Herbrandt universe (e.g. by recursion of functions), the problem of entailment in FOL is *semi-decidable* in the general case. This means that inference algorithms can determine whether a KB entails a sentence in a finite number of steps, but will not terminate in case the sentence is not entailed.

As FOL is only semi-decidable in the general case, restricted variants of FOL have been developed, which restore full decidability. Some of these logics allow inference even in polynomial time.

2.1.3 Prolog & Datalog

Restricted subsets, which are intended to be used in the same fashion as FOL is the family of *Datalog* and *Prolog* programming languages, in which KBs consist of rules and facts. Facts are required to be atomic, i.e. they must not contain any variables. Rules are required to be *definite clauses*, i.e. clauses with exactly one literal having positive polarity. By definition, such clauses with precisely one literal of positive polarity and all others being negated are effectively implications whose antecedent is given by all negative literals and the consequence given by the positive one, e.g.

$$\underbrace{A \wedge B \wedge C \rightarrow D}_{\text{implication}} \Leftrightarrow \underbrace{\neg A \vee \neg B \vee \neg C \vee D}_{\text{definite clause}}.$$

In addition to the restriction on the shape of rules in its KBs, Prolog does not support function symbols. Prolog makes use of a very efficient inference algorithm called *Backward-chaining*, which sequentially tries to prove definite clauses in a KB in the sequence they are declared in a Prolog program.

In such languages, inference can be done fairly efficiently, even in polynomial time. This increase in speed, however, comes at the cost of limited expressiveness.

2.1.4 Description Logic

Description logic (DL) is another subset of FOL, which has been mainly developed to facilitate the representation of aspects of the real world in terms of upper ontologies of concepts and instances thereof. DL is a decidable fragment of FOL, which has been designed to formally yet intuitively describe different aspects of the world, the objects and relations among them. It has evolved to the prevalent logic in the semantic web, and researchers have also recently developed powerful knowledge processing systems for robots, which are based on the DL formalism. An example of such a knowledge processing framework is, for instance, the KnowRob system (Tenorth, 2011). A KB in DL, also called an *ontology*, consists of two principle types of axioms, namely *terminological* and *assertional* axioms.



Ontology

Terminological Axioms The terminological axioms, also called the *TBox* of an ontology, define which categories of objects there are in the respective domain, which relations (e.g. specialization and generalization) between these categories hold, and what kinds of properties the objects may have. The TBox thus defines the building blocks on a conceptual level, which can be used to construct specific instances of a world. The notation of DL is slightly oriented at set calculus. If A and B are concepts, then new concepts can be defined using operators for

- concept definition: $A \doteq B$, i.e. all entities of type A are also entities of type B and vice versa,
- concept intersection: $A \sqcap B$, i.e. entities of concepts A and concept B at the same time,
- concept union: $A \sqcup B$, i.e. entities of type A or type B ,
- concept complement: $\neg A$, i.e. entities of all types but A

- concept inclusion: $A \sqsubseteq B$, i.e. all entities of type A are also of type B , but the opposite direction is not necessarily true.

Using these operators, new concepts can be defined relatively to previously defined concepts. The concept inclusion (\sqsubseteq) induces a hierarchical taxonomy relation on concepts, which allows to ‘inherit’ properties of more general concepts to more specific ones. Relations, in context of DL called *roles* are binary predicates that put two entities in relation to each other. In addition, cardinality and type constraints on relations can be expressed using the \forall and \exists quantifiers. Constraints on the arguments of a role can be made in form of $\exists R.C$ and $\forall R.C$, if R is a role and C some concept. A sentence of the form $\exists R.C$ states that for any entity e that is an instance of the concept under consideration, there must exist at least one entity e' for which the proposition $R(e, e')$ holds and whose type is C . Analogously, an expression of the form $\forall R.C$ states that all entities e' , for which $R(e, e')$ holds must be of the type C . As an example of a TBox in DL, consider the following concept definitions of terms in a rudimentary kitchen ontology:

$$\text{PhysicalThing} \doteq \neg \text{AbstractThing}$$

$$\text{UnitOfMeasure} \sqsubseteq \text{AbstractThing}$$

$$\text{ProperPhysicalThing} \sqsubseteq \text{PhysicalThing}$$

$$\text{Stuff} \doteq \neg \text{ProperPhysicalThing} \sqcap \text{PhysicalThing}$$

$$\text{Liquid} \sqsubseteq \text{Stuff}$$

$$\text{Container} \doteq \text{ProperPhysicalThing} \sqcap \exists \text{ holds.Stuff}$$

$$\sqcap \forall \text{ capacity.UnitOfMeasure}$$

$$\text{Cup} \doteq \text{Container} \sqcap \exists \text{ holds.Liquid} \sqcap \exists \text{ has.Handle}$$

A graphical representation of this ontology is depicted in Figure 2.1, where \top denotes the set of all concepts in \sqsubseteq , i.e. the most abstract concept subsuming all other concepts. This example models a dichotomization of all concepts into *AbstractThings* and *PhysicalThings*, which are mutually exclusive terms. The set of all *PhysicalThings* themselves decompose into countable *ProperPhysicalThings* and uncountable *Stuff*. A *Container* is defined as a *ProperPhysicalThing* that *holds* some *Stuff* and its *capacity* is determined by a *UnitOfMeasure*, which is an *AbstractThing*. A *Cup* is a specialization of a *Container* that *holds Liquids* and *has a Handle*.

Assertional Axioms Whilst the TBox defines the *terminological* building blocks for describing entities in the world, the ABox contains the *assertional axioms* describing a “specific state of affairs of an application domain in terms of concepts and

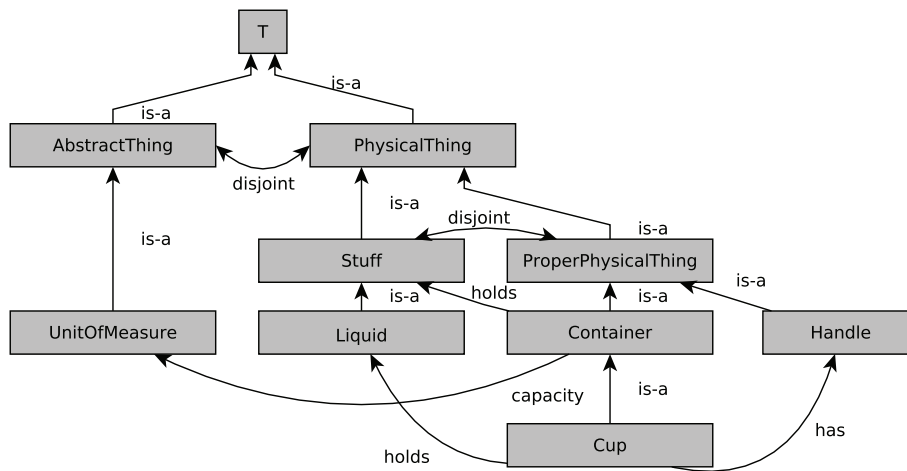


Figure 2.1: Graphical representation of the exemplary kitchen ontology

roles” (Baader and Nutt, 2003) from the TBox. Instances of concepts from the TBox are also called *individuals* and can be asserted in the form of a unary *concept assertion* of the form $C(a)$, where C is a concept symbol and a is a constant symbol identifying the individual. A second type of assertions are binary *role assertions* of the form $R(b, c)$, where R is a role of the individual b that is filled by another individual c . Using the two types of assertions, a specific world can be specified by an ABox such as

$$\begin{array}{ll} \text{Cup}(\text{cup-01}) & \text{has}(\text{cup-01}, \text{handle-01}) \\ \text{Handle}(\text{handle-01}) & \text{holds}(\text{cup-01}, \text{milk-01}) \\ \text{Liquid}(\text{milk-01}) & \end{array}$$

describing a world containing a *Cup* named *cup-01*, which *has* a *Handle* *handle-01* and *holds* the *Liquid* *milk-1*.

There are many variants of DL, which mainly differ in their expressiveness and operators allowed. For a more comprehensive overview, I refer the interested reader to Baader and Nutt (2003).

Reasoning Tasks In DL-encoded KBs, reasoning tasks of main interest include subsumption, i.e. to determine, which concepts subsume other concepts, such as “are rooms containers?”, for instance, or to determine all instances of particular concepts, such as “which individuals in the world are liquids?”.

2.1.5 Relational Algebra

Atomic facts (i.e. ground atoms) in FOL KBs have natural correspondences in relational databases. Here, predicate names correspond to *relation* or *table* names in a relational database scheme and the list of constant arguments of a ground atom represents a data *tuple* stored in that particular table. The tiny world model from the previous section thus can also be partially realized in a relational database consisting of six tables

Cup
cup-01

Handle
handle-01

Liquid
milk-01

is-a	
sub	super
Stuff	PhysicalThing
Liquid	Stuff
Cup	Container
⋮	⋮

has	
whole	part
cup-1	handle-01

holds	
container	content
cup-01	milk-01

where the identifying symbols like *cup-01* correspond to primary and foreign keys in the respective tables.

Relational algebra is an algebraic query language for retrieving tuples from a relational database. As relational algebra serves as a basis for most modern relational query languages, such as the SQL, it supports most of the operations, like the join, views, aliases and the Cartesian product. I will briefly review only the operations most relevant for this work (cf. Kemper and Eickler (2013)).

Selection The selection operator selects from a table all tuples that satisfy the *selection predicate*. The selection is denoted by σ and has attached in a subscript the selection predicate. An example is the query

$$\sigma_{\text{super=Liquid}}(\text{is-a}),$$

which selects from the taxonomy relation *is-a* all direct subconcept assertions of liquids. The selection $\sigma_p(R)$ can be regarded as an iteration over all tuples of the

argument relation R (in this case, $is-a$), evaluating the predicate ρ on each tuple. If the respective tuple satisfies ρ it is added to the relation which is returned as a result.

Projection While the selection operator extracts single rows (tuples) from a table, the *projection* operator $\Pi_\gamma(R)$ selects one or multiple columns γ from its argument relation R . Consequently,

$$\Pi_{sub}(\sigma_{super=Liquid}(is-a)),$$

projects from the subconcept assertions of liquids all concept symbols of the *sub* part of the *is-a* relation.

It can be more convenient to formulate queries to relational models in algebraic notation than in purely logical, declarative statements, since its more procedural character closer reflects a possible algorithmic implementation of a query in a relational KB.

2.2 Probabilistic Graphical Models

FOL and its variants provide powerful means for compactly representing and reasoning about deterministic knowledge. However, as shown above, the theoretical and practical constraints imposed on strictly logical KBs do not allow for inconsistency or uncertainty. Minsky stated this slightly more provocatively:

“Logical generalizations only apply to their literal lexical instances, and logical implications only apply to expressions that precisely instantiate their antecedent conditions. No exceptions are allowed, no matter how closely they match. This approach permits you to use no near misses, no suggestive clues, no compromises, no analogies, and no metaphors. To shackle yourself so inflexibly is to shoot your own mind in the foot – if you know what I mean.”
 – (Minsky, 1991)

Using joint probability distributions over atomic logical propositions, the hard logical constraints on KBs can be relaxed and uncertainty and inconsistency can be dealt with in a meaningful way. However, maintaining the full joint distributions is computationally infeasible. To reduce the computational and representational complexity in the distributions, probabilistic graphical models (PGMs) provide tools to make independence among random variables explicit in the form of topological properties of graphs.

2.2.1 Probability Theory

Bayesian probability theory provides a sound mathematical apparatus for dealing with the phenomena of uncertainty and inconsistency.

The pivotal element in probability theory is the concept of *random events* and *random variables*. An atomic event denotes a possible outcome of a random experiment. All possible outcomes of an experiment form the *sample space* Ω of an experiment. All atomic events $A \in \Omega$ are disjoint. Subsets of the sample space form a random event of the experiment. A *probability distribution* P is a function on the collection of events that additionally satisfies


1. $P(A) \geq 0, \forall A \subseteq \Omega$
2. $P(\Omega) = 1$
3. For any set of mutually exclusive events A_1, \dots, A_I :

$$P(\bigcup_{i=1}^I A_i) = \sum_{i=1}^I P(A_i)$$

A random variable $X : \Omega \mapsto E$ is a transformation of atomic events to some set of properties E that characterize the event. E is also called the domain of X , denoted as $dom(X) = E$. Obviously, Boolean random variables can be easily constructed by setting $E = \mathbb{B}$ and applying PL semantics. A probability distribution over such a random variable quantifies the degree to which an agent believes that the respective proposition holds in the real world. For example, $P(X = \top) = p$ denotes that, on a scale from 0 to 1, an agent's conviction that proposition X holds can be quantified by p . Consequently, $P(X = \perp) = 1 - p$. To keep the notation uncluttered, I also use the short forms $P(X)$ and $P(\neg X)$ instead of for $P(X = \top)$ and $P(X = \perp)$ whenever its meaning should be clear from context.

 Degree of Belief

More complex Boolean random variables can be constructed by using the full set of logical junctors of PL, such as conjunction, disjunction, negation, implication and bi-implication. For example, $P(X \wedge Y)$ denotes the probability of both X and Y being true at the same time. Probabilities of this kind are called *a-priori*, *marginal* or *unconditional* probabilities as they quantify an agent's belief in a proposition without any further constraints or conditions. Sometimes I write $P(X, Y, X)$ instead of $P(X \wedge Y \wedge Z)$. Such a prior probability distribution over multiple atomic variables is also called their *joint probability distribution*.

 Conditional Probability

Sometimes, however, the awareness of some event Y alters an agent's belief in some other event X that is possibly not directly observable but correlated with the observation. In such a scenario, the *conditional* or *posterior* probability of X , in

symbols $P(X|Y)$, denotes the probability of X given that Y is true. Technically, the conditional probability of X given Y is defined as

$$P(X|Y) = \frac{P(X \wedge Y)}{P(Y)} \quad (2.3)$$

Intuitively, the conditional probability of X given Y is defined as the proportional size of the set of all events in which both X and Y hold, relatively to the size of the set of events in which only the condition Y holds. From all events, in which Y holds, we take those in which also X holds, and compare the sizes of the event sets relatively to each other.

There are a few concepts, theorems and rules about probability distributions of random variables that I will briefly revisit in the following.

Independence In the case when knowledge about one variable Y does not alter an agent's belief of some other variable X , then $P(X|Y) = P(X)$ holds. X and Y are then called *independent* and $P(X \wedge Y) = P(X) \cdot P(Y)$. Otherwise, X and Y are called *dependent*.

Chain Rule of Probability As a corollary from the definition of conditional probabilities in Equation (2.3),

$$P(X, Y) = P(X|Y) \cdot P(Y), \quad (2.4)$$

which is also known as the product rule of probability. It allows to represent a joint probability distribution as a factorization of conditionals and marginals, which is the basis for probabilistic graphical models introduced below.

Law of Total Probability The law of total probability allows to sum out one or more random variables from a joint distribution as follows:


$$P(X) = \sum_{y \in \text{dom}(Y)} P(X, Y = y) = \sum_{y \in \text{dom}(Y)} P(X|Y = y) P(Y = y) \quad (2.5)$$

The process of applying the law of total probability is also called *marginalization*.

Bayes' Theorem One of the most fundamental theorems in probability theory is *Bayes' theorem*:

$$P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)} \quad (2.6)$$


Bayes' theorem allows to convert an abductive reasoning task into a deductive one and vice versa. It plays a fundamental role in Bayesian networks, which model uncertain causal relationships among random variables. Applying Bayes' theorem allows to turn a causal model into a diagnostic reasoning task.

 **Inference** It is obviously straightforward to convert a propositional logic KB into a probabilistic KB by introducing a Boolean random variable for every atomic proposition in Σ . The joint probability distribution $P(\Sigma)$ then can be stored in a contingency table holding the co-occurrences of the respective atomic events. Making use of the chain rule, the law of total probability and Bayes' theorem, one can compute the posterior probability $P(Q|E)$ of any arbitrary query $Q \subseteq \Sigma$ given any arbitrary evidence $E \subseteq \Sigma$. Such a posterior belief can be computed by the canonical inference equation (cf. Jain, 2012)

Posterior Inference

$$\begin{aligned} P(Q = q | E = e) &= \frac{P(Q = q, E = e)}{P(E = e)} \\ &= \frac{\sum_{u \in \text{dom}(U)} P(Q = q, E = e, U = u)}{\sum_{q' \in \text{dom}(Q)} \sum_{u \in \text{dom}(U)} P(Q = q', E = e, U = u)}, \end{aligned}$$

where $U = \Sigma \setminus Q \setminus E$ is the set of all hidden variables. However, as there must be an entry for every combination of atomic events, such a full joint probability table is hopelessly infeasible to represent for practical applications. Not only does the table grow exponentially in the number of variables, but, in order to be of statistical significance, the amount of data to be collected also grows exponentially.

 **Most Probable Explanation** Computing the posterior distribution over a set of query variables Q in the light of observed evidence variables E is only one kind of inference that is typically carried out in probabilistic KBs. Sometimes, however, one is not necessarily interested in the complete posterior distribution over the queries, but only in the *most probable* variable assignment given the evidence. In this kind of inference, also called most probable explanation (MPE) or maximum a-posteriori (MAP) inference, thus the most probable possible world \hat{x} is computed that is compatible with the observations E , i.e.

$$\hat{x} = \arg \max_{x \in \mathcal{X}} P(x | E).$$

As $P(x|E) \propto P(x, E)$, it is not required to compute a normalization constant, and it is often computationally more appealing to compute the MPE state of a posterior, when the complete distribution is not necessarily required. A very common class of such inferences are classification problems, where one is typically interested in assigning an entity the most probable class.

Learning There are, in principle, two ways for determining the quantitative specification of probabilities of specific random events: First, a knowledge engineer can enter manually probabilities into the model that expresses their belief in and knowledge about the domain of discourse. Such probabilities are called *subjective* probabilities. Another, much more frequent way of obtaining probabilities is to derive them from previously made observations. This special kind of inference, also called *inductive reasoning*, aims at determining the most suitable model parameters $\hat{\theta}$, in some model space Θ , such that the model best explains the observed data set. The training set \mathcal{D} comprises N examples of complete assignments of all variables under consideration, $\mathcal{D} = \{d_1, \dots, d_N\}$. It is typically assumed that the individual examples d_i represent samples that have been drawn independently of each other from the same ‘true’, problem-intrinsic distribution which is to be approximated by the learning procedure. This assumption is also called the *i.i.d.* (independent, identically distributed) assumption. Given the data at hand and variable model parameters θ , the *likelihood function* \mathcal{L} measures the congruence of the model defined by θ and the observations by applying the model under θ to \mathcal{D} , i.e.

$$\mathcal{L}(\theta | \mathcal{D}) = \prod_{i=1}^N P(d_i; \theta). \quad (2.7)$$

The problem of *parameter estimation* or *learning* is now to find the parameters $\hat{\theta}$ that maximize the likelihood function (2.7) with respect to θ ,

$$\hat{\theta}_{MLE} = \arg \max_{\theta \in \Theta} \mathcal{L}(\theta) = \arg \max_{\theta \in \Theta} \prod_{i=1}^N P(d_i; \theta),$$

to yield the parameters $\hat{\theta}_{MLE}$ that best explain the observations, the so-called maximum likelihood estimate (MLE). The maximum likelihood principle is the most fundamental and perhaps most commonly used technique to fit probabilistic models to observed training data. Often, it is more convenient to maximize the natural logarithm of the likelihood function. The so-called *log-likelihood* does have its maxima at



Maximum
Likelihood

same positions as the logarithm is a strictly monotonically increasing function. The log likelihood has the advantage that the logarithm of the product in (2.7) turns into a sum of the individual logarithms, which can be conveniently maximized when the model's functional form is chosen appropriately.

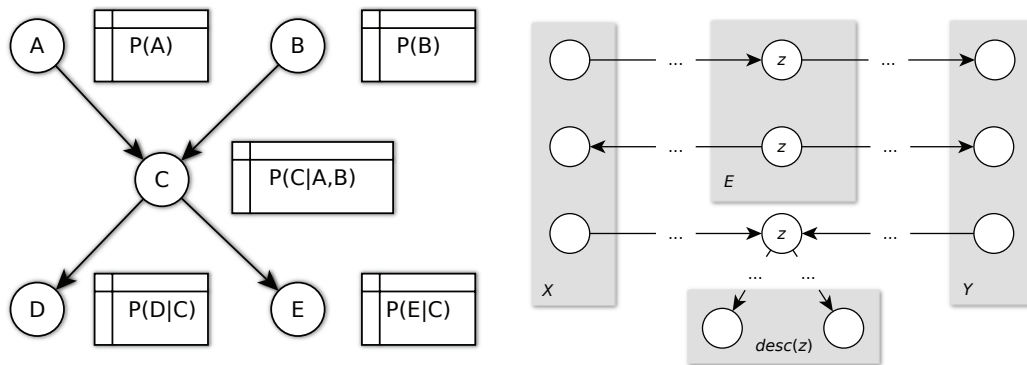
Complexity Both the problem of parameter learning and inference can be shown to be NP-complete (Russell and Norvig, 2003). Indeed, it can be shown that for many probabilistic models, the optimal model parameters can be obtained by combining model inference with a numeric optimization technique. It is therefore unlikely that there exist algorithms to compute the respective problems efficiently. Unfortunately, it seems that the only effective way to reduce the representational complexity of probabilistic KBs is rigorous exploitation of domain knowledge about independencies of variables (Koller and Friedman, 2009). Probabilistic graphical models (PGMs) are representation formalisms that make use of directed or undirected graph structures to represent the (in-)dependencies among random variables.

2.2.2 Bayesian Networks

In order to mitigate the computational complexity of probabilistic methods, Bayesian networks (BNs) provide a formalism to specify dependencies in terms of a causal relationships of variables. A Bayesian network (BN) consists of a directed, acyclic graph (DAG) structure, where there is one node for every random variable under consideration, and the (directed) edges among the nodes determine the dependency structure of the distribution. A node with an outgoing edge is called a *parent* of the node that the respective edge points at. Every node X_i with incoming edges directly depends on all of its parents, which are denoted by the set $Par(X_i)$. Every node has attached a distribution that determines the probability of the respective variable conditioned on all its direct parents. The joint distribution over all variables X_1, \dots, X_n is then given by

$$P(X_1, \dots, X_n) = \prod_{i=1}^n P(X_i | Par(X_i)) \quad (2.8)$$

The graph structure of a BN thus directly determines the factorization of the joint distribution. BNs have been successfully used for modeling causal relationships among variables, such as in expert systems for deductive and abductive reasoning in medical diagnosis (Uebersax, 2004) and risk management (Cárdenas et al., 2013),



(a) Exemplary Bayesian network with five nodes A, B, C, D, E and their conditional probability tables. (b) Graphical representation of the three cases of path blocking in the d -separation criterion.

Figure 2.2: Exemplary Bayesian network structures

among many others.

Knowledge engineering with BNs appears particularly intuitive and natural since the graph structure directly determines the joint distribution as a normalized factorization of conditionals reflecting the graph structure. In addition, the principle of a causes-and-effects model allows very intuitive top-down construction of BNs.

An example of a BN is shown in Figure 2.2a. It consists of five random variables/nodes A, B, C, D and E , and each node has attached a conditional probability table conditioned on its direct parents storing the respective conditional distributions. According to (2.8), the distribution in turn factorizes as

$$P(A, B, C, D, E) = P(D|C) \cdot P(E|C) \cdot P(C|A, B) \cdot P(B) \cdot P(A).$$

The structure of this exemplary BN is adapted from the well-known ‘alarm example’ from Russell and Norvig (2003). The random event C can have one of two causes A and B , which are independent of each other. The common effect C of A and B can itself trigger two possible events D and E . Although very simplistic, the example exposes a couple of very interesting and fundamental properties of BNs.

Parameter Estimation Parameter estimation boils down to counting relative frequencies of events in the training data and storing them in conditional probability tables, which can be shown to be the maximum likelihood estimate of the conditional probabilities. There are, however, a few subtleties that need to be taken into account when working with BNs, which I address in the following.

 Example

Conditional Independence The perhaps most subtle property of variables in a BN is the phenomenon of conditional dependence, i.e. two a-priori independent variables in a BN can become dependent given a third variable is observed. In the BN in Figure 2.2a, A and B are a-priori independent. If C is observed, however, they become dependent as knowledge about either of them might ‘explain away’ the respective other. This phenomenon of inter-causal inference makes determination of independence in BN structures particularly cumbersome.

The concept of *d-separation* defines a couple of criteria that enable to tell whether or not two sets of variables in a BN are conditionally independent by simple graph properties: Two sets of variables X and Y are conditionally independent given a set of observations E iff all (undirected) paths between X and Y are being blocked. A path between X and Y is being blocked by a node z on that path iff one of the following criteria applies:

1. there is a node z on the path with one incoming and one outgoing edge that is observed, i.e. $z \in E$,
2. there is a node z on the path with two outgoing edges that is observed, i.e. $z \in E$,
3. there is a node z on the path with two incoming edges and neither z nor any of its descendants $desc(z)$ are observed, i.e. $z \notin E$ and $desc(z) \cap E = \emptyset$.

The three criteria are depicted in Figure 2.2b. It can be shown that d-separation is equivalent to conditional independence. It remains, however, an unhandy and peculiar way of investigating dependencies in BNs.

Directedness and Acyclicity The directedness of edges conveys that dependencies are unilateral, i.e. one variable influences another variable but not vice versa. However, this is not possible in Bayesian probability theory: Let, w.l.o.g. be $P(X|Y) > P(X)$. Then, according to Bayes’ theorem, we have

$$P(X|Y) > P(X) \Leftrightarrow \frac{P(X, Y)}{P(Y)} > P(X) \Leftrightarrow \frac{P(Y, X)}{P(X)} > P(Y) \Leftrightarrow P(Y|X) > P(Y).$$

Consequently, (in-)dependence among any pair of random variables is always bilateral and the directedness of edges in a BN does not seem to have a natural correspondence in terms of probabilistic dependence.

Correlation and Causality In statistical learning and inductive reasoning, the use of causal models may be misleading and only serve better interpretability. Despite the

presence of many variables whose causations are undoubted, many others are very hard if not impossible to determine: For example, although it is commonly assumed that smoking can be a cause of lung cancer, a correlation between health and mood in humans is less evident: Does improved health lead to improved mood, or does good health lead to good mood? Or do both factors influence each other mutually? Creating a BN requires a knowledge engineer to decide on either direction of causality. If there is no or only little evidence for causality, it is hard to come up with resilient BN structures. In the worst case, unfortunately designed BN structures even leverage misinterpretations of correlations to exhibit causation. From a purely statistical point of view, for instance, it is not possible to tell if obese people tend to consume diet drinks in order to loose weight, or if the consumption of such drinks causes weight gains (“consuming diet drinks leads to obesity”).

The directed nature of knowledge bases encoded as BNs has consequences that make them practically less straightforwardly applicable as they, at first, may have seemed.

2.2.3 Markov Random Fields

As a trade-off between the generality of full joint distributions and strongly restricted BN structures, undirected graphical models are another family of probabilistic formalisms, whose semantics are more straightforward.

Undirected graphical models, also known as Markov random fields (MRFs) or Markov networks (MNs) are an alternative representation of probability distributions over complex structures of random variables. As opposed to BNs, MRFs use *undirected* graph structures for representing the dependency model of the variables. In an MRF, a pair of nodes in the network is connected by an undirected edge if the respective variables are conditionally dependent given all other nodes in the network. The factorization of the joint distribution over the variables $X = \langle X_1, \dots, X_n \rangle$ of the network G for a specific possible world x is defined as the Gibbs distribution

$$P(X = x) = \frac{1}{Z} \prod_{c \in G} \psi_c(x_{\{c\}}), \quad (2.9)$$

where the c s denote all *maximal cliques* in G , ψ_c denotes a *potential function* attached to c , and $x_{\{c\}}$ denotes a projection of the values of all variables in the specific world x that are part of c . The clique potentials ψ_c in an MRF constitute the quantitative representational part of a distribution. A potential maps every state of a clique to a non-negative real-valued measure that multiplicatively contributes to the overall

probability mass of a possible world:

$$\psi_c : X_{\{c\}} \mapsto \mathbb{R}^+$$

As the probability masses of all possible worlds may not sum to 1, it is required to normalize them by the so-called *partition function* Z ,

$$Z = \sum_{x' \in \mathcal{X}} \prod_{c \in G} \psi_c(x'_{\{c\}}),$$

in order to form a proper probability distribution. The perhaps most important observations are firstly, whereas in BNs, the individual factors in the joint distribution are local, normalized conditional distributions, in an MRF, the factors are (not necessarily normalized) clique potentials that contribute in some way to the overall probability of a possible world. Consequently, the numeric values of the potential functions in isolation do not have a direct probabilistic interpretation, but they only gain their probabilistic semantics when the contributions of *all* potentials are taken into account. Secondly, since the partition function sums over the probability masses of all possible worlds, exact inference in MRFs is intractable for all but the smallest problem instances. Therefore, approximate algorithms are typically applied in parameter learning and inference.

Independence As opposed to BNs, independence of sets of variables in MRFs can be determined by simple graph separation instead of d -separation. Two sets of variables X and Y of an MRF G are conditionally independent of each other given a third set of variables Z (with X , Y and Z being pairwise disjoint) if the removal of Z from G would separate X and Y , i.e. the removal of Z would remove any connection between X and Y . Figure 2.3 shows the exemplary probabilistic model from Figure 2.2a as an

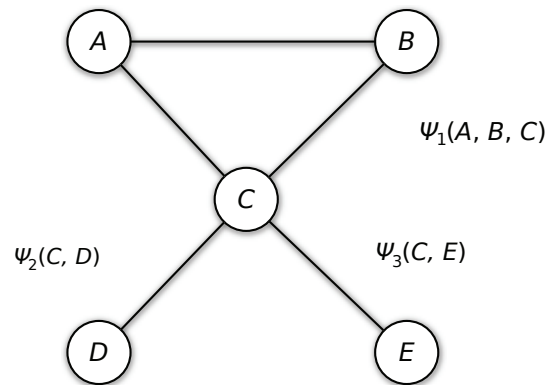


Figure 2.3: Exemplary Markov network with three maximal cliques $\{A, B, C\}$, $\{C, D\}$ and $\{C, E\}$ and their clique potentials ψ_1 , ψ_2 and ψ_3 .

MRF retaining the conditional (in)dependencies among the variables. The MRF comprises the same nodes A, B, C, D , and E . There are three maximal cliques in the network, one ternary connecting A, B , and C , and two binary connecting C and D , and C and E , respectively. Hence, there must be three clique potential functions

$\Psi_1 : A \times B \times C \mapsto \mathbb{R}$, $\Psi_2 : C \times D \mapsto \mathbb{R}$, and $\Psi_3 : C \times E \mapsto \mathbb{R}$. Note that, in order to retain the conditional dependence of A and B given C , there must be an additional edge connecting A and B . Without the edge $\{A, B\}$, A and B would be separated given C and hence would be independent. With the edge $\{A, B\}$, they can still be a priori independent as independence can also be expressed in terms of the potential values.

Log-linearity In practice, it is often beneficial to choose the clique potentials in an MRF such that they take the functional form of an exponentiated weighted sum of binary features, i.e.

$$\psi_c(x_{\{c\}}) = \exp\left(\sum_{i \in \mathcal{X}_{\{c\}}} w_{c,i} f_{c,i}(x_{\{c\}})\right),$$

where $\mathcal{X}_{\{c\}}$ denotes the set of possible configurations of the clique c and $f_{c,i}(x_{\{c\}})$ denotes a Boolean feature function returning 1 iff $x_{\{c\}}$ corresponds to the i -th configuration of the clique c . $w_{c,i}$ denotes a real-valued weight that is attached to the i -th configuration of clique c . If all clique potentials take this log-linear form, the MRF distribution has a couple of particularly appealing representational and computational properties. First, the product over all clique potentials in Equation (2.9) reduces to a sum over all weights attached to the respective clique configurations that hold in a specific possible world x . And second, the gradient with respect to a particular model parameter $w_{c,i}$ can be computed fairly efficiently, as I show below. A slightly more general representation of MRFs is obtained by relaxing the factorization of the joint distribution in (2.9) which is imposed by the topological constraints of the maximal cliques and allowing features to represent aspects of a possible world on a global scope. Such an MRF distribution, also called a *Boltzmann distribution* is given by

$$P(X = x) = \frac{1}{Z} \exp\left(\sum_i w_i f_i(x)\right), \quad (2.10)$$

where Z denotes again the partition function given by

$$Z = \sum_{x' \in \mathcal{X}} \exp\left(\sum_i w_i f_i(x')\right),$$

f_i is a feature function that is 1 iff the respective feature is present in the world x and 0 otherwise, and w_i is a real-valued weight attached to the i -th feature. The representation in (2.10) is both more compact and more general than the original definition of the MRF distribution in (2.9), since it considers features more ‘global’

characteristics of a possible world rather than a ‘local’ configuration of a clique. It also makes the MRF design more convenient as a knowledge engineer does not need to be attentive to the topological structure and the maximal cliques in a network, but can concentrate on the variables and features of the probabilistic KB.

Inference Considering an MRF representing a probability distribution over a KB in propositional logic, a canonical inference problem for computing the posterior probability of an arbitrary sentence ϕ given another sentence ψ is given by

$$P(\phi | \psi) = \frac{\sum_{x'=\phi \wedge \psi} \mu(x')}{\sum_{x=\psi} \mu(x)}. \quad (2.11)$$

$\mu(x)$ is called the *probability mass* function and assigns a positive real-valued measure to every possible world x , which determines x 's portion of the available total mass of probability given by $\sum_{x \in \mathcal{X}} \mu(x)$. It is evident that exact inference according to (2.11) requires enumerating all models of particular logical sentences, which is known to be #P-complete (Roth, 1996) and thus intractable for real-world applications. In practice, approximate algorithms therefore need to be applied. Among those, Markov chain Monte Carlo (MCMC) based methods such as Gibbs sampling (Russell and Norvig, 2003) are perhaps the most popular ones. Alternatively, there are subsets of general MRFs that impose constraints on the structural design of a model that is less expressive but allows for specialized more efficient algorithms. Examples of such MRFs include logistic regression models (Bishop, 2006) or conditional random fields (Lafferty et al., 2001).

Parameter Estimation The parameters w_i of features in an MRF taking the functional form of a log-linear Boltzmann distribution as in (2.10) are particularly elegant to obtain. Considering a set \mathcal{D} of independently drawn, identically distributed fully observed examples, the log-likelihood function is given by

$$\begin{aligned} \mathcal{L} \mathcal{L}(w) &= \log P(w | \mathcal{D}) \propto \log \prod_{d \in \mathcal{D}} P(X = d | w) \\ &= \sum_{d \in \mathcal{D}} \sum_i w_i f_i(d) - \log Z. \end{aligned} \quad (2.12)$$

As maximizing Equation (2.12) does not have a closed form solution, numeric optimization methods, such as Newton or quasi-Newton methods need to be applied in order to find the parameters w that maximize the log-likelihood. To this end, its partial derivative with respect to a particular component w_i of the weight vector w

can be obtained by computing

$$\begin{aligned} \frac{\partial \mathcal{L}(w)}{\partial w_i} &= \frac{\partial}{\partial w_i} \sum_{d \in \mathcal{D}} \left(\sum_i w_i f_i(d) - \frac{\partial}{\partial w_i} \log \left(\sum_{x' \in \mathcal{X}} \exp \left(\sum_i w_i f_i(x') \right) \right) \right) \\ &= \sum_{d \in \mathcal{D}} \left(f_i(d) - \sum_{x' \in \mathcal{X}} f_i(x') P(X = x') \right). \end{aligned} \quad (2.13)$$

The gradient in (2.13) intuitively represents for every feature the difference between the value of the feature in the data and the expectation of the feature with respect to all possible worlds subject to the model parameters. Consequently, if model parameters have been found, such that this difference disappears (equals 0), the prediction of the model for a feature and the actual value of the respective feature in the data coincide, which in turn means that the model is perfectly fitted to the training data. The likelihood is maximal in this particular set of parameters. However, as the computation of the gradient requires inference over the model as a whole, exact learning is intractable for all but the smallest examples.

It is not always necessary to represent a full joint distribution over all variables in an MRF. Instead, in many real-world applications, a knowledge engineer can identify random variables that will never appear in any query but can in any case be observed as evidence. Examples of such models are, for instance, optical character recognition systems, where the input pixels of a document are always observed but are never subject to reasoning. Such kinds of models, which are commonly referred to as *discriminative* models, are typically computationally cheaper than their *generative* counterparts as they require enumeration over fewer possible worlds.



Generative vs. Discriminative Learning

2.3 Probabilistic Relational Models

Probabilistic graphical models, such as Bayesian networks and Markov random fields, provide very elegant means for compactly representing joint probability distributions over large sets of random variables. As opposed to a naive table-based representation of random events, which was hopelessly infeasible, these models carry a graph-based structure that makes independencies among variables explicit and thus allow more compact representations and more efficient computations. In addition, these graphical structures are fairly easily interpretable and thus can be straightforwardly designed by domain experts and knowledge engineers.

However, the number of random variables in PGMs is fixed and must be known beforehand in order to instantiate a respective network. In many practical applications, however, it is desirable for KBs to support varying numbers of random variables. As an example, consider the perception component of a household robot that is to categorize different items that its sensors have identified in the environment (cf. Nyga et al. (2014)). Ideally, such a classification system should support inputs of arbitrary lengths since the number of entities to classify cannot be known at compile time. Their propositional nature thus limits the expressiveness of classical PGMs.

The field of PRMs is a research direction that aims at overcoming the obvious limitations of logics and PGMs by lifting propositional probabilistic models to first-order representations and thereby provide knowledge representation formalisms that are as expressive as FOL, but still able to deal with uncertainty and inconsistency in a meaningful way. It has been a long-standing goal in the AI and machine learning community to combine probability theory and logic in a single, coherent representation and many attempts towards this direction have been published in the recent two decades. An excellent and comprehensive survey of techniques is given by Getoor (2007a). One of the methods that have garnered most attention in recent years is perhaps Markov logic networks (MLNs), a formalism combining the expressiveness of FOL with the probabilistic semantics of MRFs.

2.3.1 Markov Logic Networks

In this section, I introduce the concept of Markov logic networks, following mainly the definitions of Richardson and Domingos (2006) and Jain (2012).

A Markov logic network L consists of an indexed set of pairs $\langle F_i, w_i \rangle$, where F_i is a formula in FOL and w_i is a real-valued weight attached to the i -th formula. Intuitively, a formula's weight determines the 'hardness' or 'correctness' of the respective formula, i.e. the larger the weight of a formula is, the more important the formula is supposed to be; or the larger the penalty for violation of the formula is, respectively. In most implementations of Markov logic, predicate arguments are *typed*, i.e. all arguments of any predicate are bound to a dedicated named set of values, their *domain*. This is in contrast to original FOL, where constant predicate arguments are in one universal set of constant symbols. Given such a finite domain of discourse, a ground Markov random field (ground MRF) can be instantiated by introducing to the MRF one Boolean variable for each ground atom reflecting the truth of the respective proposition. The set of possible worlds \mathcal{X} is in turn given by the set of

all possible combinations of truth assignments to all ground atoms and one possible world x can be considered a vector of Booleans assigning every ground atom in X a truth value. Furthermore, for each ground formula \widehat{F}_j that can be generated from the respective formula F_j , a binary feature function $\widehat{f}_j : \mathcal{X} \rightarrow \{0, 1\}$ is introduced, whose value for $x \in \mathcal{X}$ is 1 if the respective ground formula is satisfied in x , i.e. $x \models \widehat{F}_j$, and 0 otherwise, and whose weight is w_j . The ground MRF specifies a probability distribution over the set of possible worlds \mathcal{X} according to:

$$\begin{aligned} P(X = x) &= \frac{1}{Z} \exp\left(\sum_{j=1}^{|\mathcal{G}|} \widehat{w}_j \widehat{f}_j(x)\right) \\ &= \frac{1}{Z} \exp\left(\sum_{i=1}^{|\mathcal{L}|} w_i n_i(x)\right), \end{aligned} \quad (2.14)$$

where $n_i(x)$ denotes the number of true groundings of a formula F_i in the world x . Z is again a normalization constant ensuring that the distribution sums to 1 given by

$$Z = \sum_{x' \in \mathcal{X}} \exp\left(\sum_{j=1}^{|\mathcal{L}|} w_j n_j(x')\right).$$

Obviously, the probability of a possible world x is proportional to the exponentiated number of true instantiations of all formulae and their respective weights, i.e.

$$P(x) \propto \exp\left(\sum_{j=1}^{|\mathcal{G}|} \widehat{w}_j \widehat{f}_j(x)\right)$$

Intuitively, a possible world x thus becomes more probable the more true groundings it has in x and the larger its weight is, whereas formulae with close-to-zero weights only contribute little to the overall probability mass available in the universe. In contrast, if a formula has a negative weight, more true groundings turn into a penalty in terms of probability mass, such that the respective world becomes less probable.

Example As a minimal example MLN, we consider a scenario in which a probabilistic KB about students, their grades and their studying habits are to be modeled. To this end, we introduce to the MLN the five predicates with the respective semantics:

student(person)
diligent(person)
friends(person, person)
grade(person, grades?)

grades = {A,B,C,D,E,F}

The predicates **student** and **diligent** assign the respective attributes to elements of the person domain. **friends** specifies that two persons are friends, and **grade** assigns a grade to a person. The possible grades are defined in the last line and range from A to F. The question mark in the predicate declaration behind the grades argument of the **grade** predicate specifies that to every person in the domain of discourse, *at most one* of the grades can be assigned a person. Using these predicates, statements in FOL can be formulated and attached a real-valued weight determining its ‘correctness’. Deterministic logical statements (i.e. hard constraints) can be made by omitting the weight and putting a period after the formula.

For example, one can state that diligent students typically get very good grades:

1.2 **diligent**(x) ^ **student**(x) => **grade**(x, A)

Companioned students tend to have similar studying habits:

0.7 **friends**(x, y) <=> **diligent**(x) ^ **diligent**(y)

Most students are lazy:

-0.4 **student**(x) => **diligent**(y)

Only students are being graded:

!student(x) => **!grade**(x, g).

For any set of persons, e.g. persons={Ann,Bob,Cecile}, this MLN can be queried for any aspect of the domain given any other aspect as evidence. For example, one can query for the probability that Ann gets an A given that she is friends with Bob, who is diligent, $P(\mathbf{grade}(Ann,A) \mid \mathbf{friends}(Ann,Bob) \wedge \mathbf{diligent}(Bob))$, or the probability that Cecile is friends with Bob given that she is lazy, $P(\mathbf{friends}(Cecile,Bob) \mid \mathbf{!diligent}(Cecile))$.

This conceptual simplicity makes Markov logic one of the most general and most powerful yet intuitive representation languages for uncertain knowledge. In the following, I will briefly revisit the most important reasoning approaches.

Exact Inference In an MLN, the posterior of any arbitrary ground formula can be computed according to Equations (2.11) and (2.14). However, such an inference problem is impossible to solve exactly in the general case as computation of the partition function Z requires the enumeration of all possible worlds \mathcal{X} . It is therefore required to draw on approximate inference methods.

Algorithm 1 MCMC

Input: P_{trans} : the transition probability of a Markov chain
 Q : a query
 E : observations
 m : maximum number of steps

Output: an approximation of $P(Q|E)$

- 1: Initialize $x^{(0)}$, such that $P(x^{(0)}) > 0$ and $x^{(0)} \models E$ ▷ Initial world state
- 2: $i \leftarrow 0, c \leftarrow 0$
- 3: **while not converged and** $i \leq m$ **do**
- 4: Randomly sample $x^{(i+1)}$ from $P_{trans}(x^{(i+1)} | x^{(i)})$, where $x^{(i+1)} \models E$
- 5: **if** $x^{(i+1)} \models Q$ **then**
- 6: $c \leftarrow c + 1$
- 7: **end if**
- 8: **end while**
- 9: **return** $P(Q|E) \approx c/e$

Approximate Inference Among the approximate inference techniques, MCMC (Gilks et al., 1995) methods are the most popular ones. The basic idea behind MCMC is to construct a Markov chain whose stationary distribution corresponds to the probability $P(X|E)$. The desired posterior $P(Q|E)$ is approximated by drawing samples from the Markov chain and counting the frequencies of Q being true in the samples drawn.

The probably most basic MCMC algorithm is *Gibbs sampling*, where the states of the Markov chain are given by the possible states of the ground MRF of the MLN subject to the evidence. The algorithm starts with a random assignment of the variables that is consistent with the evidence. Then the algorithm randomly selects one of the query variables $X_i \in Q$ and draws a sample of the ground MRF state of the next iteration. The probability of a transition from a state $x = \langle x_i, \bar{x}_i \rangle$ to another state $x' = \langle x'_i, \bar{x}_i \rangle$ is given by $P_{trans}(X_i | \bar{X}_i = \bar{x}_i)$, where $\bar{X}_i(\bar{x}_i)$ denotes all variables (values) in the ground MRF except for $X_i(x_i)$. Whenever a sample entails the query Q , a counter c is incremented. The approximate posterior is returned as the relative frequencies of c and the total number of samples.

Gibbs sampling provably converges to the desired stationary distribution given the Markov chain is ergodic, i.e. when every state of the chain is reachable from any other state and there are no absorbing states. This constraint, however, does not always hold in MLNs as there is the possibility of hard (purely Boolean) formulae. These formulae typically are assigned comparably large weights, which in turn stops the Markov chain being ergodic and consequently voids the convergence guarantee.

MC-SAT is a variant of MCMC that is tailored to the specifics of Markov logic taking into account deterministic constraints and more sophisticated sampling techniques, which can significantly speed up the convergence of the Markov chain (Poon and Domingos, 2006).

Most Probable Explanation Computing the MPE state of the ground MRF of an MLN corresponds to solving a weighted maximum satisfiability problem (weighted MAX-SAT), where the goal is to find a complete assignment to a set of Boolean variables (in this case ground atoms), such that the total weight of a set of weighted clauses (the ground formulae) is maximized. The MAXWALKSAT algorithm is a variant of local stochastic search that randomly flips truth values of ground atoms in a simulated annealing fashion (Kautz et al., 1996). Due to its stochastic nature, however, it is not guaranteed to always find the globally best solution.

A different approach is to transform an MLN into a weighted constraint satisfaction problem (WCSP) as described by Jain et al. (2009a) and apply state-of-the-art solvers that compute exact solutions to even big problems fairly efficiently by branch-and-bound search (Clausen, 1999). An example of such a solver, which is also used in this work, is *toulbar2* (Allouche et al., 2010).

Parameter Learning Given a set of logical formulae and a database \hat{x} for training, the problem of learning the model parameters, i.e. weights for each formula is straightforward. The log likelihood is given by

$$\begin{aligned} \mathcal{LL}(w | \hat{x}) &= \log \left(\frac{1}{Z} \exp \left(\sum_i w_i n_i(\hat{x}) \right) \right) = \sum_i w_i n_i(\hat{x}) - \log Z \\ &= \sum_i w_i \cdot n_i(\hat{x}) - \log \left(\sum_{x' \in \mathcal{X}} \exp \left(\sum_k w_k n_k(x') \right) \right), \end{aligned} \quad (2.15)$$

and the gradient of the i -th weight component is

$$\begin{aligned} \frac{\partial \mathcal{LL}(w | \hat{x})}{\partial w_i} &= n_i(\hat{x}) - \sum_{x' \in \mathcal{X}} n_i(x') \cdot \frac{\exp(\sum_k w_k n_k(x'))}{\sum_{x'' \in \mathcal{X}} \exp(\sum_k w_k n_k(x''))} \\ &= n_i(\hat{x}) - \sum_{x' \in \mathcal{X}} n_i(x') \cdot P(X = x'). \end{aligned} \quad (2.16)$$

Equation (2.16) can be maximized by standard gradient-based optimization techniques such as Newton's method or quasi-Newton methods like BFGS (Broyden, 1970). In practice, however, the learning of joint probability distributions from observed training data entails severe computational challenges as exact parameter learning requires inference over the model as a whole in every iteration of the numerical optimization. This makes exact learning via maximum likelihood impossible for all but the smallest problem instances.

As a consequence, a variety of simplifications have been proposed to leverage the

computational intractability of learning in MLNs. Among them two families of approaches for approximating the exact likelihood have received particular attention in the research literature:

1. *Approximation of the gradient of the likelihood*: This approach leaves the objective function to be optimized during learning the exact likelihood, but uses approximations of its gradient to avoid heavy inference over the model, i.e. computing the expectation over possible worlds. Examples for such algorithms use for instance the MPE state of the current model ('voted perceptron') or contrastive divergence in combination with MC-SAT.
2. *Approximation of the likelihood*: The likelihood function itself is approximated by a different function, which typically makes strong independence assumptions among the variables in the models. The best-known algorithm of this kind is probably pseudo-likelihood learning, which assumes independence among ground atoms in the ground MRF and imposes conditions on their Markov blanket.

Lowd and Domingos (2007) provide an excellent overview of learning algorithms for MLNs that fall into the first category. Among those, the *voted perceptron* and *contrastive divergence* algorithm have been deemed most superior. A drawback that all of those methods have in common is that they rely on a good estimate of the gradient in each optimization step, which is, in the general case, infeasible. They mostly use MC-SAT for performing approximate inference over the model, which is an MCMC slice-sampling method dedicated for MLNs. While MCMC methods can be shown to converge to the exact MLE estimate given sufficiently large numbers of samples they often require too much computational resources to reach their stationary distributions.

Pseudo-likelihood Learning Hence, for practical applications, most researchers apply the pseudo-likelihood (Besag, 1975), which uses an approximation of the objective function itself. Pseudo-likelihood learning is computationally much cheaper than any likelihood method because it conditions every atom in the ground MRF on its Markov blanket in x :

$$\begin{aligned} \mathcal{P}\mathcal{L}\mathcal{L}(x) &= \log \prod_{k=1}^{|\mathcal{X}|} P(x_k | MB_x(X_k)) \\ &= \sum_{k=1}^{|\mathcal{X}|} -\log \left(1 + \exp \left(\sum_{i \in F_{X_k}} w_i (\bar{n}_{i,k}(x) - n_i(x)) \right) \right), \end{aligned} \quad (2.17)$$

where $\bar{n}_{i,k}(x)$ denotes the number of true groundings of the i -th formula in a modification of x where the k -th ground atom, X_k , is inverted in its truth value and $MB_x(X_k)$ denotes the Markov blanket of the ground atom X_k in x . Its gradient with respect to w_i is given by

$$\frac{\partial \mathcal{P}\mathcal{L}\mathcal{L}}{\partial w_i} = \sum_{k=1}^{|X|} (n_i(x) - P(x_k | MB_x(X_k)) \cdot n_i(x) - P(X_k = \neg x_k | MB_x(X_k)) \cdot \bar{n}_{i,k}(x)). \quad (2.18)$$

In contrast to exact parameter learning in (2.15), the computational effort of maximizing the pseudo-likelihood scales linearly instead of exponentially in the number of atoms in the ground MRF. However, the very strong independence assumptions can lead to very imprecise weight estimations. As the purpose of formulae in MRFs is often to capture the interaction between their subformulae the simplification is often too strong for the learned probabilities to be accurate.

Caveats MLNs have garnered a lot of attention in recent years. One reason for this success is undoubtedly their conceptual simplicity, generality, and straightforwardness in design. However, although they have been intensively studied since their original inception, there are various subtleties and pitfalls that make their practical application of more difficult as it may, at first, appear. The perhaps greatest difficulty is related to the semantics of the logical implication in a probabilistic setting, which might produce unexpected if not unwanted results in the posterior probabilities. According to Jain (2011), the natural correspondence to a causal relation of the form $foo(x) \rightarrow bar(x)$ in a probabilistic setting is a conditional probability $P(bar(x) | foo(x))$ as when observing $foo(x)$, one can expect to recover $bar(x)$ only with a certain probability. The conditional, however, in an MLN, cannot be established properly by simply attaching a weight to the formula $foo(x) \rightarrow bar(x)$ in order to make it a soft constraint. It rather must be constructed by a set of mutually exclusive formulae that establish a ratio of probability masses of the sentences $foo(x) \wedge bar(x)$ and $\neg foo(x) \wedge bar(x)$. Jain refers to this phenomenon (among others) as the “probabilistic implication fallacy,” and strongly discourages the use of implications when designing MLNs. Jain instead advocates the explicit representation of individual conditional cases in an MLN.



Probabilistic
Implication
Fallacy

Implementational Aspects Making structural commitments to the shape of logical formulae has further computational advantages. As Equation (2.14), (2.15) and (2.16) show, learning and inference in MLNs require counting the number of true groundings of the logical formulae in an MLN, which is known to be a #P-complete

problem in the general case (Roth, 1996). However, it is possible to identify special cases of formulae, for which model counting can be computed significantly faster than enumerating all interpretations of a sentence. As an example, consider a conjunction F , whose constituents l all are literals. For any possible world x , the number of true instantiations of F can be computed fairly efficiently since any of its instantiations \widehat{F} can only be true for precisely one truth assignment, namely the one in which all conjuncts are true. The PRACMLN learning and reasoning engine, which has been developed in context of this thesis, implements a several of such computational optimizations by exploiting frequently recurring patterns in models.

Syntactic Sugar Sometimes it is desirable to impose *functional constraints* on the set of possible worlds. A functional constraint requires that out of several ground atoms exactly one must always be true at a time and all others in turn must be false. Such constraints are called functional constraints since the value of one argument is functionally determined by the values of the other arguments. In most of the current MLN implementations, functional constraints are supported and can be specified by appending an exclamation mark (!) to the declaration of the functionally determined argument. Any possible world that violates such a constraint is automatically assigned 0 probability. Apart from that it is reasonable to use functional constraints from a modeling point of view in many cases, it is typically also computationally beneficial since functional constraints result in a partial linearization of the computational problem. In the PRACMLN implementation that I have developed in this work, there is a second type of functional constraints, so-called *soft functional-constraints*, which require maximally one ground atom out of the set of mutually exclusive ground atoms to be true instead of precisely one. This is a convenient language element in MLNs and quite useful for classification problems, for instance, in order to let the probabilistic model decline to make a decision in case of insufficient confidence, but still exploit the computational appeal of functional constraints. Soft-functional constraints are specified by appending a question mark (?) to the respective predicate argument.

2.4 Uncertainty versus Vagueness

Besides uncertainty, the concept of *vagueness* is another key concept I use in my work. Both terms refer to settings in which propositions cannot be assigned a strictly Boolean truth value, but the application domain requires finer grained specifications of beliefs

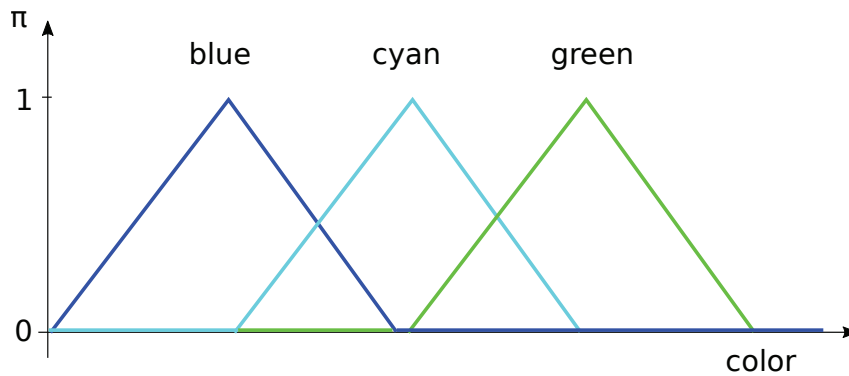



Figure 2.4: Illustration of the fuzzy membership functions π_{blue} , π_{cyan} and π_{green}

in particular propositions. Although closely related, uncertainty and vagueness fundamentally differ in their semantics and thus it is crucial to not confuse the two.

Uncertainty generally refers to “a situation which involves imperfect and/or unknown information” (Wikipedia, 2017), about specific matters in the world. A probability P of an event φ , $P(\varphi) = 0.75$, for instance, expresses the belief in φ to be true or false. It does not mean that φ is true to 75%. This degree of belief semantics thus makes the same ontological commitment as pure logic does: If one could observe φ directly, it would turn out to be *either* entirely true *or* entirely false, and in an uncertain situation an agent cannot be sure which one applies.

In contrast, the concept of *vagueness* or *fuzziness* refers to propositions whose ‘real’ truth values are not necessarily strictly Boolean but allow different shades of truth. Vagueness is a phenomenon ubiquitous in linguistics as many natural-language (NL) terms cannot be clearly demarcated from another. As an example, consider terms referring to color names, such as ‘blue’ and ‘green’. Although the two can be considered mutually exclusive colors, ‘cyan’ could be considered blue and green *to a certain degree*. Even if the precise color value could be observed, the imprecision in terminology cannot be eliminated.


Fuzzy
Logic

A well known mathematical apparatus to deal with vague propositions is fuzzy logic (FL), a multi-valued extension of propositional logic (PL). FL has its foundations in the theory of fuzzy sets, in which elements belong to a set only to a certain degree. Formally, a fuzzy subset x of a set X is a pair $\langle X, \pi_x \rangle$, where X is called the *universe* and $\pi_x : S \mapsto [0, 1]$ determines the degree to which a particular element actually belongs to x , which is called the *membership function*. In FL, the universe X is given by the set of atomic propositions and π_x is a fuzzy interpretation of X assigning every proposition in X a real-valued degree of truth. Its calculus is analogous to the calculus of PL: If A and B are propositions in FL, then the logical connectors with respect to x are defined as

$$A \wedge B := \min(\pi(A), \pi(B))$$

$$A \vee B := \max(\pi(A), \pi(B))$$

$$\neg A := 1 - \pi(A).$$

Note that the multi-valued logical calculus of FL reduces to the Boolean counterpart of PL in the extreme cases where all propositions have binary truth values.

2.4.1 Summary

In this chapter, I have laid out the technical foundations of the formalisms and techniques used in this thesis. I have reviewed the most basic elements of the areas of knowledge representation and reasoning in terms of logical representation languages. Propositional logic (PL) is a very basic formalism with only very restricted expressiveness. First-order logic (FOL) is perhaps closest to natural languages in terms of expressiveness, but the undecidability of entailment in FOL impairs its practical applicability to real-world problems. Decidable fragments of FOL exist, such as description logic (DL) and Prolog-related languages, but their deterministic semantics impedes the treatment of uncertainty and inconsistency, which are omnipresent phenomena in real-world applications.

Probabilistic graphical models (PGMs) like Bayesian networks and Markov random fields (MRFs) are formalisms that compactly represent joint probability distributions over variables and are designed to capture the uncertain and stochastic aspects of the real-world. However, their propositional nature imposes a static structure on the dependency relations constraining their applicability to domains of only fixed size. Probabilistic relational models (PRMs) relax those assumptions by combining FOL with PGM to elicit the best from two worlds: the expressiveness of first-order logical KBs and the capability of PGMs to deal with uncertainty.

Markov logic is a popular PRM language, in which logical formulae have attached real-valued weights determining the strength of the respective formula. The probabilistic semantics is thereby defined over all instantiations of possible propositions in a ground Markov random field. Most challenging in the application of MLNs is the representational and computational expense for learning and reasoning. It is therefore crucial to design models that do strongly generalize across specific problem instances to keep the implied costs as little as possible.

In the remainder of this work, I will investigate and present PRMs, in particular variants of MLNs that combine the concepts of fuzziness, uncertainty and ontological engineering in description logic, as a representation for vague instructions and their refinements, which enables to infer the most likely executable specializations thereof.

Chapter **three**

Probabilistic Knowledge Bases for Instruction Interpretation

In this chapter I will address the problem of action-specific knowledge processing, representation and acquisition for autonomous robots performing everyday activities. I will discuss the important role of action-centric knowledge representation and reasoning in human everyday problem solving and demarcate it from classical AI-driven approaches towards action intelligence. I will report on a data-driven analysis of the household domain, which has been performed on a large corpus of NL instructions from the Web and which underlines the supreme need of action-specific knowledge for robots acting in human environments. Based on this study, I will hypothesize and give estimates for the amount of knowledge that robots might need for the competent performance of envisioned robot jobs such as preparing meals. These estimates will be obtained by the application of data mining techniques to websites that provide written instructions for performing everyday activities intended for human use. I will introduce the concept of Probabilistic Action Cores (PRAC) and describe in detail the PRAC framework, a natural-language interpreter for transforming vague and incomplete NL instructions into executable robot plans by means of probabilistic first-order reasoning. I will address how such knowledge bases can be represented, acquired and used for inferring the most probable executable action plan and the problems of ambiguity, incompleteness and learning from sparse data can be tackled. The work presented in this chapter is based on prior works published in parts by Nyga and Beetz (2012, 2015b); Nyga et al. (2017b) and Lisca et al. (2015).

3.1 Probabilistic Action Cores

Robotic agents that are to be instructed in natural language need to be equipped with capabilities for understanding human task specifications. In this work, I present a novel approach towards instruction interpretation from the perspective of Bayesian cognition, formulating the problem of NL understanding as a reasoning task in probabilistic relational KBs. In this section, I give an informal definition of PRAC and demarcate it from alternative approaches.

3.1.1 Definition

In a nutshell, the basic idea of PRAC is to learn joint probability distributions over semantic networks like the ones shown in Figure 3.1, in order to (1) determine a specific semantic representation from a vague NL statement and (2) be able to infer missing aspects of an action specification from the given aspects.

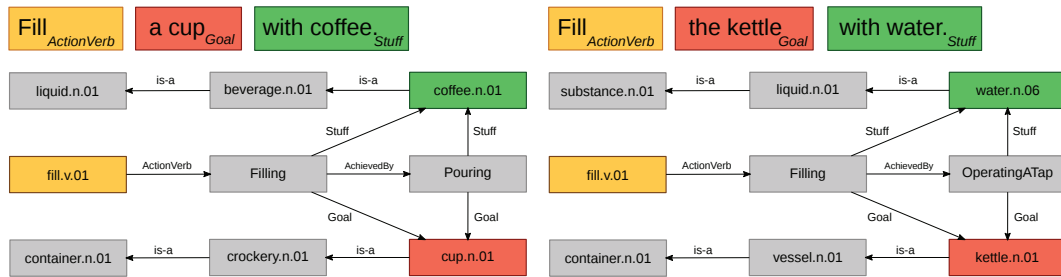
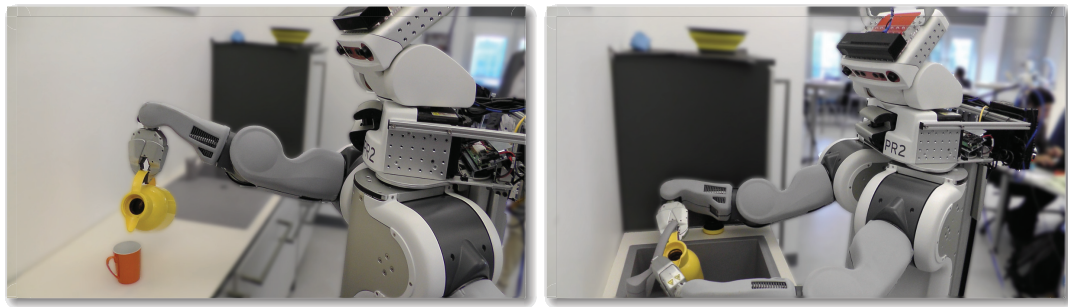
For modeling these relations, probabilistic graphical models (PGMs) are built for each relevant action verb in a specific domain. These models are inspired by semantic network (Simmons, 1963) and frame semantics (Minsky, 1974) but are intended to explicitly take into account uncertainty, vagueness and incompleteness by means of probabilistic interrelations. These graphical models are called *action cores* and can be considered generic, frequently recurring patterns of events in everyday activity. An action core is a generalized concept of an action and has attached a set of relations assigning semantics to all entities that are required to effectively *perform* the action. These relational structures of the action are called *action roles* and can be thought of as abstract, symbolic parameters of the action. The *probabilistic action core* is a probabilistic model over the relational structure of the action core and its arguments. The relational structure thereby needs to be designed by a knowledge engineer who has to decide on the set of relations and actions, their semantics and level of abstraction. Having such a relational framework the probability distributions can be learned from specific instances of action cores, which in turn generalize the knowledge to more generic action patterns that can be transferred to new situations. In this chapter, I present data structures, probabilistic models, and algorithms for representing, acquiring, storing, and using action cores for tackling the problems of under-specification and ambiguity in the interpretation of natural language (NL) instructions.

3.1.2 Alternative Approaches

Previous works on the topic of instruction understanding reported in literature, which have received particular attention, can be roughly divided into two categories, which differ in the methodologies for tackling the problem of NL understanding for robots. Approaches that fall into one category are strongly driven by the linguistics community and mainly focus on formal language analyses of written texts, investigating its building blocks and units as well as how it is structurally organized. Typical reasoning tasks in this field involve the syntactic structural analysis of sentences (e.g. parsing), the aggregation of single words into larger semantic units (e.g. chunking, named-entity recognition, compound-noun recognition) or assigning parts of speech to words (part-of-speech tagging). The syntactic structure of NL commands is then mapped into the space of robot control procedures, where mainly static, shallow mappings from action verbs to control routines are applied. The fundamental assumption underlying these approaches is that the syntactic representation of a sentence reveals sufficient information for *performing* a respective task. Typical application domains reported in literature cover robot navigation tasks in mazes or office environments. However, I argue that, for the execution of more complex, human-scale manipulation activities, such shallow mappings from NL patterns to robot control programs are too restricted in their scope to enable robust, flexible autonomous decision making in robots. As an example, consider the two instructions “fill the kettle with water” and “fill a cup with coffee.” Figure 3.1 illustrates the semantic networks of the two instructions and a PR2 robot performing them in a kitchen environment. The syntactic structures of these two instructions retrieved from the Stanford parser (Klein and Manning, 2003) are

<pre>(ROOT (S (VP (VB Fill) (NP (DT a) (NN cup)) (PP (IN with) (NP (NN coffee))))))</pre>	<pre>(ROOT (S (VP (VB Fill) (NP (DT the) (NN kettle)) (PP (IN with) (NP (NN water))))))</pre>
---	---

Comparing the two syntactic parses and the two semantic representations in Figures 3.1a and 3.1b, one can see that the two instructions are isomorphic both syntactically and semantically, i.e. their syntactic and semantic relational structures are identical except for their symbolic names. Consequently, they are indistinguishable with respect to their linguistic structure. However, considering their *execution*, the two instructions require fundamentally different manipulation skills: Filling a cup with coffee typically can be done by pouring from a coffee pot into a drinking mug, whereas filling a water kettle is normally achieved using the tap. Depending on the



(a) Top: the PR2 pouring coffee from a pitcher. (b) Top: the PR2 operating a tap. Bottom: Bottom: semantic structure of the instruction “fill a cup with coffee.” semantic structure of the instruction “fill the kettle with water.”

Figure 3.1: The PR2 robot performing two filling activities. Although the two instructions “fill a cup with coffee” and “fill the kettle with water” are syntactically and semantically isomorphic, they require fundamentally different manipulation skills.

current situation, environment and context, there might be even more possibilities of how to perform the tasks. As a consequence, understanding NL instructions of everyday manipulation tasks requires *appropriate interpretation and completion* as pure linguistic representations do not necessarily carry sufficient information for proficient execution by a robot. This goes beyond traditional machine reading and question answering – typical applications of linguistic analyses: In machine reading and question answering, ambiguity and incompleteness can often be retained since queries and the respective answers typically reside on a similar level of abstraction. If it comes to instruction execution, however, those ambiguities need to be resolved in every detail and missing information needs to be added (Beetz et al., 2015b). Thus, as Bergen et al. (2003) and Feldman and Narayanan (2004) point out, the ability to understand an instruction is tightly coupled to the capability of actually performing it.

A second class of approaches has its foundations in the AI community and formulates the problem of instruction interpretation as a classical planning problem. Approaches from this category assume that the NL commands describe goal states of the world which is to be manipulated in a way that the desired state is established. To this end,

the NL statements are transformed into a formal, semantic, logic-based representation in order to derive the goal state from an instruction. Classical methods in AI-based planning are then applied to automatically generate sequences of actions from such goal specifications, which are, in essence, based on the ideas of classical STRIPS planning (Fikes and Nilsson, 1971). Planning about action sequences and reasoning about actions, their effects and goals, has been widely studied by both the AI and the robotics community. However, for the actions are regarded as black boxes, the plans generated by such systems are often too abstract for competent execution by robots since they merely contain what is *given* but not what is *necessary*. In addition, these approaches towards action intelligence assume complete knowledge about actions and the world. They make what is commonly referred to as the *closed-world assumption*. However, service robots in human environments will have to act in open worlds, in which they will be faced with high uncertainty, ambiguity and under-specification. In such real-world scenarios, the knowledge about actions often is incomplete though indispensable (Moore, 1984).



Closed
vs. Open Worlds

Humans understand and perform such instructions with great ease, if not unconsciously, not requiring intensive planning or deliberation. They share a large amount of common background knowledge about actions and objects, which appears self-evident to them, rendering explicit statements of certain facts redundant and obsolete. As a consequence, NL action descriptions are ...

- ... *incomplete*. This originates from the common human knowledge about actions, which is so obvious that no explicit statements are required. In descriptions of activities written by humans, such as cooking recipes, vague instructions like “stir occasionally” or “serve” can be found frequently. These instructions are characterized by extreme under-specification since they lack any information about the objects involved. Humans, however, are yet very successful in correctly performing such tasks. The current context and background knowledge seem sufficient for deriving the necessary information.
- ... *ambiguous*. As an example, consider the word ‘cup’ and its use in the instructions “fill a cup with milk” and “add a cup of milk.” In the first instruction, ‘cup’ refers to a physical container that is supposed to be filled with milk. Conversely, in the second case, ‘cup’ refers to an abstract unit of measure, a specification of how much is supposed to be added to something, while the actual object to be added is still the milk and not the cup. Although this distinction may seem subtle at first, it is yet crucial to distinguish between the two meanings in order to manipulate the right objects in the right way. Common-sense knowledge, in this case, serves as a constraining mechanism that rules out inappropriate com-

binations and renders disambiguation a constraint satisfaction (or optimization) problem.

Researchers have found evidence that everyday activities are too different to playing chess or solving crypt-arithmetic puzzles – typical applications of classical, automated planning techniques. Instead, understanding NL is rather knowledge-intensive than planning-intensive and robots must be equipped with a large body of knowledge about actions, objects, and environments, and with reasoning mechanisms allowing to infer the information that is needed in order to perform an activity competently. The most convincing arguments are given by Anderson (1995), which I will summarize in the next section.

I argue that action-specific knowledge is key in dealing with complex, human-scale everyday tasks rather than classical planning and I present a framework for modeling probabilistic dependencies between actions, objects and ontological information about the world.

3.2 Everyday Rationality

According to Anderson (1995), everyday activities are characterized by mainly three qualities: First, an everyday activity typically is a complex task that is *mundane* or routine to humans. That is, everyday activities do not put forward big surprises while performing them, but primarily bring about unbiased expected results. Second, they are common and occur *frequently*. An activity thus becomes an *everyday* activity when an agent has performed it many times and thereby has acquired a great deal of knowledge about the activity, which leads to the strong expectations about its outcomes and a broad coverage of possible ramifications. And third, humans performing everyday activity typically aim for adequate or satisfactory performance rather than optimality and expert performance. They ‘satisfice’ rather than ‘optimize’ their decision making.

In classical AI, the question of how to perform a particular activity has been mainly reduced to the question of how to achieve a specific goal, i.e. finding a sequence of actions that an agent needs to take in order to establish a world state that entails the given goals. Classical planning applies general problem solvers and theorem provers, as presented by Fikes and Nilsson (1971) and Newell et al. (1972), which have been applied to logical reasoning, theorem proving, playing chess and solving for

crypt-arithmetic puzzles. However, McDermott (1992) argues that planning problems are both too simple and too complex at the same time: the computational problems that planning imposes are intractable on the one hand, but the representational black-box view on actions is too great a simplification to adequately represent a real robot's control program. Consequently, as Beetz et al. (2015b) find, the impact of such planning systems on real-world robotic platforms still remains rather limited. An excellent elaboration can also be found in Anderson (1995), who nicely puts it:

"The types of activities we commonly call everyday activities are of a fundamentally different character than this. Everyday life is not a crypt-arithmetic problem. An activity such as washing dishes seems, on the whole, to have little to do with the reasoning mechanisms used to play chess. Traveling from home to work has few commonalities with logic problems."
— (Anderson, 1995)

Firmly anchored in the science of AI is the term 'rationality,' an attempt to objectivize the notion of *intelligence* in form of an idealized, abstract concept (Russell and Norvig, 2003). There are, however, different perspectives on rationality, where rational behavior in the sense of "doing the right thing" is perhaps the one that has gained the broadest acceptance. A decision-theoretic account of the "right thing" can be given by the actions that maximize some kind of benefit, utility, probability or comfort, or minimize entropy or costs of execution, subject to the given resources (Simon, 1957). In terms of action planning, the concept of rationality simply refers to a selection of actions that attain an agent's goals (Newell, 1982). With regard to every activity, however,



Rationality

"Rationality does not mean performing the 'logical' action in the formal sense of the word; it means doing something that is compatible with the agent's past experience, future intentions, and knowledge of the situation in which the agent finds itself. This form of rationality is very different from the decision-theoretic perspective, but I argue keeps much of the spirit decision-theoretic rationality encompasses."
— (Anderson, 1995)

While still being compatible with the premise of "doing the right thing," Anderson's definition of rationality makes a slight relaxation to the classical decision-theoretic accounts to rationality. In his definition, most obviously, rational decisions are not necessarily *globally objectively* rational, but rather rational *relatively* to the agent and what the agent knows. I argue that this relaxation is not only elegant but even necessary when implementing agents performing everyday activities. As an example, consider the activity of making a pancake. Given this goal, a globally objectively

rational agent is to find a sequence of actions to conduct, which attains this goal and provably produce a pancake. The agent in turn must have a fully axiomatized set of pre- and post-conditions of actions and food-chemical processes, as well as – perhaps most difficult – a formal specification of what the goal state, i.e. a pancake, is. Creating a globally consistent knowledge base that entails such an action sequence that provably creates a pancake does not appear feasible. As a consequence, the global consistency and completeness postulated by classical planning methods and entirely decision-theoretic definitions of rationality do not seem to translate well into the domain of everyday activity. Rather, as Minsky stated it:

“[...] I do not believe that consistency is necessary or even desirable in a developing intelligent system. No one is ever completely consistent. What is important is how one handles paradox or conflict, how one learns from mistakes, how one turns aside from suspected inconsistencies. [...] “Logical” reasoning is not flexible enough to serve as a basis for thinking; I prefer to think of it as a collection of heuristic methods, effective only when applied to starkly simplified schematic plans. The Consistency that Logic demands is not otherwise usually available – and probably not even desirable – because consistent systems are likely to be too weak. I doubt the feasibility of representing ordinary knowledge effectively in the form of many small, independently “true” propositions.” – (Minsky, 1974)

Following these considerations, everyday activities are routine in a sense that they occur frequently and, as a consequence, an agent must be well-experienced in performing them. The familiarity with these activities hence results in a large amount of knowledge about *how* a particular activity is to be performed in a specific context, going along with the awareness about objects involved in these actions as well as their inter- and intra-relations. This makes executing everyday activities a very knowledge-intensive task rather than intensive to planning.

Consequently, knowledge about actions and objects can serve as a source of constraints, i.e. an agent who is to perform a particular action must act subject to the constraints given by its knowledge about this action. Constraining the space of possible action configurations by knowledge does not mean to restrict the physical or mental capabilities of an agent, but to guide the process of decision making. Considering only alternatives that have worked before rather rules out alternatives that are inappropriate or irrational in a particular context and hence makes repeated reasoning about choices obsolete.

Knowledge consequently allows an agent to adjust action parameters such as the object acted on, the utensil or tool used to manipulate the object, the direction and destination of an action etc. In other words, knowledge is key to perform the



more knowledge
= less search

appropriate action on the *appropriate* objects in an *appropriate* way. Appropriateness here does *not* mean optimality because an optimal solution often cannot be found in real-world scenarios or it would consume excessive time to compute optimal solutions. Nonetheless, the agent has to act and thus “accepts ‘good enough’ alternatives, not because he prefers less to more but because he has no choice.” (Simon, 1956) This ‘satisficing’ performance criterion in decision making is also referred to as *rational boundedness* which is imposed by cognitive limitations of an agent ultimately requires the agent to balance utility and costs of deliberation effort. Researchers have also systematically incorporated computation time as first-order items in reasoning algorithms in terms of expected utility of the deliberation effort in time-critical applications (Gershman et al., 2015).

To summarize these considerations, I argue that providing robotic agents with comprehensive knowledge about actions and objects is key in pushing robots’ cognitive skills and autonomy to more advanced levels. As a valuable complementary extension to classical planning techniques in AI, the use of NL instructions might be a promising approach. In order to account for the ambiguity and under-specificity in NL, I advocate the use of probabilistic relational models.

3.3 How much Knowledge does a Robot Need?

In studying the ways how humans explain complex activities to other humans, one can observe that the language humans tend to use is extremely under-specified and vague. As an example, consider the following recipe for making pancakes:

How to make Pancakes

1. Pour milk into a bowl.
2. Add 3 eggs.
3. Add flour and mix well.
4. Heat the greased pan.
5. Pour the batter into the pan, then wait for 2 minutes.
6. Push the spatula under the pancake and flip it.
7. Wait for another 2 minutes.
8. Place the pancake on a plate.
9. Serve.

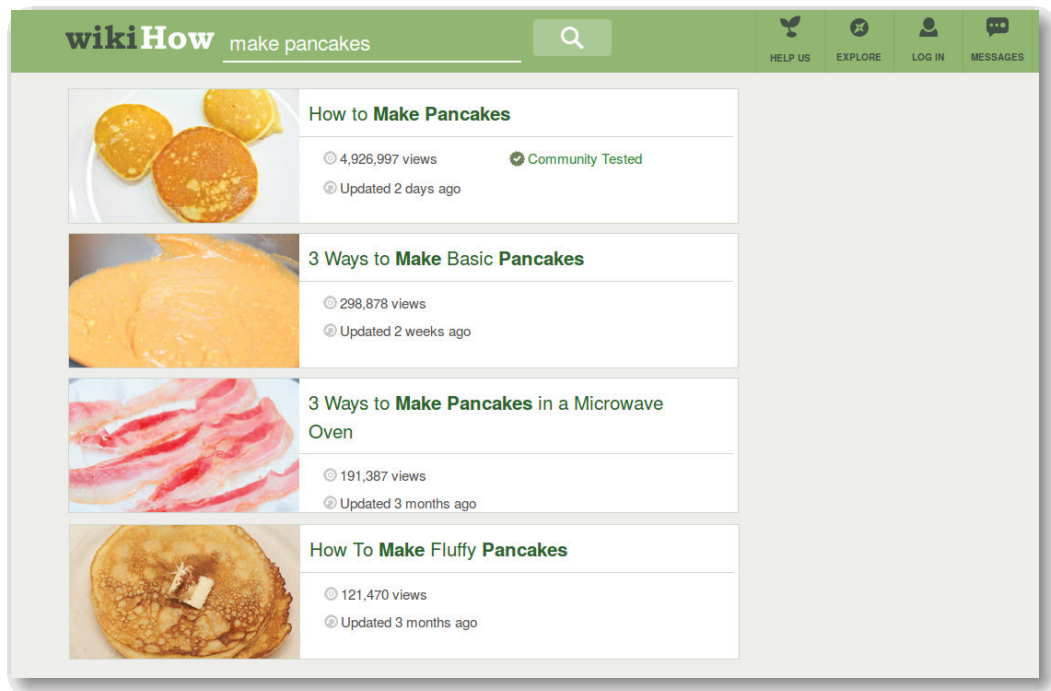


Figure 3.2: Screenshot of the *wikihow.com* website showing the result of a search for “make pancakes.” The results do not only contain basic pancake recipes, but a large amount of variants e.g. blueberry or vegan pancakes.

When formulating such directives, humans tend to omit important information, which is necessary for performing a particular action. Thus, only understanding what is explicitly *specified* by an instruction is insufficient for performing an action proficiently and successfully. The example shows that NL action specifications written by and addressed to humans are severely under-determined and ambiguous: instruction 2), for instance, does not specify what the eggs are to be added to. It also does not state that the eggs need to be cracked before and only their content must be added. Similarly, the relations between the spatula, its parts and the pancake are not explicitly referred to in instruction 6).

Typically, humans fill such information gaps with great ease. They share a large amount of common background knowledge that allows them to do such inferences quickly on demand. As Bailey (1997) points out, humans are capable of rapidly yet flexibly acquire the knowledge of how to use different action verbs in different contexts. This already happens in early childhood and by hearing just a few examples. Humans are capable of abstracting away from single instances of events to more generic event patterns and to transfer this knowledge to new, unseen situations.

In this section, I investigate how much action-specific knowledge robots should be equipped with in order to be capable of performing everyday activities successfully,

effectively and competently. I will further investigate in the subsequent sections how this knowledge can be *acquired*, *represented* and *used*. For this research I restrict myself to abstract knowledge, the knowledge about actions that is typically communicated and written down in instructions. Other kinds of essential knowledge that the abstract knowledge needs to be combined with is addressed in other works and includes naïve physics knowledge (Kunze et al., 2011), common-sense knowledge (Kunze et al., 2010) and knowledge gathered from experience (Jain et al., 2009b) or from demonstration (Fang et al., 2016).

In order to get a deeper understanding about the kind and amount of knowledge required for executing everyday activities, we did a study on a large set of natural-language instructions that we mined from the `wikihow.com` website. Wikihow contains thousands of recipes, plans and other step-by-step instruction sheets for a vast number of everyday household activities written by humans and intended for human use. Our investigations aim at answering the following questions:

- How many different actions does a robot need to know?
- How much variation do these actions exhibit?

At the time this study has been conducted, the “Food & Entertaining” category on `wikihow.com` comprised 273 subcategories consisting of 8786 articles in NL, covering basic cooking skills (e.g. cutting techniques), recipes for cooking dishes (e.g. making pancakes) and activities of wider scope, such as organizing a child’s birthday party. An exemplary search result for the query “make pancakes” is shown in Figure 3.2.

The instruction sheets have been automatically extracted from the Web using a common HTML parser. We carried out basic linguistic analyses, for which we applied the system developed by Tenorth et al. (2010) for extracting single instructions from NL text and for transforming them into a formal, logic-based representation. The system first parses an NL text using the Stanford parser (Klein and Manning, 2003) and afterwards determines the action verbs from the syntactic sentence structures. The system also implements a mapping from word meanings to concepts into the KnowRob knowledge processing framework. For a detailed description I refer the interested reader to Tenorth et al. (2010).

3.3.1 How Many Actions Are There?

The 8786 NL plans consist of more than 130,000 sentences in total. Out of those, we extracted about 53,000 relevant instructions. Instructions are considered relevant if

Action Verb	# Occurrences
Adding sth. to sth./Combining	> 7,900
Picking/Placing sth. sw.	> 4,900
Filling/Pouring sth. into/onto sth.	> 3,100
Stirring/Beating sth.	> 1,900
Removing sth.	> 1,700
Serving sth.	> 1,400
Mixing/Blending sth.	> 1,200
Cutting/Chopping/Slicing	> 1,100
Cooking/Simmering/Boiling	> 1,100
Baking	> 900
Sprinkling	> 800
Flipping/Turning over	> 700
Refrigerating/Cooling/Freezing	> 600
Shaking	> 600
Waiting	> 500

Table 3.1: Most frequent action verbs in the wikihow.com dataset and their number of occurrences.

they are goal-directed in a sense that a respective instruction effectively contributes to the overall outcome of an activity. This differentiation is important since in those recipes, beside regular instructions, also many additional explanations, comments and non-goal-directed instructions can be found, such as “enjoy your meal,” “admire your work,” or “be an artist.” Such instructions do not represent action verbs for object manipulation, which we consider here, and thus they have been filtered for this study by a stop word strategy.

We found that almost the entire set of 8786 NL plans under consideration can be represented by compositions of instructions spanned up by a space of about 100 different action verbs. These are remarkably few considering the number of 53,000 actions in total. Among those 100 action verbs, the most important (i.e. most frequent) actions are given by pick-and-place actions (e.g. “Place the place mat in front of the chair.”) and actions for combining two or more substances (e.g. “Add the eggs to the flour.”). Even more interestingly, the top 15 action verbs make more than 50% of all the 53,000 actions. Table 3.1 shows the approximate frequencies of the top 15 action verbs.



Figure 3.3: Automatically generated taxonomy of different types of “Flipping” actions using a semantic clustering of syntactic relations.

3.3.2 How Much Variation Is There?

Despite the fact that the pure number of action verbs in recipes from the household domain is rather small, the actions under consideration are still very complex to perform. On the one hand, this is due to the world to be acted in being inherently continuous and uncertain, but, on the other hand, it also results from different actions often occurring with different parameterization in different contexts.

To examine how much variation the domain-relevant action verbs exhibit, we analyzed the 53,000 instructions with respect to their parameterization given by syntactic relations such as the object acted on as well as prepositional relations that modify the respective action verb, such as ‘with’, ‘from’, ‘to’, ‘into’ relations and the like. These relations can have different meanings in different contexts and hence it is crucial to be able to resolve their meanings in order to understand and execute the respective action proficiently.

The dimensionality of each action verb configuration has been reduced by applying a semantic clustering technique to these parameterizations on the objects referred to in the respective relations. As a distance measure between concepts, the WUP similar-

Relation	Flipping	Cutting	Adding	Filling	Stirring
<i>doobj</i>	65	499	1200	188	166
<i>prep_with</i>	12	28	58	162	36
<i>prep_on</i>	2	34	42	3	4
<i>prep_into</i>	3	139	77	24	48
<i>prep_onto</i>	8	1	14	1	1
<i>prep_to</i>	2	26	460	24	18
<i>prep_from</i>	0	46	18	3	4
<i>prep_through</i>	1	31	4	0	8

Table 3.2: Sizes of clusters that have been obtained by semantically clustering the prepositional relations of action verbs.

ity (Wu and Palmer, 1994) has been applied to these symbolic, relational arguments. The corresponding word senses (i.e. concepts in the WordNet class taxonomy) have been determined first by the importer for NL instructions from Tenorth et al. (2010). Figure 3.3 exemplarily shows a taxonomy of the action verb ‘Flip’, which has been automatically generated by recursively applying the semantic clustering algorithm to the arguments of the syntactic relations in the entire set of ‘flipping’ instructions. It can be seen that our procedure generates a steep taxonomy of different action configurations for the action verb ‘Flip’, which reasonably reflects different types of that action. As an example, consider the generalization of *FlippingPancakeWithInstrumentality*, which is highlighted in yellow.

Table 3.2 shows the cluster sizes for relations with respect to five of the 15 most frequent action verbs. For example, the action ‘stirring’ appears with 166 different direct objects (denoted by the *doobj* relation), and 36 different words connected to stirring via a prepositional ‘with’ relation (denoted by the *prep_with* predicate).

The variation of prepositional arguments among action verbs varies a lot. Although flipping is one of the most frequent actions in the data set with more than 700 occurrences, there are only 65 different direct objects that the flipping action is performed on. ‘Adding’, by far the most frequent action, has more than 1,200 distinct direct objects. As the introductory exemplary recipe for pancake making also suggests, adding is a very generic action that comes with very different parameterization and can be executed in many different ways or even must be achieved in a very particular way, depending on the current context the action takes place in. On the other hand, adding eggs to a dough, adding egg yolks to a dough or adding a spice to a dough all require different execution.

The study on domain-specific action verbs shows that a large number of complex everyday household activities conducted by humans can be broken down to a relatively small number of elementary actions. It therefore gives us grounds to assume that the household activities are indeed not closed with respect to their elementary building blocks of instructions, but their parameterization allows a great combinatorial variety of whole instruction sheets.

The study thus also supports the introductory thesis that robots supposed to perform everyday activities with the same ease as humans do, need to have a substantial body of knowledge about *how* to execute the single actions involved. With implementing robot control plans for the most important action verbs we reported on above, we expect that we can cover a wide range of everyday household activities that can be performed by our robots.

3.4 Conceptual Framework

Executing NL instructions in the way they are meant often requires robots to infer missing, and disambiguate given information, which requires a large body of common and common-sense knowledge. In this section, I conceptually describe PRAC, a framework for learning of and reasoning about action-specific probabilistic knowledge bases. In PRAC, knowledge about actions and objects is compactly represented in first-order probabilistic models, which are used to learn joint probability distributions over the ways in which instructions for a given action verb are formulated. These joint probability distributions are then used to compute the plan instantiation that has the highest probability of producing the intended action given the NL instruction:

$$\arg \max_{plan} P \left(\text{intended}(plan) \left| \begin{array}{l} \text{“neutralize 75ml} \\ \text{of hydrochloric acid”} \end{array} \right. \right). \quad (3.1)$$

To solve this inference task, PRAC is equipped with a joint probability distribution over the action roles source, destination, the object acted on, the tool to be used, etc. for each action verb. To this end, I introduce the notion of *action cores*, conceptualizations of action verbs that represent formal specifications of actions and their parameters. Action cores are intended to interface executable robot plans on a symbolic, linguistic level. The resulting probabilistic first-order knowledge bases of actions and their respective roles are called Probabilistic Action Cores (PRAC).

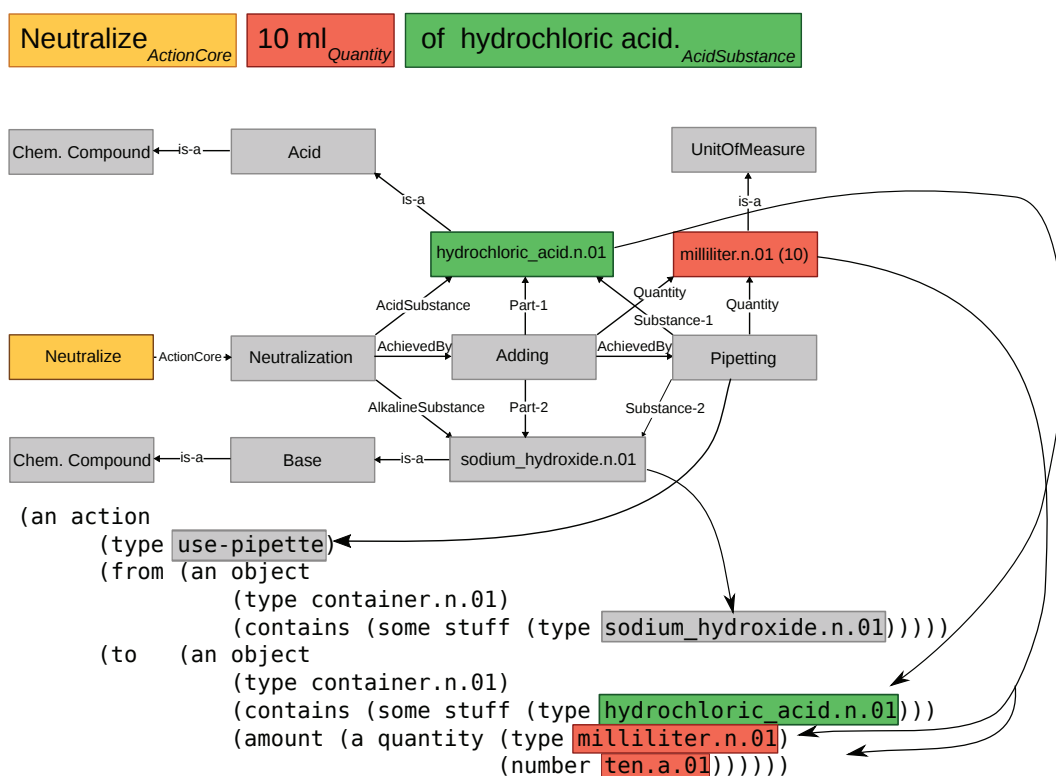


Figure 3.4: Fragment of a semantic reasoning process about executable refinements of the instruction “neutralize 10 ml of hydrochloric acid.” The instruction is refined to a pipetting action, which is used for instantiating the attached use-pipette plan schema with inferred knowledge. The instantiated plan schema can be sent to the CRAM plan executive for execution.

An *interpretation* of a natural-language instruction is defined as the most probable instantiation of a plan schema with all of its parameters assigned. The process of instantiating a plan schema is illustrated in Figure 3.4. The graph shows a fragment of a semantic network of relations and entities, which might be a representation of a reasoning process about the instruction “neutralize 10 ml of hydrochloric acid.” The original *Neutralizing* action is being refined to an *Adding* and eventually to a *Pipetting* action, and the inferred knowledge about given (hydrochloric acid) and missing (sodium hydroxide) substances and the amount specification is filled into the free slots of a plan schema attached to the *Pipetting* action. Instruction understanding can in turn be realized by retrieving the plan corresponding to the action required by the instruction and by constraining its parameterization accordingly to the instruction.

This formulation of the problem of instruction interpretation is elegant and general because by performing the inference task on a joint probability distribution over action instructions we can at the same time infer the plan that is most appropriate for performing the instruction, the refinement of the parameters of the plan schema on

the basis of the information given in the instruction, and automatically fill in missing parameters by inferring their most probable value from the distribution.

One of the core concepts in PRAC is that it is connected to the WordNet dictionary, which provides a rich taxonomy of word meanings. The use of WordNet has two essential benefits: First, as already mentioned, it provides the ontological groundwork for unambiguous representations of activity models as every concept in its taxonomy has a distinct, well-defined semantics. Second, a taxonomy like WordNet enables to establish analogies across concepts and thus allows to transfer knowledge from one concept to an unseen one by exploiting the semantic similarity of the two.

For the purpose of this thesis, I assume that robotic agents are equipped with a plan library that contains parameterizable plans for actions, which have to be refined and instantiated according to a given instruction. I call such symbolic interfaces to robot plans *plan schemata*. A plan schema is a generic, parameterizable implementation of an elementary action which is executable and guaranteed to produce meaningful behavior on a robot. Note that ‘meaningful’ does not necessarily mean ‘successful’ in the sense that its execution always results in the desired outcome. It is rather to be understood as ‘intentional’ or ‘goal-directed’ in the sense that there is a rationale behind its behavior which can be reasonably explained. As an example, consider a generic action such as pipetting liquids from a container into some other container. The signature of a respective plan schema might look as follows.



Plan Schemata

```
use-pipette(from :default (an object
                        (type container.n.01)
                        (contains (Pipetting Source)))
amount :default (Pipetting Quantity)
to :default (an object
            (type container.n.01)
            (contains (Pipetting Destination))))),
```

The plan schema has three formal parameters *from*, *to* and *amount* that specify the source container from which the substance to be transferred by a Pipetting action is to be aspirated and the goal container where the substance is to be released to. *amount* specifies the amount of this substance. Plan schemata may reside on different levels of abstraction and complexity of actions. They may represent very elementary actions such as grasping an object or putting an object down, or more complex manipulation skills like using a tool like a knife or appliances like a tap. The degree of complexity and generality a plan schema can exhibit is very task dependent and needs to be determined by the engineer who is designing it. In general, plan schemata

should be as generic as possible to be applicable to a broad range of situations and as specific as necessary to still be able to make commitments to certain effects of the action. A plan schema has slots that are free for individual parameterization. In the above example, the type of substances from the source and destination containers as well as the quantity of the substance to be transferred (highlighted in gray) can be retrieved from the respective action roles attached to the abstract concept of a *Pipetting* action.

3.4.1 Reasoning Tasks

The ultimate goal of PRAC is to resolve ambiguity and to complete an instruction to the most plausible action specification based on what is given by the instruction. In the following, I will illustrate the anticipated usage of the PRAC distribution by means of three simplified exemplary queries.

Action Role Assignment PRAC can be queried for the most likely assignment of roles for a given set of objects with respect to a particular action core. Consider an instruction, such as “add a drop of sodium hydroxide.” In order to generate a mathematical formulation of the problem of computing the most probable role assignment, we introduce two symbols o_1 and o_2 denoting two words ‘drop’ and ‘sodium hydroxide’ mentioned in the instruction. They are assigned the word senses *drop.n.02* and *naoh.n.01* in the WordNet upper ontology, respectively. In context of an *Adding* action, the PRAC distribution can be queried for the most probable assignment of the action roles attached to *Adding*. Assuming that there are three action roles attached to *Adding*, namely *Quantity*, *NewMember* and *Group*, such a query can be formulated as

$$\arg \max_{o'_1, o'_2, o'_3 \in \{o_1, o_2, \perp\}} P \left(\begin{array}{c} \text{Quantity}(o'_1, \text{Adding}), \\ \text{NewMember}(o'_2, \text{Adding}), \\ \text{Group}(o'_3, \text{Adding}) \end{array} \middle| \begin{array}{c} \text{is_a}(o_1, \text{drop.n.02}), \\ \text{is_a}(o_2, \text{naoh.n.01}) \end{array} \right) = \left\{ \begin{array}{l} o'_1 = o_1, \\ o'_2 = o_2, \\ o'_3 = \perp \end{array} \right\},$$

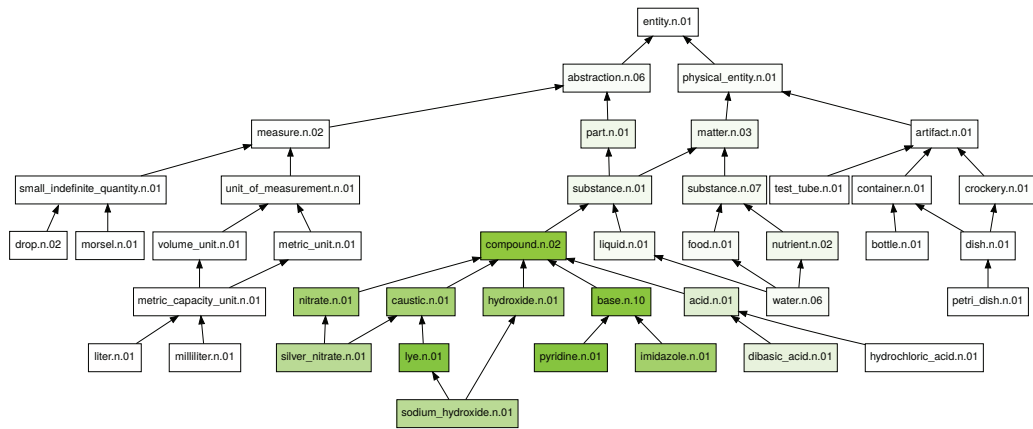
where \perp denotes no assignment. In this example, the *Quantity* role was assigned to the object of the type *drop.n.01*, the *NewMember* role to the object of the type *naoh.n.01* and the *Group* could not be assigned any of the objects mentioned in the instruction. Note that this arg max solution is not a proper interpretation of the instruction in the notion from above, because there is no *Group* specified.

Action Role Completion In order to fill missing role assignments such as the *Group* in the previous example, one can solve for a different arg max query. Consider the instruction “neutralize the hydrochloric acid” and suppose the word of the type *hcl.n.01* has already been assigned the role *AcidSubstance* of the *Neutralizing* action core analogous to the previous example. According to its definition, there must be the role *AlkalineSubstance* assigned to some concept, which is not given in the instruction. In order to infer its role assignment, a Skolem constant s can be introduced, which hypothetically fills the missing role slot of *AlkalineSubstance*. Since the taxonomy relation of the PRAC dictionary is included in the distribution, one can query for the most probable type of s :

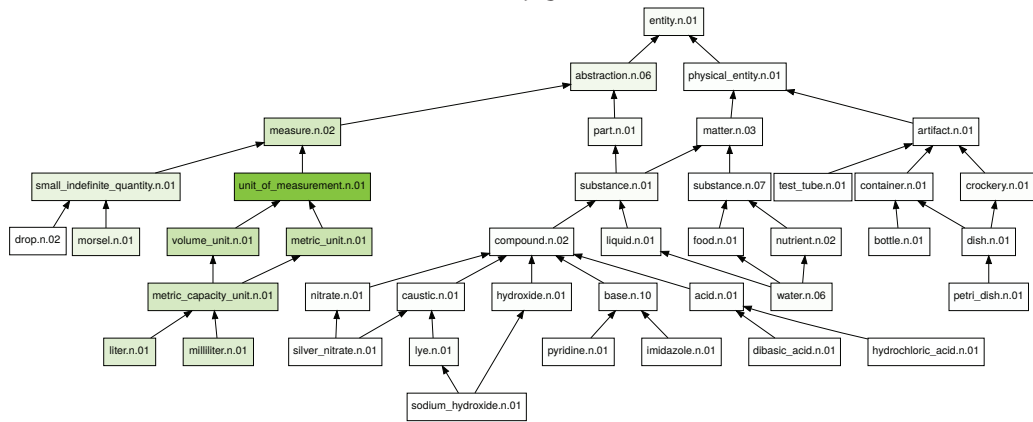
$$\arg \max_{c \in \mathcal{T}} P \left(is_a(s, c) \left| \begin{array}{l} is_a(hcl, hcl.n.01), \\ AcidSubstance(hcl, Neutralizing), \\ AlkalineSubstance(s, Neutralizing) \end{array} \right. \right) = naoh.n.01,$$

which means that sodium hydroxide (NaOH) is the most probable alkaline counterpart for *Neutralizing* hydrochloric acid (HCl).

Joint Distributions Over Taxonomies One of the key features of PRAC is the ability to perform reasoning about unmodeled concepts, i.e. concepts that have not been seen during learning and in turn are not present explicitly in any probability model. This enables (1) a compact representation of knowledge, (2) efficient transfer of the learnt knowledge to new situations and (3) filling missing information pieces in under-determined action specifications. Figure 3.5 shows two examples of conditional distributions over concepts in an excerpt of the WordNet taxonomy for potential slot fillers of the *AlkalineSubstance* and *Amount* roles of a *Neutralizing* action: From all concepts, non-acidic specializations of chemical compounds gain highest probability for the *AlkalineSubstance*. Analogously, subclasses of the *unit_of_measurement.n.01* concept represent the most likely *Amount* specifications. For PRAC maintains joint distributions over the action roles and concepts from the PRAC dictionary, any conditional distribution can be computed given any evidence, which enables context-sensitive completion of actions like in the previous example. PRAC is even able to assign probabilities to concepts that are not explicitly represented in symbols in a probabilistic KB. To this end, I have developed FUZZY-MLNs, a reasoning framework that extends MLNs by the concept of semantic similarity in taxonomies. FUZZY-MLNs are presented in detail in Chapter 4.



(a) Distribution over the WordNet taxonomy given the AlkalineSubstance action role.



(b) Distribution over the WordNet taxonomy given the Amount action role.

Figure 3.5: Posteriors distributions over the taxonomy conditioned on action roles of a neutralization action. More intense node colors indicate higher probability.

3.4.2 System Architecture

The PRAC framework for translating NL instructions into abstractly parameterized robot action plans is depicted in Figure 3.6. Its main components are the PRAC *plan library*, the PRAC *knowledge base*, the PRAC *howto library* and the PRAC *dictionary*. In a nutshell, the roles of these components are the following ones.

PRAC Dictionary The PRAC *dictionary* provides all possible meanings of all words that can occur in an NL instruction to be executed by a robot. The meanings are concepts in an ontological knowledge base defined in the online dictionary WordNet, which comprises more than 117,000 concepts. For example, the possible meanings of ‘cup’ in the PRAC dictionary include a specialization of a physical object and, more specifically, a container object, an amount specification, and a trophy (see also

Table 3.4).

PRAC Knowledge Base The PRAC *knowledge base* contains a collection of action verb-specific knowledge bases, called *action cores*, that represent how possible action configurations for a given action verb can be constructed on a conceptual level. For example, a *Pouring* action can be formalized on a conceptual level in terms of conjunctions of logical assertions over the predicates *action_core(a, Pouring)*, *Theme(a, t)*, *Source(a, s)*, *Destination(a, d)*, etc. The assertion *Theme(a, t)* states that the theme of action *a* is of the type *t*, i.e. the entity which is poured. The parameters *t*, *s* and *d* are concepts in the PRAC dictionary, while *a* denotes the action under consideration.

Probabilistic Action Cores Action-specific knowledge bases are then trained with a set of instructions stated in first-order logic in order to learn a joint probability distribution over predicate instantiations, which is induced by the given set of instructions. These distributions are called the Probabilistic Action Cores (PRAC). The learned distributions represent correlations between the concept restrictions of the parameters in instructions with respect to an action verb. For example, the PRAC of the *Pouring* action core could entail that if the *Theme* of a pouring action is the concept *wine.n.01* then it is likely that the *Source* for the pouring action will be an instance of the concept *bottle.n.01* and the *Destination* will be an instance of the concept *glass.n.02*. Conversely, if the *Theme* is of the concept *water.n.06*, then the *Source* is more likely to be a *water_faucet.n.01*.

PRAC Howto Library The PRAC *howto library* is a database containing a large amount of semantically annotated documents from any source of knowledge. The howtos might be mined from web pages containing recipes like the *wikihow.com* web page, but also text books (e.g. for basic chemical experiments), user manuals for operating devices (e.g. how to operate a microwave oven) or commands put in by a human user (e.g. “I take my coffee with milk and sugar.”) are imaginable. The howto library serves as a data corpus for analogical reasoning, which allows instruction completion to be more efficient than purely probabilistic approaches. When an incomplete instruction is encountered, PRAC searches in the howto library for the most similar instructions and uses them as a blueprint for the completion of the missing roles. As no re-learning of the probabilistic models is required, this instance-based learning approach also allows learning from only one exemplar (so-called ‘one-shot’ learning).

PRAC Plan Library Finally, the PRAC *plan library* contains action specific plans and their schemata. PRAC plans are equipped with plan signatures following the ‘design-

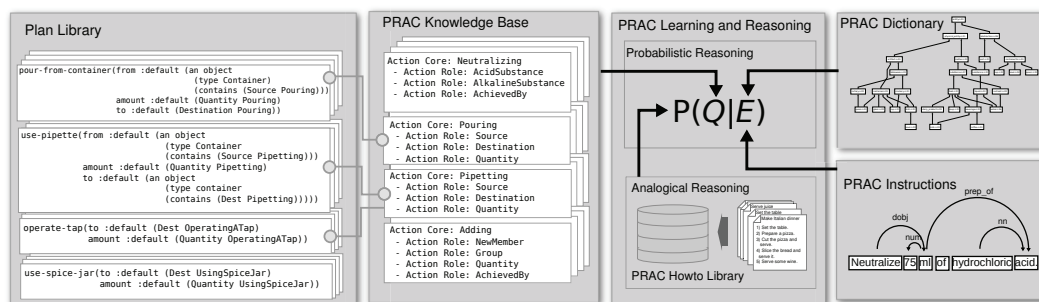


Figure 3.6: Key components of the PRAC framework and their role in inferring the most probable executable instruction.

by-contract’ principle: The plan signature specifies the formal parameters of the plan, the concept restrictions for each parameter and how the respective plan parameter can be computed from the PRAC knowledge base. An example of such a plan signature has already been given above in the form of the use-pipette plan schema.

Reasoning The probabilistic first-order knowledge base over semantic networks for solving inference problems of the form shown in Equation (3.1) has an enormous size. It contains at least the cross product of all possible word meanings squared and roles where the set of the possible word meanings include all possible meanings of the words that occur in the training data plus the number of their superconcepts in the taxonomy. To make the reasoning problem feasible we decompose it into three weakly connected subproblems and generate the probabilistic knowledge bases for each substep independently to keep the knowledge bases as small as possible: (1) inferring the relevant PRAC, (2) disambiguation and role assignment and (3) inferring missing information pieces and refinements of action cores and their associated roles. Using the components of the PRAC system introduced above, the computational process for computing the most probable executable instruction operates as follows: In a first step, a given natural-language instruction ι is translated by a natural-language parser NL-PARSE into a logical representation of the instruction’s syntactic structure I , which I call a PRAC instruction. The PRAC instruction I is then interpreted by inferring the meaning and semantic role of the individual syntactic structures and missing information pieces using the PRAC dictionary, the PRAC howto library and the action core itself. This interpretation process results in the *most probable executable instruction* of ι . Starkly simplified, reasoning in PRAC about the most probable interpretation of an NL instruction ι is implemented by the following multi-step composition of database transformations by means of probabilistic relational inference:

Predicate	Meaning
$det(w_1, w_2)$	w_2 is the determiner of w_1
$dobj(w_1, w_2)$	w_1 is the direct object of w_2
$advmod(w_1, w_2)$	w_1 is an adverbial modifier of w_2
$prep_with(w_1, w_2)$	w_1 and w_2 are connected by the preposition ‘with’
$prep_from(w_1, w_2)$	w_1 and w_2 are connected by the preposition ‘from’
$prep_to(w_1, w_2)$	w_1 and w_2 are connected by the preposition ‘to’
$nn(w_1, w_2)$	w_2 and w_1 form a compound noun
$has_pos(w, p)$	p is the part-of-speech of w

Table 3.3: Selection of logical predicates representing the syntactic structure of words in a sentence. A comprehensive list can be found in de Marneffe and Manning (2008)

$$\arg \max_{R_{A_{missing}}} P \left(R_{A_{missing}} \mid \arg \max_{R_{A_{given}}} P \left(R_{A_{given}} \mid \arg \max_A P(A \mid I) \right) \right), \quad (3.2)$$

where $I = \text{NL-PARSE}(\iota)$ is a PRAC instruction representing the syntactic structure of the NL instruction returned by NL-PARSE. A denotes the action core referred to by the instruction, $R_{A_{given}}$ are the action role assignments of A given in I , and $R_{A_{missing}}$ are the action roles of A which do not have a correspondence in I .

In the remainder of this section I will describe in greater detail the concepts and components that the learning of and reasoning about probabilistic action cores is built upon. Section 3.5 addresses the reasoning pipeline in PRAC.

3.4.3 PRAC Instructions

A PRAC instruction I is a set of assertions about the grammatical relations referring to the constituents of a NL instruction and their syntactic structure. These grammatical relations are represented by predicates including the small selection listed in Table 3.3. They are obtained for any sentence in natural language by a parser like the Stanford parser (De Marneffe et al., 2006). Using these predicates, a natural-language instruction such as $\iota =$ “neutralize the hydrochloric acid with sodium hydroxide”, for example, is transformed into the logical assertions I ,

$$\begin{aligned}
 & \text{dobj}(\text{neutralize-1}, \text{acid-4}) && \text{has_pos}(\text{neutralize-1}, \text{VB}) \\
 & \text{det}(\text{acid-4}, \text{the-2}) && \text{has_pos}(\text{acid-4}, \text{NN}) \\
 & \text{nn}(\text{acid-4}, \text{hydrochloric-3}) && \text{has_pos}(\text{hydroxide-7}, \text{NN}) \\
 & \text{nn}(\text{hydroxide-7}, \text{sodium-5}) && \text{has_pos}(\text{sodium-6}, \text{NN}) \\
 & \text{prep_with}(\text{neutralize-1}, \text{hydroxide-7}), && (3.3)
 \end{aligned}$$

which we denote by $\text{NL-PARSE}(t)$. These syntactic dependencies indicate that the second word ‘the’ depends on the fourth word ‘acid’ as a determiner, ‘hydrochloric’ and ‘acid’ represent a compound noun, which forms the direct object of the word ‘neutralize’, which is connected to the word ‘hydroxide’ via the preposition ‘with’. The logical assertions representing the syntactic structure of the instruction thus form a relational database that can serve as evidence in a probabilistic relational model as indicated in Equation (3.2). For a more detailed and exhaustive description of the syntactic dependencies, I refer to de Marneffe and Manning (2008).

The syntactic information about an instruction can carry strong evidence for understanding NL action specifications. Prepositional relations, for instance, but also the order of words in an instruction can have strong influence on its semantics. For example, in the instruction represented in (3.3), the prepositional relation *with* indicates that the sodium hydroxide can fill the action role of the *Neutralizer* for the hydrochloric acid, which in turn is the *Neutralizer*. However, the syntax alone is insufficient for robustly determining an object’s role in an instruction. For example, in the instruction “flip the pancake with a spatula,” whose parse is given by

$$\begin{aligned}
 & \text{det}(\text{pancake-3}, \text{the-2}) && \text{dobj}(\text{Flip-1}, \text{pancake-3}) \\
 & \text{det}(\text{spatula-6}, \text{a-5}) && \text{prep_with}(\text{Flip-1}, \text{spatula-6}),
 \end{aligned}$$

the preposition ‘with’ rather denotes an instrumental relationship between the action *flip* and the subsequent word *spatula*. Other prepositional relations, such as ‘into’, ‘onto’, ‘from’ or ‘off’, however, indicate *Goal* and *Source* relations in instructions like “Fill a cooking pot *with* water *from* the tap.” As the syntax alone does not carry sufficient information for a competent NL interpretation, in PRAC, the syntax of an instruction is only taken as evidence for semantic probabilistic reasoning. PRAC in turn does not rely on a correct parse, for it considers these relations just as evidence features and systematic errors of a parser could be incorporated in the learned model $P(A|I)$.

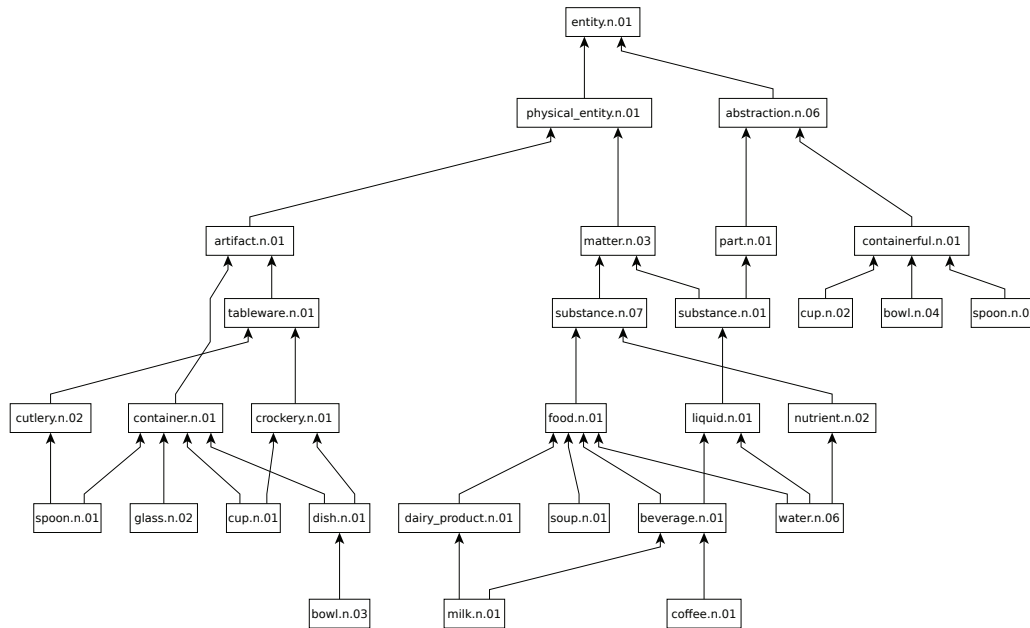


Figure 3.7: Excerpt of the WordNet upper ontology showing three major taxonomic areas: physical objects like containers (cf. bowl.n.03) and substances (cf. milk.n.01) abstract units of measure (cf. bowl.n.04)

3.4.4 PRAC Dictionary

Words can have multiple meanings causing ambiguity in NL instructions. Consider, for example, the terms ‘cup’ and ‘milk’ and their meaning in the two instructions “fill a cup with milk” and “add a cup of milk.” In the former case, ‘cup’ refers to a drinking mug, a physical object that can hold milk. In the latter case, it rather denotes a measurement unit specifying the amount of milk to be added. Though this semantic difference may seem subtle, correctly distinguishing between word meanings is crucial for successfully performing the actions.

The PRAC dictionary is a set of logical assertions that assign meaning, i.e. word senses, to words. It is filled with word senses, so-called ‘synsets’, from the online dictionary WordNet. A selection of possible meanings of the noun ‘cup’ is given in Table 3.4, a more comprehensive selection of synsets can be found in Appendix B. The word senses are organized in a taxonomy given by a directed acyclic graph, which we denote by the relation \sqsubseteq , i.e. $c_1 \sqsubseteq c_2$ denotes that the concept c_1 is a specialization of concept c_2 . Given a word and its part of speech, the possible word meanings can be obtained from WordNet.

In the PRAC dictionary, a word w is assigned a particular meaning s by means of a set

Concept Name	Definition
<i>cup.n.01</i>	a small open container usually used for drinking; usually has a handle (“he put the cup back in the saucer”; “the handle of the cup was missing”)
<i>cup.n.02</i>	cupful, the quantity a cup will hold (“he drank a cup of coffee”; “he borrowed a cup of sugar”)
<i>cup.n.05</i>	cup-shaped plant organ
<i>cup.n.08</i>	a large metal vessel with two handles that is awarded as a trophy to the winner of a competition (“the school kept the cups is a special glass case”)

Table 3.4: Selection of different meanings of the word ‘cup’ obtained from WordNet.

of atomic logical assertions

$$has_sense(w, s) \quad \text{and} \\ is_a(s, c).$$

where $has_sense(w, s)$ states that the word w has the sense s and the is_a predicate connects the sense s to a concept c in the WordNet dictionary \sqsubseteq . It is important that the word sense s is a first-class object and words are not just directly assigned a concept using the is_a predicate. This reification allows to reason about the uncertain sense assignment has_sense and at the same time specify similarities of one word sense to multiple concepts using is_a . In Chapter 4, I will motivate this modeling in more detail in context of fuzzy Markov logic network (FUZZY-MLN) inference and reasoning about unmodeled concepts.

3.4.5 PRAC Knowledge Base

The PRAC knowledge base is the central component of the PRAC system. It contains a library of data structures, which I call *action cores*. An *action core* is the conceptualization of an action which constitutes an abstract event type and assigns an action role to each entity that is needed in order to successfully perform the respective action.

Fillmore (1976) provides conceptualizations of actions that consist of an action definition (so-called *frames*) and a set of associated roles (so-called *frame elements*) that represent the parameters of a respective action in context of the *FrameNet* initiative. An action itself is represented as an abstract concept and specific action

verbs represent instances of these concepts. As an example, consider the definition of the action concept *MovingInPlace*, which is defined in FrameNet as follows:

“A Theme *moves with respect to a fixed location, generally with a certain Periodicity, without undergoing unbounded translational motion or significant alteration of configuration/shape.*” — (International Computer Science Institute, 2017-02-22)

Possible instances of this abstract event are given by the action verbs *Rotate*, *Shake*, *Spin*, *Twirl*, *Flip*, *Turn around* etc., which share the same action parameters, such as the *Theme* undergoing a non-translational motion, the *FixedLocation* or the *Periodicity* of the motion. Table 3.5 shows definitions of a selection of roles attached to the *MovingInPlace* event. In PRAC, action cores adopt the idea of frames in FrameNet but extend them by several components:

1. the joint probability distributions over the arguments of frame elements (or rather action roles)
2. the use of a dictionary and deep upper ontology assigning senses to words
3. the connection of action cores to executable robot plan schemata.

More formally, an action core AC is defined as a tuple $\langle A, R \rangle$, where A is the globally unique name of the action core and $R = \{r_{A_i}\}_{i=1}^{n_A}$ is an indexed set of its associated action roles. For an interpretation x of a PRAC instruction I , the following holds:

$$action_core(x, A) \rightarrow \exists c_1, \dots, c_{n_A} \bigwedge_{i=1}^{n_A} r_{A_i}(c_i, A), c_i \in \mathcal{T} \quad (3.4)$$

The right side of the implication in (3.4) ensures that every instantiation of an action core must have a complete assignment of its action roles to concepts in \mathcal{T} , otherwise it is not an instance of the action core. However, being able to assign all roles of an action core does not imply that it must have an instance in x . It is important to note that in PRAC, the domains of action roles and their corresponding parameter slots in the plan schemata are given by the set \mathcal{T} of all concepts from the ontological knowledge base in the PRAC dictionary. Equation (3.4) defines the space \mathcal{X} of possible interpretations of an instruction. A graphical representation of one particular interpretation of the instruction “neutralize 75 ml of hydrochloric acid” is shown in Figure 3.4: There are instances of the three action cores *Neutralizing*, *Adding* and *Pipetting* with their respective roles assigned to a concept representing the process of causing a chemical substance to take a neutral pH-value by combining it

Role	Definition
<i>Theme</i>	A physical <i>entity</i> that is participating in non-translational motion.
<i>FixedLocation</i>	The point or set of points that define the limits of motion for the <i>Theme</i> .
<i>Angle</i>	The amount of rotation that the <i>Theme</i> undergoes
<i>Periodicity</i>	The number of times the <i>Theme</i> returns to a state in a given duration.
<i>Direction</i>	The direction of rotation of the <i>Theme</i> .
<i>Place</i>	The location where the bounded motion happens.
<i>Time</i>	The time at which the <i>Theme</i> is in bounded motion.

Table 3.5: Role definitions for the Action concept *MovingInPlace*

with some other substance. For the chemical reaction itself, there must always be two components reacting, an acid and a base. The corresponding action core *Neutralizing* thus is attached two action roles, *AcidSubstance* and *AlkalineSubstance*. This ensures that all symbols have the same semantics across the different components of PRAC, the syntactic representation in the PRAC instructions, the semantic action representation of action cores as well as the plan schemata. The set of action cores and the above definition of an interpretation can thus be regarded as a template for constructing a graphical model of interpretations like the one in Figure 3.4.

There is an action core for every verb in the PRAC dictionary that represents a meaningful action. However, there are action cores that do not have a direct correspondence to a plan schema because they do not represent actions that are directly executable but are subject to further reasoning. *Neutralization* is an example of such an action core: It is not an executable action as such, but rather describes a chemical process that is triggered by *Adding* one substance to the other. *Adding* itself is an action core representing the process of making a new member part of an existing group. It has three action roles, namely the *Group*, the *NewMember* and the *Quantity*. The *Adding* action core, however, also represents a process that can be achieved in very different ways depending on the context and the objects involved. For example, “add one liter of water” could be achieved by using the tap or pouring from one container to the other, whereas “add one milliliter of water” should be performed by using a pipette. Conversely, “add a pinch of salt” can be done by using a salt cellar. Such an action core *A* that does not have a direct mapping to an executable plan schema has attached a designated action role *AchievedBy(A, A')*, which is assigned another action core *A'* representing the most likely refinement of the action represented by *A*. In

Section 3.8, I will present in detail how such refinements can be computed in PRAC. The goal in NL instruction understanding is to find the *most probable interpretation under the instruction given as evidence*. Therefore, the PRAC knowledge base has a probability distribution over all action cores and their action roles, conditioned on the PRAC dictionary and the PRAC instructions, as depicted in Figure 3.6,

$$P \left(\begin{array}{l} \text{action_core}(x, A) \rightarrow \\ \exists c_1, \dots, c_{n_A} \bigwedge_{i=1}^{n_A} r_{A_i}(c_i, A) \end{array} \middle| \top, I \right). \quad (3.5)$$

I call (3.5) the Probabilistic Action Cores (PRAC). The PRAC is a first-order probabilistic knowledge base about actions and their parameterizations that is used to disambiguate, interpret, complete and refine NL instructions.

3.4.6 PRAC Howto Library

In order to draw on a large collection of instruction sheets that serves as a source of knowledge, PRAC has a database of semantically indexed recipes available, which can be queried for instances of similar instructions that might contain valuable information for action role completion. As an example, consider the instruction “season the steak.” Browsing a large number of recipes may yield cues for most likely seasonings for steaks in particular, or at least spices for *similar* dishes. For the answers to all of such queries cannot be captured by purely probabilistic models, the howto library enables PRAC to look up the answer to a query in a library.

For storing howtos semantically in a database, a recursive data model of a *Frame* is defined as follows. As shown in Figure 3.8, the primary data structure of a *Frame* consists of an identifier of the action core the Frame is connected with, and a key/value store assigning words to action roles that are attached to the respective action core. It is important to note that at this point, the role assignment is not required to be complete, it is even allowed to be empty. Furthermore, a syntactic representation of the instruction the respective frame has originated from is stored, which consists of a table mapping syntactic relations to pairs of words in a sentence. The representation of a *Word*, on the other hand, consists of an ID that identifies the word uniquely in an instruction, a lemmatized version of the word, its part-of-speech tag, as well as its word sense. The word sense is a pointer to a synset in the PRAC dictionary.

Using this definition of the Frame data structure, a *Howto* can be defined as a

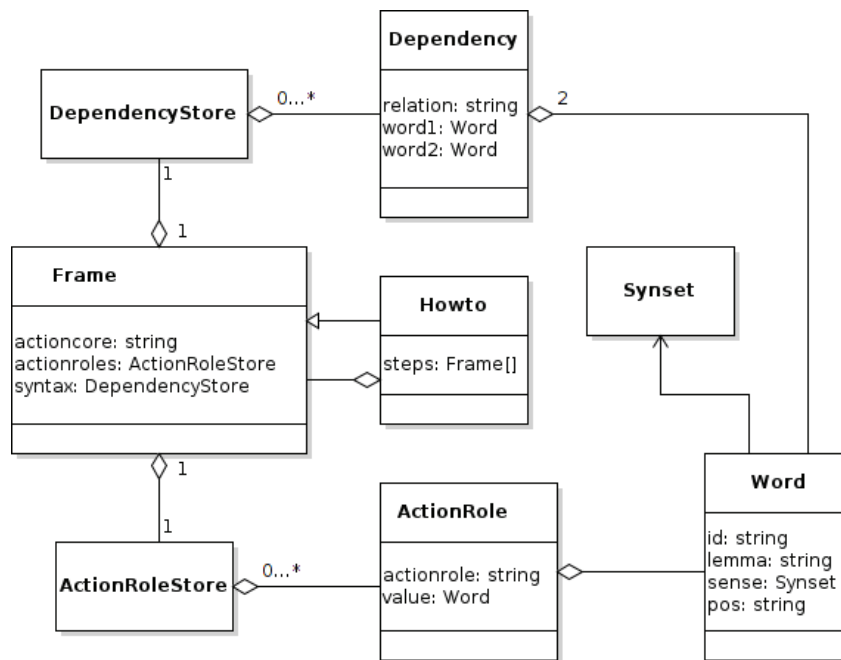


Figure 3.8: The data model used to store semantically annotated howtos in the PRAC howto library.

specialization of a Frame, which extends it by a sequence of Frames representing a sequence of subactions the respective Frame can be equivalently replaced by. A Howto thus is a Frame consisting of a sequence of subframes subsuming it. This model of semantic action representations naturally corresponds to howtos, recipes, instruction sheets, or manuals, which have a title representing a higher-level task, such as ‘how to make an Italian dinner’, and finer-grained step-by-step instructions that are to be performed to achieve the high-level goal, e.g. ‘set the kitchen table for two persons,’ ‘make a salami pizza,’ or ‘serve some wine.’ It is worth mentioning that – like in the previous example – the single instruction steps are not required to be directly executable by a robot. On the contrary, the subframes are allowed to reference other Howtos. This recursive definition allows to compose new Howtos by stacking together existing ones. In Section 3.5.1, I present the PRAC-QUERY algorithm that translates such recipes into fully instantiated, executable robot plans. There may also be different subframe sequences for the same high-level action. This is particularly useful to find alternative ways to achieve a goal if objects required in one howto are not available in the environment or the agent is physically or mentally not capable of performing single action steps. In such cases, an alternative action sequence can be retrieved from the library.

In the current implementation of PRAC, the MongoDB (Chodorow and Dirolf, 2010) database is used as the backbone of the howto library. The basic idea is to fill the

database with representations of huge amounts of step-by-step instructions to be able to efficiently issue semantic queries to it. In the current implementation about 8,400 recipes in natural-language have been mined and imported from the *wikihow.com* website, comprising more than 103,000 instruction steps in total. The procedure for semantically indexing and storing the data in the database is described in more detail in Section 3.6.

Having defined the data model for storing howtos semantically indexed, queries about action specifications can be issued, such as “what are possible utensils for flipping a pancake?”, i.e.

$$\Pi_{\text{actionroles.Instrument.sense}} \left(\sigma_{\substack{\text{actioncore}=\text{Flipping} \wedge \\ \text{actionroles.Theme.sense}=\text{pancake.n.01}}} (\text{prac.howtos}) \right), \quad (3.6)$$

or “what are possible seasonings for a steak?”, i.e.

$$\Pi_{\text{actionroles.Spice.sense}} \left(\sigma_{\substack{\text{actioncore}=\text{Flavoring} \wedge \\ \text{actionroles.Theme.sense}=\text{steak.n.01}}} (\text{prac.howtos}) \right), \quad (3.7)$$

where the selection (σ) and projection (Π) operators from relational algebra have been used. The query in (3.6) first selects from the collection *howtos* in the database *prac* those *Frames* that have been indexed with the *Flipping* action core and whose action role *Theme* is filled by a word with the word sense *pancake.n.01*. It then does a projection of word senses filling the *Instrument* action role. A possible response could be *spatula.n.01*. Analogously, (3.7) retrieves the word sense of words filling the *Spice* role in those howtos with the *Flavoring* action core.

The primary assumption is that the answers are likely to be found among the several thousands of instruction steps that we mined from the Web. A robot instruction like “season the steak.” can thus be completed by issuing such a query to the database. Section 3.8 describes in greater detail how the PRAC howto library can be used to complete instructions and transfer knowledge by analogical reasoning to new situations.

3.4.7 The PRAC Plan Library

The PRAC *plan library* contains action specific plans and their schemata. PRAC plans are equipped with plan signatures following the ‘design-by-contract’ principle: The plan signature specifies the formal parameters of the plan, the concept restrictions for each parameter and how the respective plan parameter can be computed from the PRAC knowledge base. An example of such a plan signature has already been given above in the form of the use-pipette plan schema. As another example, the signature of the plan for a pouring action looks as follows:

```
pour-from-container(from :default (an object
                                (type container.n.01)
                                (contains (Pouring Source)))
                    amount :default (Pouring Quantity)
                    to :default (an object
                                (type container.n.01)
                                (contains (Pouring Destination))))
```

The plan schema specifies that the *from* parameter has to be a specialization of the concept *container.n.01* and that it can be retrieved from the action role *Source* of the action core *Pouring* in the PRAC knowledge base. Likewise, the *amount* parameter can be obtained by querying for the *Quantity*.

It is required that all formal parameters of the plan are linked to roles in the respective action core. By providing a plan signature, the designer of the plan guarantees that for all plan refinements that satisfy the concept restrictions of the individual parameters, executing the plan generates meaningful behavior. ‘Meaningful’ here means that the plan generates behavior that makes sense but is not required to succeed. For a *Pouring* action, for instance, the plan tells the robot to grasp the *Source* container, to hold it above the destination and to tilt it. However, the execution of the parameterized plan hazards failures caused by inappropriate motor control or inaccurate perception, such as spilling the liquid because the container is held too high, off center, or the pouring angle is too steep. This requires that all parameters needed to call sub-plans are computed and none of sub-plan calls contains undefined parameters, which would cause the control system to crash.

The plans themselves are considered as black boxes in PRAC reasoning. Plan execution systems that can handle such qualitative, symbolic constraints on parameters include reactive action packages (RAP) (Firby, 1989) and procedural reasoning system (PRS) (Georgeff and Ingrand, 1989). If deeper reasoning about the ramifications

of actions is necessary, the CRAM (Mösenlechner and Beetz, 2011; Beetz et al., 2012; Mösenlechner, 2016) executive provides reasoning methods that translate qualitative constraints into PROLOG queries that use sampling and backtracking to find parameter instantiations satisfying these constraints. Kinds of such parameters include e.g. action effects, visibility, reachability and the like. The problem of plan design goes beyond the scope of this work and is therefore not further addressed in this thesis.

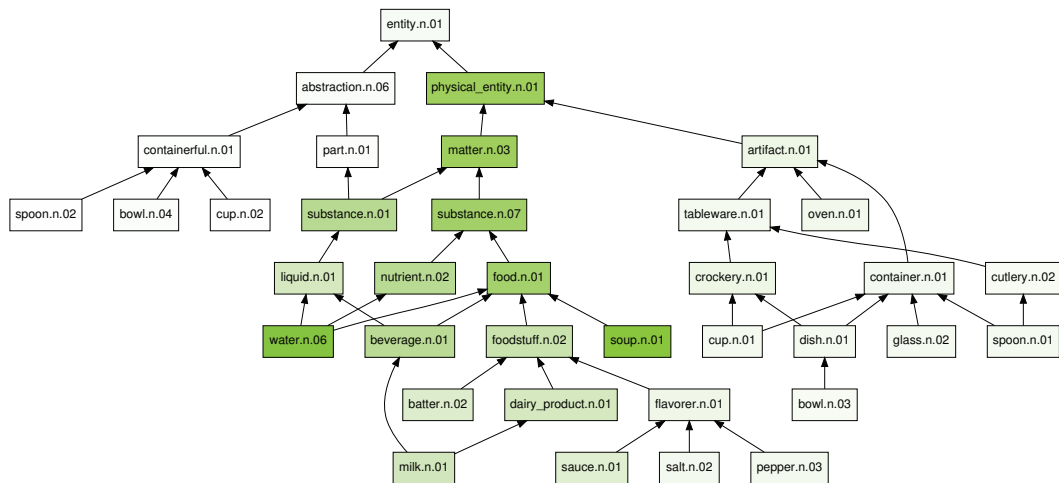
3.5 Learning and Reasoning in PRAC

In this section, I will describe in more detail how learning and reasoning is performed in the PRAC framework. I will depict the basic ideas of learning and inference in PRAC, address challenges that arise and present a processing pipeline for performing inference about interpretations and completions of natural-language instructions.

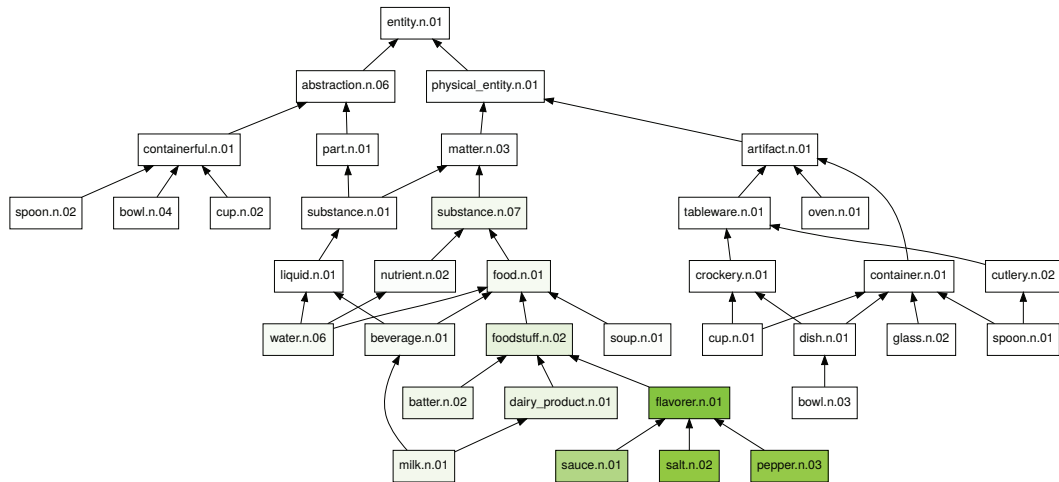
Learning through generalization Humans are capable of learning rapidly and flexibly how to use different words in different situations by only having seen very few examples. They have an efficient apparatus for generalization available, which allows them to abstract away from a very small set of specific exemplars to more generic patterns of everyday situations that they often encounter in their stereotypical form. Consider the example of a ‘filling’ action. From hearing just a few specific instances of that action verb, e.g., “fill a pot with water” and “fill a cup with milk”, humans are capable of forming a stereotyped pattern like “fill a container with a liquid.” This kind of generalization is both powerful and efficient since, on the one hand, it enables compact representation of knowledge and on the other hand, it allows to treat new, unseen examples in a meaningful way.

Inference through specialization Reasoning about new, unseen situations is done by selecting one or more generic patterns that best fit the new situation and by adapting them to reality as necessary in order to come up with an instantiated representation which is as specific and unambiguous as possible. In the example from above, an instruction like “fill a glass with juice,” for instance, is matched against the generic ‘filling’ action and is adapted accordingly by inspecting the conceptual subsumption of the terms ‘juice’, which corresponds to the liquid being poured and ‘glass’, which constitutes the goal container of the filling action.

PRAC implements these two paradigms in a coherent probabilistic framework, which



(a) Posterior distribution over the taxonomy conditioned on the Goal role of a Flavoring action.



(b) Posterior distribution over the taxonomy conditioned on the Spice role of a Flavoring action.

Figure 3.9: Posteriors distributions over the taxonomy conditioned on action roles of a seasoning action. More intense node colors indicate higher probability.

automatically finds abstractions of common situations as illustrated in the above examples by exploiting the semantic similarities of concepts in the taxonomy graph. These abstractions reasonably reflect human intuitions of how specific terms are to be used in certain situations. These principles of abstraction and generalization from examples also constitute cornerstones of human cognition (Tenenbaum et al., 2011; Bailey, 1997; Minsky, 1974).

As an implementational framework, PRAC uses MLNs to encode the knowledge about action cores, their action roles, the PRAC dictionary and the PRAC instructions. The knowledge bases are learnt from few hand-labeled instances.

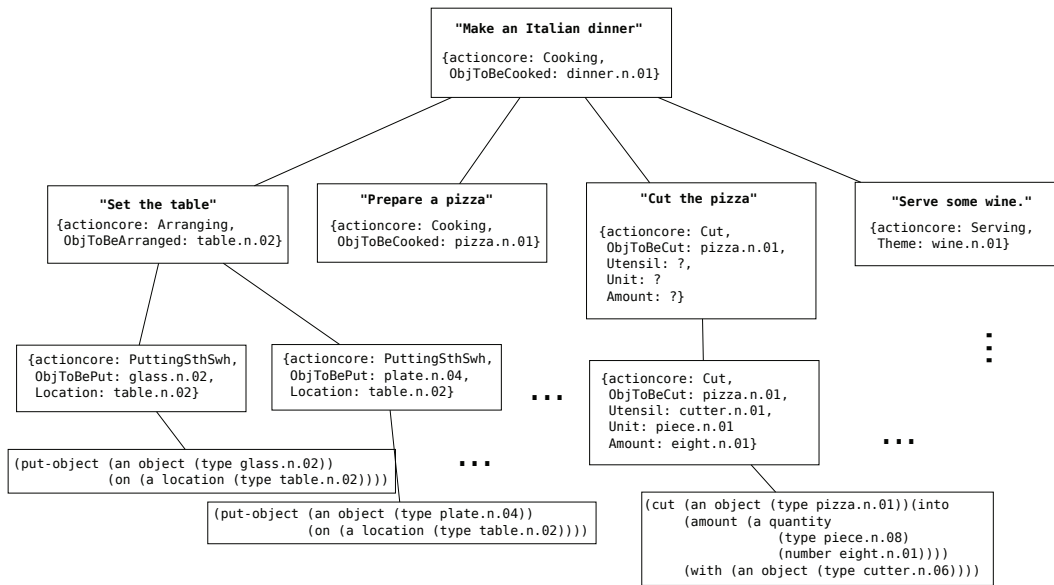


Figure 3.10: Excerpt of the inference tree of PRAC nodes spanned by the PRAC-QUERY algorithm applied to the instruction “Make an Italian dinner.” The root node holds the original, vague instruction and the leaf nodes contain instantiated calls to executable plan schemata.

An example of a distribution over an excerpt of WordNet concepts for possible seasonings is shown in Figure 3.9. The distribution in Figure 3.9a is conditioned on the *Goal* action role of the *Flavoring* action core, whereas the distribution in Figure 3.9b is conditioned on its *Spice* action role. Although the respective MLN has been learnt with only one specific training example, the distribution reasonably generalizes also to superclasses in the PRAC dictionary, i.e. different kinds of food and different subclasses of flavorers.

3.5.1 Reasoning Pipeline

Ideally, it would be desirable to have the reasoning process implemented directly as described by Equation (3.2), which starts by extracting the syntactic structure from an NL instruction, identifies the given action roles and, based on the given roles, infers the missing action roles. However, aggregating all desired inferences into one single reasoning problem of one unifying probabilistic model is infeasible in both representational and computational regards. Consequently, reasonable independence assumptions need to be introduced that allow for independent treatment of subproblems by specialized routines and models. To this end, the reasoning in

PRAC is distributed over several PRAC *modules* that operate on PRAC *nodes*. A PRAC *node* is a data structure wrapping around the results of the probabilistic reasoning, which can be passed around the reasoning modules. All reasoning modules in PRAC have the same functional signature: They accept as arguments one node holding a data structure representing the semantics of an instruction step and they return a sequence of new nodes with possibly refined action steps and augmented semantics. As a result of this one-to-many relationship, the nodes form a tree structure generated by the reasoning process, which I refer to as the PRAC *inference tree*. The root of the inference tree always contains the original PRAC instruction and the leaf nodes contain the executable, disambiguated and completed plan steps. A fragment of an exemplary inference tree is shown in Figure 3.10, which has been generated from the instruction “Make an Italian dinner.” PRAC does not impose any constraints on the shape of the semantic data structures, which makes the proposed pipeline very general. In the current implementation the logical assertions in the form of MLN databases are used, but different semantic representations, such as discourse representation structures (DRS) (Kamp, 2008), for instance, are imaginable.

The overall reasoning process in PRAC is implemented in an algorithm called PRAC-QUERY, which is listed in Algorithm 2. PRAC-QUERY performs recursive expansions of nodes into finer-grained instructions with augmented semantics using inference in probabilistic first-order KBs and analogical reasoning in the PRAC *howto* library described in the previous section. PRAC-QUERY essentially runs different reasoning modules of PRAC, whose processing order is determined by a generator called NEXT-MODULE, which is described below. PRAC-QUERY accepts a list of natural-language commands, which a robot is supposed to execute. In principle, there are no further constraints imposed on the shape, wording or structure of the NL commands. In a first step (line 5-9), the commands are parsed into a PRAC instruction using the parser NL-PARSE. The parse of each instruction represents the initial state of the algorithm, represented by the *fringe*, which is a first-in/first-out (FIFO) queue of nodes that are to be processed next. The loop in lines 10-25 iteratively takes the first *node* from the *fringe* and gets the reasoning module *mod* that is to be applied next to that *node* (line 12). If the pipeline has reached its end, i.e. NEXT-MODULE returns NULL, the state of the node, which at this point in time should not have any missing information pieces, is transformed into an executable plan call by substituting the plan parameters in the respective plan schema. If the pipeline has not reached its end and the current state has not been processed so far, the next reasoning module *mod* is executed on *node*, which yields a new list of nodes, as indicated by the EXPAND function. The returned nodes are appended to the *fringe* for subsequent processing. The check for repeated states ensures that the algorithm will not get stuck in infinite

Algorithm 2 PRAC-QUERY

Input: *instr*: a list of natural-language instructions
Output: a fully instantiated robot plan

```

1: fringe ← [ ]                                     ▷ a FIFO queue
2: steps ← [ ]                                       ▷ holds the parameterized plan calls
3: closed ← {}                                       ▷ already processed states
4: modules ← HASH-MAP()                             ▷ generates the sequence of reasoning modules
5: for i in instr do
6:   n ← MAKE-NODE(NL-PARSE(i))
7:   fringe ← APPEND(n, fringe)
8:   modules[n] = NEXT-MODULE(n)
9: end for
10: while not EMPTY?(fringe) do
11:   node ← POP(fringe)
12:   mod ← modules[node].NEXT(node)
13:   if mod = NULL then
14:     steps ← APPEND(GENERATE-PLAN(node), steps)
15:   else if STATE[node] ∉ closed then
16:     closed ← closed ∪ {STATE[node]}
17:     newnodes ← EXPAND(node, mod)
18:     fringe ← APPEND-ALL(newnodes, fringe)
19:     for each n in newnodes do
20:       modules[n] = COPY(modules[node])
21:     end for
22:   else
23:     return error
24:   end if
25: end while
26: return steps

```

recursion in case of instruction sheets mutually reference one another. Since the fringe implements a FIFO queue and reasoning modules generate new ‘child’ nodes, PRAC-QUERY effectively spans a tree of (intermediate) reasoning results in the form of nodes that are generated in a breadth-first-search (BFS) fashion.

The order in which the reasoning modules in the pipeline are being executed is controlled by a procedure NEXT-MODULE, which in essence implements a generator yielding the individual reasoning modules depending on the current state of the respective *node*. The procedure is listed in Algorithm 3. In the following, I will briefly describe the individual reasoning modules and their role in the interpretation process.

Parsing The first step in PRAC reasoning is to analyze the syntactic structure of the instruction at hand, which yields a PRAC instruction database *I* as described in Section 3.4.3. The grammatical relations retrieved from the Stanford parser and the part-of-speech tags as shown in Equations (3.3) allow to query the PRAC dictionary

Algorithm 3 NEXT-MODULE

Input: *node*: a PRAC inference node

Output: the PRAC reasoning module to be processed next on *node*,
or NULL if the pipeline has finished.

```

1: yield ACTIONCOREINFERENCE           ▶ identifies the action core
2: yield PROPERTYEXTRACTION           ▶ extracts visual properties of objects
3: yield ACTIONROLEINFERENCE         ▶ infers given action roles
4: yield COREFERENCERESOLUTION       ▶ resolves coreferences
5: actioncore ← ACTIONCORE(node)
6: if ROLESNEEDED(actioncore) \ ROLES GIVEN(actioncore) ≠ ∅ then
7:   yield ACTIONROLECOMPLETION       ▶ infer missing roles
8: end if
9: while not EXECUTABLE?(node) do   ▶ do we have a plan schema for the action core?
10:  yield ACTIONCOREREFINEMENT      ▶ refine the action and roles
11: end while

```

for all possible meanings of all words in *I*.

Action Core Inference Given the words, their parts of speech and the possible word senses from the PRAC dictionary, the first probabilistic reasoning problem in the PRAC pipeline is formulated for inferring the most probable action cores that are activated in *I*. One sentence may contain multiple action verbs denoting different instructions. An example of such a sentence is “add two eggs and mix.” The action core inference tags every action verb in a sentence which most likely refers to an actual separate instruction, such as “add_{Adding} two eggs and mix_{Mixing}” in the above example. Not every verb in a sentence, however, refers to an actual purposeful action. Consider the instruction “start with cleaning the dishes.” In this example, the word ‘cleaning’ denotes the action to be conducted, although ‘start’ is the verb in the sentence. Consequently, it is insufficient to merely attach an action core to every verb in a sentence, but more sophisticated statistical reasoning is required to robustly identify actions, which is accounted for by the action core inference module.

Property Extraction Object properties can play an important role in instruction understanding as they may carry information about visual cues of objects, such as shapes, sizes, materials or colors. These characteristics can identify objects or make them discernible from each other. Example statements are “pass me the mid-sized screw” or “my cup is the green one.” But visual properties also can give important information about how to conduct an action by specifying conditions for beginning or stopping with execution, such as “bake until golden brown” or “when the solution turns purple, pour in the imidazole.” The capability of competently interpret and execute instructions is therefore tightly coupled to the capability of extracting and detecting perceptual characteristics of objects in a robotic agent. In Chapter 5, I will

in detail address probabilistic KBs for object perception and the extraction of visual properties from textual descriptions.

Action Role Inference Having identified action cores and their activations in an instruction, the action roles attached to the respective action cores are retrieved from the PRAC knowledge base. The action role inference performs simultaneous word sense disambiguation and action role labeling using FUZZY-MLN reasoning. In Chapter 4, a more detailed treatment of this learning and reasoning task can be found.

Coreference Resolution Consecutive instructions often make references to entities mentioned in previous steps. Such references can be explicit by means of pronouns addressing a particular item, e.g. in the instruction “fill the pot with water and put *it* on the stove,” or implicit, such that references are inferable but not stated, e.g. “put the pancakes on a plate, then serve.” The coreference resolution module disambiguates possible back-references to previous objects. More details are given in Section 3.7.

Instruction Completion If not all action roles have been assigned a value by the action role inference, the instruction at hand is considered under-determined and thus needs appropriate completion. In general there are two possibilities of determining missing action role specifications in PRAC: First, learnt probabilistic first-order models can be used in the way depicted in Section 3.4.1. In this case the semantic representation of the instruction interpretation so far is taken as evidence in a probabilistic query for the most probable assignment of missing roles from all examples seen during training time. Second, as I will describe in Section 3.8, the PRAC howto library can be used for analogical reasoning in an instance-based learning fashion. Here, the knowledge about previously encountered instructions can be transferred to the new situation much more efficiently than by employing purely probabilistic reasoning approaches.

Instruction Refinement In the case that the action cores and roles inferred so far are not executable because there is no plan schema attached in the PRAC knowledge base, the actions must be refined into more specific actions or sequences of actions. In Section 3.8, I will present two complementary approaches that infer refinements of such actions descriptions.

Plan Parameterization If the interpretation does not have missing roles, PRAC checks if there is a plan schema in the PRAC library attached to the action core, which can be parameterized with the inferred roles. If so, the schema is instantiated with its parameters and sent to the plan executive for execution.

3.6 Knowledge Acquisition

In this section, I will describe two ways in which new knowledge can be acquired in PRAC. First I will describe PRAC-TELL, a procedure that can be used to populate the PRAC howto library by means of natural language statements serving as a data pool for an instance-based learning strategy, and second, a strategy for using Amazon Mechanical Turk™ (AMT) for crowdsourcing the process of data acquisition for learning the probabilistic relational models described in the previous sections.

3.6.1 One-shot Learning from Natural Language

Prevalent techniques in the field of machine learning are strongly data-driven. This means, the learning algorithms rely on training data sets of sometimes several hundreds of instances that are required to induce biases to models to generalize sufficiently well. Prominent approaches are, for instance, *deep learning* methods, which have drawn a lot of attention lately (LeCun et al., 2015). In this section I will describe PRAC-TELL, a procedure which transforms NL statements into semantic representations that form entries in the PRAC howto library.

For many learning tasks in the real world huge amounts of data are often not available for training, and if they are, they are mostly tailored to very specific problem instances like the recognition of hand-written digits, for example. In addition, the learning is often critical with respect to time and the number of training instances, because, as Krause et al. (2014) point out, agents must be able to “acquire new knowledge quickly, on the fly, during task performance. Hence, we need to augment data-driven methods with other methods that allow for online learning from possibly only a few exemplars.” Researchers in the field of Bayesian cognition have also found that

“People learning new concepts can often generalize successfully from just a single example, yet machine learning algorithms typically require tens or hundreds of examples to perform with similar accuracy. People can also use learned concepts in richer ways than conventional algorithms—for action, imagination, and explanation.” — (Lake et al., 2015)

One-shot learning, i.e. learning from very few, or even only one example, has been mainly applied to problems in the field of computer vision and image understanding. However, learning from one shot only is also highly relevant for robotic agents

Algorithm 4 PRAC-TELL

Input: *instr*: a list of natural-language instructions
Output: a sequence of nodes holding semantic representations of the action cores and roles given in *instr*

```

1: fringe ← []                                     ▶ a FIFO queue
2: modules ← HASH-MAP()                           ▶ generates the sequence of reasoning modules
3: steps ← []                                       ▶ holds the nodes
4: for i in instr do
5:   n ← MAKE-NODE(NL-PARSE(i))
6:   fringe ← APPEND(n, fringe)
7:   modules[n] = NEXT-MODULE(n)
8: end for
9: while not EMPTY?(fringe) do
10:  node ← POP(fringe)
11:  mod ← modules[node].NEXT(node)
12:  newnodes ← EXPAND(node, mod)
13:  if mod = COREFERENCE RESOLUTION then
14:    steps ← APPEND-ALL(newnodes, steps)
15:  else
16:    fringe ← APPEND-ALL(newnodes, fringe)
17:    for each n in newnodes do
18:      modules[n] = COPY(modules[node])
19:    end for
20:  end if
21: end while
22: return steps

```

and assistants in near-future applications. For example, a robot in a household environment is expected to adapt its knowledge to the needs and preferences of its users and the particularities of the environment. It is, for instance, unacceptable for such a robot to collect many examples of letting milk perish before it learns that milk and other dairy products have to be stored in the refrigerator. Likewise, it has to accommodate the knowledge where the cutlery is stored or that I take my coffee with milk and sugar on the fly, without the need to relearn and to process huge amounts of data. Even more safety-critical, for a chemical laboratory assistant, it is not tolerable at all to let it collect negative samples of experiments with dangerous substances for learning. In PRAC, one-shot learning refers to the problem of adjusting the free action roles in an under-determined instruction. The problem of using the PRAC howto library for instruction completion is described in detail in Section 3.8.5.

PRAC provides a TELL interface for additively populating the howto library with knowledge in the form of instances of the *Howto* data structure described in Section 3.4.6. The single entries in the howto library are used as the data pool in an instance-based learning fashion. An algorithm called PRAC-TELL allows to transform NL instructions into *Frame* instances, which are in turn stored in a database for

later retrieval during reasoning. PRAC-TELL is a modification of the PRAC-QUERY algorithm which only partially executes the reasoning pipeline on the respective instruction. The PRAC-TELL algorithm is listed in Algorithm 4. After NL-PARSING, it passes through the first four reasoning modules of the NEXT-MODULE generator, i.e., ACTIONCOREINFERENCE, PROPERTYEXTRACTION, ACTIONROLEINFERENCE, and stops after the COREFERENCERESOLUTION. In contrast to PRAC-QUERY, PRAC-TELL does not infer information that is unspecified in an instruction. The rationale behind this is that, as the instructions are being used as supporting points for instance-based learning, it makes sense to only include the definite knowledge that has been entered as factual knowledge. Moreover, keeping certain gaps in the action specifications gives more freedom and flexibility for the matching process during reasoning later. Using the PRAC-TELL interface, it is possible to acquire knowledge from NL instructions such as “season steaks with salt and pepper” and “season pancakes with sugar and cinnamon.”

In the current implementation, the PRAC-TELL algorithm was used to populate the PRAC howto library with the complete set of more than 8,400 recipes from the [wikihow.com](http://www.wikihow.com) website comprising in total more than 103,000 sentences.

3.6.2 Data Acquisition with Amazon Mechanical Turk™

Besides the instance-based learning approach using the PRAC howto library and the PRAC-TELL algorithm, PRAC uses conventional data-driven learning techniques for parameter estimation in MLNs to train the probabilistic models. Existing corpora with ground truth for applications in NL understanding, however, are inapplicable to PRAC as they consist of mostly newspaper or encyclopedic articles, which are of a fundamentally different nature than instruction sheets for everyday activities. In this section, I report on a study that has been conducted in collaboration with Epping (2011), who uses the crowdsourcing AMT marketplace for endowing the [wikihow.com](http://www.wikihow.com) instructions with semantic labels. In short, the approach works as follows. In a first step, a collection of NL instructions is parsed using the Stanford parser. Given the parse and part of speech, WordNet is used for obtaining a selection of possible word meanings and semantic roles that each word can have. This selection is posted as a human intelligence task (HIT) to AMT, where human workers can manually select the correct word senses for nouns, verbs, adjectives and adverbs occurring in the natural-language instructions. As a result, a semantically annotated corpus of word senses has been obtained that can be used to train the PRAC models described above.



Crowdsourcing

There are a couple of hand-labeled corpora that researchers from the field of computational linguistics have created for training and evaluating statistical language models. These corpora mainly differ in the aspects of language they cover. Some focus on the problem of word-sense disambiguation, such as the *Semantic Concordance* (Miller et al., 1993) or mere syntactic analyses of sentence structures, like the *Penn Treebank* (Marcus et al., 1993) and the *SensEval* (Kilgarri, 1998) datasets. Others aim at attaching labels to words which assign them certain semantic roles. An example is the *Proposition Bank* (Palmer et al., 2005), which is the perhaps most widely used corpus. Although these corpora provide comprehensive data sets, they all have in common that the underlying data sources are newspaper articles or lexical entries from encyclopediae. These texts are both syntactically and semantically too different to cooking recipes or instruction sheets, so they are hardly applicable to the PRAC application domains. As to the best of our knowledge, none of the available corpora accounts for the kind of knowledge that is required for learning in PRAC, we decided to acquire an own corpus of annotated robot instructions.

Crowdsourcing Conventional approaches to collect corpora of annotated data have some serious drawbacks: One way to collect the data is to set up a team of human annotators, who build up the required corpus, which is very time-consuming, tedious and costly. A new form of delegating the task of data acquisition to a wide range of people is the so-called *crowdsourcing* approach, which has emerged in the recent years. It describes the act of outsourcing a task to an unknown, large group of people or a community (i.e. the *crowd*). Members of the crowd accomplish small tasks requiring human intelligence (so-called human intelligence tasks (HITs)) for a small payment via online platforms. The most famous crowdsourcing platform is Amazon Mechanical Turk™ (AMT). These so-called *micro-task markets* provide an on-demand, scalable workforce for only small monetary costs. Consequently, this approach has the advantages of being very flexible, inexpensive and independent and while most knowledge sources on the Web are static and read-only, crowdsourcing provides the chance to actively use the Web by asking the crowd custom questions and thus allows to collect precisely the data of interest. In addition, a great number of previous works has shown promising and encouraging results.

Task Architecture A set of more than 1,400 NL instructions from the “Food & Entertaining” category of *wikihow.com* have been selected for semantic labeling. This set of instructions comprises the action verbs *add*, *cut*, *fill*, *flip*, *mix*, *place*, *pour* and *put*, which all belong to the most frequent actions verbs I reported on in Section 3.3.1. In this study, we constrained ourselves to the acquisition of word senses and skip the

semantic role labeling task. The architecture of a HIT has been designed as follows. Consider the sentence “stir/VB the/DT mixture/NN into/IN a/DT bowl/NN,” where each word has been assigned its part of speech using the Stanford parser. Given the part of speech, WordNet can be queried for all possible word senses (synsets) for each word. For example, the noun ‘mixture’ has five possible meanings in WordNet:

1. **mixture.n.01**: (chemistry) a substance consisting of two or more substances mixed together (not in fixed proportions and not with chemical bonding)
synonyms: -
examples: -
2. **concoction.n.01**: any foodstuff made by combining different ingredients
synonyms: *concoction, mixture, intermixture*
examples: *“he volunteered to taste her latest concoction”; “he drank a mixture of beer and lemonade”*
3. **assortment.n.01**: a collection containing a variety of sorts of things
synonyms: *mixed bag, miscellany, miscellanea, variety, salmagundi, smorgasbord, potpourri, motley*
examples: *“a great assortment of cars was on display”; “he had a variety of disorders”; “a veritable smorgasbord of religions”*
4. **mix.n.02**: an event that combines things in a mixture
synonyms: *mix, mixture*
examples: *“a gradual mixture of cultures”*
5. **mix.n.03**: the act of mixing together
synonyms: *mix, commixture, admixture, mixture, intermixture, mixing*
examples: *“paste made by a mix of flour and water”; “the mixing of sound channels in the recording studio”*

The list of possible meanings is transformed into a HIT, in which the subjects can just tick the most appropriate word meaning. The task is now to determine the most appropriate sense of the word ‘mixture’ in the given context. The correct answer to be expected as a label is sense no. 2, i.e. “any foodstuff made by combining different ingredients.” As the POS tagging is not 100% accurate and there might be appropriate word senses missing in WordNet, two respective additional options have been added to list of possible answers. Figure 3.11 shows a screenshot of a generated HIT available on AMT.

Pour blueberry mixture into **prepared** pie crust.

Please select the most appropriate sense for the **Adjective prepared** in the sentence above.

- Wrong part of speech provided**
- made ready or fit or suitable beforehand** "a prepared statement"; "be prepared for emergencies"
- having made preparations** "prepared to take risks"
- equipped or prepared with necessary intellectual resources** "graduates well equipped to handle such problems"; "equipped to be a scholar"
- No matching sense found**

Figure 3.11: Example of a human intelligence task on the Mechanical Turk marketplace

Evaluation and Quality Assessment Using AMT we labeled around 1,030 sentences with an average of 5.5 words per sentence. In total, 448 workers took the qualification test which we demanded, out of which 183 have passed. Every word was labeled by 3 different workers, where we applied a majority voting scheme to obtain the final decision. To estimate the accuracy to the collected data, 110 randomly selected annotations have been reviewed by an expert. The results show an accuracy of 94.5% with majority voting against 70.9 % accuracy of a single worker. A Binomial Test with significance level $\alpha = 0.05$ suggests that the overall accuracy of the whole data acquired is at least 92%.


3.7 Coreference Resolution

Typically, single instructions in NL howtos are not self-contained and thus do not make much sense when regarded in isolation. In many cases, instructions make references to previous instructions implicitly or explicitly, mostly by referring to previously mentioned objects in the form of pronouns like *it*, *they*, or *them*. In this section, I report on a novel probabilistic approach for resolving such coreferences across instructions that has been jointly developed with Meyer (2013).

Consider the following recipe for boiling pasta:

How to Cook Pasta

1. Put the pasta into a pot.
2. Fill it with water and add a pinch of salt.
3. Cook for 5 minutes.
4. When they are done, let the noodles drain in a sieve.
5. Serve on a plate.

 **Explicit Coreference** The computational linguistics literature distinguishes at least three different kinds of coreferences (cf. Crystal, 2011): *Anaphora* denote back-references from a pronoun to a noun phrase mentioned antecedently. An example of an anaphora is in instruction Step 2, where ‘it’ refers to the cooking pot introduced in Step 1. Conversely, *cataphora* denote forward-references of pronouns anteceding a noun phrase, such as ‘they’ in Step 4. A third kind of coreference, in which no pronouns are used, is given by different noun phrases that paraphrase the same entities they refer to. The term ‘noodles’ in Step 4, for example, references the same entity as the word ‘pasta’ in the first sentence. What all these types of coreferences have in common is that the coreferences are definite in the sense that there is an explicit term in the statement whose coreference needs to be resolved. However, we can identify an additional type of coreference, which is not immediately visible: Consider the instruction in Step 3, “cook for 5 minutes,” which does not contain any of the aforementioned coreferences. Yet it refers to the pasta introduced in Step 1 and the water from Step 2, which need to be boiled for 5 minutes. Likewise, adding a pinch of salt in Step 2 refers to the water in the pot from the first instruction, and the last step relates to the noodles mentioned in the preceding instruction. I refer to these kinds of coreference as *latent coreferences* for their existence, as opposed to the classical explicit coreference, is not directly evident in a sentence. In this work, I only consider explicit and latent anaphora.


 **Latent Coreference**

Figure 3.12 illustrates in minimalistic semantic networks the four action cores *Putting*, *Filling*, *Cooking* and *Serving* from the above recipe for cooking pasta. Action roles that are explicitly referred to in the instructions are highlighted in colors, whereas missing roles are denoted by gray boxes with a question mark inside. This semantic representation gives rise to a novel account to coreference resolution: Instead of relying on pronouns explicating coreferences in an NL sentence, undefined action roles from the PRAC knowledge base can serve as evidence for coreferring entities that must be resolved. This enables to not only identify explicit but also latent coreferences

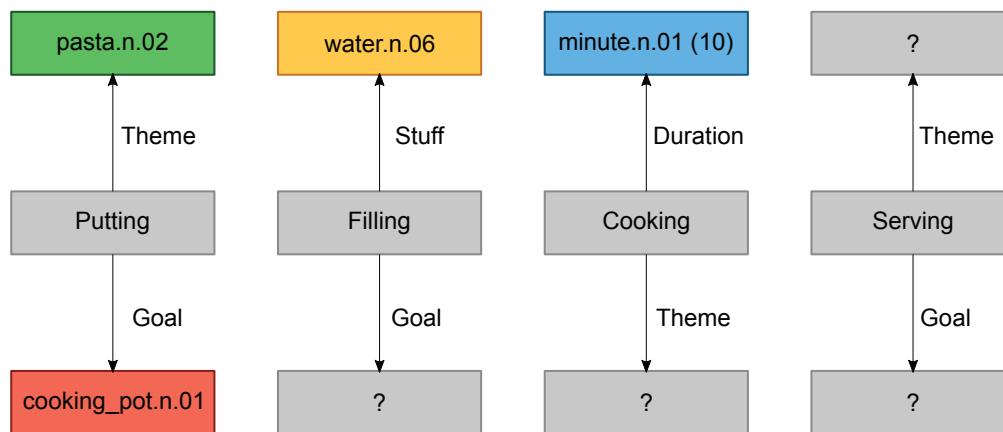


Figure 3.12: Graphical representation of the coreference resolution problem in a semantic network: The colored roles are given in the original instructions, coreferences can be identified by missing action role assignments.

that are implicit in the instructions and coreference resolution can be formulated as the problem of assigning action roles to entities in antecedent instructions. It therefore makes sense to execute the COREFERENCERESOLUTION reasoning module in the PRAC pipeline *after* the existing roles in the instruction have been assigned, but *before* the role completion using the general PRAC knowledge base and the PRAC howto library takes place (cf. Algorithm 3). The basic assumption is that action roles that cannot be assigned in the current instruction under consideration might be given in the context of surrounding instructions, and if they cannot be found in the NL context, one has to fall back to general background knowledge.

3.7.1 Probabilistic Coreference Resolution

In this section, I will describe how a probabilistic model can be constructed that is able to resolve coreferring terms in an instruction sheet by computing probability distributions over potential action role fillers in preceding instructions. I begin with introducing the constituents of the probabilistic model.

The building blocks that are used to generate the probabilistic reasoning models can be roughly divided into the following groups:

Syntactic Relations Syntactic relations are relations between words in the text or assignments of properties to those words as given in the PRAC instruction as introduced in Section 3.4.3, which also comprises the part-of-speech (POS) tags of all words in all PRAC instructions of a complete NL recipe. Moreover, several syntactic

dependencies between the words in a sentence can be identified. It is important to note that when considering multiple subsequent instructions, the symbols within a PRAC instruction generated by the Stanford parser are not globally unique anymore, since they only convey the index of a word within a particular sentence. Thus, the symbols are extended by the index of the sentence. For example, the symbol denoting the word ‘minutes’ in the above recipe is *minutes-4-3* instead of *minutes-4* as it is the fourth word in the third sentence.

Taxonomic Knowledge Taxonomy relations refer to the prior knowledge about the subsumption relation of concepts in the PRAC dictionary. For example, the concept *container.n.01* is in a hyponymy relationship with the concept *cup.n.01* since the concept of a ‘cup’ is a specialization of the concept of a more generic ‘container’.

Action Roles Action roles from the PRAC knowledge base refer to the partial evidence and queries that are instantiated in order to identify the abstract parameterization of an activity in an instruction in context of the action core referred to by the sentence.

Distance Relations Distance relationships are established between all pairs of words to determine the number of sentences that lie between the two words. For example, in our introductory example of making pasta, the word ‘pasta’ from the first action step and the word ‘it’ in the second step have a distance of 1, whereas ‘it’ and ‘water’ have distance 0 for they appear in the same sentence. It has been shown by McEnery et al. (1997) that within three sentences roughly 90% of all coreference relationships can be found. We therefore assume that, for the purpose this work, it sufficient to consider a constant ‘sliding’ window of 4 sentences, i.e. for resolving anaphora in a particular instruction, the objects appearing in the respective 3 sentences in front are considered. The abovementioned distances thus can be asserted using statements such as

$$\begin{aligned} & \textit{distance}(\textit{pasta-3-1}, \textit{it-2-2}, 1) \textit{ and} \\ & \textit{distance}(\textit{it-2-2}, \textit{water-4-2}, 0) \end{aligned}$$

for the example words from above. As the probability is expected to decay with increasing distance, a separate predicate is introduced that allows explicit incorporation of distances in the model. However, the distances 1 and 0 do not have arithmetic semantics in this case but denote plain symbols in a Boolean predicate. The decay in probability thus needs to be established by means of the model structure and parameters, as I will show in the next section.

Coreference Relations Statements about the coreference of two words can be made in the form $coreference(w_1, w_2)$, which states that the words w_1 and w_2 refer to the same entity. In order to enable reasoning about the action roles that are undetermined in an instruction, a Skolem constant in the fashion presented in Section 3.4.1 is introduced. Using this Skolem constant it is possible to explicitly reason about the roles missing and determine their coreference relationships.

In consequence, the probabilistic reasoning task is to compute the most probable assignments of atoms of the *coreference* predicate comprising all words from the three antecedent instructions as well as the Skolem constants introduced for the unassigned action roles.

Several semantic features are taken into consideration when inferring coreference. The word sense and the path through the taxonomy of two words are among the most important ones. Two words are more probable to be in coreference if the semantic distance is small. If the semantic distance is zero, two concepts are exactly the same. Different similarity measures for the semantic distance exist like the WUP similarity by Wu and Palmer (1994), which is discussed in more detail in Chapter 4.

Model Structure It is reasonable to assume that the *coreference* relation is transitive and symmetric, which can be implemented as hard formulae in MLNs, i.e. a formula that always needs to hold:

$$\begin{aligned} coreference(w_1, w_2) \wedge coreference(w_2, w_3) &\rightarrow coreference(w_1, w_3). \\ coreference(w_1, w_2) &\rightarrow coreference(w_2, w_1). \end{aligned}$$

Moreover, certain combinations of POS tags usually are not in a *coreference* relationship, e.g. it is unlikely that nouns corefer with adjectives. Conversely, it is assumed that the part of speech (POSS) of coreferring terms often coincide. Consequently, the following formula template is used in the model that attaches a weight to each possible combination of POS tags:

$$coreference(w_1, w_2) \wedge has_pos(w_1, +pos_1) \wedge has_pos(w_2, +pos_2)$$

Note that the ‘+’ operators in front of the variables pos_1 and pos_2 produce one separate formula with an individual weight for every combination of POS tags.

Coreference between two words correlates with the action roles, the word senses and the distance between the words. This relationship is captured in the formula

$$has_role(w_1, +r_1) \wedge has_role(w_2, +r_2) \wedge coreference(w_1, w_2) \wedge distance(w_1, w_2, +d)$$

The formula will be expanded over the roles and distances before the parameter learning takes place and consequently makes the roles an explicit part of the model. As a result, the model size increases quadratically in the number of roles.

3.7.2 Experiments

For the experiments, a total of 20 short yet representative texts containing a total of 57 instructions have been manually created and annotated. Each set of instructions contains on average about 12 words. It was necessary to draw on instruction sheets of this small size due to the computational complexity the number of words impose on the learning and reasoning¹. Three exemplary howtos are the following:

How to Serve a Coke	How to Make Instant Coffee	How to Serve Soda
1. Add coca cola to a glass.	1. Fill a coffee mug with water.	1. Fill a bottle with soda-water.
2. Afterwards mix with ice.	2. Add instant-coffee.	2. Add lime.
3. Serve on a tray.	3. Serve.	3. Serve.

A complete list of howtos used in the experiments can be found in Meyer (2013). The howtos comprise the action cores *Putting*, *Adding*, *Mixing*, *Filling* and *Serving*, which are among the most frequently used action verbs according to Section 3.3.1, and each sentence only contains one verb. Every howto contains a unique set of objects that do not exist in another howto. In all howtos, a total of 67 different objects are used. For inference, all predicates that are part of the evidence are assumed to fulfill the closed-world assumption, i.e. if they do not appear in the evidence, they are assumed to be *false*. In the experiments, the word senses (denoted by the *has_sense* predicate) were provided in the query databases. This is a reasonable assumption as, in the overall PRAC reasoning pipeline, the word senses are jointly inferred by the

¹The experiments have been conducted before the inception of FUZZY-MLN reasoning. Since they allow more compact model representations, it is expected that more complex cases can be covered using FUZZY-MLNs.

\emptyset # of formulae	# Test DBs	# Training DBs	\emptyset F1	\emptyset Precision	\emptyset Recall
1635	2	18	0.690	0.617	0.851
1615	10	10	0.600	0.522	0.762
1183	15	5	0.558	0.473	0.782
1021	17	3	0.513	0.428	0.778
816	18	2	0.442	0.363	0.752

Table 3.6: Experimental results of the coreference resolution experiments.

ACTIONROLEINFERENCE module (cf. Algorithm 3).

The 20 databases are split in sets of different sizes, on which we performed cross validation with differently large portions of training and test sets. The results are shown in Table 3.6. In the experiment, the results show that expectedly, the run with 18 training databases receives the highest scores for recall as well as precision and the lowest for the run with only two training databases. The experiments show that the simple model introduced in this section is in principle able to effectively learn the coreferences in NL howtos.

Limitations Extremely challenging scenarios are instructions in which not all word senses are available or even no word sense is available at all. For example, in the howto

How to Make a Fruit Salad (1)

1. Mix the fruits. Location?
2. Serve. Theme?

As the location of the *Mixing* activity is not explicitly mentioned, e.g. a bowl, it is impossible for the coreference model to infer coreferences of the subsequent *Serving* action. An additional difficulty with coreference exists when identifying individual objects and a word occurs multiple times in a set of instructions but they refer to different object entities. For example, the instruction set:

How to Make a Fruit Salad (2)

1. Add fruits to a bowl.
2. Mix. Place?
3. Serve in a bowl.

One interpretation of the instructions could be that the word ‘*bowl*’ in the first sentence refers to a mixing-bowl where the ingredients are mixed. The word ‘*bowl*’ in the third sentence on the other hand might refer to an eating-bowl that is used for serving. This is difficult to handle and particularly difficult to handle only with syntactic features. Consequently, the semantic information is a key element in the correct resolution of the coreference. In the given example a possible resolution is that the term ‘*bowl*’ in the first sentence has the sense *mixing_bowl.n.01* and in the last sentence it has the sense *salad_bowl.n.01*. This already indicates that these two are not the same real world object. Moreover, even in the absence of the senses, the roles of the two words can indicate whether it is probable that they are in coreference or not.

3.8 Instruction Completion & Refinement

One of the key challenges in instruction interpretation addressed by PRAC is the specification and refinement of incomplete information in NL statements. Consider an instruction like “season a steak,” or “season the soup” as parts of cooking recipes. A robotic agent needs to know that it should use salt and pepper for spicing steaks and that it can achieve this goal by pouring from a spice jar or a pepper mill, respectively. Conversely, the topping of a pancake should be sugar rather than salt and pepper, for instance. In addition, the robot should also be

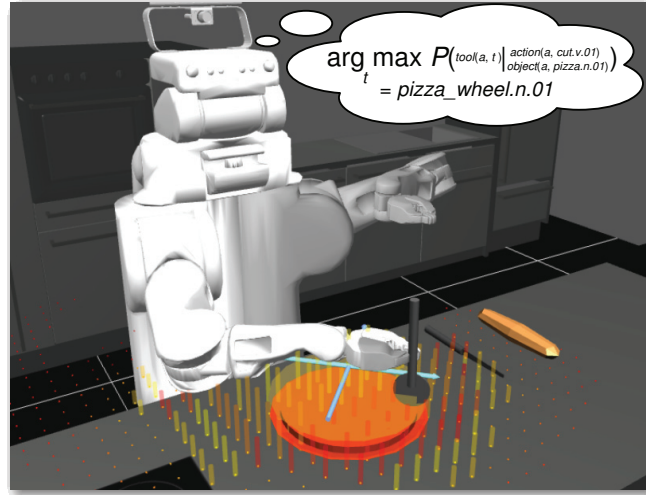




Figure 3.13: “Cut the pizza!” – the simulated PR2 robot generating and solving for a query for the most probable completion with respect to the cutting tool.

able to transfer its knowledge about specific actions and objects to new situations and new objects that it has not seen before. The knowledge about pancake toppings, for instance, could be transferred to waffles and donuts as well. In Section 3.4.1 and Section 3.5.1, I have already shown how distributions over ontological concepts can be used to infer missing action roles. However, the enormous size of these models and the computational expense in learning and reasoning often impedes the practical applicability of purely probabilistic models to real-world domains. I will review two approaches for the tasks of completing and refining instructions. The first approach is a purely probabilistic one, making use of trained MLN KBs. I propose a novel method of inferring missing information pieces for instruction completion, which implements knowledge transfer by semantic analogical reasoning in the PRAC howto library of NL instruction sheets, while instruction completion can be achieved through fast database queries. Parts of the work presented in this section build upon the work by Koralewski (2016) and have been published in Nyga et al. (2017b).

3.8.1 Probabilistic Completion & Refinement

 *Instruction Refinement* As described in Section 3.4.5, not all action cores are directly executable and have a plan schema attached, but need further semantic analysis. An obvious example is the *Adding* action core, which is, with respect to execution, too generic an action to be directly executable. Consider the many ways in which different types of objects can be added, e.g. water, milk, salt, pepper, or baking soda. Consequently, action cores with certain roles assigned need to be refined in order to describe more concrete activities. For example, an instruction “add_{Adding} a pinch_{Amount} of salt_{NewMember}” should be refined to a *Scooping* or *UsingSpiceJar* action, whereas an instruction “add_{Adding} a splash_{Amount} of milk_{NewMember}” can be most likely performed by *Pouring*. To account for this kind of action refinement, PRAC maintains a special binary predicate *AchievedBy*(*a*, *a'*), which is used to state that an action *a* can be achieved by a different action *a'*. PRAC maintains probability distributions over the *AchievedBy* relation, which have been learned from manually annotated modal clauses found in NL recipes. Examples of such modal clauses are “flip the pancake *using a spatula*,” or “*use a pipette* to add 5 drops of HCl.” An example of a distribution over the *AchievedBy* predicate for the *Adding* action is shown in Table 3.7. The table shows three possible refinements of the *Adding* action core, namely *Pipetting*, *Pouring*, and *OperatingTap* (the query to the PRM of the *AchievedBy* relation), and different concepts from the PRAC dictionary filling the *NewMember* and *Amount* role slots of the *Adding* action core. The probability distributions show that PRAC learns reasonable biases with respect to the role arguments: Depending on the amount and the type of substance to be added, the distributions suggest different refinements: small amounts of chemical substances (*NewMember*(*hcl.n.01*), *Amount*(*milliliter.n.01*)) can be added with a *Pipetting* action, larger amounts of beverages like juice (*NewMember*(*fruit_juice.n.01*), *Amount*(*liter.n.01*)) with a *Pouring* action, but the tap is also a possible source of water (*NewMember*(*water.n.06*), *Amount*(*liter.n.01*)). Having computed a refinement of the action, the roles of the respective refined action core are assigned in a separate reasoning module.

 *Instruction Completion* Although formulating the problem of action completion as a query in form of a conditional probability distribution seems natural and elegant at first, such a purely probabilistic approach suffers from the representational and computational costs incurred by the learning and reasoning. Consequently, the generality of KBs and the transferability of knowledge to new situations is only given within limits. Figure 6.4d shows a scenario, in which a PR2 robot is to slice a pizza. Depending on the current context, the robot needs to select an appropriate action to conduct, which is not explicitly mentioned in the instruction, however. Symbolic probabilistic models need



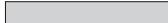










E	Q	P(Q E)	
			
<i>NewMember(water.n.06)</i> <i>Amount(milliliter.n.01)</i>	<i>AchievedBy(Adding,Pipetting)</i>		0.01
	<i>AchievedBy(Adding,Pouring)</i>		0.68
	<i>AchievedBy(Adding,OperatingTap)</i>		0.31
<i>NewMember(water.n.06)</i> <i>Amount(liter.n.01)</i>	<i>AchievedBy(Adding,Pipetting)</i>		0.00
	<i>AchievedBy(Adding,Pouring)</i>		0.40
	<i>AchievedBy(Adding,OperatingTap)</i>		0.60
<i>NewMember(fruit_juice.n.01)</i> <i>Amount(liter.n.01)</i>	<i>AchievedBy(Adding,Pipetting)</i>		0.00
	<i>AchievedBy(Adding,Pouring)</i>		0.96
	<i>AchievedBy(Adding,OperatingTap)</i>		0.04
<i>NewMember(hcl.n.01)</i> <i>Amount(milliliter.n.01)</i>	<i>AchievedBy(Adding,Pipetting)</i>		0.99
	<i>AchievedBy(Adding,Pouring)</i>		0.01
	<i>AchievedBy(Adding,OperatingTap)</i>		0.00

Table 3.7: Probability distribution over the *AchievedBy* predicate for different parameterizations of *Adding*: Different specifications of the *Liquid* and the *Amount* result in different probabilities of action core refinements.

to explicitly represent in the model every symbol subject to reasoning. For example, a probabilistic model that is supposed to consider the *bread_knife.n.01* as the most probable tool for cutting bread, and the *pizza_wheel.n.01* the most probable tool for cutting pizza, more formally,

$$\begin{aligned} \arg \max_t P \left(\text{tool}(a, t) \mid \begin{array}{l} \text{action}(a, \text{cut.v.01}) \\ \text{object}(a, \text{bread.n.01}) \end{array} \right) \\ = \text{bread_knife.n.01} \end{aligned} \quad (3.8)$$

and

$$\begin{aligned} \arg \max_t P \left(\text{tool}(a, t) \mid \begin{array}{l} \text{action}(a, \text{cut.v.01}) \\ \text{object}(a, \text{pizza.n.01}) \end{array} \right) \\ = \text{pizza_wheel.n.01}, \end{aligned} \quad (3.9)$$

must at least contain representations of the respective combinations of atoms in Equations (3.8) and (3.9) in its symbolic structures. If we consider real-world applications such as human household and cooking scenarios, constructing probabilistic KBs

over such large domains is hopelessly infeasible as the number of required symbolic combinations would be intractable. Consequently, *deep knowledge transfer* (Davis and Domingos, 2009) with purely probabilistic symbolic models to new situations is not always straightforward. As another related example, consider an instruction like “pour milk into a glass,” which a probabilistic KB has been trained with. A desirable answer from this KB to a query for the most probable destination container for a “pour juice” instruction is also the glass. However, unless the combination of *juice* and *glass* is not represented explicitly in the model, the desired result cannot be computed in such a model. In fact, the model is inapplicable to such queries. The FUZZY-MLN framework introduced in Chapter 4 exploits the semantic similarity of concepts in a taxonomy in order to support off-domain reasoning, i.e. reasoning about symbols that are not present during model construction. Examples of such distributions are shown in Figure 3.9. FUZZY-MLNs are able to compute posterior beliefs about unseen concepts, but they are unable to efficiently and constructively compute novel solutions in open domains. Consequently, this kind of reasoning does not bring about the desired results since the completion still can only be performed on concepts in the model.

Following these considerations, employing probabilistic models for the deep transfer from one scenario to another does not seem as straightforward as it may, at first, have appeared.

3.8.2 Reasoning by Analogy

A different way of refining an under-determined action which is not directly executable is applying role completion, plan expansion, and plan adaptation by analogical reasoning. The role completion simply looks up potential slot fillers in a large library of instructions that have worked before in similar situations, i.e. the PRAC howto library. The plan expansion resolves references across different howtos by replacing one particular step in an instruction sheet by the steps that PRAC finds an instance of a howto in the howto library. For example, when a command “cook the pasta” is encountered as part of a recipe, PRAC recursively expands this command to the single steps “fill a pot with water,” “put it on the stove” and “switch on the stove, then insert the pasta,” for instance, which can be found in the PRAC howto library.

The basic assumption of the analogical reasoning approach is that the answers to the queries can be found elsewhere in a large collection of recipe documents and can be easily ‘looked up’. Moreover, accessing semantically not only action roles

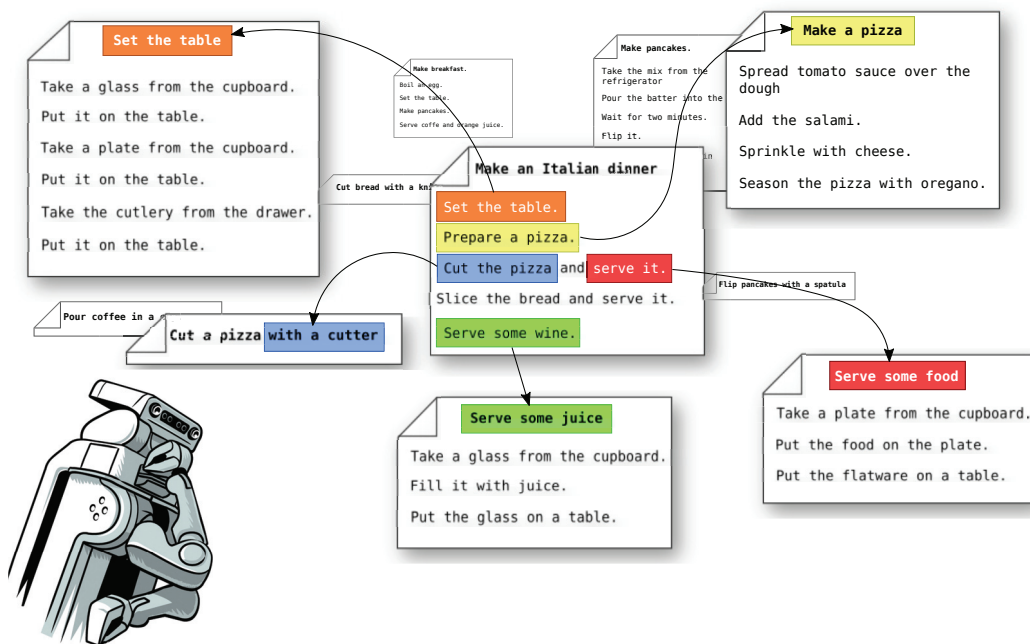


Figure 3.14: Illustration of the analogical reasoning: information pieces that are missing in an instruction can be found in other documents. Instructions can be refined by looking up whole howtos in the howto library (e.g. “set the table”), some other howtos have to be adapted to the current situation (e.g. “serve some food.”) or selected action roles (e.g. the instrument for cutting a pizza) can be retrieved from other documents.

of single actions but also instruction sheet titles allows to recursively refine single instruction steps referring to whole instruction sheets. Examples of such reasoning tasks are illustrated in Figure 3.14. Starting from an instruction sheet for making an Italian dinner, several steps in this howto may refer to other instruction sheets, such as the directive to “set the table,” for which probably a separate document exists. Other howtos that exist are written for manipulating different but similar objects and can be adapted to the current recipe under consideration. The “Serve some juice” howto, for instance, can be easily adapted to also work with wine. By establishing such analogies on a semantic level, the knowledge about specific scenarios can be transferred to new situations with possibly unknown objects. The ultimate advantage over purely probabilistic reasoning methods lies in the scalability of learning and the ease of extensibility: While the probabilistic models for understanding a *given* sentence can be represented parsimoniously, new knowledge can be easily taught by adding new instances to a database and undesired or flawed knowledge can be ‘forgotten’ by removing the faulty instances.

Researchers from the cognitive science and cognitive psychology found that “Analogical reasoning – the ability to perceive and use relational similarity between two situations or events – is a fundamental aspect of human cognition,” (Gentner and

Smith, 2012) and is frequently used by humans essentially in three ways. One use of analogies aims at explaining a situation that cannot be directly seen by establishing an analogy to a familiar situation, so it becomes easier to grasp. A frequently used analogy to explain the concept of an algorithm is, for example: “think of an algorithm as a cooking recipe for computers.” The projection of the new concept ‘algorithm’ into the familiar domain ‘cooking’ makes it easier to grasp for most non-computer scientists. A second use case is in argumentation, where analogies are used to reinforce propositions or to refute claims by reducing it to absurdity (e.g. “You’re comparing apples to oranges!”). A third use case is *abstraction*: From several instances of relational structures, one might abstract away to more generalized schemata that are transferable to different domains.

“Similarity and analogy are fundamental in human cognition. They are crucial for recognition and classification, and have been associated with scientific discovery and creativity. Successful learning is generally less dependent on the memorization of isolated facts and abstract rules than it is on the ability to identify relevant bodies of knowledge already stored as the starting point for new learning.”

— (Vosniadou and Ortony, 1989)

An excellent overview on the structure and systematic use of analogies in human reasoning is given by Gentner and Smith (2012). According to them, analogical reasoning “involves identifying a common relational system between two situations and generating further inferences driven by these commonalities.” In general, the reasoning process can be divided into three parts: (1) *Retrieval* denotes the current new situation in memory, which triggers reminders to potential analogs in the long-term memory. (2) *Mapping* involves the alignment of two representations and the inference by projecting structural parts of the reminded into the new structure, and in (3) *Evaluation* the analogical mapping and inference is judged. The central aspect in analogy reasoning is the process of mapping from a well-known *source/base* situation to a less familiar *target* situation by similarity of the *relational structures* of the two situations. Thereby, inference happens as a byproduct of the structural alignment process:

“Once the base and target have been aligned and their common relational structure found, if there are additional parts of the relational pattern in the base that are not present in the target, then this missing pattern will be brought over as a candidate inference [...]. Thus, one way to think about inference generation is as a process of relational pattern completion.”

— (Gentner and Smith, 2012)

The structural alignment and inference process in the “season the steak” example

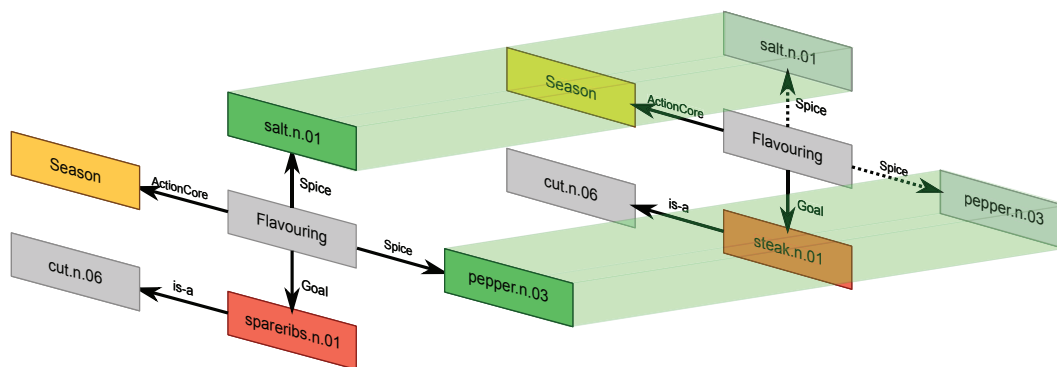


Figure 3.15: Exemplary role completion of the instruction “season the steak.” by analogical reasoning: The action verb and the Goal role are given in the target structure (orange/red), whereas the appropriate seasonings like salt and pepper need to be inferred. Inference is being achieved by projecting the missing roles from a structurally similar source network from the PRAC howto library, which provides the missing roles salt and pepper. The inferred Spice roles are drawn with dashed arrows.

is illustrated in Figure 3.15. It shows two semantic network structures, namely the source (left) and target (right) networks. The relational structures are determined by the action cores that have attached their sets of action roles, in this case the *Flavouring* action core with its action roles *Goal* and *Spice*. In the target network, no *Goal* relation is specified, so a matching source structure is recalled from long-term memory (i.e. the howto library), which has the missing roles assigned. In this example, the source network might have originated from an instruction “season the spareribs with salt and pepper.” The projection of the inference candidates *salt.n.01* and *pepper.n.03* is indicated by the green projection corridors.

Researchers have identified three broad kinds of qualities that largely determine the expressive power of analogies. At first, *structural consistency* denotes the property of an analogy where every constituent of the source analog is in a one-to-one relationship with a constituent of the target analog. Consequently, analogies are unambiguous with respect to their structural alignment and no entity in the source can match more than one thing in the target. Second, humans tend to prefer large, deeply connected structures to just single relations. This preference is referred to as the *systematicity principle* and indicates humans’ implicit preference for structures with high information content and strong inferential power. A third principle, called *transparency* assesses the congruence of two analogs and describes how similar corresponding entities in the source and target analog are. Highly transparent matches refer to analogies with both very similar structures and entities. The instructions “fill a cup with coffee” and “fill the kettle with water” from the beginning of this chapter are an example of a high-transparency analog: The relational structures of the two instructions, determined by the action core *Filling* and its action roles, are the same

in both cases, and also the entities, the liquids water and coffee, and the containers cup and kettle, are also quite similar. High-transparency analogies are generally more natural and easier to process for humans than low-transparency analogies, in which also the relational structures as such need to be aligned. Analogies that are used in this work are mostly highly transparent, in some cases even *literally similar*, i.e. matching entities are equal and also fill the same relational slots.

3.8.3 Analogical Completion & Refinement

As mentioned above, maintaining a joint distribution as in Equations (3.8) and (3.9) is intractable for all combinations of concepts. Therefore, we intend to keep this kind of knowledge outside the probability distribution and rather maintain a large collection of example instructions that can be found web pages and have PRAC semantically annotate them using the PRAC-TELL algorithm. In order to answer questions about missing action roles, the PRAC howto library can then be queried for the most similar instance in the KB and adapt it to the current situation. The knowledge acquisition in this fashion can be regarded as an instance-based learner as the acquired knowledge is mapped against and adapted to new unseen situations. To enable fuzzy mappings from target to source structures, a similarity measure is needed which uses deep taxonomic prior knowledge in order to retrieve from the PRAC howto library the instructions that are semantically most similar to the given incomplete task description. The system in turn is queried for the semantically annotated sentence in the PRAC howto library that is most similar to the given incomplete NL instruction. The howto library can be queried much faster than computing the joint distribution and thereby enables the deep transfer of knowledge to unprecedented situations. The method thus implements an instance-based learning technique employing *reasoning by analogy* as introduced above and is thereby also in line with Minsky's notion of frames:

“When one encounters a new situation (or makes a substantial change in one’s view of the present problem) one selects from memory a structure called Frame. This is a remembered framework to be adapted to fit reality by changing details as necessary.” — (Minsky, 1974)

The approach presented in this section complements the PRAC reasoning pipeline implementing the reasoning modules ACTIONROLECOMPLETION and ACTIONCORREFINEMENT as presented in Section 3.5.1. The data model of *Frames* in the PRAC howto library allows for queries targeted at specific action cores whose action roles

must precisely match the query. In such cases, the KB must contain howtos or frames that entail the desired answer for successfully finding a completion of an instruction. However, this is not always the case although the knowledge from a very similar frame could be used as an answer to the query. As an example, consider the query for possible seasonings of a steak from above. If the KB does not contain a frame attached to the action core *Flavoring* with an action role *theme=steak.n.01*, the query in Equation (3.7) does not return a result at all. If, however, the KB contains a similar frame with action roles *theme=cutlet.n.01*, *spice={salt.n.02, pepper.n.03}*, then it would yet be desirable if the system were able to transfer this knowledge about seasonings of a cutlet to seasonings of a steak.

3.8.4 Similarity of Frames

To enable fuzzy reasoning about the frames stored in the PRAC howto library, a notion of *frame similarity* is required in order to enable a fuzzy mapping from the source to the target frames in a way that not only exactly matching frames are selected from the knowledge base, but also frames which are *similar* to the query. The semantic similarity of two concepts in a taxonomy can be characterized in terms of the relative location of the two concepts in the taxonomy. Popular measures take into account the lengths of the shortest paths between two concepts in the respective taxonomy graph. The shorter the paths connecting the two nodes in the graph are, the more similar the respective concepts are assumed to be. Among those similarity measures, the WUP similarity (Wu and Palmer, 1994), $sim_{WUP} : \mathbb{T} \times \mathbb{T} \rightarrow [0, 1]$, is the most widely used, which I will discuss in more detail in Chapter 4. It defines the semantic similarity measure on concepts in a class taxonomy as

$$sim_{WUP}(c_1, c_2) := \frac{2 \cdot depth(lcs(c_1, c_2))}{depth(c_1) + depth(c_2)},$$

where $lcs(\cdot, \cdot)$ denotes the lowest common super-concept of two concepts. We define the semantic similarity of a frame f_1 and another frame f_2 as the harmonic mean of WUP similarities of all roles in f_1 to the respective roles in f_2 , i.e.

$$sim(f_1, f_2) = \frac{|R(f_1)|}{\sum_{r \in R(f_1)} sim_{WUP}(r(f_1), r(f_2))^{-1}}, \quad (3.10)$$

where $R(f)$ is the set of all given action roles in the frame f . A similarity value of 1 denotes maximal similarity, whereas a value of 0 denotes maximal dissimilarity. For

the harmonic mean of two numbers tends to be closer to the lower of the two, all concepts of action roles of f_1 and f_2 must be reasonably similar for the two frames to expose significant similarity. Note that our frame similarity measure is not symmetric, i.e. $sim(f_1, f_2) \neq sim(f_2, f_1)$ in general, for the number of assigned roles can vary among different frames, such that $|R(f_1)| \neq |R(f_2)|$.

3.8.5 Action Role Completion

Having defined the conceptual and computational apparatus of the action role completion engine, it is straightforward to formulate queries to the PRAC howto library that infer missing roles, expand single instructions to sequences of plan steps, and adapt existing howtos to new, even unseen situations and thus transfer knowledge about manipulating specific objects to unknown objects by analogical reasoning.

Consider the task of finding a reasonable completion of an action, which is given by the instantiation of a frame, as shown in Figure 3.15. As described in Section 3.4.6, the mapping phase in the analogical reasoning process can be efficiently implemented through database queries that select the frames from the PRAC howto library that match the roles in the target frame. In the following, let f_{source} and f_{target} denote the source and target frames. Let further $R(f)$ denote the action roles assigned to the frame f and let $M(f)$ be the roles that are missing in f . $prac.howto$ denotes the collection named *howtos* in the database *prac*, which denotes the howto library in PRAC. The analogical inference for role completion can be realized by the canonical query

$$\Pi_{M(f_{source})} \left(\sigma_{\wedge_{r \in R(f_{target})} r(f_{target})=r(f_{source})} (prac.howtos) \right) \quad (3.11)$$

to the howto library, which selects all frames whose action roles correspond to the given roles of f_{target} , and does a projection of the role *Spice*, which is unspecified in the source frame. However, queries of this kind are too limited to implement knowledge transfer as they only apply to their literal action core instantiations in the library. Using the frame similarity, we can formulate a more general inference task.

Given a partially instantiated frame f_{target} , its missing roles can be completed by formulating a query for the *most similar* frame f_{source} in the howto library, and projecting the roles that are unspecified in f_{target} . We can formulate the canonical inference problem of completing action roles in f_{target} by means of analogical reasoning as

$$\Pi_{M(f_{target})} \arg \max_{f_{source} \in \text{prac.howtos}} \left(\text{sim}(f_{target}, f_{source}) \right). \quad (3.12)$$

Note that, in contrast to (3.11), this query does not select the frame from the database that literally matches the action roles of the query, but for the most similar one. Thereby, the selection of the most similar source frame f_{source} corresponds to the matching phase of the analogical reasoning, and the projection corresponds to the ‘relational pattern completion’. The fuzzy matching enables to generalize knowledge about roles of specific objects in terms of their semantic similarity as explained in the previous section and to transfer this knowledge to different types of objects as shown in Figure 3.15

In its current implementation, the inference in (3.12) always returns the most similar frame found in the database. In some cases or domains, however, when the robot is manipulating delicate or even dangerous objects, e.g. experimenting in a chemical laboratory, one might want to restrict the robot’s freedom to extrapolate its knowledge to unseen objects. To account for such scenarios, it is also possible to introduce an absolute threshold θ of minimum similarity that frames in the database are required to expose in order to satisfy the query:

$$\Pi_{M(f_{target})} \arg \max_{f_{source} \in \{f \mid f \in \text{prac.howtos}, \text{sim}(f_{target}, f) \geq \theta\}} \left(\text{sim}(f_{target}, f_{source}) \right). \quad (3.13)$$

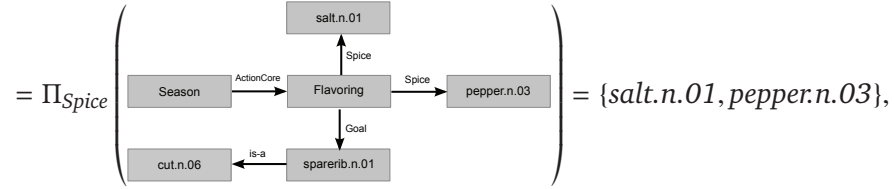
Note that, in the case $\theta = 0$, the query in (3.13) reduces to (3.12) and the case $\theta = 1$ is equivalent to the literal matching scheme in (3.11).

Example Using the query in the running example of the instruction “season the steak” can be completed using the PRAC howto library in the fashion depicted in Figure 3.15. Plugging the semantic network representing the given information into Equation (3.12) yields

$$\Pi_{Spice} \arg \max_{f_{source} \in \text{prac.howtos}} \text{sim} \left(\begin{array}{c} \text{?} \\ \uparrow \text{Spice} \\ \text{Season} \xrightarrow{\text{ActionCore}} \text{Flavouring} \\ \downarrow \text{Goal} \\ \text{cut.n.06} \xleftarrow{\text{is-a}} \text{steak.n.01} \end{array}, f_{source} \right).$$

Let us assume that there is a frame stored in the PRAC howto library representing the knowledge that spareribs can be seasoned with salt and pepper, which is most similar

to the target frame. This is a plausible assumption since the two concepts *sparerib.n.01* and *steak.n.01* is $sim_{WUP}(sparerib.n.01, steak.n.01) \approx 0.82$. Consequently, solving for the source frame and projecting the *Spice* action role missing in the target frame yields



i.e. the action role completion returns the concepts *salt.n.01* and *pepper.n.03* from the PRAC dictionary as possible seasonings for a steak.

3.8.6 Plan Expansion

Very often single steps in a recipe cross-reference other complete instruction sheets. Examples of such references can be found in abundance, for instance, in the example of making an Italian dinner illustrated in Figure 3.14. The instruction to “set the table” refers to a different howto which details the respective action. Likewise, the directive to “serve some wine” also can be refined by looking up a howto for serving juice and adjusting the respective parameters accordingly. In PRAC, it is assumed that such instructions can be replaced by the sequence of steps of the respective referred howto to obtain a more detailed description of what actions to perform. One can thus think of this kind of instruction completion as ‘expanding’ a directive to its associated plan steps. The ability to proficiently handle such references is a crucial skill in resolving and detailing the vagueness in NL instruction interpretation.

The reasoning pipeline in PRAC has been designed to account for such plan expansions directly by the functional signature of every reasoning module, which must map one single inference node in the inference tree to a sequence of inference nodes, which can contain refinements of action specifications.

In terms of the analogical reasoning framework it is straightforward to query the PRAC howto library for an instruction representing a high-level activity, such as “make a pizza” and access the sequence of action steps by querying for the *steps* attribute of a corresponding howto in the database,

$$\Pi_{f_{source}.steps} \left(\sigma \wedge_{r \in R(f_{target})} r(f_{target}) = r(f_{source}) (prac.howtos) \right).$$

$R(f)$ again denotes the set of roles specified in the frame f . As already described in Section 3.4.6, every howto in the PRAC howto library has an attribute *steps*, which holds a sequence of frames determining the single finer-grained actions to be conducted in order to achieve the higher-level goal represented by f_{source} . It is important to note that, during plan expansion, all action roles r in the returned frame f_{source} must literally match the roles in the target frame f_{target} , i.e. $r(f_{source}) = r(f_{target})$, $\forall r \in R(f_{target})$. The rationale behind this is that during action refinement of a specific task, a user would not want to give a robot the freedom to replace a high-level task by a different one, even if the source is a high-transparency match. For example, if a recipe contains a command to “bring water to a boil,” it is unacceptable to let the robot autonomously replace water by milk. In a chemical laboratory setting, if a robotic assistant is instructed to fetch a certain substance, it would be considered a flaw to bring a different substance instead. Therefore, I argue that it is reasonable to require the source to literally match the preconditions imposed by the target.

However, it could be rational to assume that plans written for certain objects can be *transferred* to similar entities. For example, an instruction sheet for heating up milk in a microwave oven can also be applied to boil water if the antecedent conditions from the original instruction are propagated through the plan. Likewise, a plan for fetching a flask holding a substance can also be adapted to work with different substances. This kind of adjusting parameters of a howto to work in new, previously unseen scenarios is a kind of analogical reasoning and key in human cognition and skill transfer.

3.8.7 Plan Adaptation

During plan expansion, all roles of the query and answer frames must coincide. This is a reasonable restriction, since the action sequence *steps* of a howto is only applicable if all objects occurring in the high-level action frame also are referred to by the same symbols in the action sequence. If one wants to relax this restriction, one additional step needs to be appended, which replaces objects referred to in the source frame by the objects in the target frame. This substitution must be applied to every frame in the sequence of steps. More formally, one can query the PRAC howto library for the most similar howto that matches the target,

$$\Pi_{f_{source}.steps} \left(\arg \max_{f_{source} \in \text{prac.howtos}} \left(\text{sim}(f_{target}, f_{source}) \right) \right)_{[r(f_{source})/r(f_{target})]_{r \in R(f_{target})}}, \quad (3.14)$$

project the single action steps, and apply for each action role specified in f_{target} , $r \in R(f_{target})$, in every action step a substitution of the object satisfying r in f_{source} by the respective object satisfying r in f_{target} . The substitution operator $(\cdot)_{[x/y]}$ replaces all occurrences of x in its argument term by y . Consequently, the substitution $[r(f_{source})/r(f_{target})]_{r \in R(f_{target})}$ denotes the application of the substitution to every role $r \in R(f_{target})$. The substitution of objects in the plan steps thus adapts a howto to new objects and implements the transfer of knowledge across concepts. It does so by establishing analogies from the current target situations at hand to the source situations that are recalled from memory.

Example Let a simple example illustrate the process of recalling a complete howto from the PRAC howto library and adapting it to the requirements of the current situation by analogical reasoning. Consider the following 3-step instruction sheet for serving fruit juice:

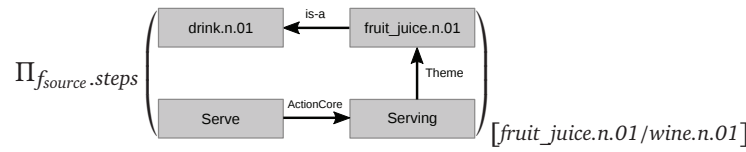
How to Serve Juice

1. Take a glass from the cupboard.
2. Fill it with juice.
3. Bring the glass to the table.

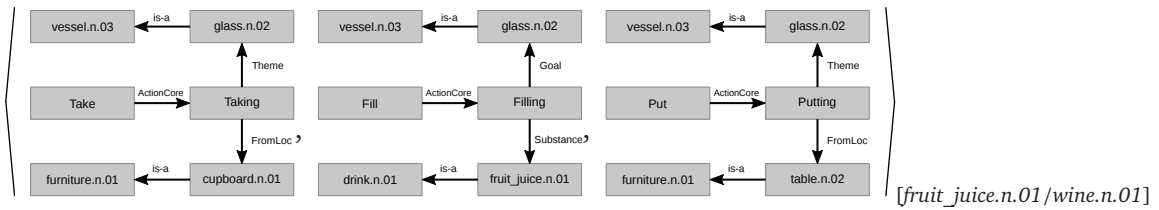
A robotic waiter is now instructed to serve a glass of wine. As the original plan expansion cannot be applied since the howto is not a literal match to the query, the robot is supposed to transfer the knowledge about serving juice to serving wine. Using the plan adaptation scheme in Equation (3.14), the PRAC howto library can be used to adjust the parameters of the instruction sheet as follows. First, the query in (3.14) is instantiated by plugging in the semantic network representation of the target frame, i.e.

$$\Pi_{f_{source-steps}} \left(\arg \max_{f_{source} \in \text{prac.howtos}} \text{sim} \left(\begin{array}{c} \begin{array}{cc} \text{drink.n.01} & \xleftarrow{\text{is-a}} & \text{wine.n.01} \\ \text{Serve} & \xrightarrow{\text{ActionCore}} & \text{Serving} \end{array} \\ \uparrow \text{Theme}, f_{source} \end{array} \right) \right)_{[\text{Theme}(f_{source})/\text{wine.n.01}]} .$$

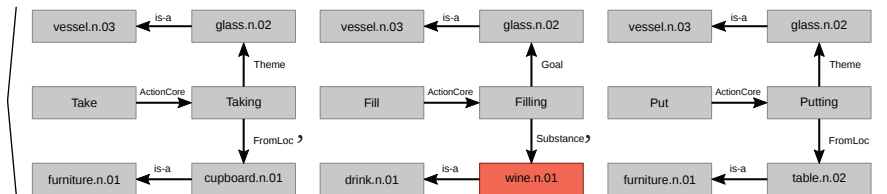
Assuming that the howto for serving juice is the most similar source candidate in the library, we obtain



having resolved the $Theme(f_{source})$ to the *fruit_juice.n.01* concept. Subsequently, the projection of the sequence of single action steps is applied, which returns the vector



Finally, in the last step, the substitution operator is applied to all occurrences of *fruit_juice.n.01* in the source frame, which are replaced by the respective concept *wine.n.01*.



The presented approach to instruction completion is fully integrated in the PRAC natural-language interpreter. The PRAC howto library has been populated with more than 103,000 single instruction steps extracted from the ~8,400 NL recipes from Wikihow.

As a test case, we consider the instruction “flip the pancake,” which is lacking information about which utensil to be used for the flipping action. The task is to infer the most appropriate utensil by establishing analogies to similar situations from the large body of instructions. Out of more than 7100 flipping instructions, ~210 have complete role assignments and thus are potential candidates for the role completion. Table 3.8 shows the top 5 results of the frames most similar to the instruction, their similarity score, the NL instruction they originated from, as well as their role assignments. It can be seen that the examples are indeed semantically close with respect to the similarity of their role assignments.

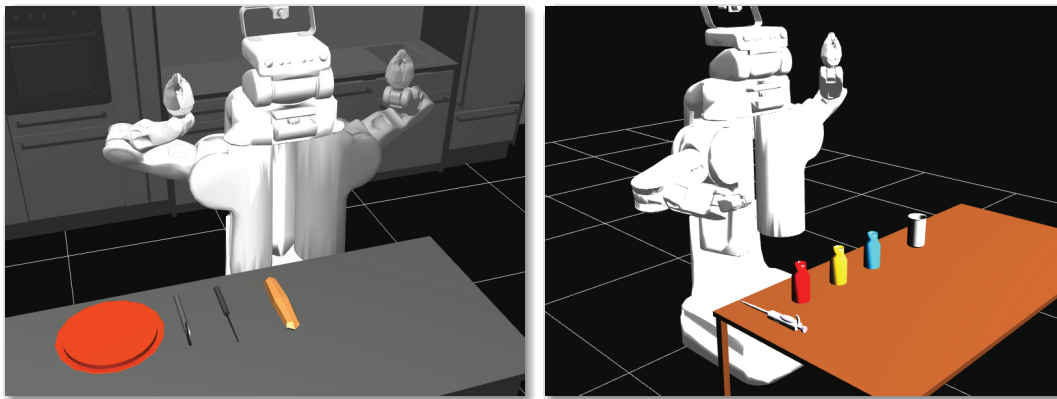
Score	Sentence	Roles	Senses
97%	"If you're skillful, you can flip the crepe with a quick action of the wrist and no spatula."	action verb: object: utensil:	<i>flip.v.08</i> <i>crepe.n.01</i> <i>wrist.n.01</i>
93%	"To flip a crumpet, remove the ring using tongs, and then flip the crumpet with a spatula ."	action verb: object: utensil:	<i>flip.v.08</i> <i>crumpet.n.01</i> <i>spatula.n.01</i>
92%	" Turn the pancake with the help of turner and let it cook on the other side till get brown spots."	action verb: object: utensil:	<i>turn.v.10</i> <i>pancake.n.01</i> <i>turner.n.08</i>
87%	"Do not try to flip food with a skillet that is too heavy for you to easily control."	action verb: object: utensil:	<i>flip.v.08</i> <i>food.n.02</i> <i>frying_pan.n.01</i>
77%	" Flip the steak over and repeat steps 1-3 with the other side ."	action verb: object: utensil:	<i>flip.v.08</i> <i>steak.n.01</i> <i>side.n.05</i>

Table 3.8: Exemplary queries for semantically most similar frames in the PRAC instruction sheet database: The top 5 scored frames for the query "flip a pancake."

3.9 Execution in Simulation

Researchers in the fields of neuroscience and cognition have found evidence indicating that the capability to *understand* a directive in NL is tightly coupled to the ability of actually *performing* it (Feldman and Narayanan, 2004). They view simulation as a mental process for language understanding (Kaup et al., 2010) and physics reasoning (Markman et al., 2008), and experiments have also shown that similar regions in the human brain are active during actual task performance in the real world and during just imagining the task being performed (Decety and Michel, 1989). The ultimate goal of the PRAC system is therefore to be able actually perform the analyzed instructions in simulation. To this end, PRAC is connected to the CRAM plan executive (Mösenlechner, 2016) that can run the plans generated by PRAC in a robot simulator. CRAM is connected via a bridge that has been developed in collaboration with Pomarlan et al. (2017).

Figure 3.17 shows an architectural overview of the connected systems. Both PRAC and CRAM are running on a web server machine and communicate over a remote procedure call (RPC) interface. The user interface is PRACWEB, a web-based client to the PRAC system running in a web browser, which I will present in Section 6.1. The disambiguated, completed and refined semantic representations of NL instructions computed by PRAC are sent to the CRAM server in the form of so-called *action designators*, i.e. symbolic descriptions of actions consisting of the respective action name and role-value pairs for the identified action parameters.



(a) The simulated kitchen environment.

(b) The simulated chemical laboratory.

Figure 3.16: Two Worlds in the Gazebo-based Robot Simulator.

On the CRAM side, a component called the *simulation manager* accepts a list of action designators. As it takes several seconds to start a Gazebo simulator (Koenig and Howard, 2004), the simulation manager maintains a pool of initialized simulated worlds that have been prespecified to keep the system more responsive. The simulation manager selects a world appropriate for the command. At the time this thesis is being written, two different static worlds have been implemented, which corresponds to the two application scenarios considered in this thesis, i.e. a kitchen and a chemical laboratory. The two worlds are shown in Figure 3.16.

The received action designators are composed into a coherent plan and interpreted and executed by CRAM. The components of the simulated robot, its sensors, controllers, and the cognitive component, are all implemented as ROS nodes (Quigley et al., 2009). A live visualization of the state of the simulated world is streamed back to the PRACWEB client using the Robot Web Tools (Alexander et al., 2012), a toolkit for remotely connecting robot middlewares with web technologies. The visualization allows the users to see the simulated robot and the objects in the environment, as well as additional visualization markers created by the robot plan. Using the mouse, users can also change the position and orientation of the camera or zoom in and out.

The CRAM system is connected to the OPENEASE cloud robotics platform (Beetz et al., 2015c), which records and stores information collected by the robot during runtime. Such information includes the intentions of the robot while acting, why it manipulated which object in which way and why, and what kinds of errors happened during task performance. The logged information can be used by the robot to learn and improve its future behavior.

In this section, I have presented a system integrating simulation of robot plans with probabilistic reasoning about natural-language instructions in PRAC. PRAC in

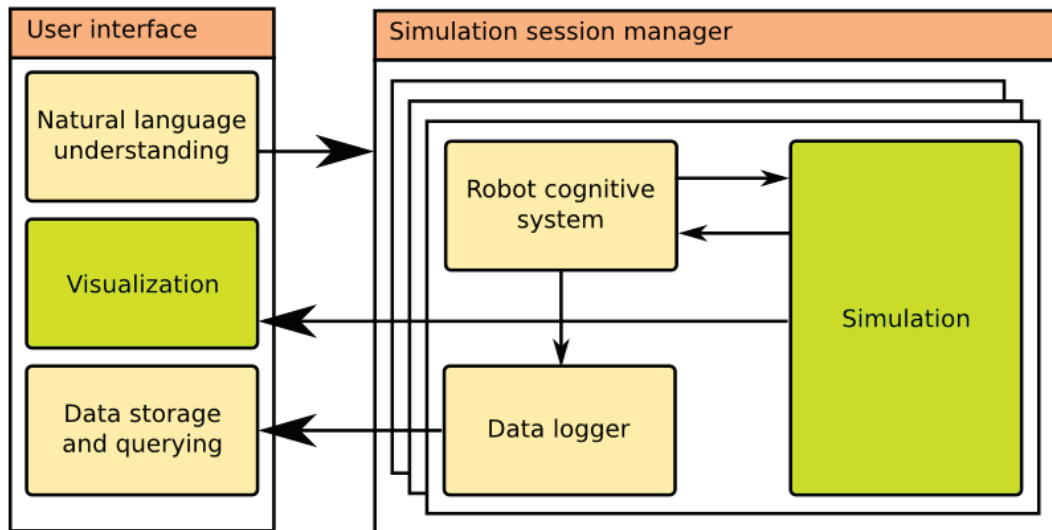


Figure 3.17: Architecture overview of the connection between PRAC and CRAM. For reasons of responsiveness, several simulated worlds are running concurrently, waiting for a command from the natural-language understanding component. The simulation produces logs of data about the robot’s processes, and a live visualization for the user. (By courtesy of Mihai Pomarlan)

turn implements a complete pipeline from underspecified, vague NL instructions to execution. This integration allows the robotic system to efficiently infer knowledge about the physical world that would be tedious to specify by hand in a collection of logical statements. It promises great potential for simulation as a learning, reasoning and validation method for robots.

3.10 Related Work

In recent years, much work has been done in order to make knowledge sources available to robots, which are originally indented for human use (Tenorth et al., 2010; Ryu et al., 2010), and to generate robot plans out of natural-language instructions (Matuszek et al., 2010; Tellex et al., 2011; Dzifcak et al., 2009; Neo et al., 2008; Tenorth et al., 2010). Dzifcak et al. (2009) use combinatorial categorical grammars for deriving a goal formulation in temporal logic in order to find an action sequence that achieves this goal. Matuszek et al. (2010) use statistical machine translation techniques to match natural-language navigation directives against a formal path description language. Tellex et al. (2011); Ryu et al. (2010) use probabilistic models to derive plans to be executed by a robot. Misra et al. (2014b) take into account

the context of the environment for grounding objects in an instruction to objects in the environment. They solve the ambiguity in instructions using an energy function corresponding to a conditional random field.

What these approaches have in common is that they do not take into account that natural-language instructions typically are severely underspecified, ambiguous and often not directly executable. They make what is commonly referred to as the *closed-world assumption* postulating that all knowledge about the world is given and complete. Additionally, most approaches to teach robots by means of natural language are designed to capture and execute what is specified by an instruction using shallow mappings to robot control, but they are not intended to accumulate semantic action knowledge that can be recalled in and adapted to different situations. Artzi and Zettlemoyer (2013) and Kim and Mooney (2012) learn probabilistic context-free grammars for robot navigation tasks. Their approach is inspired from a more linguistic point of view, where such grammars are typically induced from large corpora of text consisting of sequences of navigational directives. The problems inferring completions and executable refinements of instructions have received surprisingly little attention so far.

PRAC pursues the goal of accounting for the variational complexity and richness of human-scale manipulation tasks with everyday objects. In PRAC, the result of a linguistic analysis of an instruction is taken only as evidence in a probabilistic first-order knowledge base which allows us on the one hand to include any syntactic characteristics of a sentence as evidence in a query, and on the other hand it enables tight integration with the robot's belief state, high-level knowledge base, executive and perception system, which can provide comprehensive context information, such as the objects perceived in a scene, for instance. In addition, PRAC makes use of a rich taxonomy of concepts, the PRAC dictionary, which allows to transfer the learned knowledge to new, unseen concepts. In PRAC the goal is not to find action sequences given a particular goal, but to infer *how* to perform complex everyday activities in presence of partial and incomplete information. It is inspired by and closely related to frame semantics (Minsky, 1974) and is partially adapted from the FrameNet initiative Baker et al. (1998) of action verbs but extended and adapted for including knowledge necessary for robot action execution. ROBOFRAMENET (Thomas and Jenkins, 2012) is a system that uses natural language to command robot action through the intermediary of semantic frames. It does not account for uncertainty, ambiguity or vagueness. Rather static mappings on a word level are used to connect NL with robot control routines and no deeper semantic representation is used.

A number of previous works deal with inferring executable plans using probabilistic models (Branavan et al., 2009; Matuszek et al., 2013; Misra et al.; Tellex et al., 2011).

However, to the best of our knowledge, only a few address the problem of inferring missing information pieces and under-specificity. Similarly to PRAC, some of them are able to infer reasonable plans even on unseen objects using taxonomic knowledge. *Tell Me Dave* (Misra et al.) is a similar system, which is provided with training sentences such as “throw the drinks in the trash bag.” Given a new instruction the system is able to infer missing roles and finds that the “trash bag” is suitable as well to “throw away the chips”. Sung et al. (2014) learn Markov random fields and iteratively infer the most probable next action step to conduct and its parameters.

An example for a system that extracts knowledge from NL documents on the Web is *Text Runner* (Etzioni et al., 2008) which is a self-supervised learning system generating its own training data using a dependency parser to parse a corpus of documents. The extracted knowledge is encoded by pairs representing relationships between two respective entities. During the information extraction process, TextRunner applies POS tagging on each sentence in the corpus the output of which is given as evidence to the classifier which evaluates every sentence if it contains a relation considered trustworthy, which will then be stored in a KB.

PRISMATIC (Fan et al., 2012) is another approach to extract knowledge from NL documents. It was developed by IBM for creating knowledge bases for the Watson system which became popular for winning the Jeopardy! competition (Ferrucci et al., 2010b). *PRISMATIC* extracts shallow knowledge from the corpora based on which it performs semantic inference using aggregate statistics. To process the corpora, the system uses dependency parsing to extract the grammatical relations of the words in the sentences and represent them as frames, which are attribute-value pairs incorporating a predicate and its immediate participants. With a sufficiently large data set, it is possible to infer relations that are not explicitly mentioned but implicitly encoded in the corpus. However, using statistical aggregation to determine the results comes with the side effect that the outcome may be biased if the information in the corpus is unbalanced.

Never-ending Language Learner (NELL) (Carlson et al., 2010) is an information extraction system. The system iteratively updates its knowledge base with new corpora it is provided with. New knowledge is acquired using a number of subsystem components serving different purposes. One component extracts knowledge of NL documents based on the categories and relations in the knowledge base while another component extracts new entities and relations of semi-structured documents like HTML and XML. A third component is a uses sets of logistic regression models determining which category noun phrases belong to. NELL learns new relations based on relations already contained in the knowledge base. The components assign scores to their extracted knowledge which is then used to decide which new facts will

be added to the KB. Like PRAC, NELL uses an initial ontology to extract knowledge for the information extraction process.

The *Semantic Similarity Engine* (Boteanu and Chernova, 2015) learns analogies of the form “*A* is to *B* as *C* is to *D*, like they are commonly used in standardized tests for evaluating humans’ abilities to grasp and complete the relational structures of terms.


Chapter **four**

Reasoning in Large-scale Taxonomies

A key problem in the application of first-order probabilistic methods is the enormous size of the graphical models they imply. The size results from the number of possible worlds that have to be generated from a domain of objects and relations. One of the reasons for this explosion is that so far the approaches do not sufficiently exploit the structure and similarity of possible worlds in order to encode the models more compactly, as shown in Chapter 2. In this chapter, I introduce fuzzy Markov logic networks (FUZZY-MLNs), an inference mechanism in Markov logic networks, which enables the exploitation of taxonomic knowledge as a source of imposing structure onto possible worlds. I show that by exploiting this taxonomic structure, probability distributions can be represented more compactly and that the reasoning systems become capable of reasoning about concepts not contained in the probabilistic knowledge base. More specifically, I first present an approach for reasoning about unknown concepts by exploiting semantic similarity to known concepts in Markov logic, which typically impedes a compact representation of classes that are hierarchically organized in a taxonomy. Second, I present a reasoning framework for Markov logic networks (MLNs) that enables inference in presence of vague evidence, which allows a very compact representation of knowledge in MLNs and learning from sparse data. And third, I demonstrate the strengths of the FUZZY-MLN approach by the example of a word-sense disambiguation task and showcase its strong generalization abilities.


4.1 Motivation

Many real-world reasoning problems require the combination of relational representations with inference mechanisms that can solve the problems by reasoning from incomplete, ambiguous, inaccurate and even contradictory information. Examples of such reasoning tasks are the interpretation of natural language (Beltagy and Mooney, 2014), robot perception (Nyga et al., 2014) or intent recognition in human-robot interaction (Sukthakar et al., 2014). Probabilistic relational models (PRMs) (Getoor, 2007b) have great potential to serve as powerful problem-solving tools for such application domains: joint probability distributions over the instantiated relations that describe the possible worlds in the respective domain can be queried for any aspect Q contained in the model given any evidence E , $P(Q | E)$. This kind of reasoning is often called *shallow transfer*, which transfers the learnt knowledge to different variations of the same domain, for example considering different numbers of objects of the same kinds (Davis and Domingos, 2009).


*Shallow
Transfer*

These powerful reasoning capabilities, however, come at the cost of computational complexity in learning and reasoning as the size of the domain under consideration grows. As a consequence, practical applications are mostly bound to small application domains with limited complexity. Many knowledge systems, however, have to work in open worlds: they are equipped with knowledge bases (KBs) that have to answer queries about unseen situations not accounted in their design like the examples mentioned above. Hence, the application of expressive probabilistic representation methods requires the inference mechanisms to support *off-domain reasoning* – reasoning about concepts that are not explicitly represented in the KB. Most of the symbolic probabilistic KBs, however, do not support off-domain reasoning: They require every symbol subject to reasoning to be explicitly represented in the model. On the other hand, learning a probability distribution with all possible concepts is hopelessly infeasible.

We therefore aim at developing reasoning mechanisms that are able to rapidly yet flexibly generalize and learn from very few examples, which has also been identified as key features in human cognition (Tenenbaum et al., 2011; Bailey, 1997). This form of knowledge transfer is often called *deep transfer* (Davis and Domingos, 2009). One possible idea to tackle this is to take into account knowledge about the taxonomic structure of the reasoning domain, which is captured by ontological knowledge representations such as description logic (DL). To this end, the correlation between the semantic similarity of concepts, and the similarity of their relational structure can be exploited for reasoning in probabilistic relational models to transfer the learned


*Deep
Transfer*

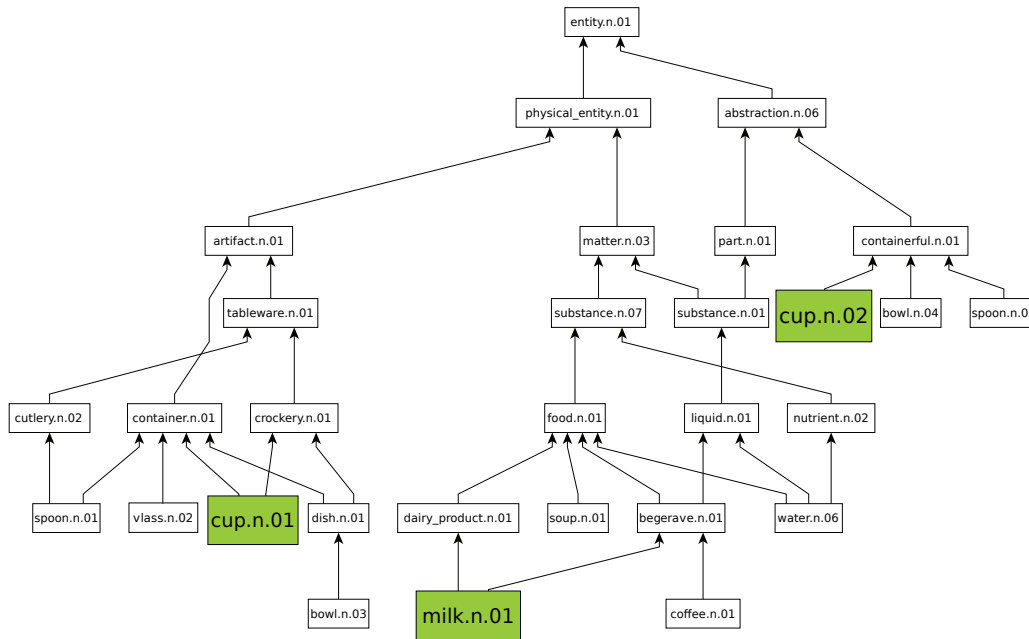


Figure 4.1: Excerpt of the WordNet taxonomy of concepts for the ‘containers-&-liquids’ example.

knowledge to classes unseen in the training data.

4.1.1 Running Example

Let word-sense disambiguation (WSD) and semantic role labeling (SRL), which are challenging and widely studied problems in natural-language processing, be our running examples. Solving these problems enables software systems to interpret incomplete and ambiguous instructions and transform them into well-defined action specifications. More specifically, take the terms ‘cup’ and ‘milk’ and their usage in the two instructions “fill a cup with milk” and “add a cup of milk.” In the former case, ‘cup’ refers to a drinking mug, a physical object that can hold milk. In the latter case, it refers to a measurement unit specifying the amount of milk to be added to something not further specified.

Figure 4.1 shows a small excerpt of the WordNet taxonomy of possible word senses covering this example. Using the taxonomy we can represent the two instructions using the logical assertions shown in Table 4.1. The assertions assign a word sense (*has_sense*) to each word. The word sense is linked to the taxonomy using the *is_a* predicate. In addition, the predicate *sem_role* states the semantic role that the word

“Fill a cup with milk”	“Add a cup of milk.”
<code>has_sense(Fill, fill-sense)</code>	<code>has_sense(Add, add-sense)</code>
<code>is_a(fill-sense, fill.v.01)</code>	<code>is_a(add-sense, add.v.01)</code>
<code>has_sense(cup, cup-sense₁)</code>	<code>has_sense(cup, cup-sense₂)</code>
<code>is_a(cup-sense₁, cup.n.01)</code>	<code>is_a(cup-sense₂, cup.n.02)</code>
<code>has_sense(milk, milk-sense)</code>	<code>has_sense(milk, milk-sense)</code>
<code>is_a(milk-sense, milk.n.01)</code>	<code>is_a(milk-sense, milk.n.01)</code>
<code>sem_role(cup, goal)</code>	<code>sem_role(cup, amount)</code>
<code>sem_role(milk, theme)</code>	<code>sem_role(milk, theme),</code>

Table 4.1: Semantic representation of the natural-language instructions “fill a cup with milk” and “add a cup of milk.”

takes in the instruction, whether it is the object acted on, the source of the stuff to be transferred, the destination, the action verb, and so on.

Now suppose we have a taxonomy and two exemplary instructions to learn from: “fill a cup with milk” and “add a cup of milk.” For the sentence “fill water into the pot” a probabilistic reasoner should infer that water is the stuff to be added and the pot the destination, even when ‘water’ and ‘pot’ are not contained in the probabilistic knowledge base. The reason is that ‘water’ is a liquid like ‘milk’ and therefore semantically similar and that a ‘pot’ is a container and therefore similar to a cup. Current first-order probabilistic reasoning frameworks cannot perform this pattern of reasoning as they are restricted to concepts contained in their probabilistic knowledge base.

In the following sections I will explain how reasoning in MLNs can be extended to perform such reasoning tasks. Note that the reasoning tasks addressed here are not whether or not two concepts are similar. This is already asserted in the taxonomy and assumed factual knowledge. We rather want to infer the concepts that entities belong to and the role they take in actions.

4.1.2 Semantic Similarity

In order to implement the principle of deep knowledge transfer in terms of the proximity of symbols, a notion of similarity or distance in the symbol space is needed. However, as there is no natural intrinsic total order on symbols in general, it is nontrivial to define such a proximity measure for every symbol space. The key

idea of FUZZY-MLNs is to learn joint probability distributions conditioned on large taxonomic knowledge bases that are assumed to be given as factual knowledge. Indeed, a number of comprehensive high-quality taxonomies exist that have been carefully designed to reflect the relational structure of concepts. Examples are the WordNet dictionary (Fellbaum, 1998) and the Cyc ontology (Lenat, 1995). In this work, I use the WordNet taxonomy as an implementational basis. Using WordNet is particularly appealing since it has been carefully designed by linguists since the early 1990's and is perhaps the world's most comprehensive and mature ontology of terms of the English language, comprising more than 117,000 concepts.

The semantic similarity of two concepts in DL-based representations can be characterized in terms of the relative location of the two concepts in the taxonomy. Popular measures take into account the lengths of the shortest paths between two concepts in the respective taxonomy graph. The shorter the paths connecting the two nodes in the graph are, the more similar the respective concepts are assumed to be. Among those similarity measures, the WUP similarity (Wu and Palmer, 1994) is the most widely used. Given a subsumption relation \sqsubseteq , it defines the semantic similarity on concepts in a class taxonomy as a function \sim_{\sqsubseteq} mapping from the Cartesian product of two concepts $c_1, c_2 \in \mathcal{T}$ to a real-valued measure of the similarity of c_1 and c_2 ,

$$\sim_{\sqsubseteq}: \mathcal{T} \times \mathcal{T} \mapsto [0, 1], \quad c_1 \sim_{\sqsubseteq} c_2 := \frac{\text{depth}(\text{lcs}(c_1, c_2))}{1/2 (\text{depth}(c_1) + \text{depth}(c_2))}. \quad (4.1)$$

$\text{lcs}(\cdot, \cdot)$ denotes the lowest common super-concept of two concepts in \sqsubseteq . A value of 1 indicates maximal similarity, whereas a value of 0 indicates maximal dissimilarity. There are two basic assumptions that lead to Equation (4.1). The first assumption is that the depth on which a concept resides in the taxonomy graph reflects its abstractness or concreteness, respectively. Consequently, the leaf concepts are assumed to be the most specific, and the root concept \mathcal{T} is assumed to be the most abstract concept. The second assumption is that the abstractness of the most specific concept that is both a superconcept of c_1 and c_2 (i.e. the lowest common superconcept (LCS)) is an indicator for their similarity. This means, the more abstract the LCS of c_1 and c_2 is, the more dissimilar they are assumed to be.

An example of the WUP similarity applied to a small excerpt of the WordNet taxonomy is shown in Figure 4.2. The figure visualizes the similarity of the concept *coffee.n.01* to the two concepts *milk.n.01* and *cup.n.01*. It is most intuitive to think of the LCS in terms of properties that two concepts have in common. If the LCS is very concrete, the concepts have many properties in common as properties in DL are being inherited along the subsumption relation \sqsubseteq to more specific concepts (cf. Figure 4.2a).

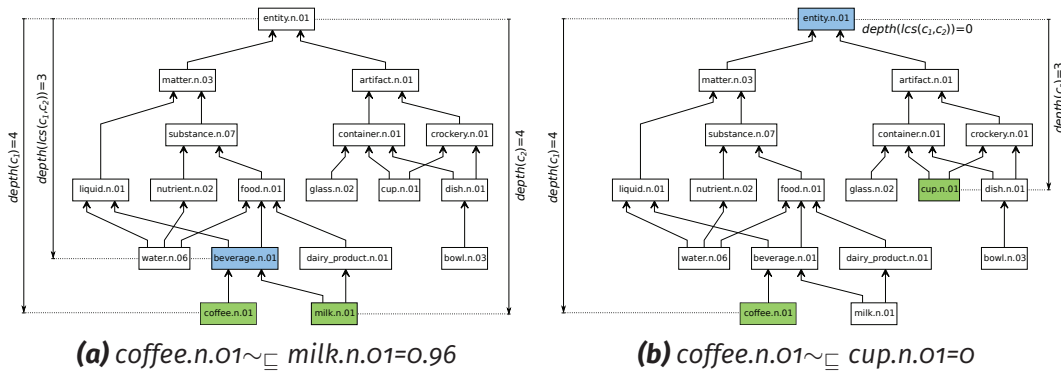


Figure 4.2: Visualization of the WUP similarity of concepts (a) *coffee.n.01* and *milk.n.01*; (b) *coffee.n.01* and *cup.n.01*: the two concepts c_1 and c_2 are highlighted in green, their lowest common superconcept $lcs(c_1, c_2)$ is highlighted in blue.

Conversely, if the most specific common property of two objects is to be instances of *something* (i.e. an *entity.n.01*), it is reasonable to assume that those concepts are fundamentally different in nature (cf. Figure 4.2b).

Following these considerations, the WUP similarity implements a reasonable distance metric on the space of concepts in an ontology. There are, however, subtleties with similarity measures that only take into account the path lengths connecting two concepts, in particular in connection with properties that cannot be reflected by the graph structure of the taxonomy. I will address these issues in greater detail in Chapter 5.

4.1.3 Fuzzy Logic

Semantic similarity measures like the WUP similarity offer a convenient means for referring to any concept in an upper ontology in terms of its similarity to any other concept. Assuming a symbolic KB representing knowledge about the concepts *cup.n.01*, *cup.n.02* and *milk.n.01* as indicated in Figure 4.1, it is possible using the WUP similarity \sim_{\sqsubseteq} to refer to a *bowl.n.03* as something very similar to a *cup.n.01* and very dissimilar to *milk.n.01* and *cup.n.02*, and to refer to *soup.n.01* as something similar to *milk.n.01* but dissimilar to *cup.n.01* and *cup.n.02*, and thus to reason about unknown concepts without explicitly introducing symbols for them.

As the goal is to perform reasoning about concepts that are not contained in a probabilistic relational model (PRM), a representational means is needed, in order to express the expectations about the properties of an unknown concept, which we are uncertain about. To this end, the Boolean truth values in MLNs can be replaced by

 see Appendix B for Reference

degrees of truths about whether or not relations hold for a concept not contained in the probabilistic model. Fuzzy logic (FL) provides an elegant calculus for this purpose, a multi-valued extension of propositional logic (PL). As introduced in Chapter 2, fuzzy logic (FL) has its foundations in the theory of fuzzy sets, in which elements belong to a set only to a certain degree.

Formally, a fuzzy subset x of a set X is a pair $\langle X, \pi_x \rangle$, where X is called the *universe* and $\pi_x : X \mapsto [0, 1]$ determines the degree to which a particular element actually belongs to x , which is called the *membership function*. In FL, the universe X is given by the set of atomic propositions and π_x is a fuzzy interpretation of X assigning every proposition in X a real-valued degree of truth. Its calculus is analogous to the calculus in PL: If A and B are propositions in FL, then the logical connectors with respect to x are defined as

$$\begin{aligned} A \wedge B &:= \min(\pi_x(A), \pi_x(B)), \\ A \vee B &:= \max(\pi_x(A), \pi_x(B)), \text{ and} \\ \neg A &:= 1 - \pi_x(A). \end{aligned} \tag{4.2}$$

Note that the multi-valued logical calculus of FL reduces to its binary counterpart of PL in the extreme cases where all propositions have Boolean truth values. By replacing the strictly Boolean calculus of first-order logic (FOL) in MLNs by a multi-valued calculus of FL, finer-grained statements about class memberships are possible by stating that an object is an instance of some concept only *to a certain degree*.

4.2 FUZZY Markov Logic Networks

In contrast to existing probabilistic methods incorporating class hierarchies, FUZZY-MLNs do not target reasoning about the taxonomic structure as such. This comes with the advantage that the concepts subject to reasoning do not need to be exhaustively modeled in the probabilistic model. This enables (1) a compact representation of knowledge, (2) a powerful generalization from sparse training data and (3) a reduced complexity of learning and inference.

4.2.1 Definition

A FUZZY-MLN F is a pair $\langle L, \sqsubseteq \rangle$, where L is an MLN and \sqsubseteq is a taxonomy of concepts, such that L represents a conditional probability distribution

$$P(\text{has_sense}(\cdot, \cdot), \dots \mid \pi_x(\text{is_a}(\cdot, \cdot)), \dots). \quad (4.3)$$

In addition, the following conditions hold:

1. an entity e in the domain of discourse D is connected to a concept c in the taxonomy \sqsubseteq always by a proposition

$$\text{has_sense}(e, s) \wedge \text{is_a}(s, c), \text{ where } s, c \in \mathcal{T},$$



Semantic Similarity

2. all ground atoms of the form $\text{is_a}(s, c)$, where $s, c \in \mathcal{T}$ take real-valued degrees of truth $\in [0, 1]$, which we call *semantic similarity*. Ground atoms of all other predicates take strictly binary truth values $\in \{0, 1\}$.



Possible Worlds

3. The set \mathcal{X} of *possible worlds* represented by F is the set of all fuzzy subsets of all ground atoms X , where the membership functions for every ground atom $\text{is_a}(s, c)$ is equal across all possible worlds and is defined as the semantic similarity of s and c with respect to \sqsubseteq , i.e. for all $x \in \mathcal{X}$ and for all $s, c \in \mathcal{T}$

$$\pi_x(\text{is_a}(s, c)) = s \sim_{\sqsubseteq} c$$

holds.

4.2.2 Semantics

According to the second condition in our definition, the semantics of FUZZY-MLNs differs from the original in Equation (2.14) in two aspects: First, a possible world x is no longer a strictly Boolean vector assigning a truth value to every ground atom but also allows for real-valued degrees of truth. The ground Markov random field (ground MRF) of a FUZZY-MLN thus contains Boolean *and* numerical random variables; one real-valued variable for every ground atom of the form $\text{is_a}(\cdot, \cdot)$ and a Boolean one for every other ground atom. Second, as a consequence, the semantics of the Boolean logical features $\hat{f}_j : \mathcal{X} \mapsto \{0, 1\}$ in the ground MRF is not applicable

anymore. Therefore, the features associated to every ground formula \widehat{F}_j in the Markov random field (MRF) are defined to take the form $\widehat{f}_j : \mathcal{X} \mapsto [0, 1]$, where each feature $\widehat{f}_j(x)$ evaluates to the truth value of its ground formula \widehat{F}_j in x by applying the fuzzy logic calculus as defined in Equation (4.2), i.e. $\widehat{f}_j(x) = \pi_x(\widehat{F}_j)$. Hence the distribution of F becomes

$$P(X = x) = \frac{1}{Z} \exp \left(\sum_{j=1}^{|\mathcal{G}|} \widehat{w}_j \pi_x(\widehat{F}_j) \right), \quad (4.4)$$

where the transition from formulas F_i in FOL to ground formulas \widehat{F}_j is achieved by the ordinary grounding process in MLNs. Condition no. 3 in the definition ensures that the probability distribution in (4.4) corresponds to the conditional distribution in (4.3): since the truth value of a ground atom of the *is_a* predicate is required to be equal across all possible worlds x , the distribution $P(X = x)$ in (4.4) is effectively conditioned on every atom of the form *is_a*(\cdot, \cdot).


In FOL, the equality operator ('=') is used to express that two symbols refer to the same entity in the real world. Its semantics therefore corresponds to the identity of two entities rather than equality. As I cannot think of any situation in which it would be reasonable to discern several degrees of identity, I argue that it makes sense to leave the equality operator its original semantics, i.e. a term $y = z$ evaluates to 1 iff y is identical to z , and 0 otherwise.


Since the FOL semantics of conventional MLNs do not account for real-valued truth values, the calculus of the logical rules in the MLN, which are being transformed to clique potential functions in the ground MRF during the grounding process needs to be adapted appropriately. To do so, it is straightforward to make the step from strictly Boolean FOL to a multi-valued logical calculus that reduces to its Boolean counterpart in the extreme cases where all predicates have Boolean truth values. Consequently, we can require every feature function in the ground MRF to take the form $f_j : \mathcal{X} \rightarrow [0, 1]$ and taking the FL calculus as a basis.

4.2.3 Knowledge Representation

A FUZZY-MLN contains two dedicated predicates, *has_sense* and *is_a*, which provide means to incorporate knowledge from the class taxonomy into the probabilistic model. In short, *is_a* encodes the taxonomic knowledge and *has_sense* is used for expressing uncertainty about which concepts entities belong to. By discerning the

two predicates, we can model that one is *certain about the taxonomic structure* of the domain subject to reasoning, but possibly *uncertain about which concept an entity belongs to*.

 In contrast to conventional MLNs, FUZZY-MLNs do not require all predicates to be Boolean and they do not require all predicates to be real-valued. Variables (ground atoms) of the form $is_a(s, c)$ in the ground MRF take real-valued degrees of truth $\in [0, 1]$, which express the degree to which s is similar to c . Here, a value of 1 denotes maximal similarity, whereas 0 denotes maximal dissimilarity. This allows to represent entities that belong to concepts not contained in the probabilistic knowledge base by referring to them in terms of their similarity to known concepts. Note that in FUZZY-MLNs, the semantic similarities are not computed by the inference algorithm as in other formalisms. Rather, they are always given by the taxonomy structure and exclusively appear as evidence. This makes the representation of the conditional distribution in (4.3) very compact, since the taxonomy structure may be collapsed into single numeric values that scale the contribution of every single ground formula to the probability mass (4.4) by the similarities of its constituents to concepts that are contained in the model. This allows to generalize the learnt knowledge also to classes unseen in the learning data. In addition, realizing FUZZY-MLNs without having to equip them with the capability of reasoning about the similarity relation \sim_{\sqsubseteq} as such, enables to escape a complexity monster: without making this restriction, inference and learning would require to compute integrals over those variables, rendering computational complexity infeasible for practical applications.

 Since the distribution of a FUZZY-MLN is conditioned on the taxonomic structure of the domain, the second predicate, has_sense , is used to link any entity in the domain of discourse to a concept in \sqsubseteq . Unlike is_a , has_sense is Boolean and may be subject to inference. Propositions about class memberships of an entity e are made in the form $has_sense(e, s) \wedge is_a(s, c)$.

4.2.4 Example

Let a minimalistic example illustrate how inference about unknown concepts can be achieved in FUZZY-MLNs: Suppose we want to represent the conditional distributions that parrots can fly and that mammals can not. In Markov logic, we can establish these distributions in an MLN with, for example, the two weighted formulas

$$w_1 = \ln .9/.1 \quad \text{flies}(e) \wedge \text{has_sense}(e, s) \wedge \text{is_a}(s, \text{Parrot})$$

$$w_2 = \ln .1/.9 \quad \text{flies}(e) \wedge \text{has_sense}(e, s) \wedge \text{is_a}(s, \text{Mammal}).$$

In conventional MLNs, reasoning can only be performed about instances of either of the concepts *Parrot* and *Mammal* because for any other concept, none of the formulas are applicable. Using a FUZZY-MLN with the same model structure and an underlying taxonomy \sqsubseteq , however, reasoning tasks outside the model domain can be tackled, such as

$$\hat{p} = P(\text{flies}(F) \mid \text{has_sense}(F, \text{Turkey})).$$

In this example, there are two ground atoms of the *is_a* predicate, *is_a(Turkey, Parrot)* and *is_a(Turkey, Mammal)*, which are, for instance, assigned the truth values

$$\pi_x(\text{is_a}(\text{Turkey}, \text{Parrot})) = 0.90$$

$$\pi_x(\text{is_a}(\text{Turkey}, \text{Mammal})) = 0.01$$

in every possible world x according to a similarity \sim_{\sqsubseteq} . Consequently, the influence of the two ground formulae

$$\begin{aligned} \pi_x(\hat{F}_1) &= \text{flies}(F) \wedge \text{has_sense}(F, \text{Turkey}) \\ &\quad \wedge \text{is_a}(\text{Turkey}, \text{Parrot}) = \min(\pi_x(\text{flies}(F)), 0.9) \\ \pi_x(\hat{F}_2) &= \text{flies}(F) \wedge \text{has_sense}(F, \text{Turkey}) \\ &\quad \wedge \text{is_a}(\text{Turkey}, \text{Mammal}) = \min(\pi_x(\text{flies}(F)), 0.01) \end{aligned}$$

on the distribution in (4.4) is scaled down by the similarity of concepts resulting in a posterior distribution of $\hat{p} \approx \langle 0.876, 0.124 \rangle$. In the extreme case, where there is maximal dissimilarity of two concepts, the contribution of every ground formula vanishes resulting in a uniform distribution. This is reasonable since in this case the model cannot make any well-informed statement about entities that are maximally dissimilar to everything that is contained in the model.

Learning & Reasoning FUZZY-MLN is an extension of *reasoning* in MLNs. As the only modification is in the semantics of the features \hat{f} to allow for soft truth values, there

is no need for adaptations of the learning and reasoning methods since the original inference methods like Gibbs sampling are defined over the ground MRF of the MLN.

Running example continued Let us now continue with the running example and explain how FUZZY-MLNs solve the respective reasoning tasks. We consider again the two training databases corresponding to the instructions (1) “fill a glass with milk” and (2) “add a cup of milk.” In order to model word sense and role/sense co-occurrences, we construct a FUZZY-MLN consisting of one single weighted template formula,

$$\begin{aligned} & has_sense(w_1, s_1) \wedge is_a(s_1, +c_1) \\ & \wedge has_sense(w_2, s_2) \wedge is_a(s_2, +c_2) \\ & \wedge sem_role(w_1, +r_1) \wedge sem_role(w_2, +r_2) \wedge w_1 \neq w_2, \end{aligned}$$

which has been trained with the two databases introduced at the beginning. As described in Chapter 2, the ‘+’ operator expands the respective variable to a separate formula for each value of the variable.

In order to illustrate that the learned MLN indeed reasonably generalizes across classes, we visualize the posterior distributions over the WordNet taxonomy for two exemplary queries. Figure 4.3 shows the posteriors of two queries for the meaning of a word representing the *theme* of a ‘filling’ activity and its *goal*, respectively, i.e.

$$\begin{aligned} P \left(\begin{array}{l} has_sense(w_1, s_1), \\ has_sense(w_2, s_2) \end{array} \middle| \begin{array}{l} has_sense(w', fill.v.01), \\ sem_role(w', action_verb), \\ sem_role(w_1, theme), \\ sem_role(w_2, goal), \\ \pi_x(is_a(fill.v.01, fill.v.01)), \dots, \\ \pi_x(is_a(s_1, milk.n.01)), \dots \\ \pi_x(is_a(s_2, milk.n.01)), \dots \end{array} \right). \end{aligned} \quad (4.5)$$

The condition contains the real-valued assertions of semantic similarities of all word senses s_1 and s_2 under consideration of all concepts in the MLN (i.e. *fill.v.01*, *add.v.01*, *milk.n.01*, *cup.n.01* and *cup.n.02*), such that the *is_a* predicate’s truth values are fully specified for all of its ground atoms.

The distributions show that, conditioned on the semantic role of a word, two clearly

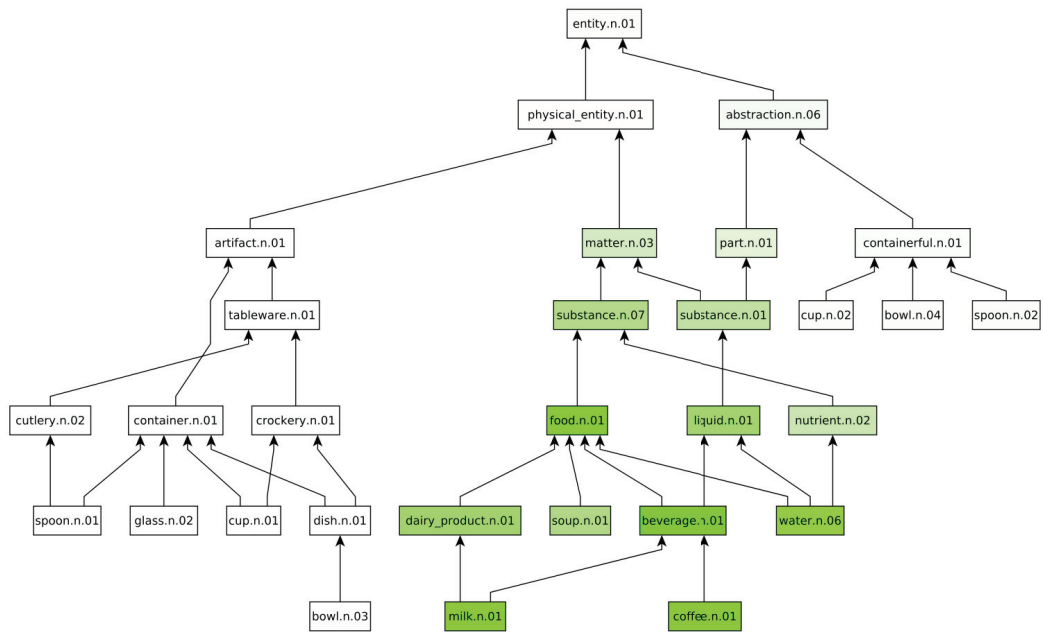
separable clusters of concepts appear in the taxonomy. For the *theme* role of a filling action, all substances/liquids gain considerably high probability (Figure 4.3a), whereas the *goals* of such an action are represented by all types of containers (Figure 4.3b). Note that also categories *not* explicitly modeled, such as *water.n.06*, *soup.n.01*, or *spoon.n.01* and *bowl.n.03* and *glass.n.02*, respectively, have been assigned significant probability masses indicating that the model indeed reasonably generalizes knowledge across unseen object categories.

A slightly different query is visualized in Figure 4.4. Here, the goal is to show how the distributions over the possible senses of the word ‘bowl’ differs in the two instructions “add a bowl of water” and “fill a bowl with water”. To this end, one can query for the distribution over all possible senses s and condition on different action verbs, for example

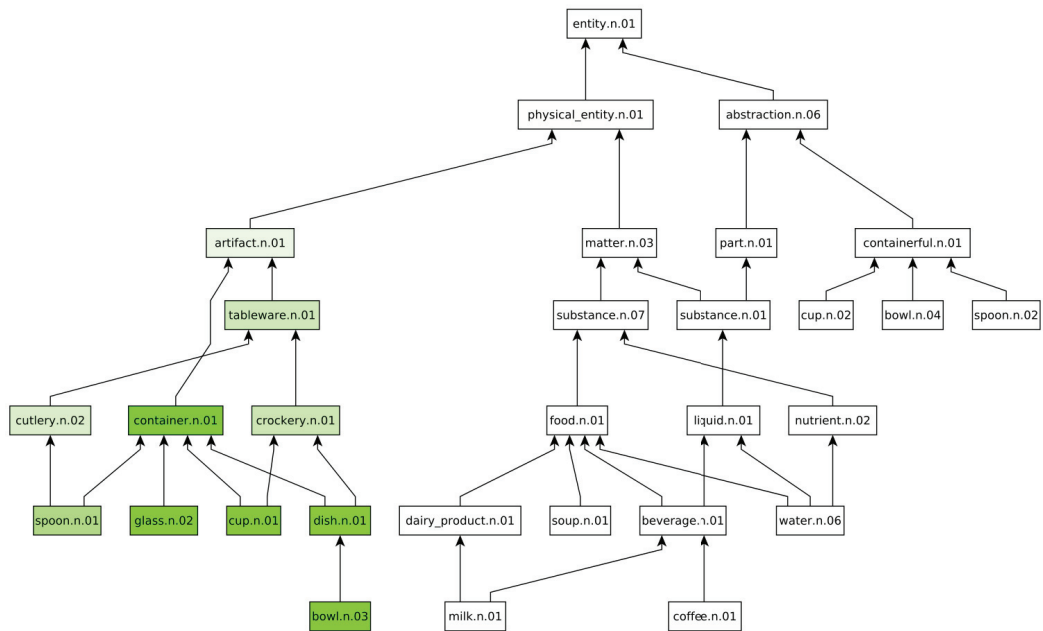
$$P \left(\begin{array}{l} \text{has_sense}(w, s) \mid \text{has_sense}(w', \text{fill.v.01}), \\ \text{sem_role}(w', \text{action_verb}), \\ \pi_x(\text{is_a}(\text{fill.v.01}, \text{fill.v.01})), \dots, \\ \pi_x(\text{is_a}(s, \text{milk.n.01})), \dots, \\ \pi_x(\text{is_a}(s, \text{cup.n.01})), \dots \end{array} \right), \quad (4.6)$$

where the senses s are from the set of actually possible senses of the word ‘bowl’. Here, a sense is considered a possible sense if the concept is returned by WordNet as a synset for the word. Therefore, all other nodes in the taxonomy graph can be assigned zero probability. Figure 4.4b shows the distribution of such a query. Obviously, the paths of hypernyms of the concept *bowl.n.03* has highest probability representing a physical container object. Figure 4.4a shows how the posterior is altered by conditioning on a different action verb, namely an ‘adding’ action (*add.v.01*). The distribution changes in favor of the *bowl.n.04* concept, which represents the abstract unit of measure.

The examples of simultaneous word-sense disambiguation and semantic role labeling show that FUZZY-MLNs are a promising concept for reasoning in PRMs and successfully transfer learnt knowledge to new, unseen concepts by exploiting the taxonomic structure of the domain.

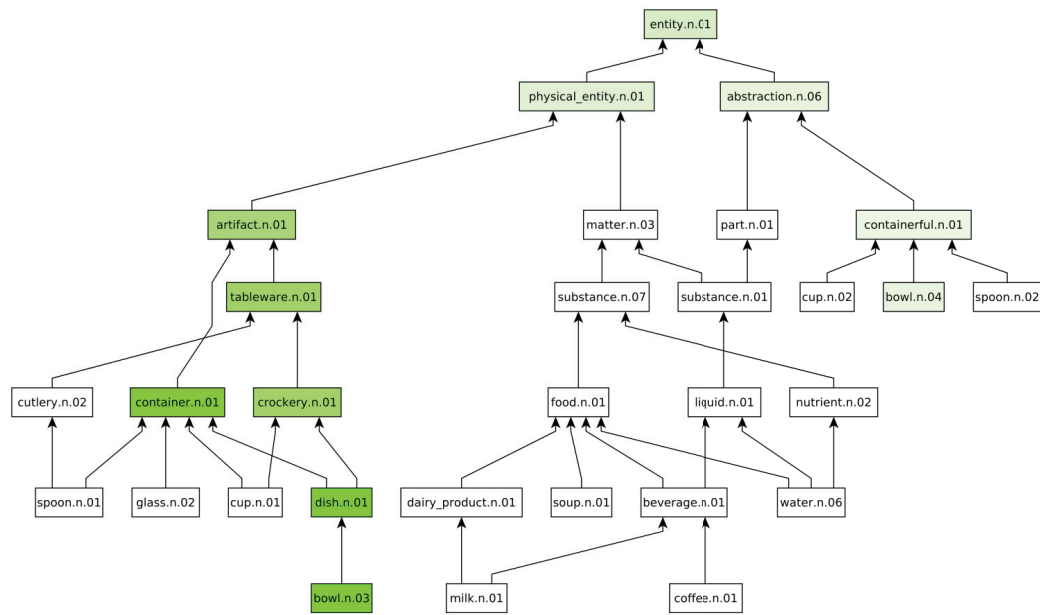


(a) Posterior of Equation (4.5) for w_1 given $\text{sem_role}(w_1, \text{theme})$

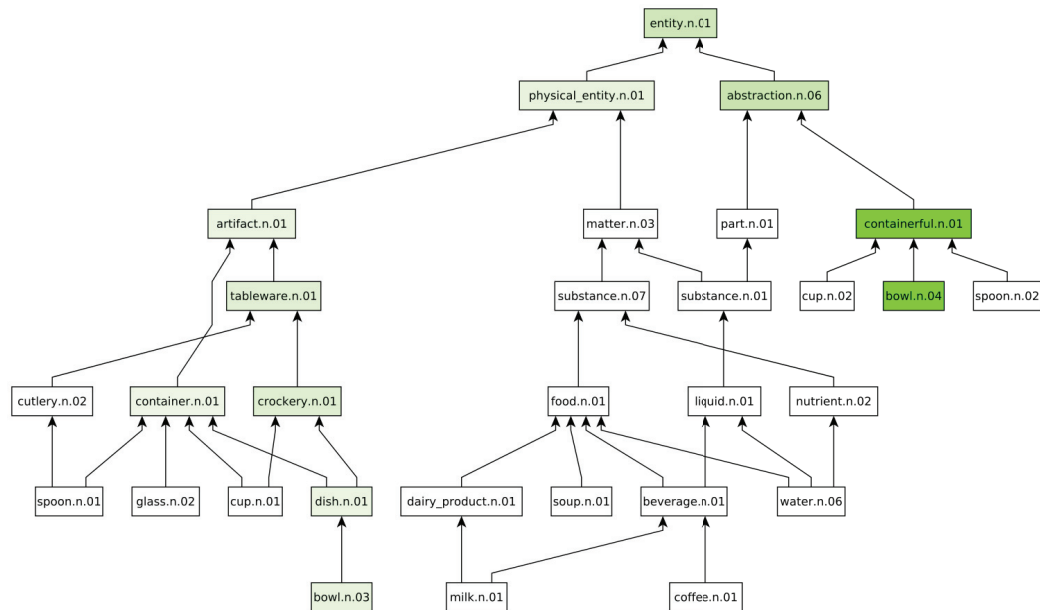


(b) Posterior of Equation (4.5) for w_2 given $\text{sem_role}(w_2, \text{goal})$

Figure 4.3: Posterior distributions over the taxonomy conditioned on semantic roles of a filling action according to Equation (4.5). More intense node colors indicate higher probability. w_1 given $\text{sem_role}(w_1, \text{theme})$ Also for word meanings that are unknown to the MLN, all posteriors represent reasonably well the appropriate word meanings.



(a) Posterior of the meaning of 'bowl' given a 'filling' action.



(b) Posterior of the meaning of 'bowl' given an 'adding' action.

Figure 4.4: Posterior distributions of word-sense disambiguation queries according to Equation (4.6) for the word 'bowl' in the sentences "add a bowl of water", and "fill a bowl with water". Also for word meanings that are unknown to the MLN, all posteriors represent reasonably well the appropriate word meanings.

4.3 Evaluation

We chose the problem of word-sense disambiguation (WSD) as a showcase for reasoning with FUZZY-MLNs since it is a difficult, widely studied and highly relevant problem in Artificial Intelligence (AI) and language understanding. We evaluate the FUZZY-MLN reasoning framework on a real-world data set of natural-language instructions that have been mined from the [wikihow.com](http://www.wikihow.com) web site and manually annotated with correct word senses. We intend to contrast its performance against ordinary MLNs with FOL calculus and – as a baseline comparison – the state-of-the-art WSD algorithm “It Makes Sense” (‘IMS’) (Zhong and Ng, 2010). IMS operates on the WordNet database of senses. However, it does not take into account the class hierarchy. Instead, it learns support vector machine classifiers on features like part-of-speech tags, surrounding words and local collocations. It has been selected as a baseline comparison as it is one of the few algorithms specifically designed for WSD, whose implementation is publicly available for training and testing.

Experimental Setup In our setup, we provide the learning and reasoning algorithms with word co-occurrences and part-of-speech tags of words. In order to emphasize the generalization capabilities of the considered methods, we chose the hardest experimental setup we can imagine:

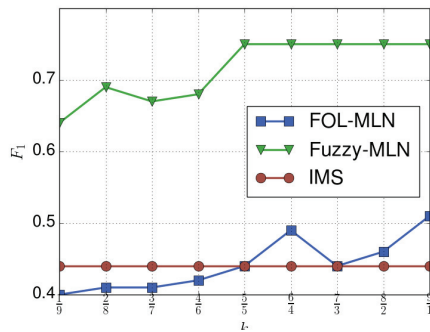
1. the datasets have been selected to exhibit maximal entropy with respect to the concepts that are contained in the examples, such that they are as dissimilar as possible. Indeed, none of the respective word senses occurs twice in the data.
2. the models were trained with only very small portions of data and tested on larger, unseen portions of data.

k -fold cross-validation is a popular and well-acknowledged method for classifier evaluation. In classical k -fold cross-validation, all data available are randomly partitioned into k subsets of equal size. Subsequently, the classifier to be evaluated is trained using $k - 1$ of the subsets and tested on the remaining k -th set. This procedure is repeated k times with every subset serving as the test set once.

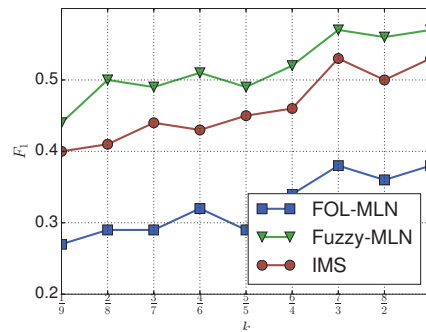


*Inverse
Cross-validation*

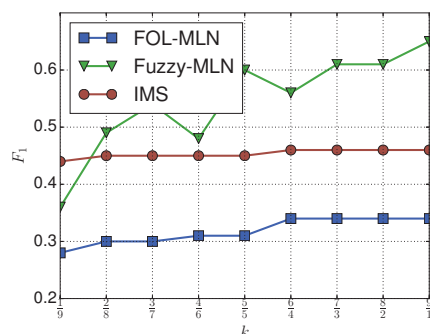
We conduct a modification of classical cross-validation, which we call ‘inverse’ k -fold cross-validation. We call it ‘inverse’, because inverse proportions of training and test set sizes are considered as well. For $k = 1/9$, for example, only 10% of the available data is used for training, and the remaining 90% serve for evaluation. Conversely, $k = 9/1$ corresponds to classical 10-fold cross validation. This makes the evaluation



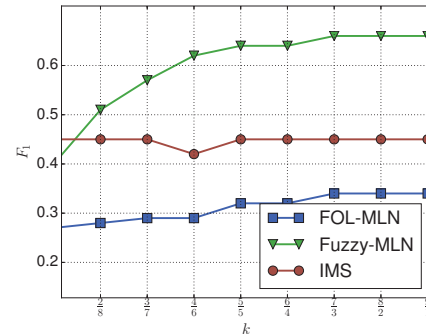
(a) Filling (fill.v.01)



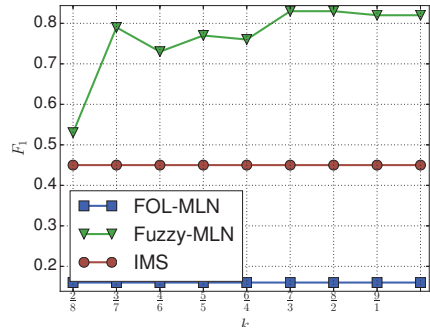
(b) Adding (add.v.01)



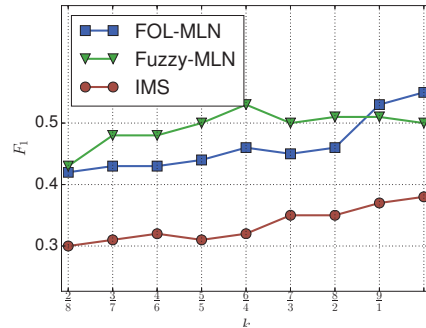
(c) Slicing (slice.v.03)



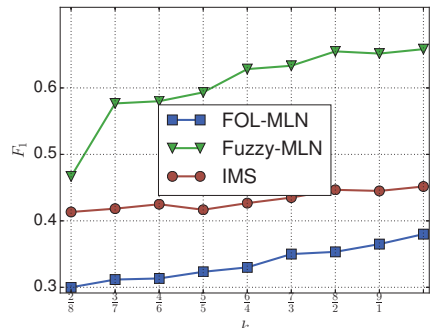
(d) Cutting (cut.v.01)



(e) Stirring (stir.v.01/08)



(f) Putting (put.v.01)



(g) Average over (a)-(f)

Figure 4-5: F_1 scores for inverse k -fold cross validation for $k = 1/9 \dots 9/1$ using classical MLNs with FOL semantics and Fuzzy-MLNs applied to a WSD problem of 20 examples per action verb.

Action Verb		k								
		1/9	2/8	3/7	4/6	5/5	6/4	7/3	8/2	9/1
Filling	FOL-MLN	0.40	0.41	0.41	0.42	0.44	0.49	0.44	0.46	0.51
	IMS	0.44	0.44	0.44	0.44	0.44	0.44	0.44	0.44	0.44
	FUZZY-MLN	0.64	0.69	0.67	0.68	0.75	0.75	0.75	0.75	0.75
Adding	FOL-MLN	0.27	0.29	0.29	0.32	0.29	0.34	0.38	0.36	0.38
	IMS	0.40	0.41	0.44	0.43	0.45	0.46	0.53	0.50	0.53
	FUZZY-MLN	0.44	0.50	0.49	0.51	0.49	0.52	0.57	0.56	0.57
Slicing	FOL-MLN	0.28	0.30	0.30	0.31	0.31	0.34	0.34	0.34	0.34
	IMS	0.44	0.45	0.45	0.45	0.45	0.46	0.46	0.46	0.46
	FUZZY-MLN	0.36	0.49	0.54	0.48	0.60	0.56	0.61	0.61	0.65
Cutting	FOL-MLN	0.27	0.28	0.29	0.29	0.32	0.32	0.34	0.34	0.34
	IMS	0.45	0.45	0.45	0.42	0.45	0.45	0.45	0.45	0.45
	FUZZY-MLN	0.40	0.51	0.57	0.62	0.64	0.64	0.66	0.66	0.66
Putting	FOL-MLN	0.42	0.43	0.43	0.44	0.46	0.45	0.46	0.53	0.55
	IMS	0.30	0.31	0.32	0.31	0.32	0.35	0.35	0.37	0.38
	FUZZY-MLN	0.43	0.48	0.48	0.50	0.53	0.50	0.51	0.51	0.50
Stirring	FOL-MLN	0.16	0.16	0.16	0.16	0.16	0.16	0.16	0.16	0.16
	IMS	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45
	FUZZY-MLN	0.53	0.79	0.73	0.77	0.76	0.83	0.83	0.82	0.82

Table 4.2: Left: F_1 scores averaged over all action verbs. Right: F_1 scores for inverse k -fold cross validation for $k = 1/9, \dots, 9/1$.

particularly challenging as models have to greatly generalize their knowledge to unseen concepts.

Results The instructions are grouped with respect to the action verbs they contain and use 20 examples per action verb in each fold because we observe that senses of words in a sentence strongly depend on the respective action verb. Splitting with respect to verbs thus allows easier interpretation of the results. A juxtaposition of F_1 scores of an ordinary FOL-MLN, the IMS algorithm and FUZZY-MLN is given in Table 4.2. FUZZY-MLNs clearly outperform the classical MLNs as well as the baseline algorithm in almost *every* test case. Moreover, FUZZY-MLNs achieve F_1 scores significantly above 0.5 even with very small portions of training data.

An obvious example is given by the ‘filling’ action depicted in Figure 4.5a, where the FUZZY-MLN outperforms the competitors by at least 20 percentage points for $k = 1/9$.

The F_1 score measures the classification accuracy with respect to word meanings from the taxonomy assigned to each word in the respective natural-language (NL) instruction. In cases of ‘putting’, all methods perform comparably poorly. An explanation for this is that the verb ‘put’ by its nature is very generic and thus does not impose a bias on its co-occurring words, which would be sufficient to effectively discern word senses.

It is interesting to note that, while only moderate improvements in ordinary MLNs and IMS can be observed with increasing amounts of training databases, the most significant performance jumps with FUZZY-MLNs are made when only sparse training data is used, i.e. for $k \in \{1/9, 2/9, 3/9\}$. In these extreme cases, where concepts occur in the test data that are not contained in the training data, classical MLNs (and all other approaches mentioned in the related work) are inapplicable to perform meaningful reasoning but are forced to randomly guess.

The experiments, although not carried out on a large-scale corpus, are sufficient to show that fuzzy inference in MLNs can perform adequate reasoning about concepts in the taxonomy that are not explicitly represented in the probability distribution and have not been seen during training and, in these cases, clearly outperform the state-of-the-art methods.

4.4 Related Work

A couple of frameworks have been proposed to incorporate concept taxonomies and similarity in probabilistic models, such as probabilistic description logics (Lukasiewicz, 2008; Niepert et al., 2011), tractable Markov logic (TML) (Domingos and Webb, 2012) and probabilistic similarity logic (PSL) (Brocheler et al., 2010), which FUZZY-MLNs differ from in basically two fundamental ways:

1. FUZZY-MLNs do *not* postulate uncertainty among the taxonomy structure as such, i.e. the structure itself is not subject to reasoning
2. FUZZY-MLNs do not model the whole taxonomy in the probabilistic model, but only the concepts seen during training.

This makes the FUZZY-MLN concept a more compact reasoning framework.

Tractable Markov logic (*TML*) is a subset of MLNs. TML introduces the idea of concept taxonomies in MLNs, but in order to perform reasoning about superclasses,

the inheritance relationship of concepts is explicitly represented in the model. By employing semantic similarity as evidence, the taxonomy relation is more compactly encoded in FUZZY-MLN.

Probabilistic similarity logic (PSL) uses a formalism similar to FUZZY-MLN. However, their probabilistic semantics differ fundamentally: By definition, all constituents of a ground formula in PSL are subject to the *degree-of-truth* semantics of multi-valued logics. This comes at the cost that in PSL, the partition function Z is required to integrate over all real-valued degrees of truth in the interval $[0, 1]$ of all variables in the ground MRF, which makes the models computationally even harder than in Boolean-valued MRFs. Conversely, in FUZZY-MLNs, we have both real-valued variables in the ground MRF that follow the degree-of-truth semantics, and boolean variables following the *degree-of-belief* semantics, whereas only the boolean variables are subject to reasoning and real-valued variables are required to appear as evidence only. This restriction is motivated by the assumption that ontology structures as such are given and do not exhibit any uncertainty, and at the same time it allows using existing inference algorithms for ordinary MLNs with only boolean variables avoiding the intractability of computing integrals over real-valued variables in the partition function. In addition, unlike FUZZY-MLNs, the goal of PSL is rather to reason about the degree to which a set of entities are similar to each other. Conversely, in FUZZY-MLNs the taxonomy is fixed and serves for filling gaps in the probabilistic KB.

Hybrid MLNs (HMLN) (Wang and Domingos) extend MLNs to reason about continuous variables. They discern features in hard FOL and numeric features that may be expressed as ‘soft’ (in)equality constraints. Those constraints are typically connected in a multiplicative way, such that, if a logical constraint evaluates to false, then also a connected numeric feature will have no influence on the probability of the respective possible world. Hence representing semantic similarities in HMLNs does not appear straightforward. The concept of soft evidence (Jain and Beetz, 2010) is closely related to the idea of vague evidence, though it has fundamentally different semantics for it still assumes boolean truth values and soft evidences serve as prior probability constraints on ground atoms.

To the best of my knowledge, none of these approaches can deal with entities that are not part of the probabilistic model in any meaningful way. This is a severe limitation, because they are not capable of exhaustively modeling joint probability distributions of realistic domain sizes. Since learning in first-order probabilistic models remains intractable in the general case, inference and generalization across concepts is essential and outstandingly important for probabilistic relational models to be scalable and applicable to real-world problems.

4.5 Discussion

A key feature in human cognition is the remarkable ability to rapidly yet flexibly learn and generalize from specific examples of how a particular word or concept is to be used in a certain situation, by just seeing very few examples of that word:

“In coming to understand the world – in learning concepts, acquiring language, and grasping causal relations – our minds make inferences that appear to go far beyond the data available.”
– (Tenenbaum et al., 2011)

This highly effective learning process is particularly impressive because it takes place already in very early childhood (Bailey, 1997). Children grasp the meaning of new words very quickly, categorize them into subsets and learn how to use them appropriately in new situations.

As an example, consider again the terms ‘cup’ and ‘milk’ and imagine how they are used in the context of the two instructions “fill a cup with milk” and “add a cup of milk,” from an NL recipe. In both cases the word ‘milk’ refers to the white nutritious beverage, which can typically be found in the refrigerator. The term ‘cup’, however, in the former case refers to a drinking mug, a physical object that is supposed to hold the milk, whereas in the latter case, it rather refers to an abstract entity, a unit of measure, specifying the amount of milk which is to be added to something not further specified. Now suppose a human is faced with an instruction “add a spoon of sugar,” without having ever encountered the word spoon before. Typically, humans are able to exploit their taxonomic knowledge to determine it can be either interpreted as a piece of cutlery or as a measuring unit. I argue that in such a case it is rational to choose the option which is most similar to a situation we have already seen, as one can expect it to behave similarly.

In knowledge representation and reasoning, the usage of ontologies and class taxonomies, hierarchically organized sets of concepts, have a long tradition as they appear to naturally represent a way of how humans organize their commonsense knowledge. Subsuming categories of entities in super-classes allows efficient reasoning about everyday situations for all entities being affected in a single symbol. Representation and reasoning about concept taxonomies in FOL has been widely studied in AI (Thau and Ludäscher, 2007) and so has been DL, a subset of FOL dedicated to represent and reason about such hierarchies. Due to the inherently uncertain nature of many real-world problems, however, the deterministic character of those formalisms leads to difficulties with respect to their practical applicability.

With the field of statistical relational learning (SRL), a variety of new languages and formalisms was given rise, which account for both the relational and the probabilistic character of many application domains (Getoor, 2007a). Among those, MLNs (Richardson and Domingos, 2006) have gained a lot of popularity and attention in recent years. They combine the expressiveness of FOL with probabilistic graphical models (PGMs)' ability to deal with uncertainty and have been proven successful in a variety of applications, such as collective classification, natural-language processing, entity resolution, image processing and many more. It therefore appears straightforward to incorporate reasoning about concept hierarchies in Markov logic.

As an example of knowledge representation that could be encoded in MLNs, consider a framework in the spirit of Minsky's frame structures (1974) or the PRAC framework presented in Chapter 3, which can be thought of as data structures for representing stereotyped situations, networks of nodes and relations, which represent a human's expectations about a certain situation. When encountering a new situation, a frame from memory is recalled and adapted as necessary to fit reality. Following this idea, one can elicit at least three fundamental requirements such a reasoning formalism must meet. The formalism must be

1. capable of representing abstract, stereotyped knowledge,
2. this knowledge must be adaptable to (and possibly acquired from) more specific, concrete situations and
3. new, unseen situations and concepts must be treated in a meaningful way.

From a representational point of view, a probabilistic logic approach seemingly puts itself forward for encoding such structures. Since the model is supposed to abstract away from specific occurrences of entities, it is also desirable to take into account a hierarchy of class concepts and thus enabling the representation of more general knowledge about particular situations. A limitation that many probabilistic methods for categorical variables have in common is that such models can perform inference only about propositions that are explicitly represented in the model, which comes with the two drawbacks:

1. Generalization on a conceptual level is difficult. In continuous probability spaces, for example, distributions are very compactly represented by a few parameters allowing to perform inference on a typically infinite number of propositions (i.e. numbers in \mathbb{R}). Such distributions often have a compact closed-form representation because of the existence of a total order on all

propositions allows to interpolate the probability density between two propositions. Since categorical domain values typically cannot be arranged in such an order, interpolation of the density is no longer possible and hence there is no way of assigning a probability mass to a proposition not explicitly represented in the model.

2. Consequently, a compact representation of generic knowledge is challenging, since every concept a probabilistic model is supposed to take into account needs to be explicitly incorporated. It can be argued that, in the realm of statistical relational learning, these limitations are mitigated since those models are supposed to generalize well across domains and hence are in this regard superior to their propositional counterparts. However, this only holds for non-conceptual domains, supporting the principle of shallow transfer. If we want to meaningfully discern different concepts in a model, we need to introduce symbols for every concept under consideration and explicitly represent them in the model.

This problem becomes clear when comparing models from continuous and symbolic probability spaces. Consider a Gaussian mixture model (Bishop, 2006) as shown in Figure 4.6a, which depicts a mixture of two Gaussian distributions, represented by their means μ_1 and μ_2 , as well as their variances σ_1^2 and σ_2^2 . The final distribution $P(x)$ is then defined as the weighted sum of Gaussians,



Interpolation in Symbolic Spaces

$$P(x) = \sum_{k=1}^K \pi_k \mathcal{N}(x; \mu_k, \sigma_k^2),$$

where π_k are mixing coefficients determining the strength of the influence of the k -th Gaussian, and \mathcal{N} is the normal distribution given by

$$\mathcal{N}(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right).$$

The parameters that fully determine the model are given by the π_k , μ_k and σ_k^2 and reasoning about any number $x \in \mathbb{R}$ can be made by referring to x in terms of its squared distance to the model parameters, $(x - \mu_1)^2$ and $(x - \mu_2)^2$. In the example shown in Figure 4.6a, the sample at \hat{x} is closer to μ_1 than to μ_2 , such that the red distribution will contribute more to the probability mass assigned to \hat{x} than the blue one. This is possible because the density function of a Gaussian allows to interpolate between the supporting points specified in the model and thus to generalize the knowledge from few supporting points to the whole space of real numbers. This

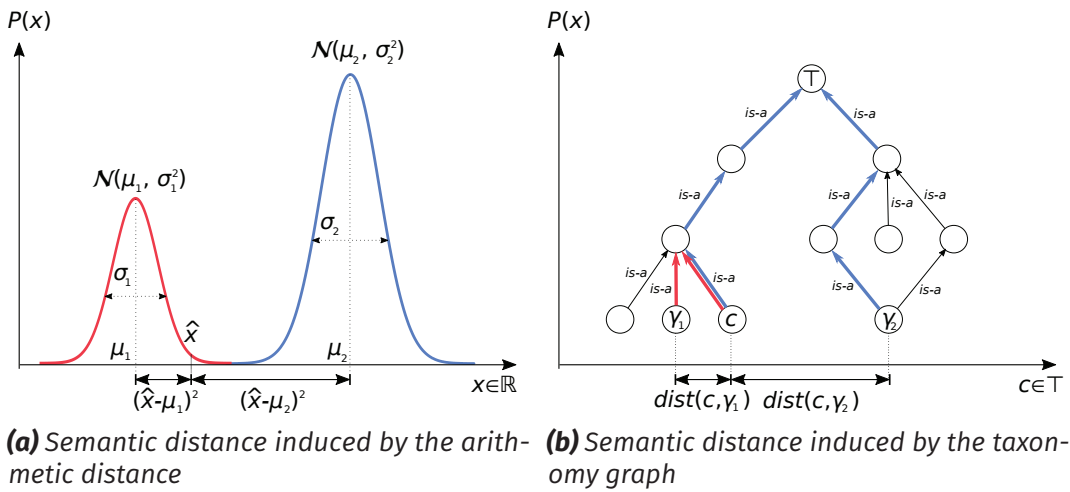


Figure 4.6: Interpolation in the Probability Density Function based on semantic distances in class taxonomies and real numbers.

makes such representations very compact.

In contrast, ordinary *symbolic* probabilistic models, such as MLNs, do not have as powerful generalization capabilities. Instead, they rely on every symbol subject to reasoning to have appeared in the training data and thus represent an explicit supporting point in the model. Sometimes, such models are based on contingency tables and reasoning is performed by a mere lookup. This impedes the deep transfer of knowledge to different domains, even when they are very similar.

The principal idea of FUZZY-MLN reasoning is to exploit the taxonomic structure of symbols that comes from an upper ontology and imposes a distance metric on the symbols, as shown in Figure 4.6b. The illustration shows a taxonomy of concepts encoded in DL, with the most abstract concept \top at its root. Two concepts γ_1 and γ_2 denote two (hypothetical) object classes that are explicitly modeled in the KB and therefore can be considered the ‘supporting points’ of the distribution. Given a distance metric $dist$, any arbitrary concept c can be referred to in terms of its distance to both γ_1 and γ_2 , $dist(c, \gamma_1)$ and $dist(c, \gamma_2)$, which are shown in form of the paths in red and blue, respectively.

This analog of continuous mixture models makes a key problem of symbolic KBs evident and offers a possible solution to it: the disregard of the structure among symbols in ordinary KBs impedes the generalization and deep transfer of knowledge from one domain to another. In turn, taking the class taxonomy from an upper ontology into account may help to exploit the semantic distance (or rather similarity) of symbols in order to ‘interpolate’ the probability mass to regions of the taxonomy that a model does not have explicit supporting points for.



Recent developments in statistical relational learning have tackled the problem of incorporating concept hierarchies into probabilistic logical models (Domingos and Webb, 2012). They incorporate uncertainty in inheritance relationships of taxonomies following the “degree of belief” paradigm which postulates sharp boundaries between concept classes in a hierarchy. However, as has been shown earlier, it can be beneficial to regard class memberships being not mutually exclusive, assuming that there are no sharp boundaries clearly demarcating our notions of categories and no general criteria for telling whether a particular entity belongs to one category and not to the other, exist.

Instead, as Varzi (2001) points out, class memberships rather seem to be gradual and their range of application have vague boundaries. In many practical cases, we do not have available data from which we can induce models that are sufficiently comprehensive and exhaustively cover all relevant cases. Instead, we experience data that are sparse and noisy and we cannot expect our data at hand to completely cover all concepts that are contained in a particular taxonomy. Hence, probabilistic KBs should be able to perform reasonably well on classes that have not been seen in the training set.

I argue that a key concept for powerful generalization from such severely sparse data is a notion of conceptual similarity. Employing a measure of semantic similarity comes with a fundamental advantage: there is no need for representing superconcepts explicitly in a model, the taxonomic inheritance relationship (the structure of the hierarchy) can be very compactly and efficiently captured by a numeric value, which aggregates knowledge about how a particular concept behaves with respect to concepts that have already been seen.

When performing probabilistic inference, one does not need to explicitly take into account superclasses in the inference process, just conditioning on the similarity of a concept to the known concepts suffices. This mechanism is both powerful and efficient with respect to the requirements mentioned in our introductory example of knowledge representation and reasoning. For reasoning about unknown concepts, which have no correspondence to a symbol or proposition covered by the model, this concept just needs to be related to the known concepts by application of some semantic similarity measure.

In this work, we have described the design and the implementation of FUZZY-MLNs, a reasoning framework for MLNs that allows to represent probability distributions over open domains compactly – if complete ontologies are available for these domains. FUZZY-MLNs exploit the semantic similarity of concepts in a taxonomy in order to handle off-domain concepts in previously unseen situations in a meaningful way

and hence allow efficient generalization from very sparse data whilst the original representation formalism of MLNs remains unchanged. The key idea of FUZZY-MLNs is to learn joint probability distributions conditioned on large taxonomic knowledge bases that are assumed to be given as factual knowledge. In contrast to existing more general probabilistic methods incorporating class hierarchies or similarities, FUZZY-MLNs do not target reasoning about the taxonomic structure as such. This comes with the advantage that the concepts subject to reasoning do not need to be exhaustively modeled in the probabilistic KB. This enables (1) compact representation of knowledge, (2) powerful generalization from sparse training data and (3) reduced computational complexity of learning and inference.

I have shown that FUZZY-MLNs can perform different probabilistic reasoning tasks in a way that matches our intuitions and can outperform probability distributions learned in the ordinary MLN framework as well as state-of-the-art classification systems both significantly and substantially.

Probabilistic Knowledge Bases for Robot Perception

Performing everyday activities in open human environments requires a robotic agent to robustly perceive objects of daily use. NL instructions can contain indispensable information about how the objects to be used look (e.g. “Bring me the green box with cornflakes.”), how they change while they are being manipulated (e.g. “Bake until golden.”) and what ramifications their appearance might have (e.g. “If the test strip turns red, the solution is acidic.”). Although the perceptual characteristics of objects hence play a crucial role in instruction understanding, the two problems are largely treated independently of each other. In this chapter, I investigate how the task of object perception can be grounded in natural language (NL), and how NL object descriptions can be transformed into semantic representations that can be used by ROBOSHERLOCK, a state-of-the-art perception system. To this end, I first report on a novel approach for learning probabilistic first-order KBs that combine many highly diverse perception routines to an ensemble of experts and therefore leverages synergies across multiple algorithms with different expertises. Based on this principle, I present an approach to transform phrases stated in NL that describe such objects by their visual appearance into formal, semantic representations of their perceptual cues, which in turn can be used in ROBOSHERLOCK in order to identify objects that the robot has never encountered before. The results of this chapter have been published in Nyga et al. (2014) and Nyga et al. (2017a).

5.1 Ensemble Learning with Markov Logic Networks



Figure 5.1: PR2 looking at a breakfast table.

A key problem implied by the task of perceiving objects of daily use is the variety of perceivable properties of objects, such as their shape, texture, color, size, text pieces and logos, transparency and shininess, that go beyond the capabilities of individual state-of-the-art perception methods. A promising alternative is to employ combinations of more specialized perception methods. In this section,

I report on a novel combination

method, which structures perception in a two-step process, and apply this method in the ROBOSHERLOCK perception system (Blodow, 2014; Beetz et al., 2015a). In the first step, specialized methods annotate detected object hypotheses with symbolic information pieces. In the second step, the given query Q is answered by inferring the conditional probability $P(Q | E)$, where E are the symbolic information pieces considered as evidence for the conditional probability. In this setting, Q and E are part of a probabilistic first-order model of scenes, objects and their annotations, which the perception method has beforehand learned a joint probability distribution of. The work presented in this section has been jointly conducted with Ferenc Bálint-Benczédi, who has contributed the acquisition of the perception data, the application of annotators to object hypotheses using ROBOSHERLOCK and the integration of the approach into the ROBOSHERLOCK perception framework. The author has contributed the design of the probabilistic models, their implementation, learning and evaluation. The results presented in this section have been published in Nyga et al. (2014).

5.1.1 Overview

One of the big challenges in object perception is that in most situations, the scenes that a robot has to perceptually interpret include objects with different perceptual characteristics. A scene on a breakfast table, for example, typically includes textured

objects such as jelly jars and cereal boxes, objects characterized by their shapes such as bowls and cups, translucent objects such as glasses, and small objects such as knives and forks. In the past, it has proven difficult to equip robots with perception algorithms that can handle objects with very different perceptual characteristics. In most cases, scenes are being drastically simplified to account for the perceptual capabilities of the individual methods.

A promising alternative is the development of perception systems that are capable of combining more specialized algorithms in a synergistic manner to better scale towards realistic environments and scenes. An example of such a perception system is the ROBOSHERLOCK framework (Blodow, 2014; Beetz et al., 2015a).

In the following, I report on a novel approach to combine the outputs of several diverse routines that are specialized on different perceptual characteristics of everyday objects in different scenes. The primary idea is to implement object perception in a two-step process: In the first step, specialized algorithms operate on perceived object hypotheses, extract perceptual information from the sensor data, and semantically annotate the respective object hypotheses with these pieces of information. Object hypotheses, in this application, are RGB-D point clusters as they can be obtained from a Kinect sensor, for instance. In a second step, the symbolic annotations of objects and the background knowledge about the whole scene are used as evidence to perform probabilistic reasoning about information that the perception system is requested for. To this end, the robot has previously learned a joint probability distribution over object classes, their annotations, the co-occurrences of objects, and the occurrence of objects in different kinds of scenes.

Possible queries are illustrated in Figure 5.1. The robot asks queries, such as “is the category of the object hypothesis c_1 a cereal box and what is the expected logo on the object hypothesis c_3 .” These queries are transformed into a query to a relational probabilistic model $P(\text{category}(c_1, \text{cereal}), \text{logo}(c_3, ?\text{logo}) \mid E)$ that take the observed scene as their evidence. Formulating the task of object perception as a relational probabilistic reasoning problem has several advantages over alternative approaches:

1. using perception algorithms for collecting perceptual evidence rather than making decisions makes the use of multiple specialized algorithms straightforward: they simply add their findings as annotations.
2. as inferences take into account uncertainty on the basis of collected evidence, the system can also handle inconsistent annotations in a meaningful way and learn the systematic errors of individual methods.
3. the system can answer queries about any aspect contained in the probabilistic

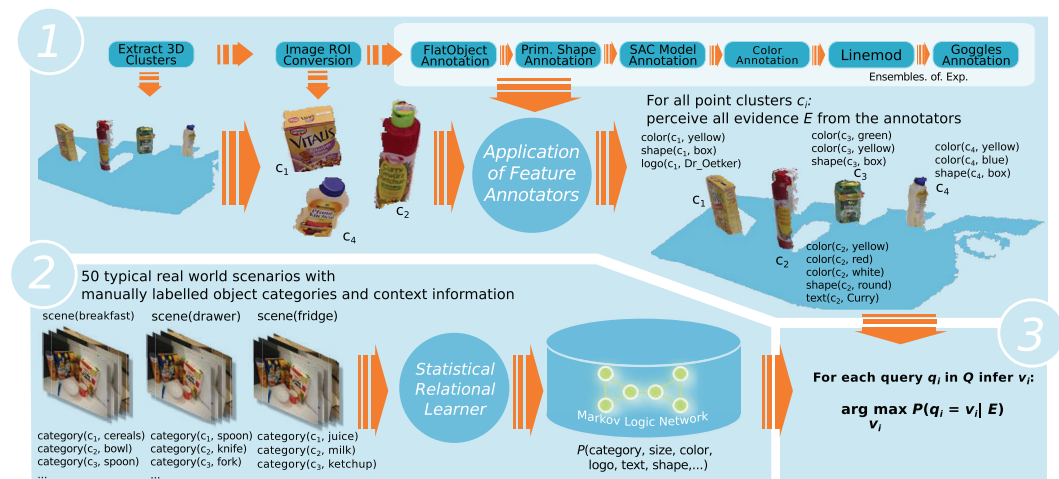


Figure 5.2: Architecture of the system: (1) segmentation of point clouds into regions of interest, (2) the statistical relational learning and (3) reasoning system.

model given any evidence.

- the perception system can also exploit the regularities of the domain with respect to objects and their appearance and the occurrence/co-occurrence of objects in scenes.

5.1.2 Processing Pipeline

(By courtesy of Ferenc Bálint-Benczédi)

Unstructured Information Management

ROBOSHERLOCK formulates the problem of object recognition in the paradigm of unstructured information management (UIM) (Ferrucci et al., 2010b). Following the UIM principle, ROBOSHERLOCK treats RGB-D scenes as a kind of unstructured documents, which it strives to enrich with semantic information. To this end, it searches for objects in pre-specified regions of interest, such as the top of counters, the content of drawers, and the content of refrigerators. It generates object hypotheses in these regions that might correspond to objects and annotates each cluster with symbolic information pieces such as the color, size, shape, text, and the logo on the respective objects. In this work, we describe how a probabilistic relational model can be constructed that uses these annotations as evidence E for probabilistic reasoning. Using a learned joint probability distribution over annotated scenes, ROBOSHERLOCK can then infer answers to perception-related queries Q by computing the conditional probability $P(Q | E)$.

Annotator	Condition	MLN Predicate	Description
Color	Always	color(cluster, color)	The color annotator returns semantic color annotation based on color distribution in HSV color space. Colors: <i>blue, red, black, green, yellow, white</i> . Depending on the distribution, one object can have multiple colors.
Size	Always	size(cluster, size)	The size annotator classifies objects into <i>small</i> or <i>big</i> depending on distance between extreme points normalized with the distance to the camera. Values returned: <i>big, small</i> .
Goggles	If Google goggles returns text or logos	logo(cluster, logo) text(cluster, text) texture(cluster, t)	The annotator sends the image region of interest to the Google Goggles servers and parses the answer to extract text, logo, and texture information.
FlatObject	If there are objects that are too flat to be found by the general 3D clustering	shape(cluster, shape)	After extraction 3D clusters from the table this annotator looks for additional object hypotheses in color space (e.g., napkins, ...).
PrimShape	Always	shape(cluster, shape)	This annotator fits lines and circles to 3D point clusters projected on to the 2D plane using RANSAC (Goron et al., 2012). Values returned: <i>box, round</i> .
LineMod	Confidence that <i>c</i> is one of the objects looked for exceeds threshold	linemod(cluster, category)	This annotator matches each object hypothesis to a set of object models that the robot should actively look for using the Linemod algorithm (Hinterstoisser et al., 2011).
SACmodel	If enough inliers for a model are found	shape(cluster, shape)	This annotator recognizes cylindrical objects in 3D space. If the number of inliers found exceeds the given threshold (60% of the total points in a cluster) the annotator accepts the match. Value returned: <i>cylinder</i> .
Location	Always	scene(scene!)	This annotator interprets object positions in terms of a semantic environment map (Pangercic et al., 2012) and returns places such as <i>counter tops, tables, fridges, and drawers</i> . The <i>depth-</i> and <i>RGB-</i> image are being filtered leaving only pixels in a pre-defined region of interest.

Table 5.1: Description of the annotators, the conditions under which they work and the predicate declarations in the Markov logic network.

Figure 5.2 gives a more detailed picture of the operation of the approach. It consists of three main components: (1) an image annotation component that segments point clouds into regions of interests that correspond to object hypotheses and annotates

the individual hypotheses, (2) a statistical relational learning engine that learns joint probability distributions over annotated scenes, and (3) the probabilistic reasoning system.

The image annotation component employs object segmentation mechanisms that detect objects on supporting planes, in drawers, and in refrigerators. As the focus of this paper is the boosting of object perception through method combination, this work is restricted to scenes in which the individual objects are clearly separable. Works that deal with object recognition in more cluttered scenes include Richtsfeld et al. (2012), Mian et al. (2006), and Marton et al. (2012).

Simple Euclidean clustering in 3D space for objects located on a supporting plane are applied. Segmentation methods using prior knowledge about the static environment in the form of semantic 3D object maps (Pangercic et al., 2012) filter out everything except for the region of interest. They in turn generate object hypotheses in the remaining region of interest. Some of the annotators (such as *SacModel*, *Size*, *PrimitiveShape*) use point clouds (Rusu and Cousins, 2011) as their input, whereas others use RGB images (*Linemod*, *Color*, *Goggles*). A converter is used to find the region of interest corresponding to the extracted clusters. Having a representation of object candidates in both 3D and RGB space, all annotators can be executed on the object hypotheses in order to attach their semantic information to those.

Annotators are specialized perception routines that perceive specific aspects of information. For example, the color annotator asserts the fact $color(c, col)$ for the cluster c . Another annotator uses Google Goggles, an internet service that retrieves web pages containing images similar to a given image (Blodow, 2014). Google Goggles works well for distinctively textured images, logos, and texts. This annotator labels object hypotheses with text (i.e. $text(c, txt)$) and logos (i.e. $logo(c, l)$). A short overview of the annotators used and the semantics of their outputs can be found in Table 5.1.

An example of a pipeline of annotators is depicted in the upper part of Figure 5.2. The pipeline first evaluates the flatness of the individual object hypotheses. Subsequently, object hypotheses are annotated with a simple shape symbol and their color. They are also compared to existing object models using the *Linemod* algorithm (Hinterstoisser et al., 2011). Finally, text and logo annotations are generated using the Google Goggles web service. A detailed performance analysis of the annotators will be presented in Section 5.1.4.

Given the annotations of objects E , the probabilistic reasoning component of the perception system can be used to answer queries about any aspect of the respective probabilistic model. The PRM represents the joint probability distribution over the combinations of the object classes and all possible annotations. The answer to the

query is then the $\arg \max_q P(Q = q | E)$.

The learning process of the joint probability distribution over manually labeled scenes is depicted in the lower part of Figure 5.2. The learning of the the probability distribution and the probabilistic reasoning mechanisms are detailed in Section 5.1.3.

5.1.3 Ensemble Learning

The previous section has described how the raw sensory input data is being transformed into semantically more meaningful object hypotheses by the application of experts, the so-called annotators. However, since most of the annotators producing those object hypotheses are applied independently of each other, their outputs are not guaranteed to be globally consistent and they typically do not take into account object interactions in the current scene. In fact, their annotations might even be incorrect or contradictory. Therefore, in order to come up with a final ensemble decision, a strategy for combining all the annotations is needed.

To this end, we learn joint probability distributions over the annotator outputs in PRMs, more specifically, in MLNs. In MLNs, we can capture complex object interactions, represent and reason about object properties, their attributes and the relations that hold between them. Most notably, the ultimate strength of PRMs is their capability of allowing to reasoning about *all* observations simultaneously, taking into account interactions between objects and thus achieving a posterior belief that is guaranteed to be probabilistically sound and globally consistent.

Maintaining a joint probability distribution over observations, their class labels and the robot's current task context and belief state has several advantages over classical approaches and makes the systems' reasoning capabilities go far beyond traditional classifier systems:

Collective Classification MLNs are able to simultaneously take into account any arbitrary but finite number of objects for classification. This is an important feature for a perception system, since it captures interactions between objects in a scene. If a classification system is aware of the probability of jointly encountering two objects of particular types, this can tremendously boost the classification accuracy in real-world scenes. Encountering milk and cornflakes together on a table, for instance, is much more likely than finding cornflakes and ketchup.

Confidence-rated Output Since the MLN for compiling annotations to a final decision is stacked upon the independent application of experts, such a probabilistic model is able to compensate for inconsistent annotations or uninformative features. If, for example, an annotator systematically confuses the shapes of clusters, the MLN will learn this erroneous hypotheses and treat them in a meaningful way.

Generative Models An MLN representing a joint probability distribution can be used to infer answers to arbitrary queries about any aspect represented in the model. As our experiments will show, the MLN can also be used to reason about the most informative visual features when looking for a particular type of object in scene, for example.

Extensibility integration of additional task-specific context information, or new specialized perception routines is straightforward. They just need to add their annotations to each of the object hypotheses and can be declaratively incorporated in the MLN.

From a logical point of view, the outputs of the feature annotators can be regarded as tables in a relational database and thus naturally correspond to predicates in FOL, and the segmented clusters represent the domain of discourse of entities we wish to apply probabilistic, logical reasoning to. Furthermore, we can think of the final class label, i.e. the object category we wish to predict, as an additional predicate. As an example, consider a scene of two objects c_1 and c_2 , where the ensemble of experts have identified c_1 being a yellow-ish box with a “Kellogg’s” brand logo on it, and c_2 being a round, blue thing. We can capture such a scene in a relational database as follows:

$shape(c_1, Box)$	$color(c_2, Blue)$
$color(c_1, Yellow)$	$shape(c_2, Round)$
$logo(c_1, 'Kellogg's')$	$shape(c_2, Round)$
$category(c_1, Cereal)$	$category(c_2, Bowl),$

where we have manually added information about object classes in the *category* predicate. It is straight-forward to construct an MLN relating object attributes with class labels. Assuming, for instance, that an object’s category correlates with its shape, a set of weighted formulae such as

$w_1 = \ln 0.66$	$shape(?x, Round) \wedge category(?x, Bowl)$
$w_2 = \ln 0.33$	$shape(?x, Box) \wedge category(?x, Bowl)$

can be added to the MLN, which naturally represent the rules “everything is a round bowl” and “everything is a box-shaped bowl.” By default, all variables are universally quantified in MLNs. Of course, the above rules do not hold for most of the entities we encounter in the real world and, in fact, they can be even considered mutually exclusive. However, according to the MLN distribution in Equation (2.14), the probability distribution defined by this MLN indicates that any world in which we encounter a round bowl is twice as likely as a world in which we find a box-shaped bowl (assuming that all other aspects in the respective possible worlds are identical). Following these considerations, abstract, coarse “rules of thumb” can be added to the MLN, modeling connections between the ensemble experts and the final ensemble decision. The weight parameters of the resulting MLN can be learned in a supervised learning manner.

For the MLN used to obtain the final ensemble decision experts, the logical predicates described in Table 5.1 are used, which naturally correspond to the annotator outputs in ROBOSHERLOCK. Two additional predicates are used for specifying knowledge about the current context (i.e. the type of scenario) the perceptual task is performed in and for assigning a class label to each of the clusters in the scene under consideration:

- *scene(scene)*: represents background knowledge about the current context in which the perceptual task is being performed. Possible contexts are $dom(scene) = \{Breakfast, Cooking, Drawer, Fridge\}$ denoting a breakfast table scene, a cooking scene, a view into a drawer and a view into a fridge.
- *category(cluster, object!)*: assigns a class label to each cluster in the scene. In the experiments, we distinguished 21 different object categories (see also Figure 5.3).

In the MLN syntax, the ‘!’ operator in a predicate declaration specifies that this predicate is to be treated as a functional constraint for the respective domain, i.e. for every cluster $c \in dom(cluster)$, there must be exactly one true ground atom among the possible ground atoms with respect to the *category* predicate. In other words, we require exactly one object category association for each cluster. Since a particular cluster or entity cannot be of two different categories at a time, this model constraint is a reasonable assumption.

The following MLN has been designed in order to model correlations between annotator outputs and the object classes:

$$w_1 \quad size(?c, +?sz) \wedge shape(?c, +?sp) \\ \wedge color(?c, +?cl) \wedge category(?c, +?obj)$$

w_2	$linemod(?c, +?ld) \wedge category(?c, +?obj)$
w_3	$logo(?c, +?logo) \wedge category(?c, +?obj)$
w_4	$text(?c, +?t) \wedge category(?c, +?obj)$
w_5	$scene(+?s) \wedge category(?c, +?obj)$
w_6	$category(?c_1, +?t_1) \wedge category(?c_2, +?t_2) \wedge ?c_1 \neq ?c_2,$

where the ‘+’ operator specifies that the respective formula will be expanded to one individual formula for every value in the respective domain.

The domain *text* of the predicate requires a special treatment: since its output is based on OCR text recognition by Google Goggles, this domain is potentially infinite and noise-prone. Thus, a mechanism is needed for transforming arbitrary strings into a proper set of symbolic constants. To this end, we applied a SAHN (sequential, agglomerative, hierarchical, non-overlapping) clustering technique to the values of the *text* domain in the training data before running the learning process. As a distance measure, the well-known Levenshtein distance (Brill and Moore, 2000) has been chosen. Subsequently, every string (in both the training and test data) has been replaced by its nearest cluster centroid. This mechanism mitigates the negative effects of noise in the OCR annotations, since every unknown text is mapped to a known string which is explicitly represented in the model.

5.1.4 Experiments and Results

With the experiments presented in this section, we will demonstrate the following properties of the proposed system. We will show that

1. hypotheses of individual annotators can be significantly boosted by applying probabilistic first-order KBs,
2. it is beneficial to take into account object-object co-occurrences in a perceptual classification model,
3. the proposed method is robust towards inconsistent annotations, which can be treated in a meaningful way,
4. the capabilities of the probabilistic KBs go far beyond ordinary classifier systems, which are mainly given by discriminant functions with dedicated in- and output variables.

Object	Accuracy	Precision	Recall	F₁-Score
Bowl	1.00	1.00	1.00	1.00
Cereal	0.98	0.80	0.80	0.80
Chips	0.99	0.83	0.71	0.77
Coffee	0.99	1.00	0.75	0.86
Cup	0.98	0.77	1.00	0.87
Fork	0.90	0.09	0.07	0.08
Juice	0.96	0.75	0.71	0.73
Knife	0.89	0.29	0.32	0.30
Ketchup	1.00	0.88	1.00	0.93
Milk	0.97	0.77	0.67	0.71
Mondamin	0.98	0.64	1.00	0.78
Oil	0.98	0.78	0.64	0.70
Pancake maker	1.00	1.00	1.00	1.00
Pitcher	1.00	1.00	1.00	1.00
Plate	0.95	0.77	0.82	0.79
Popcorn	0.99	0.83	0.83	0.83
Pot	0.99	0.75	1.00	0.86
Salt	0.99	0.75	0.75	0.75
Spatula	0.93	0.71	0.45	0.56
Spoon	0.91	0.35	0.38	0.36
Toaster	0.99	0.57	1.00	0.73

Table 5.2: Class-specific error measures for 10-fold cross-validation.

To this end, we arranged and recorded 50 realistic scenes, each comprising 5-10 instances of 21 different object categories, which can typically be found in kitchen scenarios. We discern four different kinds of scenarios: a breakfast table, a cooking scenario, a look into a refrigerator and a look into a kitchen drawer. The types of scenarios have been incorporated into each data set and can be regarded as task-specific knowledge about the current context of an activity. This is a reasonable presumption, since the location the robot is currently looking at can be assumed to be known from e.g. a map of the environment as described by Pangercic et al. (2012), and co-occurrences of objects are highly correlated in real-world scenarios. The object categories for each object have been manually labeled.

The weights have been determined by supervised learning of the manually labeled data using pseudo-log-likelihood learning with a Gaussian zero-mean prior of $\sigma = 10$,

Pred./Truth	Bowl	Cereal	Chips	Coffee	Cup	Fork	Juice	Ketchup	Knife	Milk	Mondamin	Oil	PancakePan	Pitcher	Plate	Popcorn	Pot	Salt	Spatula	Spoon	Toaster	
Bowl	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Cereal	0	8	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
Chips	0	0	5	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
Coffee	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Cup	0	0	2	2	20	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0
Fork	0	0	0	0	0	1	0	0	7	0	0	0	0	0	0	0	0	0	0	0	3	0
Juice	0	1	0	0	0	0	12	0	0	1	0	1	0	0	0	1	0	0	0	0	0	0
Ketchup	0	0	0	1	0	0	0	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Knife	0	0	0	0	0	9	0	0	6	0	0	0	0	0	0	0	0	0	0	1	5	0
Milk	0	0	0	0	0	0	3	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0
Mondamin	0	0	0	0	0	0	0	0	0	1	7	1	0	0	0	0	0	0	0	2	0	0
Oil	0	0	0	0	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	2	0	0
PancakePan	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	0	0
Pitcher	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0
Plate	0	0	0	0	0	2	0	0	0	0	0	0	0	0	23	0	0	0	3	2	0	0
Popcorn	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	5	0	0	0	0	0	0
Pot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	1	1	0	0	0
Salt	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	3	0	0	0	0
Spatula	0	0	0	0	0	0	0	0	0	0	0	1	0	0	3	0	0	0	0	10	0	0
Spoon	0	0	0	0	0	3	0	0	6	0	0	0	0	0	0	0	0	0	0	2	6	0
Toaster	0	1	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	4

Table 5.3: Confusion Matrix for 10-fold cross-validation on the data set of 50 scenes. The rows represent the predictions, ground truth is given in the columns. Objects with the most misclassifications are knives, forks, spoons and spatulas, which means that they are visually hardly discernible.

Pred./Truth	Bowl	Cereal	Chips	Coffee	Cup	Fork	Juice	Ketchup	Knife	Milk	Mondamin	Oil	PancakePan	Pitcher	Plate	Popcorn	Pot	Salt	Spatula	Spoon	Toaster
Pred./Truth	Bowl	Cereal	Chips	Coffee	Cup	Fork	Juice	Ketchup	Knife	Milk	Mondamin	Oil	PancakePan	Pitcher	Plate	Popcorn	Pot	Salt	Spatula	Spoon	Toaster
Bowl	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Cereal	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Chips	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Coffee	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Cup	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Fork	10	0	4	9	20	15	0	1	19	7	4	8	6	3	28	5	6	4	16	16	4
Juice	0	2	0	0	0	0	14	0	0	3	1	2	0	0	0	1	0	0	3	0	0
Ketchup	0	0	1	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0
Knife	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Milk	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0
Mondamin	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0
Oil	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
PancakePan	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Pitcher	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Plate	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Popcorn	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Pot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Salt	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Spatula	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	3	0	0
Spoon	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Toaster	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

(a) Cross-validation using only the Goggles annotator. It is strong in **(b)** Cross-validation using only the shape annotator. It is strong in classifying textured cereal boxes, juice and ketchup, but poor on bowls, classifying textureless bowls, plates and cups, but poor on textured plates and cups.

(b) Cross-validation using only the Goggles annotator. It is strong in **(a)** Cross-validation using only the shape annotator. It is strong in classifying textured cereal boxes, juice and ketchup, but poor on bowls, classifying textureless bowls, plates and cups, but poor on textured plates and cups.

Annotator	# total Annotations	# correct Annotations	Predictive perf. (acc.)
Color	289	231 (79.9%)	17.5%
Goggles	80	—	21.2%
Prim. Shape	336	233 (69.3%)	26.1%
SACmodel	38	31 (81.5%)	
FlatObject	142	116 (81.6%)	19.6%
Linemod	90	45 (50%)	

Table 5.5: Evaluation of annotators in isolation: Correctness of their annotations and their predictive performance when applied in 10-fold cross-validation. Shape, SACmodel and FlatObject have been aggregated since they all contribute to the ‘shape’ predicate.

which serves regularization purposes.

For evaluating the model’s performance in identifying object classes of entities given the observations from the annotators as evidence, we performed 10-fold cross-validation on the 50 scenes we recorded. The results are shown in Figure 5.3 and 5.2. It can be seen that the model achieves reasonably high classification accuracies with respect to precision, recall and F_1 -score. Indeed, the system achieves F_1 -scores significantly above 70% for all objects except for the cutlery. The comparatively low performance on cutlery can be explained by the limited resolution of the sensor, which is insufficient for the annotation experts to distinguish between the marginally observable differences between forks, spoons and knives.

However, I argue that the overall performance of the system is remarkable compared to the individual performances of the single annotators in isolation. Figure 5.5 shows an evaluation for each of the annotators, counting the number of times an expert has annotated an object, the correctness of its annotations as well as the predictive performance an MLN consisting of only one formula containing the respective predicate would achieve. Note that Linemod’s low performance is due to the fact that its main strength is recognizing untextured objects, but we created models for textured objects as well.

Figure 5.4 shows the confusion matrices for MLNs that have been trained with only one annotator each, in particular the *Goggles* and the *shape* annotator. It can be seen that the individual annotators perform poorly on the entire data set, but each achieves quite good results in a particular subset. The *Goggles* annotator, for example, shows good performance on product images and textured objects like the cereal boxes and the juice cartons, but fails on untextured object like cups and plates. On the other hand, the *shape* annotator fails on most of the products, but succeeds

Ground Atom	Cereal	Chips	Cup	Pot
<i>color(c, black)</i>	0.3302	0.3476	0.2864	0.3582
<i>color(c, blue)</i>	0.2954	0.3316	0.2186	0.3148
<i>color(c, red)</i>	0.3852	0.3656	0.3452	0.3388
<i>color(c, white)</i>	0.3508	0.4216	0.2806	0.3768
<i>color(c, yellow)</i>	0.4264	0.3484	0.4422	0.2936
<i>text(c, VITALIS_A)</i>	0.623	0.0000	0.0000	0.0004
<i>logo(c, DrOetker)</i>	0.136	0.0006	0.0000	0.0000
<i>logo(c, Kellogg's)</i>	0.3734	0.0000	0.0000	0.0008
....
<i>linemod(c, PfannerIce)</i>	0.0004	0.0000	0.0008	0.0002
<i>linemod(c, Popcorn)</i>	0.7392	0.0006	0.0000	0.0010
<i>linemod(c, Pot)</i>	0.0008	0.0004	0.0004	0.9994
<i>linemod(c, PringlesVin)</i>	0.0000	0.0000	0.0004	0.0006
<i>linemod(c, PringlesSalt)</i>	0.0002	0.4986	0.0010	0.0006
....
<i>shape(c, box)</i>	0.4806	0.3870	0.2810	0.3556
<i>shape(c, cylinder)</i>	0.3722	0.4540	0.4010	0.4266
<i>shape(c, flat)</i>	0.3226	0.3682	0.2864	0.3862
<i>shape(c, round)</i>	0.3176	0.4092	0.5182	0.4068
<i>size(c, big)</i>	0.368	0.3442	0.3768	0.3292
<i>size(c, small)</i>	0.2626	0.2686	0.3148	0.2836

Table 5.6: (Partial) probabilities for different queries about visual features conditioned on the object class.

in identifying plates, cups and bowls. Hence, the single annotators can indeed be regarded complementary with respect to their individual expertise, though neither of them is strong enough to perform well on all of the object classes. The ensemble given by the MLN, however, adapts to the individual strengths and weaknesses of the experts and thus can treat contradictory annotator outputs competently in order to come to a final decision.

Inferring the most probable categories given the observed properties of each objects is only one possible kind of queries the system can answer. Indeed, the learned joint probability distribution over objects and their attributes allows reasoning about *arbitrary* queries with respect to any variable that is contained in the model. Our approach can also be used to reason about the perceptual features to be expected

when looking for a particular object in a scene. If the robot is supposed to find a box of cereals on a breakfast table, for instance, the following query can be formulated in order to retrieve the most informative features for distinguishing the cereal box from other objects:

$$P \left(\begin{array}{l} \mathit{shape}(c, ?sh), \mathit{color}(c, ?c), \\ \mathit{size}(c, ?s), \mathit{logo}(c, l?), \mathit{text}(c, ?t) \end{array} \middle| \begin{array}{l} \mathit{scene}(\mathit{breakfast}), \\ \mathit{category}(c, \mathit{Cereal}) \end{array} \right).$$

Figure 5.6 shows an excerpt of the probability distribution computed for the above query. where the most probable solution is printed bold. Querying for the most probable solution (i.e. the most probable explanation (MPE) state), symbolic descriptions of the expected visual properties objects can be deduced, which are highlighted: The cereals are expected to be big, yellow and red boxes, on which one can read the text 'VITALIS _A', and the Linemod annotator would consider it popcorn (which is another example of how the MLN learns and compensates for the errors of individual experts).

5.1.5 Discussion & Related Work

Most autonomous manipulation robots employ perception systems that are trained with appearance models of the objects they are to detect, recognize, and localize. In operation, the robot uses a database of trained objects to match against the perceived sensor data. Successful examples of such robot object perception systems are described by Aldoma et al. (2012) using point features of 3D opaque objects, or *MOPED*, which uses visual keypoint descriptors for the learned textured objects (Collet et al., 2011) and specialized perception systems for translucent objects (Lysenkov et al., 2012). In the presented work we encapsulate these methods as annotators, use them as experts, and therefore boost the performance by exploiting these algorithms.

The ensemble-based learning approach presented in this section has been fully integrated into the ROBOSHERLOCK framework and applied on a real robot performing everyday activities. Figure 5.3 shows a look on a kitchen table from the perspective of a robot making pancakes. The detected object hypotheses are marked by their bounding boxes. The logical annotations of the individual perception routines are attached to these bounding boxes and so are the final class labels assigned by the MLN.

A variety of methods exist that can handle reasonably well some of the subproblems

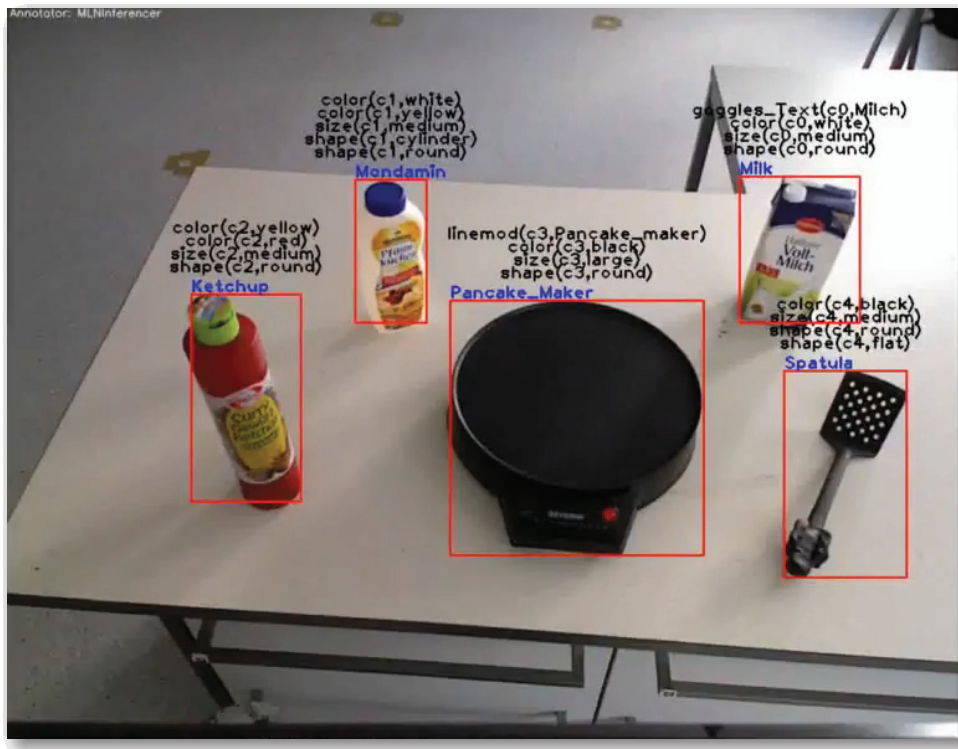


Figure 5.3: Object classification tasks in a household scenario: the annotations from individual perception routines are printed in black, the classification result from the MLN in blue (By courtesy of Ferenc Bálint-Benczédi)

of perception. Many of these methods are complementary and could be combined to enhance performance. Examples of such methods are door handle detectors (Rühr et al., 2012). Using the proposed approach, it is straightforward to include such methods in the ensemble.

With respect to its operation, the approach of the ROBOSHERLOCK system falls into the category of unstructured information management (UIM) systems – systems that look for segments of unstructured information that have a deeper structure. In 3D perception, RGB-D point clouds can be considered as unstructured information that contain object hypotheses as nuggets of more structured information. Object hypotheses have several perceptual features as well as symmetry and compactness properties. UIM primarily facilitates hypothesis generation, testing, and ranking and the use of ensembles of expert methods. It is primarily investigated in the area of webscale information systems, most prominently in the context of the Watson system (Ferrucci et al., 2010a). In ROBOSHERLOCK, this technology is transferred and used for robot perception. For a deeper view on ROBOSHERLOCK, I refer the reader to Blodow (2014) and Beetz et al. (2015a).

Ensemble of expert-based systems have proven to be very successful in the area of machine learning and computational intelligence and hold great promise for boosting the perceptual capabilities of robots. Polikar (2006) presents an excellent overview of ensemble-based systems. A categorization of ensembles is provided by Jain et al. (2000) based on architecture, trainability, level of information the members produce and adaptability.

An example of a robotic perception system employing the ensemble of experts idea is presented by Okada et al. (2007). They combine multiple detectors and views based on particle filters. The probabilistic fusion of different results corresponds to a simple rule ensemble, i.e. one that is not trainable. MRFs have been successfully used for object detection based on human interaction (Wu and Aghajan, 2010), and in object segmentation (Kumar et al.). In the context of robotics, Sun et al. (2013b) introduce the combination of appearance attributes and object names in order to identify objects in a scene. Pronobis and Jensfelt (2012) describe a framework for semantic mapping based on a combination of object attributes, room appearance and human input. Our approach can be regarded as a combination of their approaches, having as inputs only visual cues of the objects and the domain knowledge (e.g. the scene type).

5.2 Interpretation of Natural-language Object Descriptions

As described in the introductory section, the recognition and competent interpretation of perceptual characteristics of everyday objects can yield important information about which objects to be used during an activity and in which ways their visual appearance may change as an effect of the activity. In particular, identifying objects, their properties and inferring how they relate to an instruction

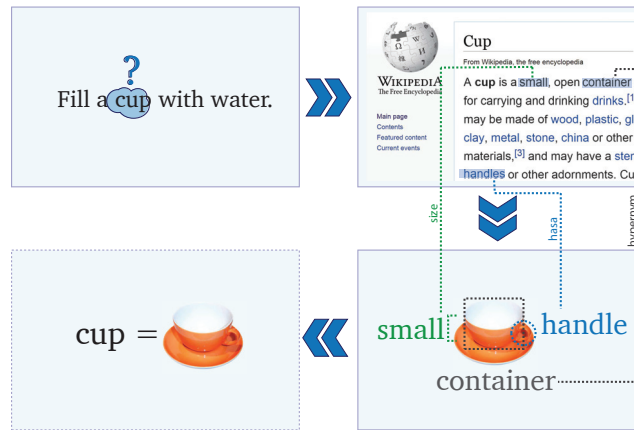


Figure 5.4: Pipeline how a robot can detect objects it has never seen before.

is a crucial ability of robots for competently performing everyday activity. Consider, for instance, the instructions “pour flour into the metal bowl” and “put the sugar bowl on the table.” In the former example, the word ‘metal’ denotes the material that the bowl is made of, whereas in the latter case, the word ‘sugar’ refers to the stuff which is held by the bowl. Though the two cases are syntactically not discernible, their semantics are fundamentally different. Likewise, an instruction like “bring me the carton with the text ‘orange juice’ on it” requires a robot to understand that it has to search for a box with the label ‘orange juice’ on it. In this section I present mechanisms for appropriately interpreting textual descriptions of objects of daily use given in natural language and for identifying them in real-world scenarios. To this end, the PRAC system is combined with probabilistic perception models described in Section 5.1. Parts of the work presented in this section have been conducted jointly with Mareike Picklum (2015) and published in Nyga et al. (2017a).

The overall idea of the approach is illustrated in Figure 5.4. Imagine a situation in which a robot is tasked with the instruction “fill a cup with water,” but it is missing a perceptual model for detecting a cup. The robot might then look up articles about cups in online sources like Wikipedia, which it analyzes for perceptual attributes that are typical for cups, such as “it is a relatively small container which has a handle.” Those primitive visual attributes can then be used by the robot’s perception system in order to identify cups in its environment. To realize such a system, I propose a novel *probabilistic, knowledge-driven* approach to attribute-based object recognition from natural language (NL).

The task of identifying objects from an NL description is formulated as a two-step probabilistic reasoning process: First, a phrase in NL is interpreted by transforming it into a formal semantic representation of the objects of discourse with respect to the perceptual attributes mentioned in the phrase. Second, this semantic representation is used in the ROBOSHERLOCK perception framework and matched against the annotator outputs attached to object hypotheses in the scene. To this end, every symbol considered by the reasoning system, be it a word in the NL description or a visual attribute in the perception framework, is grounded in a taxonomy of concepts, in particular the WordNet taxonomy. This comes with two major advantages:

1. All symbols subject to reasoning have a globally consistent and well-defined semantics: all concepts referring to object attributes such as *oval.s.01*, *box-shaped.s.01* and *cylindrical.s.01*, or *red.n.01*, *green.n.01* and *yellow.n.01* for example, correspond to concepts in the WordNet taxonomy and are used consistently in all components of the system. This means that the meanings of words in the NL description can be directly mapped to the output of perception algorithms that detect, for instance, shapes or colors in a scene.
2. The taxonomy relation *is_a(·, ·)* makes it possible to exploit the semantic similarity. This enables powerful generalization across concepts, since words that have not been seen during training can still be treated in a meaningful way: If the probabilistic interpreter for object descriptions has been trained with only a small number of colors, it can transfer the learned knowledge about how colors are used in object descriptions to unseen words denoting a color, since they are more similar to each other with respect to their location in the taxonomy.

This is in contrast to previous approaches towards attribute-based object recognition: A common approach to relating NL descriptions of objects is to collect large amounts of images and textual descriptions of them. Rudimentary feature extraction is applied to the raw image data, which are used to learn shallow mappings between features and NL terms. The feature attributes need to be learned from vast amounts of data and the data in turn needs to cover the space of features sufficiently exhaustively. In particular, every word that the system is to process needs to be seen beforehand in order to be able to match it against what is represented in the model. Learning and reasoning in the proposed approach is less data-intensive since it exploits *knowledge* about super-concepts of objects.

The remainder of this chapter is organized as follows. First, I introduce the notion of *knowledge-based* attributes in object perception – attributes that are not just abstract symbols but are linked to a rich taxonomy of concepts giving them abstract symbolic

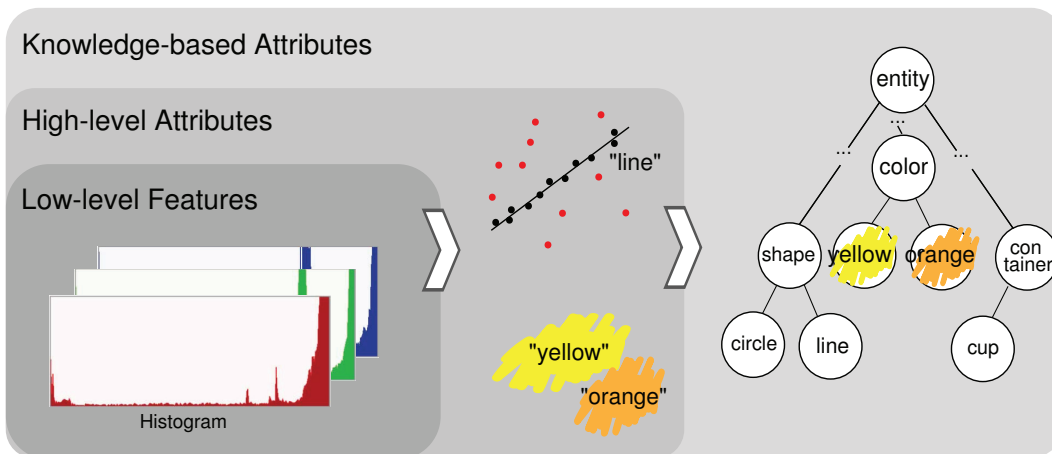


Figure 5.5: Overview over the different attribute types. Low-level features: mostly given by numeric feature vectors like (color-, gradient-) histograms High-level attributes: Attach symbolic identifiers to regions in feature spaces. Knowledge-based attributes: ground high-level features into an ontology enabling symbolic reasoning about them.

meaning. Based on these attributes, I explore the construction of probabilistic semantic models for object recognition in first-order probabilistic models, which can be used to identify objects that have never been part of any training data of a perception system. Then, I evaluate a proof-of-concept implementation of the approach, on a set of 14 different object types.

5.2.1 Semantic Models for Robot Perception

In this section, I will introduce the notion of *knowledge-based* attributes and present how they can be used to interpret and identify objects in a real-world scene. There are two layers of abstraction on object attributes reported in literature.

Low-level Features are features that are often closely related to individual instances, that means they denote characteristics not necessarily of an object category, but rather of one specific image or object. Features such as color or gradient histograms (Chang and Krumm, 1999), SIFT (Lowe, 2004) or HOG (Dalal and Triggs, 2005) are examples of these kinds of features. They often occur in form of vectors of numerical values and are used as inputs for training classifiers like SVMs or decision trees. The values of low-level attributes can be easily compared by a suitable norm to the feature values of previously seen objects and therefore allow recognizing objects that have been part of the training data. However, they lack a well-defined semantics, which makes them hard to be interpretable for humans.



Figure 5.6: A kitchen scene in ROBOSHERLOCK with segmented regions of interest (clusters) and their logical annotations created by specialized perception routines.

High-level Attributes represent the current state of the art and have shown promise in various applications for robot perception (Nyga et al., 2014) and object identification from natural language (Sun et al., 2013a). They abstract away from low-level attributes in the way that they aggregate regions in the space of low-level attributes by assigning a symbolic identifier to those regions, which are used for further processing. High-level attributes are more general than low-level attributes in the sense that they can be used to describe object categories in a more abstract way. For example, a certain region in color space may be assigned to a symbol *red*, and likewise a region in gradient space may be assigned a symbol *box*. This enables to refer to an object by its attributes on an abstract symbolic level instead of less meaningful histogram values and has proven superior to low-level features (Farhadi et al., 2009). They are intuitively human-understandable (Duan et al., 2012) and are typically learned from a number of examples via classifier learning, manually engineered or are computed by model fitting algorithms such as RANSAC (Goron et al., 2012).

Knowledge-based Attributes I propose an additional layer of abstraction, which I call *knowledge-based attributes*. Knowledge-based attributes add another layer of abstraction to high-level attributes by grounding the abstract symbols into a taxonomy of concepts. This comes with the advantage that abstract symbols have a globally unique and well-defined semantics and are not just unrelated categorical identifiers. In turn, they can be related to each other in terms of their superconcepts in the taxonomy. This enables for example, to state that the symbols *oval*, *elliptical*, *round* and *circular* are not precisely synonymous, but **similar** to each other because they are all subclasses of the concept *shape*, but more dissimilar to the concepts *red*, *small*

or *container*. On the internet one can find a number of high-quality taxonomies of concepts. One of the most popular ones is the WordNet lexical database. A summary of the different layers of object attributes is depicted in Figure 5.5.

5.2.2 The ROBOSHERLOCK Perception Framework

As a perception framework, the ROBOSHERLOCK system is used as an implementational basis for performing robot perception. As described in Section 5.1.2, ROBOSHERLOCK works in a two-step process which segments the perceived RGB-D image into multiple regions of interest, i.e. point clusters in the image representing object hypotheses in the scene, and executes perception routines that attach symbolic annotations to clusters. Annotators can wrap any arbitrary perception algorithm and thus can be thought of as an ‘expert’ with respect to the attributes of that algorithm. The annotator outputs attached to clusters are represented by atomic propositions in first-order logic. For example,

$$\begin{array}{ll} \text{color}(s_1, \text{red}) & \text{shape}(s_1, \text{cylinder}) \\ \text{color}(s_1, \text{yellow}) & \text{logo}(s_1, \text{Hela}) \end{array}$$

states that the cluster s_1 has been identified as a red and yellow cylinder, which has a ‘Hela’ brand logo on it. A typical scene with clusters and their annotations is shown in Figure 5.6. Having learned a joint probability distribution in the form of an MLN, the object classification task can be formulated as a probabilistic query

$$\arg \max_{c_1, c_2, c_3, c_4, c_5} P \left(\begin{array}{l|l} \text{object}(s_1, c_1) & \text{color}(s_1, \text{red}) \\ \text{object}(s_2, c_2) & \text{color}(s_1, \text{yellow}) \\ \text{object}(s_3, c_3) & \text{shape}(s_1, \text{cylinder}) \\ \text{object}(s_4, c_4) & \text{logo}(s_1, \text{Hela}) \\ \text{object}(s_5, c_5) & \text{color}(s_2, \text{black}) \\ & \dots \end{array} \right) \quad (5.1)$$

i.e. the most probable object class assignments c_1, \dots, c_5 of the 5 exemplary clusters s_1, \dots, s_5 given their object attributes as evidence. Like any other classification system, the MLN performing the classification task in (5.1) needs to be trained with manually labeled data, i.e. the learning algorithm needs to be provided with real instances of those objects that the system is to perform reasoning about. This comes with the

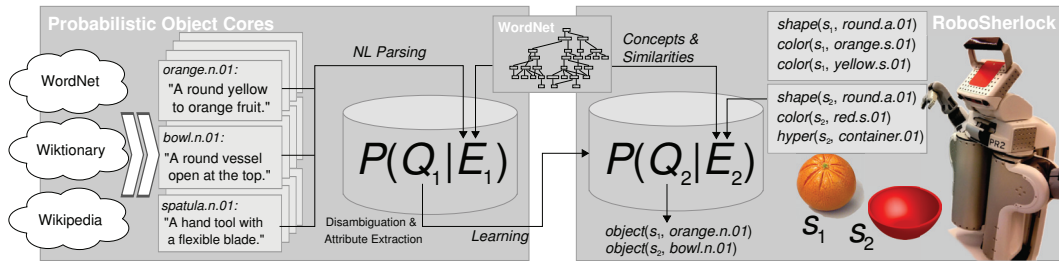


Figure 5.7: High-level architecture of the the proposed system system and its interaction with ROBOSHERLOCK: object descriptions in NL are mined from web pages like Wikipedia. The parsed NL sentences and ontological knowledge from WordNet serve as evidence E_1 in the the proposed system, which performs word sense disambiguation and extraction of visual attributes Q_1 . A second probabilistic first-order KB is trained with analyzed object descriptions, which is used in ROBOSHERLOCK for object recognition on the robot. It receives object properties of clusters as evidence E_2 and performs object classification Q_2 .

disadvantage that ROBOSHERLOCK can only identify objects of classes that it has been provided with during training time.

5.2.3 Identifying Objects from Descriptions

The key idea of this work is to generate the MLN representing the probability distribution in (5.1) from natural-language descriptions of objects that can be found on the web instead of scenes with perceived real objects. This adds to a perception system substantially more generality and flexibility, since the diversity of objects that a robot can recognize is not fixed at design time, but can be extended during runtime by describing objects in a naturalistic manner. Consequently, a robot would be able to recognize objects that it has never seen before and the necessity to acquire large amounts of data for training is obsolete. To this end I propose the PRAC system to extract object attributes from natural-language descriptions that in turn serve as training data for the MLN performing object classification in ROBOSHERLOCK. The general architecture of the system is depicted in Figure 5.7.

Semantic Similarity

Most of the symbolic perceptual attributes that are assigned to clusters in ROBOSHERLOCK have a natural correspondence in natural-language terms, such as primitive shapes (*box, cylinder, flat, ...*), colors (*red, blue, ...*) or sizes (*big, small, ...*). It

appears therefore straightforward to extract these attributes from natural-language descriptions of objects and match them against the attributes computed by annotators in ROBOSHERLOCK. However, those symbols as such lack an explicit semantics and hence do not naturally relate to each other. That is, the terms *box*, *box-like*, *cuboid* and *cube*, for example, represent different symbols that are entirely unrelated, though their implicit meaning is very similar.

Previous works by Sun et al. (2013a) have tackled this problem by collecting large bodies of textual descriptions from human subjects, and by applying unsupervised learning techniques in order to find relations between the words contained in the descriptions to features extracted from images by image processing algorithms. This approach comes with a few drawbacks. The efforts accompanied by the data acquisition process are huge and one might run into subtle issues when this process is not carried out with great care, which can lead to undesired results. For example, it is crucial that the pictures the subjects are being faced with and asked to describe are selected carefully to cover the feature space as exhaustively and as evenly as possible in order to keep the unsupervised learner from overfitting. This approach represents a rather ‘shallow’ mapping from words to object attributes.

In the proposed system, a different approach is taken, which is less data-intensive, but instead exploits knowledge that is already available in online dictionaries like WordNet. By raising the symbols used in ROBOSHERLOCK to the level of knowledge-based attributes given by ontological concepts, they can be directly related to word senses that have been assigned by PRAC to words in natural-language descriptions. As described in Chapter 4, the subsumption relation $is_a(\cdot, \cdot)$ induces an order on the symbols, which enables to exploit their *semantic similarity* during reasoning. Two concepts are considered similar if they reside in close areas in the graph spanned by the is_a relation, i.e. the shorter the path connecting two concepts in the graph is, the more similar they are assumed to be. Conversely, a long path connecting two concepts postulates dissimilarity. Hence the WUP similarity introduced in Chapter 4 can be used. The semantic similarities of concepts enable to treat *unknown* terms, i.e. concepts that have not been seen during training of the system, in a meaningful way. If an annotator in ROBOSHERLOCK assigns the attribute $color(s, purple.n.01)$ to a cluster s , for example, the symbol $purple.n.01$ can be related to the concept $violet.n.02$ by means of their similarity $purple.n.01 \sim_{\square} violet.n.02 \approx 0.94$ given by the taxonomy, which means that the two concepts are very similar. Conversely, the similarity of the concepts $box.n.01$ and $purple.n.01$ is only $purple.n.01 \sim_{\square} box.n.01 \approx 0.13$. Note that from now on, WordNet concepts are used instead of plain symbols to denote object attributes.

Interpretation of Natural-language Object Descriptions

Natural language is typically severely unstructured and ambiguous, i.e. there may exist different meanings for one word. As an example, the term ‘orange’ can refer to a fruit on the one hand and to a color on the other. In order to successfully *interpret* an object description in natural language, we thus need a method to determine the most appropriate sense of a word, depending on the context, i.e. the *word sense disambiguation* (WSD) problem. However, for understanding it is insufficient to merely assign a concept to every word in an object description. It is also necessary to determine which attribute is denoted by which word, which can be mapped to the problem of *action role labeling* in context of the PRAC interpreter.

From a probabilistic point of view, the task of object recognition from natural language can be formulated as a two step inference process: First, the object description in NL is analyzed with respect to the objects and their perceptual characteristics mentioned and second, the respective objects need to be found in a given scene under consideration, which is done according to (5.1). In this section, I focus on the first step of extracting visual attributes from NL sentences and present the probabilistic model that is used to accomplish that. As an example, consider the phrase ‘A yellow or orange fruit.’ The reasoning task of extracting visual attributes from an object description into a more semantic representation is given by

$$\arg \max_{sh, c, s, \dots} P \left(\begin{array}{c} color(o, c) \\ shape(o, sh) \\ size(o, s) \\ \dots \end{array} \middle| \begin{array}{c} \text{“A yellow or} \\ \text{orange fruit.”} \\ \dots \end{array} \right), \quad (5.2)$$

i.e. we infer the most probable attribute assignments given the NL description, among all possible worlds. For representing the conditional probability distribution in (5.2), we learn a probabilistic first-order KB, which is encoded in an MLN. The MLN contains a dedicated predicate for each perceptual attribute of objects we want to perform reasoning about. In the current implementation, there are five different attribute types, in particular three property types representing visual attributes which have also been found informative by other authors (Sun et al., 2013a; Wang et al., 2009; Alomari et al., 2016) namely the $color(\cdot, \cdot)$, the $size(\cdot, \cdot)$ and the $shape(\cdot, \cdot)$ of an object and two relational attributes $has-a(\cdot, \cdot)$ and $hypernym(\cdot, \cdot)$ denoting meronym (‘part-of’) and hypernym relations, respectively. The $hypernym$ predicate is used to extract attributes that relate an object to a more abstract super-concept in an NL description, which may contain valuable information about its perceptual

characteristics. An example is the term ‘container’, which is commonly used to describe specializations of the respective concept. ROBOSHERLOCK has a specialized annotator for detecting objects that can contain stuff. This annotator can, for instance, be used to directly match this attribute in a scene. The *has-a* predicate is used to specify proper physical parts of objects, such as handles, knobs, caps, lids or the like, for which also specialized annotators exist in ROBOSHERLOCK.

The processing pipeline for computing (5.2), which is depicted in Figure 5.7 and works as follows. First, an NL sentence subject to reasoning is analyzed for its syntactic structure using a NL parser such as the Stanford Parser (Klein and Manning, 2003). The result of the parse is a set of logical atoms representing syntactic dependencies and part-of-speech (POS) tags of the words in the sentence, which are the initial evidence database E_1 for the inference process. For each word and its part of speech, WordNet is then queried for all possible word senses for the respective word in the sentence, which is added to the query Q_1 as hypotheses of the form *has_sense(word,sense)*. Like in PRAC, the possible word senses are linked to the taxonomy of WordNet via the *is_a(sense,concept)* predicate. The crucial point in this step is that word senses that are not contained in the MLN as symbols, i.e. concepts that have been part of the training data, are asserted as vague evidence in terms of their semantic similarity to known concepts via the *is_a* predicate. Subsequently, for every word in the sentence, we add to Q_1 every possible atom relating a word in the sentence to a visual attribute it might represent. The arg max solution is then the most probable word sense and -attribute assignment in Q_1 conditioned on the taxonomic knowledge and the syntactic structure of the sentence in E_1 .

5.2.4 Semantic Similarity Measures

As described in the previous section, a key feature in the proposed system is its ability to reason about object attributes stated in natural language in a meaningful way, even if the respective terms are not contained in the probabilistic KB. This is possible because every symbol is grounded into the WordNet ontology of senses, which allows the exploitation of taxonomic knowledge about concepts and in turn enables to refer to unknown concepts in terms of their similarity to known concepts. As a measure of similarity, I have already introduced the WUP-similarity \sim_{\square} .

However, since the WUP similarity only takes into account relative path lengths between two concepts, it might yield unexpected results when applied to concepts corresponding to visual attributes. This is based on the peculiarities of the WordNet

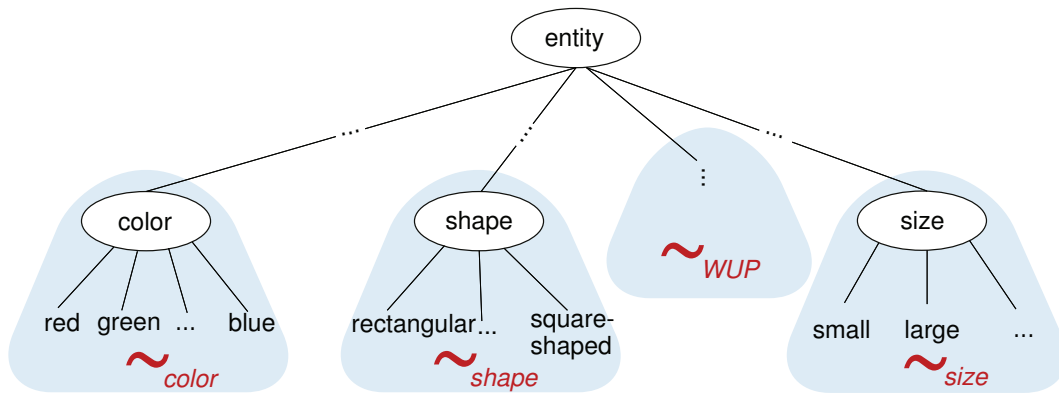


Figure 5.8: For concepts of the categories *shape*, *size* and *color*, customized similarity calculations are used, the default measure is the WUP similarity.

ontology which primarily represents hypernym relations, as these do not necessarily reflect the desired concept relations adequately. As an example, consider the concepts representing symbolic colors, such as *blue.n.01*, *yellow.n.01*, and *orange.n.02*. For they all have the concept *chromatic_color.n.01* as a direct hypernym, \sim_{\square} yields

$$\begin{aligned} & \textit{blue.n.01} \sim_{\square} \textit{yellow.n.01} \\ & = \textit{blue.n.01} \sim_{\square} \textit{orange.n.02} \\ & = \textit{yellow.n.01} \sim_{\square} \textit{orange.n.02} = 0.875, \end{aligned}$$

i.e. with respect to the taxonomy relation, all colors are equally similar to each other. This is counter-intuitive, since one would expect the concept of the color orange to be more similar to yellow than to blue. The same effect can be observed with shapes and sizes. It thus shows impractical to let the similarity of concepts exclusively depend on their taxonomic relatedness without further consideration of deeper semantic meaning. We have therefore developed customized similarity measures for particular subregions of the taxonomy, which have been carefully designed in order to mitigate the negative effects of purely path-based similarity measures. In particular, we present \sim_{color} , a customized similarity for color concepts, \sim_{shape} , a customized similarity for shape concepts, and \sim_{size} , a customized similarity for size concepts.

They are combined in a unifying similarity measure \sim , which is defined as

$$x \sim y := \begin{cases} 0 & \text{if } \neg is_a(x, t) \vee \neg is_a(y, t) \\ x \sim_t y & \text{if } is_a(x, t) \wedge is_a(y, t) \\ x \sim_{\square} y & \text{otherwise.} \end{cases}, \quad t \in \{color, shape, size\} \quad (5.3)$$

\sim selects the respective customized similarity measure \sim_t if both objects x and y are a subtype of $t \in \{color, shape, size\}$, or assigns 0 similarity if only one is of type t . \sim_{\square} is used as the default. (5.3) can easily be extended with additional custom similarities to account for more diverse perceptual characteristics in the taxonomy. We will describe our customized variants in the following. A graphical representation of \sim is shown in Figure 5.8.

Colors The semantic similarity of concepts representing colors is defined in terms of numeric values in the HSV (Hue-Saturation-Value) color space, which is illustrated in Figure 5.9. To every color concept in WordNet, we have manually assigned a representative HSV color value as indicated for a subset in the table in Figure 5.9.

Note that for chromatic colors, the values mostly differ in the hue. Achromatic colors like white, black and gray on the other hand are easier to discern looking at their saturation and value. Therefore we treat chromatic and achromatic colors differently. Intuitively, one would not consider a certain chromatic color, say blue, to be more similar to gray, white or black. At the same time, another chromatic color is not more or less similar to white or black than blue. The similarity between a chromatic and an achromatic color is therefore defined as a fixed value, while the similarity between two colors of the same type is defined with respect to their assigned values.

The similarity of two chromatic colors x and y is defined as the residual of the Euclidean distance of their assigned HSV color vectors $x = (x_h, x_s, x_v)$ and $y = (y_h, y_s, y_v)$,

$$x \sim_{color} y = 1 - \sqrt{(x_h \ominus y_h)^2 + (x_s - y_s)^2 + (x_v - y_v)^2},$$

where we have introduced a customized difference operator for the hue values, \ominus ,

$$x \ominus y := \begin{cases} \frac{|x-y|}{180} & \text{if } |x-y| \leq 180 \\ 1 - \frac{|x-y| \bmod 180}{180} & \text{otherwise.} \end{cases}$$

Since the hue (H) value is typically arranged in a circle and ranges in $[0, 360]$, \ominus ensures that maximally dissimilar colors lie on opposite sides of the circle and that the distance is normalized.

Sizes The size of an object can be regarded as a one-dimensional property. We can therefore easily assess similarities of size attributes by establishing a total order on concepts denoting sizes. We do so by assigning a numerical value to each adjectival concept denoting size in WordNet, where smaller numbers refer to smaller sizes. As

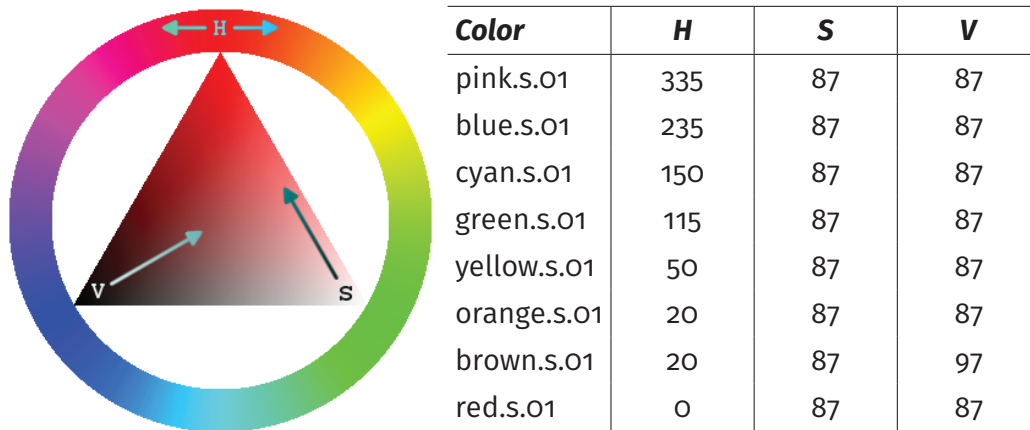


Figure 5.9: Left: HSV Color wheel: The HSV color model can be represented as a wheel, the hues (H) continuously arranged along the circular region, the saturation (S) along the vertical axis of the inner triangle and the value (V) along the horizontal axis of the triangle (Wikimedia Commons, 2017) Right: Feature vectors for colors: each color symbol is assigned a vector containing the respective values for the corresponding color in the HSV color model.

an example, the term *little* is assigned a small value, say 2, and the term *huge* is assigned a larger value, such as 8 or 10. The similarity of concepts is then defined as the residual of the normalized Euclidean distance, $x \sim_{size} y := 1 - \|x_n - y_n\|$. It is, of course, only a rough approximation of size values, which does not always reflect sizes appropriately since statements about object sizes are often relative to the object type (a small car is larger than a big spoon). However, as the intended application domain involves mainly objects of daily use, it is sufficient for this specific use case.

Shapes Geometric shapes are more challenging to assess with respect to their similarity, since they can be characterized along multiple dimensions. Unfortunately, there is not a single, coherent similarity measure which is applicable to determine similarities of concepts of shapes, since the individual manifestations of shapes can vary a lot. Using only the taxonomic relations of shapes is not suitable to determine their respective relatedness as their categorization may vary depending on the interpretation of their properties. In our setting we do not discern 2D and 3D shapes as we found it is common in NL object descriptions to use terms like *rectangular* and *boxlike* or *round* and *spherical* interchangeably. Using a path-based similarity calculation in a taxonomy purely based on hypernym relations does not necessarily reflect the intuitive relatedness of the incorporated concepts. An example is illustrated in Figure 5.10. The *cylinder* in the lower left corner is a subcategory of *tapered* which itself is a subclass of *round* due to its circular base. According to the standard WUP similarity the *cylinder* would now be considered closely related to other subclasses

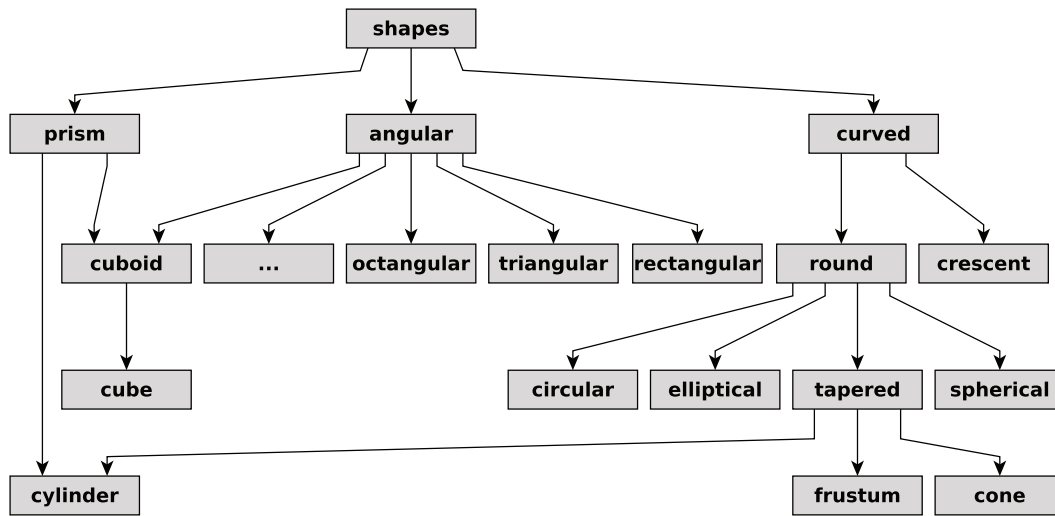


Figure 5.10: The shapes structured in a taxonomy: shapes can be subordinates of multiple super-concepts and may be similar to shapes in different taxonomy branches even if the hypernym relation does not reflect this intuitive relatedness.

of *round* (e.g. *spherical*) which does not correspond to the intuitive understanding of their respective similarity. Therefore we define a relation representing mutually exclusiveness of certain subcategories, in this case of *tapered* and *spherical* shapes. Conversely, two shapes may be intuitively considered very similar although they are located in different taxonomy branches so the path-based similarity calculation does not reflect this relatedness. As an example, a polygon approximates a *circle* with increasing number of edges but is considered very dissimilar to any kind of *round* shape as its sub-branch is split from the *curved* sub-branch in the taxonomy. It is therefore necessary to employ relations linking these concepts together. We manually define mappings for these special cases and use the WUP similarity as default to calculate the relatedness of two shapes.

The color, size and shape features can also be learned from data as presented shown by Kay and Regier (2003), who have learned color terms in different languages by transferring colors of a certain stimulus array to the CIEL*a*b color space (Hunter, 1958), averaging over color centroids named by speakers of certain languages and then coerced back to a color most similar to it in the stimulus array. Another approach would be to learn feature terms from data collected from Amazon’s Mechanical Turk, as reported by Guadarrama et al. (2013), where the collected data is used to learn spatial relation propositions to train a multi-class logistic regression model. The focus of this work, however, is the use of available knowledge and ontological information to interpret natural-language object descriptions and to align it with the output of a perception framework used on a real robot which avails itself of the same knowledge

sources.

5.2.5 Experiments

In this section I report on the experiments that have been conducted in order to demonstrate as a proof of concept the proposed system's ability to recognize objects from natural language.

We trained an MLN representing the conditional distribution in (5.2) with a training set of 39 descriptions from WordNet labeled with word senses that have been manually extended with object attributes for performing word sense disambiguation and extraction of object attributes. 56 object descriptions of 14 different object classes that have been extracted from different sources on the Web. In particular, we used descriptions from WordNet, Wikipedia, FreeBase and Wiktionary ¹. From these sources, we took object descriptions in natural language, such as “A round yellow to orange fruit of any of several citrus trees” as a description of an orange (*orange.s.01*) and applied the MLN in (5.2) in order to transform them into the formal, semantic representation in first-order logic, e.g. *color(o,yellow.s.01)*, *color(o,orange.s.01)*, *shape(o,round.a.01)*. These formal representations of object attributes serve as training data for the second MLN, which performs the actual object classification in ROBOSHERLOCK. We performed 10-fold cross-validation on the object recognition task, the results of which are shown in a confusion matrix in Figure 5.11. The type of objects have been selected such that they exhibit both very similar objects, such as forks, knives and spoons, which are hard to distinguish, and object categories that we expected to be easily separable, such as bananas and plates. It can be seen that, using the MLN generated from natural language, the system achieves reasonably high recognition rates among different kinds of object families, such as the fruits, containers and cutlery. It is interesting to note that the only object classes that are confused do indeed have very similar descriptions. As an example, consider the concepts *pot.n.01* and *bowl.n.01*. For *pot.n.01*, the description in WordNet states “a round vessel that is open at the top”, and *bowl.n.01* is described as a “metal or earthenware cooking vessel that is usually round and deep”. Our experiments show that as a proof-of-concept, our approach is able to discern objects only provided with natural-language descriptions. However, the descriptions that can be typically found in dictionaries and encyclopedic articles are insufficiently precise in order to separate similar objects.

¹<http://www.wikipedia.org>, <https://www.wiktionary.org>, <https://www.freebase.com>

Prediction/Ground Truth	banana.n.02	bowl.n.01	cherry.n.03	coffee.n.01	cup.n.01	fork.n.01	knife.n.01	lemon.n.01	orange.n.01	pan.n.01	plate.n.04	pot.n.01	spatula.n.02	spoon.n.01
banana.n.02	4	0	0	0	0	0	0	1	2	0	0	0	0	0
bowl.n.01	0	2	0	0	0	0	0	0	0	0	0	2	0	0
cherry.n.03	0	0	4	0	0	0	0	0	0	0	0	0	0	0
coffee.n.01	0	0	0	4	0	0	0	0	0	0	0	0	0	0
cup.n.01	0	0	0	0	4	0	0	0	0	2	0	0	0	0
fork.n.01	0	0	0	0	0	0	2	0	0	0	0	0	0	0
knife.n.01	0	0	0	0	0	4	2	0	0	0	0	0	0	2
lemon.n.01	0	0	0	0	0	0	0	2	0	0	0	0	0	0
orange.n.01	0	0	0	0	0	0	0	1	2	0	0	0	0	0
pan.n.01	0	0	0	0	0	0	0	0	0	2	0	0	0	0
plate.n.04	0	1	0	0	0	0	0	0	0	0	4	0	0	0
pot.n.01	0	1	0	0	0	0	0	0	0	0	0	2	0	0
spatula.n.02	0	0	0	0	0	0	0	0	0	0	0	0	4	0
spoon.n.01	0	0	0	0	0	0	0	0	0	0	0	0	0	2

Figure 5.11: Confusion matrix for 10-fold cross validation on our data set of 56 NL object descriptions of 14 different object categories.

The proof-of-concept implementation has been integrated into ROBOSHERLOCK, which we run on a PR2 robot to test the system in a real-world setting. Figure 5.12 shows a snapshot of a scene from the perspective of the robot looking at a table. The robot has not seen any of those objects before.

5.2.6 Related Work

The field of recognizing objects from natural language has gained a lot of attention in recent years.

Matuszek et al. (2012) present an approach to build a joint model of language and perception for grounded attribute learning. They learn how natural language is represented and extract the meaning of it to ground it in the physical world. Guadarrama et al. (2014) introduce an approach for mapping object images to a set of descriptive words, which are then compared to a natural-language query. Several works deal with classification tasks that make use of semantic attributes Jayaraman et al. (2014); Yu et al. (2013), especially in the context of recognizing unseen objects Su et al. (2010) or activities Cheng et al. (2013). Duan et al. (2012) exercise fine-grained recognition of categories closely related to each other. Farhadi et al. (2009) use descriptions of objects instead of their name to identify one specific instance. In particular, they categorize objects based on their semantic attributes, such as color, material or shape. Their system allows them to report the absence

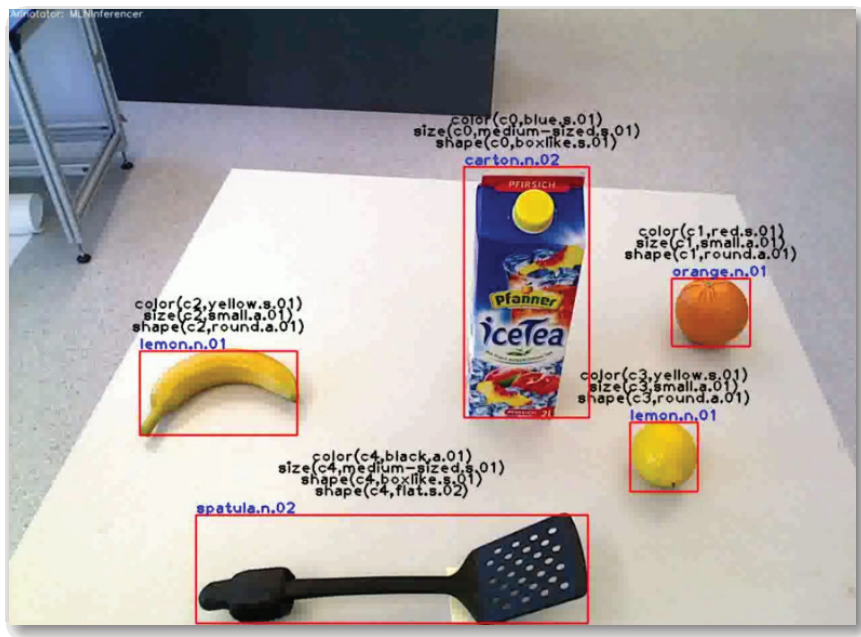


Figure 5.12: Snapshot of a table-top scene from the perspective of the robot. The ROBOSHERLOCK annotator outputs (black) and the object categories inferred by the MLN (blue). The banana is misclassified since it is described as an “elongated crescent-shaped yellow fruit with soft sweet flesh,” but ROBOSHERLOCK currently cannot recognize the attribute *crescent.a.01*.

of attributes of familiar objects (such as a missing head of a bird) or, on the other hand, the presence of atypical attributes. Lampert et al. (2009) present methods to classify unseen objects by using attribute-based classification, using high-level descriptions of the objects. Wang et al. (2009) use fixed patterns to parse natural-language descriptions to extract attributes. Classification is then done by maximizing the likelihood of the given image under each model. Tellex et al. (2014) present an approach to soften the requirement of annotations specifying mappings between symbols in natural language instructions and their corresponding groundings in the external world by unsupervised learning of word meanings from an unaligned parallel corpus. Sun et al. (2013a) present an attribute-based approach for object identification and use sparse coding techniques to learn rich RGB-D features from manually labeled datasets. Alomari et al. (2016) learn visual characteristics of objects, such as colors and shapes, by connecting language with visual observations of basic manipulation actions in video sequences. Liang et al. (2013) introduce a semantic representation highlighting a parallel between dependency syntax and the evaluation of logical terms to be able to map questions in natural language to answers via latent logical forms. Kong et al. (2014) use NL descriptions to improve 3D semantic parsing. They use an MRF to reason about the type of a scene, 3D object classes and which

visual concept a (pro-)noun refers to.

Most of these approaches have in common that they are data-intensive and learn rather direct shallow mappings between natural-language terms and perceptual attributes in mostly unsupervised manners. To the best of our knowledge, our approach is the first that makes heavy use of available knowledge and ontological information, which leverages costly data acquisition and preprocessing steps. The proposed system can deal with concepts and terms that they have never encountered in any perceptual training data, since every term in the system has a well-defined semantics, and disambiguation, extraction and mapping of visual attributes takes into account their similarities that are encoded in the underlying taxonomy of concepts.

5.3 Conclusions

In this chapter, I have reported on two extensions of the ROBOSHERLOCK perception framework for robots acting in everyday human environments.

The first extension is a novel way of combining arbitrary perception routines that are specialized to particular perceptual cues of objects. In contrast to existing systems, which mainly focus on employing a specific perceptual algorithm, the novel approach follows the paradigm of ensembles of experts – sets of diverse and highly specialized algorithms that are strategically combined in order to draw a well-informed final conclusion. To this end, the proposed system instead makes use of MLNs, a powerful relational probabilistic learning and reasoning framework allowing to take into consideration also object-interactions like they are encountered very frequently in real-world scenarios. I have proven the strength of this approach by a profound evaluation of our system’s performance on a data set of 50 typical kitchen scenarios. The evaluation shows that the PRMs employed are well-suited for the computation of posterior beliefs over given queries, though the expressiveness of the individual perception routines is quite poor.

The second extension is a novel knowledge-driven approach for interpreting object descriptions in natural language, which are used in the ROBOSHERLOCK system. Unlike previous approaches, the proposed system makes use of already available knowledge by raising visual object attributes to the level of concepts in the comprehensive taxonomy WordNet, which I call *knowledge-based attributes*. The use of knowledge enables to deal with previously unseen concepts in a meaningful way by referring to them in terms of their similarity to known concepts, which makes the

proposed system highly efficient for learning from sparse data. The experiments show that semantic object models for robot perception can be learned from NL descriptions mined from the web and used for perceiving objects of daily use on a real robot. The PRM for identifying perceptual properties in instructions is also used in PRAC.

Evaluation

It is a challenge on its own to come up with an evaluation of an AI system, which is convincing, objective, and fair. Not least because our notion of intelligence in terms of “doing the *right* thing” always leaves room for interpretation and, as I have elaborated in Chapter 3, intelligent behavior is “something that is compatible with the agent’s past experience, future intentions, and knowledge of the situation in which the agent finds itself” (Anderson, 1995). Our ultimate goal is a complete robotic agent that is able to successfully *perform* complex real-world manipulation tasks formulated in vague natural language (cf. Anderson (1995); Barker (1968)). Assessing the performance of such a robotic system in a quantitative manner, however, is practically impossible at the current state of the art due to the absence of established benchmarks or approved evaluation criteria and methods. Commonly used linguistic corpora of instructions do not account for the behavior that the analyzed instruction produces. This makes a large-scale corpus-based evaluation of PRAC nearly impossible.

The previous chapters already provided quantitative evaluations of individual sub-components where available. This chapter examines the contributions of this work on a larger scale, depicting the current capabilities of the PRAC system and showcasing its use in some of the real-world demonstrators that have been implemented at the Institute for Artificial Intelligence.



Figure 6.1: The browser-based web interface to PRAC on a desktop and a mobile client: the natural-language instruction “slice the bread” is transformed into the executable robot plan and executed in the web-based simulator. (courtesy of Mareike Picklum)

6.1 Open-source Software

PRACWEB The PRAC system has been released as an open-source software project and made accessible to the public in a browser-based web application called PRACWEB¹, which has been jointly developed with Mareike Picklum. The applications are also available as mobile-optimized variants for smart phone devices. Figure 6.1 shows screen shots of the systems in operation. The results of the single inference steps are visualized in an interactive graph structure representing the semantic networks of entities and relations as they are presented at several points throughout this thesis. The overall process of the pipeline can be followed by means of a flowchart diagram highlighting the PRAC reasoning module which is currently executed. The users can

¹<http://prac.open-ease.org>

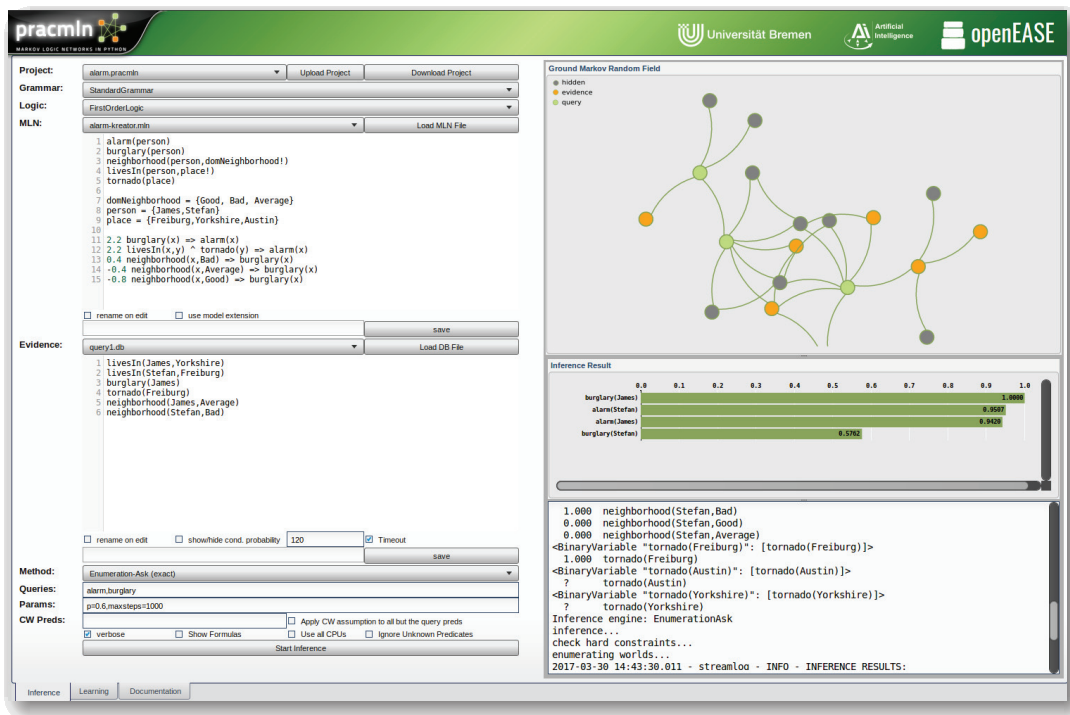


Figure 6.2: Screenshot of the PRACMLN Web Application: The left pane contains the MLN model specification and evidence, the right pane shows a graph visualization of the ground MRF as well as the inference results and a console log.

either pick one of the provided exemplary instructions from the drop-down field or type in an instruction manually. By default, the whole inference process will run through to the end but it may also be investigated in a step-by-step fashion, allowing the user to inspect the reasoning problems generated in more detail, view the used inference settings for a particular step and inspect the trained MLNs and generated evidence databases. After an inference run has been completed, the generated robot plan is displayed in a popup window, which can then be executed live in a Gazebo-based simulation, if plan schemata for the respective plan steps are available in the PRAC plan library.

WEBMLN The MLN learning and reasoning engine PRACMLN has also been made available in a web application called WEBMLN². A user can upload and download model files or investigate a selection of exemplary MLNs and test different learning and reasoning algorithms online. The PRACMLN library provides comprehensive MLN functionality for the Python programming language and is actively used by the community and used for teaching in AI courses at the Institute for Artificial Intelligence. A Screenshot of the application is shown in Figure 6.2.

²<http://pracmln.open-ease.org>

6.2 Comparison to Related Systems

In this section, I evaluate PRAC with respect to existing AI systems that have garnered a lot of attention in the recent years.

IBM Watson The Watson system by IBM (Ferrucci et al., 2010b) is targeted at answering questions stated in natural language using knowledge acquired from a huge amount of unstructured data from the Internet. In 2011, it has received a lot of publicity also in the mainstream media since it, as the first AI system ever, has beaten human champions in the quiz 'Jeopardy!'. To realize Watson, a new implementational paradigm for the design and programming of AI systems has been introduced, named *DeepQA*. In a nutshell, the DeepQA principle abstains from the constructive generation of an answer to a query, which is in many cases infeasible, but uses a large collection of very diverse methods to compute candidate solutions (hypotheses), which are rated and weighted by a confidence estimation component. This is in contrast to traditional methods in machine learning and AI, where largely monolithic, self-contained and closed methods are used. More precisely, Ferrucci et al. (2010b) chart a list of overarching principles in the DeepQA architecture: At first, massive parallelism is exploited to enable parallel generation of multiple interpretations and hypotheses. The use of many expert systems facilitates the integration, application, and contextual evaluation of a wide range of loosely coupled probabilistic question answering and content analytics methods. Thereby no component commits to an answer; all components produce features and associated confidences, scoring different question and content interpretations. An underlying confidence-processing substrate learns how to stack and combine the scores. DeepQA integrates shallow and deep knowledge as it advocates a balance in the use of strict semantics and shallow semantics, leveraging many loosely formed ontologies. In contrast, PRAC uses exclusively strong semantics with defined semantic models. It remains unclear, however, to what degree Watson makes use of strongly semantic, carefully engineered models like PRAC.

Virtual Personal Assistants Virtual personal assistants (VPAs) have become popular applications on modern mobile phones. Having originally started in the CALO (Cognitive Agent That Learns and Organizes) project (Stanford University, 2006), the *Siri* agent (Tom Gruber, 2015) was the first mainstream VPA on the consumer market shipped with Apple's iPhone devices. Since then, the number of VPAs has grown drastically and nearly all of the world's biggest IT companies have one in their portfolio,

such as Microsoft Cortana, Google Now, or Amazon Echo. The main target application domain of VPAs is to give access to core functionality of mobile phones by use of natural language. For instance, speech interfaces to the contacts, SMS, email and calendar apps, but also to online web services such as ticket booking and traveling agencies, food delivery services and navigation systems are popular applications. The systems have also widespread access to sensory information provided by the device, such as the GPS location and motion sensor, such that the processing of queries can take into account the current context of the users, like the city they are currently in, or whether they are at work or at home. Machine learning techniques are being applied to adapt to user preferences. Since all of the abovementioned systems are proprietary, only little is known about the internals. I therefore have to restrict myself to a comparison on feature level. The main targets of VPAs can be split into three bigger functional areas, i.e. conversational interfaces, personal context awareness, and service delegation. The most obvious difference to the PRAC system is that VPAs typically come with voice recognition and generation software, such that spoken language can be used to operate the systems and feedback to the user is also given in natural language. It is not known whether current VPAs effectively use deep knowledge representation and acquisition methods, or if reasoning is being applied. There is also no evidence that more extensive, connected text sources can be processed. Therefore, as recent consumer tests indicate, the quality of semantic analyses of today's VPAs is not powerful enough to significantly go beyond speech interfaces to search engines (Michael Rupp, 2015).

Mise en Place is another system that aims at determining which actions are to perform on which objects and in what order, given instruction sheets or recipes in NL (Kiddon et al., 2015). Unlike PRAC, it is an unsupervised approach to learn to interpret instructional recipes using text only. As an underlying semantic representation, the approach uses *action graphs*, a network structure similar to action cores. In action graphs, the nodes represent words or phrases in a recipe and edges represent the flow of arguments assigning a POS tag, a semantic role, as well as the start and end points of the phrase in the text. Kiddon et al. consider three types of semantic roles, namely meronymy relations, locations, and food. They represent a joint probability distribution over the connections in action graphs C and the textual recipe R , $P(R, C)$, by a factorization $P(R|C)P(C)$, where $P(R|C)$ can thus be interpreted as an 'emission' model of a human who authors a recipe. $P(C)$ is an a-priori distribution over the connections. The distribution is learned in an unsupervised Expectation-Maximization (EM) fashion: Starting from a uniform distribution, a local search method finds an instantiation of an action graph with the highest likelihood with

respect to all recipes in the training set. Given this action graph, the probability distributions are adjusted accordingly. These two steps are executed alternately, until the probabilities have converged. In contrast to PRAC, their approach does not use probabilistic relational models but the representational problems are coerced to propositional representations with strong independence assumptions. PRAC does only maintain a joint distribution over the semantic representations of action cores. Distributions over the syntactic elements of recipes are taken as given observations. This allows PRAC to fit the semantic representations more flexibly. In addition, Kiddon et al. do not use background knowledge or upper ontologies, but only plain textual representations. This makes it hard to handle subtle semantic differences in word ambiguities, which can be resolved by PRAC. In general, PRAC can be considered as having available stronger semantic background knowledge and more explicit models. It is unclear if their approach does account for the incompleteness and executability of instructions.

Tell Me Dave (Misra et al., 2014a) is a system for learning the meaning of high-level verbs to execute recipes in natural language. The underlying semantic model consists of the verbs and the syntactic prepositional relations appearing in the textual representation of an instruction. The syntax is obtained from the Stanford parser. A model based on conditional random fields represents sequential dependencies of actions. Tell Me Dave provides an online simulator, in which humans can perform sequences of activities in a virtual environment, which are used to train the CRF. Conceptually, Tell Me Dave and PRAC differ in the ontological components they model: While in PRAC, the representation, learning, and reasoning is concerned with a *conceptual* account of actions and their parameterization, Tell Me Dave considers specific *instances* of a world and the objects therein. However, it does not use deep semantic models or upper ontologies to account for the ambiguity of language. Furthermore, in PRAC the goal is not to reason about sequences of actions, but about how actions are to be parameterized and executed.

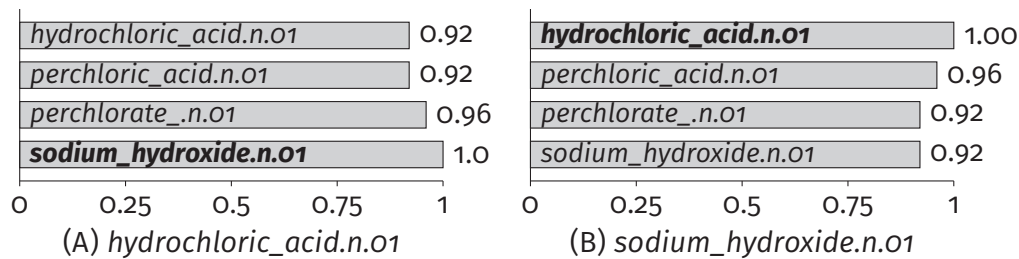


Figure 6.3: Completion of a Neutralizing instruction with different substances.

6.3 Robot Demonstrators

The methods and systems that have been developed in this thesis have been applied in the final demonstrators of the four-year European research projects ACAT³ and ROBOHOW⁴. In those demonstrators, the PRAC system has been applied to transform NL instructions into robot plans, which have then been executed in parts on the real robot platforms ‘Boxy’ and a PR2 robot, which are shown in Figure 1.3, and in Gazebo-based simulated environments as described in Section 3.9. I will first evaluate and showcase the performance of PRAC in disambiguating and completing challenging natural-language instructions. Second, I give estimates of how much knowledge PRAC comprises at the time I was writing this thesis.

In this section, I demonstrate the capabilities and performance of PRAC by applying it to a selection of challenging NL instructions. All examples are operational and can be tested in the online web-applications.

6.3.1 Chemical Laboratory

In the chemical laboratory scenario, a robotic assistant is to help a human researcher conducting experiments like making a DNA extraction. As part of the procedure, the robot has to handle different substances and objects. The robot has to decide, depending on the command it was given by the human, which actions to take and which objects to manipulate.

“Neutralize the hydrochloric acid.” Let us consider the instruction “neutralize the hydrochloric acid.” Given this task, the robot has to recognize that a base is missing

³<http://www.acat-project.eu>

⁴<http://www.robohow.eu>

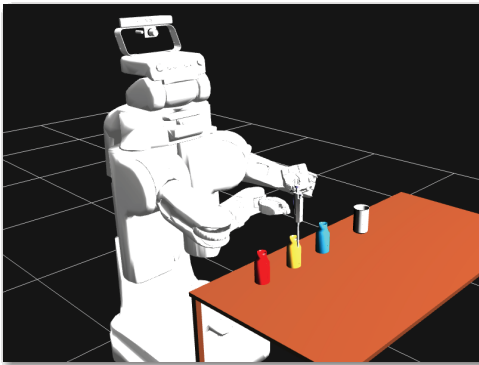
to neutralize the hydrochloric acid. To determine the correct base, it issues a query to the PRAC howto library, which provides a ranking of the substances hydrochloric acid, perchloric acid, perchlorate, and sodium hydroxide. Figure 6.3 shows the list of substances that are ranked by the matching scheme of the analogical reasoning engine, which qualify most for serving as a *Neutralizer* substance for the respective *Neutralizee*. According to that, the sodium hydroxide is the most appropriate neutralizer for the hydrochloric acid and vice versa, in each case with a matching score of 1, which means that the system has found the answer to exactly that questions in the PRAC howto library.

“Start with neutralizing 4 drops of pyridine.” is a complex instruction from the chemical application domain. It is particularly challenging because there are two verbs in the instruction (‘start’ and ‘neutralizing’), where the actual action verb is not even given in its infinite form but in the gerund. In this example, the *Neutralizee* action role of the *Neutralizing* action core is given but the *Neutralizer* substance, (hydrofluoric acid), must be inferred. The inference of the most specific instruction involves two refinement steps, first a mapping from *Neutralizing* to *Adding*, and second from *Adding* to *Pipetting*. The final plan call inferred by PRAC is

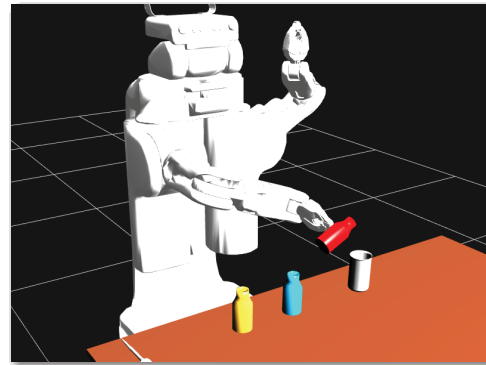
```
(an action (type use-pipette)
  (source (an object (type container.n.01)
    (contains (a substance (type hydrofluoric_acid.n.01))))))
  (count (unit drop.n.02)
    (number four.n.01))
  (destination (an object (type container.n.01)
    (contains (a substance (type pyridine.n.01))))))
```

The robot executing the plan is shown in Figure 6.4a.

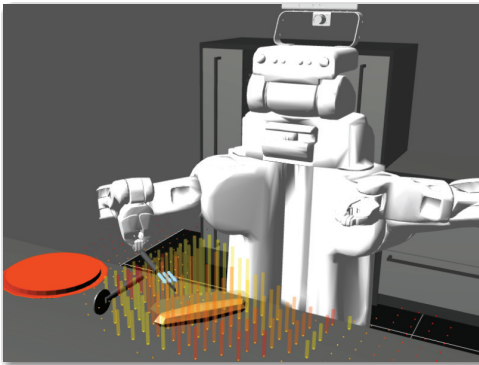
“Neutralize the methacrylic acid with 100 milliliters of cyanuramide.” Although all action roles are specified in this instruction, this is a vivid example of how the effective manipulation actions can differ fundamentally, though two original NL instructions are very similar. As in the previous example, the inference of the most specific executable plan involves two action core refinement steps. In the first step, the *Neutralizing* action core is refined to the *Adding* action core. However, since the quantity of the substance to be added is larger than before (cmp. 4 drops versus 100 ml) the second refinement step yields a *Pouring* action instead of a *Pipetting* action:



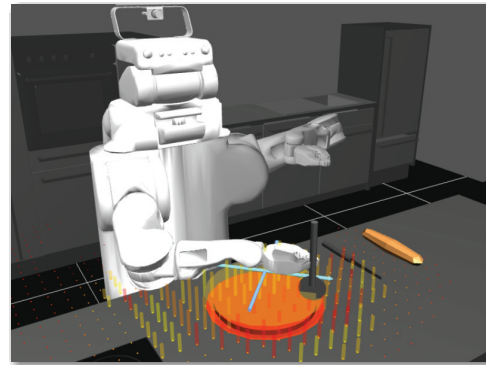
(a) “Start with neutralizing 4 drops of pyridine.”



(b) “Neutralize the methacrylic acid with 100 milliliters of cyanuramide.”



(c) “Cut the bread into 4 pieces.”



(d) “Slice the pizza.”

Figure 6.4: Simulated execution of plans generated by PRAC from different natural-language instructions

```
(an action (type pour-from-container)
  (source (an object (type container.n.01)
    (contains (a substance (type stuff))))))
(count (unit milliliter.n.01)
  (number hundred.n.01))
(destination (an object (type container.n.01)
  (contains (a substance (type stuff))))))
```

The robot executing the pouring plan in simulation is shown in Figure 6.4b.

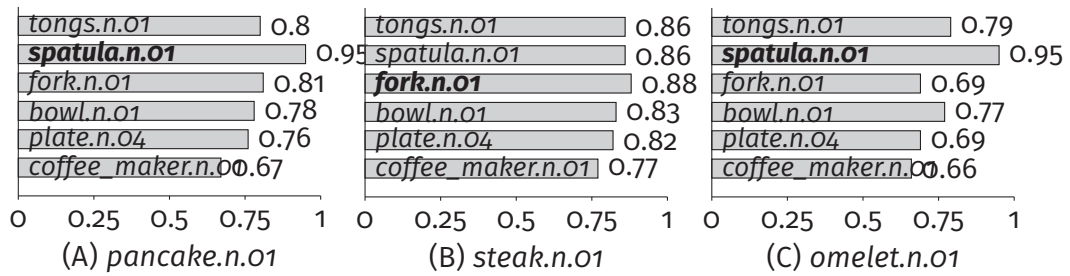


Figure 6.5: Completion of a Flipping instruction with different objects and utensils.

6.3.2 Assistive Household

In the assistive household scenario, service robots are investigated, which are able to perform complex activities in human environments like the kitchen. Therefore, the scenarios of making pancakes as a part of a breakfast preparation, and preparing an Italian dinner are considered. The latter involves more complex manipulation activities such as setting a table, baking a pizza, cutting bread and serving wine.

As a test case, we consider the action of flipping different kinds of food where the task is to infer an appropriate utensil to be used by using the analogical reasoning approach described in Section 3.8. Figure 6.5 shows the responses from the PRAC howto library to queries for (A) a pancake, (B) a steak and (C) and omelet. As the concepts *pancake.n.01* and *omelet.n.01* are semantically very similar in the taxonomy, it is expected that the responses for the two are equal. In this case, the concept *spatula.n.01* is returned as the closest match. For a steak, however, the flipping utensil with the highest score is the concept *fork.n.01*.

“Cut the bread into 4 pieces.” is an exemplary query with limited complexity. The task in this example is to correctly identify the action *Cutting* and the action roles *ob_to_be_cut*, *unit*, and *amount*. The utensil, which is not mentioned has to be inferred from the PRAC howto library using the analogical reasoning module. Given this instruction, PRAC generates and instantiates the plan call

```
(an action (type cut-object)
  (an object (type bread.n.01))
  (count (unit piece.n.01)
    (number four.n.01))
  (utensil (an object (type knife.n.01))))
```

which can be executed in the CRAM simulator. A snapshot of the robot cutting the bread using a knife is shown in Figure 6.4c.

“Slice the pizza.” is a slightly more challenging instruction because there are three action roles missing that need to be inferred, namely the unit and amount determining the number of cuts to be made, and an appropriate utensil. In addition, the action verb differs from the original verb ‘cut’. The plan returned by PRAC is

```
(an action (type cut-object)
  (an object (type pizza.n.01))
  (count (unit piece.n.01)
    (number eight.n.01))
  (utensil (an object (type cutter.n.06))))
```

A screenshot of the robot performing the cutting action in simulation is shown in Figure 6.4d. Note that it has chosen the cutter wheel as a utensil instead of the knife and a different number of pieces compared to the previous instruction.

“Put a cooking pot on the stove. Fill with water. Turn it on.” is a challenging example for coreference resolution and instruction refinement. It consists of three individual instructions, where the filling action in the second one is lacking information about the container that is to be filled with water. As there is not even a pronoun referencing the container, this is a form of *latent* anaphora that I have described in Section 3.7. The third step contains an explicit anaphora denoted by the word ‘it’, which refers to the stove that must be turned on. It is particularly challenging as the referent word ‘stove’ is mentioned even two sentences before. In addition, reasonable action refinements need to be found in order to fill the pot and to switch on the stove. To this end, the PRAC howto library has been provided with a simplistic manual for operating a centrifuge consisting of only one step,

How to Start the Centrifuge

1. Press the start button on the centrifuge,

which has been imported using PRAC-TELL. Using this instruction sheet, PRAC is able to transfer the knowledge of starting a centrifuge to turning on a stove. The inferred plan steps are

```
(an action (type put-object)
  (an object (type pot.n.01))
  (target (a location (on stove.n.02))))
(an action (type operate-tap)
  (source (an object (type faucet.n.01))
```

```

                                (contains (a substance (type water.n.06))))))
(count (unit nil)
      (number nil))
(destination (a location (in pot.n.01))))
(an action (type press-object)
          (an object (type push_button.n.01))
          (target (a location (on stove.n.02))))

```

As can be seen, PRAC correctly completes the missing roles in the second and third instruction. The action of filling the pot is refined to operating the tap, and the action to turn on the stove is refined to pressing a button on the stove. Note that the amount specifications in the *operating-tap* plan call are unspecified (*nil*). This makes sense for a filling action as the quantity of the liquid is typically determined by the size of the container.

“**Make an Italian dinner.**” is the perhaps most challenging example that the PRAC system is able to handle at the time of writing this thesis. The challenge is to use five different instruction sheets and recipes, which have been imported to the PRAC howto library, namely the following ones.

How to make an Italian dinner

1. Set the table.
2. Prepare a pizza.
3. Cut the pizza and serve.
4. Slice the bread and serve it.
5. Serve some wine.

How to make a pizza

1. Spread tomato sauce over the dough.
2. Add the salami.
3. Sprinkle with cheese.
4. Season the pizza with oregano.

How to serve a drink

1. Take a glass from the cupboard.
2. Fill it with the drink.
3. Put the glass on the table.

How to serve food

1. Set the table.
2. Prepare a pizza.
3. Cut the pizza and serve.
4. Serve some wine.

How set the table

1. Take a glass from the cupboard.
2. Put it on the table.
3. Take a plate from the cupboard.
4. Put it on the table.
5. Take the cutlery from the drawer.
6. Put it on the table

Obviously, these howtos are not sufficiently complete and detailed enough to actually prepare a real Italian dinner when being performed in a real environment. However, interpreting the respective instructions to prepare a dinner and make use of the knowledge contained in the howtos is extremely challenging for an NL interpreter,

since they exhibit all kinds of ambiguity, incompleteness and vagueness I have addressed in this thesis. The PRAC interpreter computes a robot plan comprising 21 action steps in total. I consider this task therefore the ultimate showcase and demonstration of the reasoning capabilities I have developed and implemented in PRAC. Not all of the substeps are currently executable on a robot because implementations of the respective plan schemata are not available yet. Implementing parameterizable plans, however, goes beyond the scope of this work and is on the future research agenda.

The web interface to PRAC offers a few more predefined examples. I have presented here only the most interesting and the most challenging ones. Walking through all of them is not possible in this work, but I invite the interested readers to convince themselves by trying out the online tools presented in Section 6.1.

6.3.3 Statistics

It is very complex, if not impossible to quantitatively compare a system like PRAC to existing approaches, as neither benchmarks nor established methods exist. In this section, I present estimates of the representational and computational dimensions of PRAC to complement the qualitative evaluation in the previous section.

Probabilistic Knowledge Bases First, I give a rough idea of how large the probabilistic knowledge bases are and how many of them are contained in PRAC. At the time I am writing this thesis, PRAC contains implementations and probabilistic models for the recognition, analysis and completion of 25 action cores. The largest number of action roles for one action core is five, the smallest is two. As there are six reasoning modules that rely on reasoning in probabilistic first-order models, i.e. ACTIONCOREINFERENCE, PROPERTYEXTRACTION, ACTIONROLEINFERENCE, COREFERENCERESOLUTION, ACTIONCOREREFINEMENT, and ACTIONROLEREFINEMENT, PRAC requires roughly $25 \times 6 = 150$ probabilistic KBs. The number of training instances used to learn the models vary. This is due to the fact that the respective action cores reside on different levels of abstraction: In the applications considered in this work, the *Adding* action core comes with many different parameterizations as *Adding* is a very generic action that needs to be performed in different ways, depending on which objects it is to be executed on. In order to learn models that are able to discern multiple cases, more training data is needed, which in turn implies larger models. By contrast, the *Cooking* action core is a fairly domain-specific action, such that only one example suffices to achieve reasonable accuracy. Likewise, the number of action roles prevalingly influences the

Action Core	# Action Roles	Training Examples	# MLN formulas	# Howto Lib
Adding	5	30	153	3477
Arranging	2	1	4	62
Cooking	2	1	4	1044
Cutting	5	2	125	299
Evaluating	3	1	9	490
Filling	5	11	63	142
Flavoring	3	9	176	218
Flipping	3	5	38	39
Lifting	2	1	4	25
Neutralizing	5	15	400	7
Opening	2	3	7	74
Pipetting	5	2	266	0
Pouring	5	10	95	1604
Preheating	5	3	120	224
Pressing	3	1	16	157
Putting	3	5	24	2657
Serving	2	2	6	619
Shaking	4	1	16	349
Spreading	3	1	9	289
Sprinkling	3	1	9	54
Starting	2	2	8	77
Storing	3	20	81	176
Taking	3	3	36	311
Turning	2	1	4	47
Waiting	2	1	9	20

Table 6.1: Overview of the sizes of individual MLN knowledge bases (i.e. of ACTIONROLEINFERENCE reasoning module) per action core and the number of instances in the PRAC howto library.

number of formulas in the MLN. The action roles are attached to an action core, the more pairwise combinations of concepts need to be captured by logical formulae in the MLN.

Table 6.1 shows a juxtaposition of the number of action roles, the number of training examples, the resulting number of formulas in the learned MLNs and the number of instances of the respective action cores in the howto library. The table only refers to the models learned for the ACTIONROLEINFERENCE as they are most representative.

Instruction	Reasoning Module	∅ Runtime (in sec)	# Calls	Total Runtime (in sec)
"Slice the bread into 4 pieces."	NL Parsing	2.01	1	2.01
	Action Core Inference	1.93	1	1.93
	Property Extraction	0.53	1	0.53
	Action Role Inference	4.61	1	4.61
	Coreference Resolution	0.11	1	0.24
	Prob. Refinement	2.31	1	2.31
	Analog. Role Refinement	0.38	1	0.38
	Plan Generation	0.10	1	0.10
"Cut the pizza."	NL Parsing	1.93	1	1.93
	Action Core Inference	4.78	1	4.78
	Property Extraction	1.47	1	1.47
	Action Role Inference	13.82	1	13.82
	Coreference Resolution	0.50	1	0.5
	Plan Generation	0.54	1	0.54
"Start with neutralizing 4 drops of pyridine."	NL Parsing	2.06	1	2.06
	Action Core Inference	5.22	1	5.22
	Property Extraction	7.65	1	15.30
	Action Role Inference	45.12	1	45.12
	Coreference Resolution	0.59	1	0.59
	Prob. Refinement	6.00	2	12.01
	Prob. Role Refinement	0.83	2	0.83
	Plan Generation	0.50	1	0.50
"Neutralize the methacrylic acid with 100 milliliters of cyanuramide."	NL Parsing	2.01	1	2.01
	Action Core Inference	2.87	1	2.87
	Property Extraction	0.97	1	0.97
	Action Role Inference	23.37	1	23.37
	Coreference Resolution	0.24	1	0.24
	Prob. Refinement	2.84	2	5.68
	Prob. Role Refinement	3.33	2	6.66
	Plan Generation	0.20	1	0.20
"Make an Italian dinner."	NL Parsing	2.03	1	2.03
	Action Core Inference	4.76	1	4.76
	Property Extraction	1.95	1	1.95
	Action Role Inference	13.34	1	13.34
	Coreference Resolution	0.61	1	0.61
	Prob. Refinement	2.82	4	11.29
	Prob. Role Refinement	1.82	4	4.29
	Analog. Plan Expansion	0.019	6	0.11
	Plan Generation	0.06	21	0.24

Table 6.2: Runtime analysis of five natural-language instructions interpreted with PRAC.

Runtime Statistics With respect to the runtime of the inferences required by PRAC, five representative exemplary instructions of varying complexity have been analyzed. To get a more precise picture of the complexity of the reasoning processes in PRAC, every reasoning module in the pipeline has been measured individually, as the module can be executed multiple times due to possible action refinement processes.

Table 6.2 shows the five examples, where the processing times are broken down to the individual respective reasoning modules with regard to the number of calls, the average processing time per call, the total time consumed by the modules in the respective instructions, and the total processing time for the instructions as a whole. The time required for the NL parsing is almost constant by roughly 2 seconds per instruction, as syntactically, the instructions have similar length.

As expected, the modules performing probabilistic reasoning take most of the processing time, of which the ACTIONCOREINFERENCE, and the ACTIONROLEINFERENCE modules are the most time consuming ones. As the reasoning takes into account all possible word senses, the time highly depends on the number of WordNet concepts associated with a word. The word ‘slice’, for example, has only four possible senses, whereas the word ‘make’ has 49 possible meanings (cmp. Appendix B), which is reflected in the runtimes for the action core inferences for the two examples.

Table 6.2 also exposes that the action refinement by means of the analogical reasoning strategy is significantly faster than the probabilistic reasoning. In the third instruction, “Start with neutralizing 4 drops of pyridine,” the probabilistic reasoning about action and role refinement has been applied twice. The first refinement step transforms the *Neutralizing* action into an *Adding* action, the second transforms the *Adding* action into a *Pipetting* action. Conversely, in the fifth instruction, “Make an Italian dinner,” action refinement by means of analogical reasoning has been applied six times to expand cross-referencing howtos to finer grained steps. Although the reasoning pipeline in the latter instruction performs 10 refinement steps in total, the overall processing time is twice as fast. The experiments show that the analogical reasoning with database indexing can be ~150 times faster than the probabilistic refinement.

Chapter seven

Conclusions

This thesis has proposed PRAC, a probabilistic framework for interpreting natural-language instructions for everyday activity tasks, which near-future service robots will have to accomplish. Such instruction sheets constitute a valuable source of knowledge about how to perform complex high-level activities. They are available in abundance on the World Wide Web and thus are a promising alternative to classical approaches to action planning. However, the task of interpreting natural language is extremely challenging as humans describing activities to other humans tend to omit lots of information they assume their recipients to know or be able to infer. Consequently, such instruction sheets are severely vague, unspecific and even incomplete. I have argued that robots that are to understand and perform such activities must be equipped with a substantial body of action-specific knowledge in order to resolve the problems of ambiguity and underspecification, which are ubiquitous when dealing with natural language. I have reported on a study on a large corpus of natural-language recipes mined from the [wikihow.com](http://www.wikihow.com) web page, which supports this statement.

This work has presented PRAC (Probabilistic Action Cores), an interpreter for natural-language instructions which is able to resolve vagueness and ambiguity in natural language and infer missing information pieces that are required to transform an instruction into an executable robot plan. I have shown that the interpretation of ambiguous and incomplete natural-language instructions can be tackled by formulating it as the problem of computing the most probable complete and unambiguous instruction in action specific knowledge bases. Within the PRAC framework, the most probable complete and unique instruction enables robots to find the most appro-

priate plan with the most general refinement of the formal plan parameters given the instruction. As an inferential framework, PRAC makes use of two complementary paradigms, namely reasoning in probabilistic first-order knowledge bases and reasoning by analogy.

To perform inferences about action core identification, word-sense disambiguation and action role assignments, the PRAC framework learns joint probability distributions over the possible ways in which instructions for a given action verb are typically formulated. The probabilistic knowledge bases are encoded in Markov logic networks (MLNs), a formalism for uncertain knowledge representation that is particularly attractive due to its conceptual simplicity and expressiveness. However, the practical applicability of MLNs often is limited as the models grow exponentially in the number instances of subject to reasoning. To tackle this problem, I have proposed FUZZY-MLNs, an extension of MLNs that allows to represent probability distributions over open domains compactly — if complete ontologies are available for these domains. The basic idea underlying FUZZY-MLNs is to explicitly represent only the small subset of formulae that is contained in the training databases. After having learned the probability distribution, FUZZY-MLNs can reason about concepts that are not contained in the graphical model but in the taxonomy. They do so by exploiting the fact that the relational structure of concepts in the taxonomy is correlated with the relational structures of the explicitly represented concepts weighted by a notion of semantic similarity. FUZZY-MLNs implement this bias by generalizing the taxonomic assertions for out-of-domain concepts from Boolean truth to real-valued degrees of truth. The degree of truth is then computed based on the semantic similarity of the off-domain concept to those concepts contained in the graphical model.

For accomplishing the task of instruction completion and refinement, a novel instance-based learning approach called PRAC-TELL has been proposed in this work, which uses the principle of analogical reasoning in a knowledge base of several thousands of semantically indexed instruction sheets. The underlying assumption is that reasonable completions of unspecific instructions can be found in a large collection of related instruction sheets, the knowledge about which can be transferred and adapted to the needs of a new situation. To this end, I have presented methods to complete action parameters, expand instructions to finer-grained action sequences and adapt existing procedures for manipulating objects to similar objects by using fast database queries. The use of instance-based learning strategies is advantageous in several regards. It avoids the restriction of purely probabilistic models, which may grow too large in size and computational expense with increasing numbers of variables and thus can quickly become intractable. Furthermore, knowledge about how to perform tasks can be simply accommodated or retracted by adding or removing instruction sheets

to or from the collection without the need for expensive relearning.

A pipeline of individual reasoning methods for transforming incomplete and ambiguous natural-language instructions to fully specified, executable plans are implemented in a single algorithm called PRAC-QUERY, which has been introduced in this work. The algorithm performs recursive expansion of instructions into sequences of more specific actions which are analyzed by different reasoning modules of the PRAC system. Every module is specialized to a particular reasoning task, such that each instruction is getting incrementally enriched by semantic knowledge until it can be executed. PRAC-QUERY in turn spans a tree of intermediate reasoning results which is built up in a breadth-first fashion. The PRAC framework provides an attractive alternative to other instruction interpretation approaches, in particular for the interpretation of complex manipulation tasks. One important advantage is that PRACs are not limited to inferring which sequences of actions should be executed but also *how* the individual actions are to be executed. A second advantage is that the use of taxonomic reasoning in the PRAC inference results in the inference of the most general concept refinements of the plan parameters. This generates least commitment calls of plans that keep maximal flexibility at execution time and avoids the necessity of grounding symbolic names that are generated in the interpretation process (symbol grounding problem). The current implementation comprises a set of 25 PRACs and plan schemata from two application domains, the household/cooking domain and the domain of conducting chemical experiments, which are being continuously extended.

This thesis has further presented a novel knowledge-driven approach for interpreting object descriptions in natural language, which is integrated in the PRAC reasoning pipeline for the identification of perceptual characteristics of objects in instructions, which can also be used in the ROBOSHERLOCK perception framework for object detection. To this end, visual attributes of object hypotheses detected by means of individual specialized perception routines are enhanced to provide their outputs as concepts in the WordNet taxonomy. Grounding the visual features of objects into a taxonomy gives abstract symbolic meaning to those features so they share the same semantics across all components and can be equally referred to by the natural-language and perception components. The use of these knowledge-based features enables to deal with previously unseen concepts in a meaningful way by referring to them in terms of their similarity to known concepts, which makes the proposed method highly efficient for learning from sparse data and enables a robot to recognize objects from purely textual descriptions without the need to retrain statistical models when new objects arise. Our experiments show that using PRAC, semantic object models for robot perception can be learned from natural-language descriptions mined from the web and used for perceiving objects of daily use on a

real robot.

Furthermore, a novel strategy for combining multiple diverse perception algorithms in one coherent perception system using probabilistic relational models has been introduced. As most perception routines for detecting objects of daily use are strongly tailored to very specific perceptual cues of objects, none of them shows promise in robustly detecting all kinds of objects. Markov logic networks are used to learn an ensemble of sets of diverse and highly specialized algorithms that are strategically combined in order to draw a well-informed final conclusion. The experiments show that the MLNs indeed learn probability distributions over the strengths and weaknesses of the individual algorithms and successfully combine them to come to globally consistent and probabilistically sound posterior belief. Not only does the proposed approach leverage synergies among available perception algorithms for object detection, but it can also be queried for the most descriptive features of an object. Using the resulting information will enable the systems to choose the feature detectors that are most appropriate, and use these for detection and tracking. I have argued the use of ensemble-based systems and specialized perception routines is a key paradigm for pushing the perceptual capabilities of our today's robots to more versatile and advanced applications.

The PRAC system has been implemented and released as an open-source software framework which is publicly accessible as a browser-based web and smart-phone application, which allows users to inspect and investigate the reasoning processes in PRAC and to directly execute the generated sequences of robot plans in a Gazebo-based simulator, which is connected to PRAC via the CRAM plan executive. The evaluation shows that the proposed methods indeed scale to realistic domain sizes and show promise in pushing future robots towards accomplishing more and more versatile tasks and competent decision making.

In addition, PRACMLN has been released as open source as the standalone learning and reasoning engine for Markov logic networks.

I believe that equipping robots with action-specific knowledge is a key paradigm for implementing more flexible, cognitive robot behavior and pushing autonomous robots to perform more advanced everyday activities. Although several approaches towards understanding natural-language directives for robots exist, to the best of my knowledge the PRAC system is the first that implements the whole pipeline starting from vaguely stated natural-language instructions, disambiguating and completing them to formally specified plan calls and executing these plans on a real robot execution system.

Prior Publications

This thesis is in parts based on prior work that has been published in different international conferences and books. The parts of this work drawing on content from prior publications referenced the prior works where appropriate. For the sake of completeness, this section charts a complete list of my prior publications.

Book Chapters

Daniel Nyga, Michael Beetz, “Cloud-based Probabilistic Knowledge Services for Instruction Interpretation”, *In Springer Proceedings in Advanced Robotics, Vol. 2, ROBOTICS RESEARCH, Editors: Wolfram Burgard and Antonio Bicchi, 2017.*

Michael Beetz, Hagen Langer, Daniel Nyga, “Planning Everyday Manipulation Tasks–Prediction-based Transformation of Structured Activity Descriptions”, *Chapter in Exploring Cybernetics, Springer, pp. 63-83, 2015.*

Michael Beetz, Ferenc Bálint-Benczédi, Nico Blodow, Christian Kerl, Zoltán-Csaba Márton, Daniel Nyga, Florian Seidel, Thiemo Wiedemeyer, Jan-Hendrik Worch, “RoboSherlock: Unstructured Information Processing Framework for Robotic Perception”, *In Handling Uncertainty and Networked Structure in Robot Control, Springer International Publishing, Cham, pp. 181-208, 2015.*

Conference Papers

Daniel Nyga, Mareike Picklum, Sebastian Koralewski, Michael Beetz, “Instruction Completion through Instance-based Learning and Semantic Analogical Reasoning”, *In International Conference on Robotics and Automation (ICRA), Singapore, 2017. Accepted for publication*

Daniel Nyga, Mareike Picklum, Michael Beetz, “What No Robot Has Seen Before – Probabilistic Interpretation of Natural-language Object Descriptions”, *In International Conference on Robotics and Automation (ICRA), Singapore, 2017. Accepted for publication*

Mihai Pomarlan, Daniel Nyga, Mareike Picklum, Sebastian Koralewski, and Michael Beetz. “Deeper Understanding of Vague Instructions through Simulated Execution” (extended abstract). *In Proceedings of the 2017 International Conference on Autonomous Agents & Multiagent Systems, AAMAS '17. International Foundation for Autonomous Agents and Multiagent Systems, 2017. Accepted for publication*

Daniel Nyga, Michael Beetz, “Cloud-based Probabilistic Knowledge Services for Instruction Interpretation”, *In International Symposium of Robotics Research (ISRR), Sestri Levante (Genoa), Italy, 2015.*

Daniel Nyga, Michael Beetz, “Reasoning about Unmodelled Concepts – Incorporating Class Taxonomies in Probabilistic Relational Models”, *In Arxiv.org, 2015. Preprint*

Gheorghe Lisca, Daniel Nyga, Ferenc Bálint-Benczédi, Hagen Langer, Michael Beetz, “Towards Robots Conducting Chemical Experiments”, *In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 2015.*

Michael Beetz, Ferenc Balint-Benczedi, Nico Blodow, Daniel Nyga, Thimo Wiedemeyer, Zoltan-Csaba Marton, “RoboSherlock: Unstructured Information Processing for Robot Perception”, *In IEEE International Conference on Robotics and Automation (ICRA), Seattle, Washington, USA, 2015. Best Service Robotics Paper Award*

Daniel Nyga, Ferenc Balint-Benczedi, Michael Beetz, “PR2 Looking at Things: Ensemble Learning for Unstructured Information Processing with Markov Logic Networks”, *In IEEE International Conference on Robotics and Automation (ICRA),*

Hong Kong, China, 2014.

Nicholas Hubert Kirk, Daniel Nyga, Michael Beetz, “Controlled Natural Languages for Language Generation in Artificial Cognition”, In *IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, China, 2014.

Daniel Nyga, Michael Beetz, “Everything Robots Always Wanted to Know about Housework (But were afraid to ask)”, In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vilamoura, Portugal, 2012.

Daniel Nyga, Moritz Tenorth, Michael Beetz, “How-Models of Human Reaching Movements in the Context of Everyday Manipulation Activities”, In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, 2011.

Moritz Tenorth, Daniel Nyga, Michael Beetz, “Understanding and Executing Instructions for Everyday Manipulation Tasks from the World Wide Web”, In *IEEE International Conference on Robotics and Automation (ICRA)*, Anchorage, AK, USA, pp. 1486-1491, 2010.

Other Publications

Moritz Tenorth, Daniel Nyga, Michael Beetz, “Understanding and Executing Instructions for Everyday Manipulation Tasks from the World Wide Web”, *Technical report, IAS group, Technische Universität München, Fakultät für Informatik*, 2009.

WordNet Concepts

In this appendix, the full list of possible WordNet synsets are charted for a selection of frequently used ambiguous words.

Lemma:	cup
Part of Speech:	n
Synset	Definition
cup.n.01	a small open container usually used for drinking; usually has a handle (he put the cup back in the saucer;the handle of the cup was missing)
cup.n.02	the quantity a cup will hold (he drank a cup of coffee;he borrowed a cup of sugar)
cup.n.03	any cup-shaped concavity (bees filled the waxen cups with honey;he wore a jock strap with a metal cup;the cup of her bra)
cup.n.04	a United States liquid unit equal to 8 fluid ounces ()
cup.n.05	cup-shaped plant organ ()
cup.n.06	a punch served in a pitcher instead of a punch bowl ()
cup.n.07	the hole (or metal container in the hole) on a golf green (he swore as the ball rimmed the cup and rolled away;put the flag back in the cup)
cup.n.08	a large metal vessel with two handles that is awarded as a trophy to the winner of a competition (the school kept the cups is a special glass case)

Appendix B. WordNet Concepts

Lemma:	water
Part of Speech:	n
Synset	Definition
water.n.01	binary compound that occurs at room temperature as a clear colorless odorless tasteless liquid; freezes into ice below 0 degrees centigrade and boils above 100 degrees centigrade; widely used as a solvent ()
body_of_water.n.01	the part of the earth's surface covered with water (such as a river or lake or ocean) (they invaded our territorial waters;they were sitting by the water's edge)
water.n.03	once thought to be one of four elements composing the universe (Empedocles) ()
water_system.n.02	a facility that provides a source of water (the town debated the purification of the water supply;first you have to cut off the water)
urine.n.01	liquid excretory product (there was blood in his urine;the child had to make water)
water.n.06	a liquid necessary for the life of most animals and plants (he asked for a drink of water)

Lemma:	bowl
Part of Speech:	n
Synset	Definition
bowl.n.01	a round vessel that is open at the top; used chiefly for holding food or liquids; ()
bowl.n.02	a concave shape with an open top ()
bowl.n.03	a dish that is round and open at the top for serving foods ()
bowl.n.04	the quantity contained in a bowl ()
stadium.n.01	a large structure for open-air sports or entertainments ()
bowling_ball.n.01	a large ball with finger holes used in the sport of bowling ()
bowl.n.07	a wooden ball (with flattened sides so that it rolls on a curved course) used in the game of lawn bowling ()
bowl.n.08	a small round container that is open at the top for holding tobacco ()
roll.n.15	the act of rolling something (as the ball in bowling) ()

Lemma:	mil <u>k</u>
Part of Speech:	n
Synset	Definition
mil <u>k</u> .n.01	a white nutritious liquid secreted by mammals and used as food by human beings ()
mil <u>k</u> .n.02	produced by mammary glands of female mammals for feeding their young ()
mil <u>k</u> .n.03	a river that rises in the Rockies in northwestern Montana and flows eastward to become a tributary of the Missouri River ()
mil <u>k</u> .n.04	any of several nutritive milklike liquids ()

Lemma:	milliliter
Part of Speech:	n
Synset	Definition
milliliter.n.01	a metric unit of volume equal to one thousandth of a liter ()

Lemma:	add
Part of Speech:	v
Synset	Definition
add.v.01	make an addition (to); join or combine or unite with others; increase the quality, quantity, size or scope of (We added two students to that dorm room;She added a personal note to her letter;Add insult to injury;Add some extra plates to the dinner table)
add.v.02	state or say further ('It doesn't matter,' he supplied)
lend.v.01	bestow a quality on (Her presence lends a certain cachet to the company;The music added a lot to the play;She brings a special atmosphere to our meetings;This adds a light note to the program)
add.v.04	make an addition by combining numbers (Add 27 and 49, please!)
total.v.02	determine the sum of (Add all the people in this town to those of the neighboring town)
add.v.06	constitute an addition (This paper will add to her reputation)

Appendix B. WordNet Concepts

Lemma:	neutralize
Part of Speech:	v
Synset	Definition
neutralize.v.01	make politically neutral and thus inoffensive (The treaty neutralized the small republic)
neutralize.v.02	make ineffective by counterbalancing the effect of (Her optimism neutralizes his gloom;This action will negate the effect of my efforts)
counteract.v.03	oppose and mitigate the effects of by contrary actions (This will counteract the foolish actions of my colleagues)
neutralize.v.04	get rid of (someone who may be a threat) by killing (The mafia liquidated the informer;the double agent was neutralized)
neutralize.v.05	make incapable of military action ()
neutralize.v.06	make chemically neutral (She neutralized the solution)

Lemma:	slice
Part of Speech:	v
Synset	Definition
slit.v.01	make a clean cut through (slit her throat)
slice.v.02	hit a ball and put a spin on it so that it travels in a different direction
slice.v.03	cut into slices (Slice the salami, please)
slice.v.04	hit a ball so that it causes a backspin

Lemma:	make
Part of Speech:	v
Synset	Definition
make.v.01	engage in (make love, not war;make an effort;do research;do nothing;make revolution)
make.v.02	give certain properties to something (get someone mad;She made us look silly;He made a fool of himself at the meeting;Don't make this into a big deal;This invention will make you a millionaire;Make yourself clear)

make.v.03	make or cause to be or to become (make a mess in one's office;create a furor)
induce.v.02	cause to do; cause to act in a specified manner (The ads induced me to buy a VCR;My children finally got me to buy a computer;My wife made me buy a new sofa)
cause.v.01	give rise to; cause to happen or occur, not always intentionally (cause a commotion;make a stir;cause an accident)
produce.v.02	create or manufacture a man-made product (We produce more cars than we can sell;The company has been making toys for two centuries)
draw.v.04	make, formulate, or derive in the mind (I draw a line here;draw a conclusion;draw parallels;make an estimate;What do you make of his remarks?)
make.v.08	compel or make somebody or something to act in a certain way (People cannot be made to integrate just by passing a law;;Heat makes you sweat)
create.v.05	create by artistic means (create a poem;Schoenberg created twelve-tone music;Picasso created Cubism;Auden made verses)
gain.v.08	earn on some commercial or business transaction; earn as salary or wages (How much do you make a month in your new job?;She earns a lot in her new job;this merger brought in lots of money;He clears \$5,000 each month)
do.v.08	create or design, often in a certain way (Do my room in blue;I did this piece in wood to express my love for the forest)
form.v.02	to compose or represent:"This wall forms the background of the stage setting" (The branches made a roof;This makes a fine introduction)
reach.v.07	reach a goal, e.g., "make the first team" (We made it!;She may not make the grade)
make.v.14	be or be capable of being changed or made into (He makes a great host;He will make a fine father)
make.v.15	make by shaping or bringing together constituents (make a dress;make a cake;make a wall of stones)
make.v.16	perform or carry out (make a decision;make a move;make advances;make a phone call)

Appendix B. WordNet Concepts

construct.v.01	make by combining materials and parts (this little pig made his house out of straw;Some eccentric constructed an electric brassiere warmer)
make.v.18	change from one form into another (make water into wine;make lead into gold;make clay into bricks)
make.v.19	act in a certain way so as to acquire (make friends;make enemies)
name.v.03	charge with a function; charge to be (She was named Head of the Committee;She was made president of the club)
have.v.17	achieve a point or goal (Nicklaus had a 70;The Brazilian team got 4 goals;She made 29 points that day)
reach.v.01	reach a destination, either real or abstract (We hit Detroit by noon;The water reached the doorstep;We barely made it to the finish line;I have to hit the MAC machine before the weekend starts)
lay_down.v.01	institute, enact, or establish (make laws)
make.v.24	carry out or commit (make a mistake;commit a faux-pas)
make.v.25	form by assembling individuals or constituents (Make a quorum)
hold.v.03	organize or be responsible for (hold a reception;have, throw, or make a party;give a course)
make.v.27	put in order or neatened (make the bed;make up a room)
take.v.27	head into a specified direction (The escaped convict took to the hills;We made for the mountains)
stool.v.04	have a bowel movement (The dog had made in the flower beds)
make.v.30	undergo fabrication or creation (This wool makes into a nice sweater)
make.v.31	be suitable for (Wood makes good furniture)
make.v.32	add up to (four and four make eight)
make.v.33	amount to (This salary increase makes no difference to my standard of living)
make.v.34	constitute the essence of (Clothes make the man)
make.v.35	appear to begin an activity (He made to speak but said nothing in the end;She made as if to say hello to us)
make.v.36	proceed along a path (work one's way through the crowd;make one's way into the forest)

make.v.37	reach in time (We barely made the plane)
make.v.38	gather and light the materials for (make a fire)
cook.v.02	prepare for eating by applying heat (Cook me dinner, please;can you make me an omelette?;fix breakfast for the guests, please)
seduce.v.01	induce to have sex (Harry finally seduced Sally;Did you score last night?;Harry made Sally)
make.v.41	assure the success of (A good review by this critic will make your play!)
make.v.42	represent fictitiously, as in a play, or pretend to be or act like (She makes like an actress)
make.v.43	consider as being (It wasn't the problem some people made it)
make.v.44	calculate as being (I make the height about 100 feet)
make.v.45	cause to be enjoyable or pleasurable (make my day)
make.v.46	favor the development of (Practice makes the winner)
make.v.47	develop into (He will make a splendid father!)
make.v.48	behave in a certain way (make merry)
make.v.49	eliminate urine (Again, the cat had made on the expensive rug)

Acronyms

- FUZZY-MLN** fuzzy Markov logic network 14–17, 72, 77, 90, 101, 107, 125, 128, 131–137, 140, 142–144, 148–150
- PRAC** Probabilistic Action Cores ii, 7, 8, 14, 16, 17, 53, 54, 67, 68, 70–74, 80, 83–85, 87, 89
- AI** Artificial Intelligence 2, 4, 13, 19, 42, 53, 56–59, 61, 137, 145, 187, 189, 190
- AMT** Amazon Mechanical Turk™ 91, 94–96
- BN** Bayesian network 34–38, 41, 51
- DL** description logic 25–27, 51, 52, 126, 129, 145, 148
- FL** fuzzy logic 50, 51, 130, 131, 133
- FOL** first-order logic 19, 20, 23–25, 28, 29, 42, 44, 51, 131, 133, 140, 145, 158
- ground MRF** ground Markov random field 42, 45, 46, 48, 51, 132–135, 143
- HIT** human intelligence task 94–96
- KB** knowledge base 3, 5, 7, 14, 15, 19, 21–25, 27–29, 32, 34, 37, 40, 42, 43, 51, 54, 72, 88, 90, 104, 106, 107, 111, 112, 123, 126, 130, 144, 148, 149, 151, 160, 176, 177, 199
- LCS** lowest common superconcept 129
- MAP** maximum a-posteriori 5, 32
- MCMC** Markov chain Monte Carlo 40, 45, 47
- MLE** maximum likelihood estimate 33

- MLN** Markov logic network 20, 42–46, 48, 49, 51, 52, 72, 85, 87, 94, 100, 104, 125, 128, 130, 131, 133–136, 140, 142, 143, 145–147, 149, 150, 157–159, 164–166, 173, 174, 176, 182, 185, 189, 200
- MN** Markov network 37
- MPE** most probable explanation 32, 33, 46, 166
- MRF** Markov random field 37–42, 51, 132, 144, 168, 184
- NL** natural-language 5–7, 10, 11, 13–15, 17, 50, 53–57, 61, 63–65, 67, 72, 74–76, 92–95, 97–99, 104, 105, 111, 115, 119, 120, 123, 142, 145, 151, 170, 176, 177, 184, 185, 193, 194, 198, 202
- NL** natural language 3, 5, 7, 8, 11–14, 54–56, 58, 61, 63, 80, 86, 91, 93, 94, 102, 119, 122, 151, 169, 170, 176, 177, 191
- PGM** probabilistic graphical model 20, 29, 31, 34, 41, 42, 51, 54, 145
- PL** propositional logic 20–23, 30, 32, 40, 50, 51, 130, 131
- POS** part-of-speech 99, 101, 123, 177, 191
- POS** part of speech 101
- PRM** probabilistic relational model 7, 19, 20, 42, 51, 52, 91, 105, 126, 130, 137, 156, 157, 185, 192
- PRS** procedural reasoning system 84
- RAP** reactive action packages 84
- SRL** statistical relational learning 19, 145
- VPA** virtual personal assistant 190, 191
- WCSP** weighted constraint satisfaction problem 46
- WSD** word-sense disambiguation 137, 140

Bibliography

- A. Aldoma, Z.-C. Marton, F. Tombari, W. Wohlkinger, C. Potthast, B. Zeisl, R. B. Rusu, S. Gedikli, and M. Vincze. Tutorial: Point Cloud Library – Three-Dimensional Object Recognition and 6 DoF Pose Estimation. *Robotics & Automation Magazine*, 19(3):80–91, September 2012.
- B. Alexander, K. Hsiao, C. Jenkins, B. Suay, and R. Toris. Robot web tools [ros topics]. *IEEE Robotics & Automation Magazine*, 19(4):20–23, 2012.
- D. Allouche, S. de Givry, and T. Schiex. Toulbar2, an open source exact cost function network solver. Technical report, Technical report, INRIA, 2010.
- M. Alomari, E. Chinellato, Y. Gatsoulis, D. Hogg, and A. Cohn. Unsupervised grounding of textual descriptions of object features and actions in video. In *KR 2016*, pages 505–508. Association for the Advancement of Artificial Intelligence, March 2016. URL <http://eprints.whiterose.ac.uk/95572/>. © 2016, Association for the Advancement of Artificial Intelligence. This is an author produced version of a paper published in Proceedings, 15th International Conference on Principles of Knowledge Representation and Reasoning (KR 2016).
- J. Anderson. *Constraint-directed Improvisation for Everyday Activities*. PhD thesis, 1995.
- Y. Artzi and L. Zettlemoyer. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1(1):49–62, 2013.
- F. Baader and W. Nutt. Basic Description Logics. In *Description logic handbook*, pages 43–95, 2003.
- D. Bailey. *When push comes to shove: A computational model of the role of motor control in the acquisition of action verbs*. PhD thesis, UNIVERSITY of CALIFORNIA, 1997.

- C. F. Baker, C. J. Fillmore, and J. B. Lowe. The berkeley framenet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1*, ACL '98, pages 86–90. Association for Computational Linguistics, 1998. doi: 10.3115/980845.980860. URL <http://dx.doi.org/10.3115/980845.980860>.
- R. Barker. *Ecological psychology: Concepts and methods for studying the environment of human behavior*. Stanford Univ Pr, 1968.
- M. Beetz, U. Klank, I. Kresse, A. Maldonado, L. Mösenlechner, D. Pangercic, T. Rühr, and M. Tenorth. Robotic Roommates Making Pancakes. In *11th IEEE-RAS International Conference on Humanoid Robots*, Bled, Slovenia, October, 26–28 2011.
- M. Beetz, D. Jain, L. Mösenlechner, M. Tenorth, L. Kunze, N. Blodow, and D. Pangercic. Cognition-enabled autonomous robot control for the realization of home chore task intelligence. *Proceedings of the IEEE*, 100(8):2454–2471, 2012.
- M. Beetz, F. Balint-Benczedi, N. Blodow, D. Nyga, T. Wiedemeyer, and Z.-C. Marton. RoboSherlock: Unstructured Information Processing for Robot Perception. In *IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, Washington, USA, 2015a. URL <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7139395>. Best Service Robotics Paper Award.
- M. Beetz, H. Langer, and D. Nyga. Planning everyday manipulation tasks—prediction-based transformation of structured activity descriptions. In *Exploring Cybernetics*, pages 63–83. Springer, 2015b.
- M. Beetz, M. Tenorth, and J. Winkler. Open-EASE – a knowledge processing service for robots and robotics/ai researchers. In *IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, Washington, USA, 2015c. Finalist for the Best Cognitive Robotics Paper Award.
- M. Beetz, D. Beßler, J. Winkler, J.-H. Worch, F. Balint-Benczedi, G. Bartels, A. Billard, A. K. Bozcuoglu, Z. Fang, N. Figueroa, A. Haidu, H. Langer, A. Maldonado, A.-L. Pais, M. Tenorth, and T. Wiedemeyer. Open Robotics Research Using Web-based Knowledge Services. In *International Conference on Robotics and Automation (ICRA)*, Stockholm, Sweden, 2016.
- I. Beltagy and R. J. Mooney. Efficient markov logic inference for natural language semantics. In *Proceedings of the Fourth International Workshop on Statistical Relational AI at AAAI (StarAI-2014)*, pages 9–14, Quebec City, Canada, July 2014.

- B. Bergen, S. Narayan, and J. Feldman. Embodied verbal semantics: Evidence from an image-verb matching task. In *Proceedings of the Twenty-Fifth Annual Conference of the Cognitive Science Society*, pages 139–144, 2003.
- J. Besag. Statistical Analysis of Non-Lattice Data. *Journal of the Royal Statistical Society. Series D (The Statistician)*, 24(3):pp. 179–195, 1975. ISSN 00390526. URL <http://www.jstor.org/stable/2987782>.
- C. M. Bishop. Pattern recognition. *Machine Learning*, 128, 2006.
- N. Blodow. *Managing Belief States for Service Robots: Dynamic Scene Perception and Spatio-temporal Memory*. PhD thesis, Intelligent Autonomous Systems Group, Department of Informatics, Technische Universität München, 2014.
- M. Bollini, J. Barry, and D. Rus. BakeBot: Baking Cookies with the PR2. In *The PR2 Workshop, from International Conference on Intelligent Robots and Systems (IROS)*, 2011.
- A. Boteanu and S. Chernova. Solving and explaining analogy questions using semantic networks. In *AAAI*, pages 1460–1466, 2015.
- S. R. Branavan, H. Chen, L. S. Zettlemoyer, and R. Barzilay. Reinforcement learning for mapping instructions to actions. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 82–90. Association for Computational Linguistics, 2009.
- E. Brill and R. C. Moore. An improved error model for noisy channel spelling correction. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 286–293. Association for Computational Linguistics, 2000.
- M. Brocheler, L. Mihalkova, and L. Getoor. Probabilistic similarity logic. *Conference on Uncertainty in Artificial Intelligence*, 2010.
- C. G. Broyden. The convergence of a class of double-rank minimization algorithms 1. general considerations. *IMA Journal of Applied Mathematics*, 6(1):76, 1970. doi: 10.1093/imamat/6.1.76. URL [+http://dx.doi.org/10.1093/imamat/6.1.76](http://dx.doi.org/10.1093/imamat/6.1.76).
- I. C. Cárdenas, S. S. Al-jibouri, J. I. Halman, and F. A. van Tol. Capturing and integrating knowledge for managing risks in tunnel works. *Risk analysis*, 33(1): 92–108, 2013.

- A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka Jr, and T. M. Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, volume 5, page 3, 2010.
- P. Chang and J. Krumm. Object recognition with color cooccurrence histograms. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 2. IEEE, 1999.
- N. Chater, J. B. Tenenbaum, and A. Yuille. Probabilistic Models of Cognition: Conceptual Foundations. *Trends in Cognitive Sciences, Special Issue: Probabilistic Models of Cognition*, 10(7), 2006.
- H.-T. Cheng, F.-T. Sun, M. Griss, P. Davis, J. Li, and D. You. Nuactiv: Recognizing unseen new activities using semantic attribute-based learning. In *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*, pages 361–374. ACM, 2013.
- K. Chodorow and M. Dirolf. *MongoDB: The Definitive Guide*. O’Reilly Media, Inc., 1st edition, 2010. ISBN 1449381561, 9781449381561.
- J. Clausen. Branch and bound algorithms-principles and examples. *Department of Computer Science, University of Copenhagen*, pages 1–30, 1999.
- A. Collet, M. Martinez, and S. S. Srinivasa. The moped framework: Object recognition and pose estimation for manipulation. *The International Journal of Robotics Research*, 30(10):1284–1306, 2011.
- J.-B. Coumau, H. Furuhashi, and H. Sarrazin. A smart home is where the bot is. *McKinsey Global Institute*, January 2017.
- D. Crystal. *Dictionary of linguistics and phonetics*, volume 30. John Wiley & Sons, 2011.
- N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893 vol. 1, June 2005.
- J. Davis and P. Domingos. Deep transfer via second-order markov logic. In *Proceedings of the 26th annual international conference on machine learning*, pages 217–224. ACM, 2009.
- M. De Marneffe, B. MacCartney, and C. Manning. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454, 2006.

- M.-C. de Marneffe and C. D. Manning. The Stanford Typed Dependencies Representation. In *Coling 2008: Proceedings of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation*, CrossParser '08, pages 1–8. Association for Computational Linguistics, 2008. ISBN 978-1-905593-50-7. URL <http://dl.acm.org/citation.cfm?id=1608858.1608859>.
- J. Decety and F. Michel. Comparative analysis of actual and mental movement times in two graphic tasks. *Brain and cognition*, 11(1):87–97, 1989.
- P. Domingos and W. Webb. A tractable first-order probabilistic logic. In *Proceedings of the Twenty-Sixth National Conference on Artificial Intelligence*, 2012.
- K. Doya, S. Ishii, A. Pouget, and R. P. N. Rao, editors. *Bayesian Brain*. MIT Press, 2007.
- K. Duan, D. Parikh, D. Crandall, and K. Grauman. Discovering Localized Attributes for Fine-grained Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.
- J. Dzifcak, M. Scheutz, C. Baral, and P. Schermerhorn. What to do and how to do it: Translating natural language directives into temporal and dynamic logic representation for goal management and action execution. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 4163–4168. IEEE, 2009.
- S. Epping. Web-enabled Learning of Models for Word Sense Disambiguation. *Bachelor's Thesis*, 2011.
- O. Etzioni, M. Banko, S. Soderland, and D. S. Weld. Open information extraction from the web. *Communications of the ACM*, 51(12):68–74, 2008.
- J. Fan, A. Kalyanpur, D. Gondek, and D. A. Ferrucci. Automatic knowledge extraction from documents. *IBM Journal of Research and Development*, 56(3.4):5–1, 2012.
- Z. Fang, G. Bartels, and M. Beetz. Learning models for constraint-based motion parameterization from interactive physics-based simulation. In *International Conference on Intelligent Robots and Systems (IROS)*, Daejeon, South Korea, 2016.
- A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth. Describing Objects by their Attributes. In *CVPR*, 2009.
- J. Feldman and S. Narayanan. Embodied meaning in a neural theory of language. *Brain and Language*, 89(2):385 – 392, 2004. ISSN 0093-934X. doi: [http://dx.doi.org/10.1016/S0093-934X\(03\)00355-9](http://dx.doi.org/10.1016/S0093-934X(03)00355-9). URL

- <http://www.sciencedirect.com/science/article/pii/S0093934X03003559>.
Language and MotorIntegration.
- C. Fellbaum. *WordNet: an electronic lexical database*. MIT Press USA, 1998.
- D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. Prager, N. Schlaefer, and C. Welty. Building Watson: An overview of the DeepQA project. *AI Magazine*, 31(3):59–79, 2010a. ISSN 0738-4602. URL <http://www.aaai.org/ojs/index.php/aimagazine/article/view/2303>.
- D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. Prager, et al. Building watson: An overview of the deepqa project. *AI magazine*, 31(3):59–79, 2010b.
- R. O. Fikes and N. J. Nilsson. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. Technical Report 43r, AI Center, SRI International, 1971. URL <http://www.ai.sri.com/shakey/>.
- C. Fillmore. Frame semantics and the nature of language. *Annals of the New York Academy of Sciences*, 280(1):20–32, 1976.
- J. Firby. *Adaptive Execution in Complex Dynamic Worlds*. Technical report 672, Yale University, Department of Computer Science, January 1989.
- D. Gentner and L. Smith. Analogical reasoning. *Encyclopedia of human behavior*, 130: 130, 2012.
- M. Georgeff and F. Ingrand. Decision making in an embedded reasing system, 1989.
- S. J. Gershman, E. J. Horvitz, and J. B. Tenenbaum. Computational rationality: A converging paradigm for intelligence in brains, minds, and machines. *Science*, 349 (6245):273–278, 2015.
- L. Getoor. *Introduction to statistical relational learning*. MIT press, 2007a.
- L. Getoor. *Introduction to statistical relational learning*. MIT press, 2007b.
- W. R. Gilks, S. Richardson, and D. Spiegelhalter. *Markov chain Monte Carlo in practice*. CRC press, 1995.
- L. C. Goron, Z. C. Marton, G. Lazea, and M. Beetz. Segmenting cylindrical and box-like objects in cluttered 3D scenes. In *7th German Conference on Robotics (ROBOTIK)*, Munich, Germany, 2012.

- T. L. Griffiths, C. Kemp, and J. B. Tenenbaum. *The Cambridge Handbook of Computational Cognitive Modeling*, chapter Bayesian Models of Cognition. Cambridge University Press, 2008.
- S. Guadarrama, L. Riano, D. Golland, D. Gouhring, Y. Jia, D. Klein, P. Abbeel, and T. Darrell. Grounding spatial relations for human-robot interaction. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 1640–1647. IEEE, 2013.
- S. Guadarrama, E. Rodner, K. Saenko, N. Zhang, R. Farrell, J. Donahue, and T. Darrell. Open-vocabulary object retrieval. RSS, 2014.
- S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit. Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In *IEEE International Conference on Computer Vision (ICCV)*, 2011.
- R. S. Hunter. Photoelectric color difference meter. *Josa*, 48(12):985–995, 1958.
- International Computer Science Institute. FrameNet Data.
<https://framenet.icsi.berkeley.edu/fndrupal/frameIndex>, 2017-02-22.
Accessed: 2017-02-22.
- A. K. Jain, R. P. W. Duin, and J. Mao. Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–37, 2000.
- D. Jain. Knowledge Engineering with Markov Logic Networks: A Review. In *DKB 2011: Proceedings of the Third Workshop on Dynamics of Knowledge and Belief*, 2011.
- D. Jain. *Probabilistic Cognition for Technical Systems: Statistical Relational Models for High-Level Knowledge Representation, Learning and Reasoning*. PhD thesis, Technische Universität München, 2012. URL
http://mediatum.ub.tum.de/node?id=1096684&change_language=en.
- D. Jain and M. Beetz. Soft Evidential Update via Markov Chain Monte Carlo Inference. In *KI 2010: Advances in Artificial Intelligence, 33rd Annual German Conference on AI*, volume 6359 of *Lecture Notes in Computer Science*, pages 280–290. Springer, 2010. ISBN 978-3-642-16110-0.
- D. Jain, P. Maier, and G. Wylezich. Markov Logic as a Modelling Language for Weighted Constraint Satisfaction Problems. In *Eighth International Workshop on Constraint Modelling and Reformulation, in conjunction with CP2009*, 2009a.

- D. Jain, L. Mösenlechner, and M. Beetz. Equipping Robot Control Programs with First-Order Probabilistic Reasoning Capabilities. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3626–3631, 2009b.
- D. Jayaraman, F. Sha, and K. Grauman. Decorrelating semantic visual attributes by resisting the urge to share. In *CVPR*, 2014.
- H. Kamp. A theory of truth and semantic representation. *Formal semantics: The essential readings*, 1:189–222, 2008.
- B. Kaup, J. Lüdtke, and C. Maienborn. “the drawer is still closed”: Simulating past and future actions when processing sentences that describe a state. *Brain and Language*, 112(3):159–166, 2010.
- H. A. Kautz, B. Selman, and Y. Jiang. A general stochastic approach to solving problems with hard and soft constraints. *Satisfiability Problem: Theory and Applications*, 35:573–586, 1996.
- P. Kay and T. Regier. Resolving the question of color naming universals. *Proceedings of the National Academy of Sciences*, 100(15):9085–9089, 2003.
- A. Kemper and A. Eickler. *Datenbanksysteme*. Oldenbourg Wissenschaftsverlag, 2013. In German.
- C. Kiddon, G. T. Ponnuraj, L. Zettlemoyer, and Y. Choi. Mise en Place: Unsupervised Interpretation of Instructional Recipes. In *EMNLP*, pages 982–992, 2015.
- A. Kilgarri. Senseval: An exercise in evaluating word sense disambiguation programs. In *Proc. of the first international conference on language resources and evaluation*, pages 581–588, 1998.
- J. Kim and R. J. Mooney. Unsupervised pcf induction for grounded language learning with highly ambiguous supervision. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 433–444. Association for Computational Linguistics, 2012.
- R. D. King, K. E. Whelan, F. M. Jones, P. G. Reiser, C. H. Bryant, S. H. Muggleton, D. B. Kell, and S. G. Oliver. Functional genomic hypothesis generation and experimentation by a robot scientist. *Nature*, 427(6971):247–252, 2004.
- D. Klein and C. D. Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics, 2003.

- D. C. Knill and A. Pouget. The Bayesian Brain: The Role of Uncertainty in Neural Coding and Computation. *Trends in Neurosciences*, 27(12), 2004.
- N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3, pages 2149–2154. IEEE, 2004.
- D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.
- C. Kong, D. Lin, M. Bansal, R. Urtasun, and S. Fidler. What are you talking about? text-to-image coreference. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 3558–3565. IEEE, 2014.
- S. Koralewski. Scaling Probabilistic Completion of Robot Instructions through Semantic Information Retrieval. *Master's Thesis*, 2016.
- E. Krause, M. Zillich, T. Williams, and M. Scheutz. Learning to recognize novel objects in one shot through human-robot interactions in natural language dialogues. In *Proceedings of Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.
- M. P. Kumar, P. H. S. Torr, and A. Zisserman. Objcut: Efficient segmentation using top-down and bottom-up cues. *IEEE Trans. Pattern Anal. Mach. Intell.*, pages 530–545.
- L. Kunze, M. Tenorth, and M. Beetz. Putting People's Common Sense into Knowledge Bases of Household Robots. In *33rd Annual German Conference on Artificial Intelligence (KI 2010)*, pages 151–159, Karlsruhe, Germany, September 21-24 2010. Springer.
- L. Kunze, M. E. Dolha, and M. Beetz. Logic Programming with Simulation-based Temporal Projection for Everyday Robot Object Manipulation. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, San Francisco, CA, USA, September, 25–30 2011. Best Student Paper Finalist.
- J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the eighteenth international conference on machine learning, ICML*, volume 1, pages 282–289, 2001.
- B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

- ISSN 0036-8075. doi: 10.1126/science.aab3050. URL
<http://science.sciencemag.org/content/350/6266/1332>.
- C. H. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *CVPR*, 2009.
- Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.
- D. Lenat. CYC: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38, 1995. ISSN 0001-0782.
- P. Liang, M. I. Jordan, and D. Klein. Learning dependency-based compositional semantics. *Computational Linguistics*, 39(2):389–446, 2013.
- G. Lisca, D. Nyga, F. Bálint-Benczédi, H. Langer, and M. Beetz. Towards Robots Conducting Chemical Experiments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Hamburg, Germany, 2015.
- E. Loper and S. Bird. Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ETMTNLP '02, pages 63–70, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1118108.1118117. URL <http://dx.doi.org/10.3115/1118108.1118117>.
- D. Lowd and P. Domingos. Efficient Weight Learning for Markov Logic Networks. In *PKDD 2007, 11th European Conference on Principles and Practice of Knowledge Discovery in Databases*, volume 4702 of *Lecture Notes in Computer Science*, pages 200–211. Springer, 2007.
- D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004. ISSN 0920-5691. URL <http://dx.olddoi.org/10.1023/B%3AVISI.0000029664.99615.94>.
- T. Lukasiewicz. Expressive probabilistic description logics. *Artificial Intelligence*, 172(6–7):852 – 883, 2008. ISSN 0004-3702. doi: <http://dx.doi.org/10.1016/j.artint.2007.10.017>. URL <http://www.sciencedirect.com/science/article/pii/S0004370207001877>.
- I. Lysenkov, V. Eruhimov, and G. Bradski. Recognition and Pose Estimation of Rigid Transparent Objects with a Kinect Sensor. In *Proceedings of Robotics: Science and Systems*, Sydney, Australia, July 2012.

- J. Manyika, M. Chui, J. Bughin, R. Dobbs, P. Bisson, and A. Marrs. Disruptive technologies: Advances that will transform life, business, and the global economy. *McKinsey Global Institute*, May 2013.
- M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993.
- K. Markman, J. A. Suhr, and W. M. P. Klein. *Handbook of Imagination and Mental Simulation*. Psychology Press, 2008.
- Z.-C. Marton, F. Balint-Benczedi, F. Seidel, L. C. Goron, and M. Beetz. Object Categorization in Clutter using Additive Features and Hashing of Part-graph Descriptors. In *Proceedings of Spatial Cognition (SC)*, Abbey Kloster Seeon, Germany, 2012.
- C. Matuszek, D. Fox, and K. Koscher. Following directions using statistical machine translation. In *Proceeding of the 5th ACM/IEEE international conference on Human-robot interaction*, pages 251–258. ACM, 2010.
- C. Matuszek, N. FitzGerald, L. Zettlemoyer, L. Bo, and D. Fox. A Joint Model of Language and Perception for Grounded Attribute Learning. *Proceedings of the 29th International Conference on Machine Learning (ICML 2012)*, 2012.
- C. Matuszek, E. Herbst, L. Zettlemoyer, and D. Fox. Learning to parse natural language commands to a robot control system. In *Experimental Robotics*, pages 403–415. Springer, 2013.
- D. McDermott. Robot Planning. 13(2):55–79, 1992.
- A. McEnery, I. Tanaka, and S. Botley. Corpus annotation and reference resolution. In *Proceedings of a Workshop on Operational Factors in Practical, Robust Anaphora Resolution for Unrestricted Texts*, ANARESOLUTION '97, pages 67–74, Stroudsburg, PA, USA, 1997. Association for Computational Linguistics. URL <http://dl.acm.org/citation.cfm?id=1598819.1598829>.
- F. Meyer. Grounding Words to Objects: A Joint Model for Co-reference and Entity Resolution Using Markov Logic Networks for Robot Instruction Processing. *Diploma Thesis*, 2013.
- A. S. Mian, M. Bennamoun, and R. Owens. Three-dimensional model-based object recognition and segmentation in cluttered scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1584–1601, October 2006. ISSN 0162-8828.

- Michael Rupp. Siri vs. Cortana vs. Google Now: Das Duell. <http://www.pc-magazin.de/vergleich/siri-cortana-google-now-test-vergleich-3195396.html>, 2015. Accessed: 2017-03-31.
- G. A. Miller, C. Leacock, R. Teng, and R. T. Bunker. A semantic concordance. In *Proceedings of the workshop on Human Language Technology*, pages 303–308. Association for Computational Linguistics, 1993.
- M. Minsky. A Framework for Representing Knowledge. Technical Report Memo 306, MIT-AI Laboratory, 1974.
- M. L. Minsky. Logical versus analogical or symbolic versus connectionist or neat versus scruffy. *AI magazine*, 12(2):34, 1991.
- D. Misra, J. Sung, K. Lee, A. Saxena, J. Sung, B. Selman, A. Saxena, J. Sung, C. Ponce, B. Selman, et al. Tell me dave: Context-sensitive grounding of natural language to mobile manipulation instructions. In *Robotics: Science and Systems, RSS*, 2014a.
- D. K. Misra, K. Tao, P. Liang, and A. Saxena. Environment-driven lexicon induction for high-level instructions.
- D. K. Misra, J. Sung, K. Lee, and A. Saxena. Tell me dave: Context-sensitive grounding of natural language to manipulation instructions. In *Proceedings of Robotics: Science and Systems*, Berkeley, USA, July 2014b.
- R. Moore. A formal theory of knowledge and action. Technical report, DTIC Document, 1984.
- L. Mösenlechner and M. Beetz. Parameterizing Actions to have the Appropriate Effects. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, San Francisco, CA, USA, September 25–30 2011.
- L. Mösenlechner. *The Cognitive Robot Abstract Machine*. Dissertation, Technische Universität München, München, 2016.
- E. Neo, T. Sakaguchi, and K. Yokoi. A natural language instruction system for humanoid robots integrating situated speech recognition, visual recognition and on-line whole-body motion generation. In *Advanced Intelligent Mechatronics, 2008. AIM 2008. IEEE/ASME International Conference on*, pages 1176–1182. IEEE, 2008.
- A. Newell. The knowledge level. *Artificial intelligence*, 18(1):87–127, 1982.
- A. Newell, H. A. Simon, et al. *Human problem solving*, volume 104. Prentice-Hall Englewood Cliffs, NJ, 1972.

- M. Niepert, J. Noessner, and H. Stuckenschmidt. Log-linear description logics. In *IJCAI*, pages 2153–2158, 2011.
- D. Nyga and M. Beetz. Everything Robots Always Wanted to Know about Housework (But were afraid to ask). In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vilamoura, Portugal, October, 7–12 2012.
- D. Nyga and M. Beetz. Reasoning about Unmodelled Concepts – Incorporating Class Taxonomies in Probabilistic Relational Models. In *arxiv.org*, 2015a. Preprint: <http://arxiv.org/abs/1504.05411>.
- D. Nyga and M. Beetz. Cloud-based Probabilistic Knowledge Services for Instruction Interpretation. In *International Symposium of Robotics Research (ISRR)*, Sestri Levante (Genoa), Italy, 2015b.
- D. Nyga, F. Balint-Benczedi, and M. Beetz. PR2 Looking at Things: Ensemble Learning for Unstructured Information Processing with Markov Logic Networks. In *IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, China, May 31-June 7 2014.
- D. Nyga, M. Picklum, and M. Beetz. What No Robot Has Seen Before – Probabilistic Interpretation of Natural-language Object Descriptions. In *International Conference on Robotics and Automation (ICRA)*, Singapore, 2017a. Accepted for publication.
- D. Nyga, M. Picklum, S. Koralewski, and M. Beetz. Instruction Completion through Instance-based Learning and Semantic Analogical Reasoning. In *International Conference on Robotics and Automation (ICRA)*, Singapore, 2017b. Accepted for publication.
- K. Okada, M. Kojima, S. Tokutsu, T. Maki, Y. Mori, and M. Inaba. Multi-cue 3D object recognition in knowledge-based vision-guided humanoid robot system. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*., pages 3217–3222, 2007.
- Open Source Initiative. The 3-Clause BSD License. <https://opensource.org/licenses/BSD-3-Clause>, 2017. Accessed: 2017-02-22.
- M. Palmer, D. Gildea, and P. Kingsbury. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics*, 31(1):71–106, 2005.
- D. Pangercic, M. Tenorth, B. Pitzer, and M. Beetz. Semantic object maps for robotic housework - representation, acquisition and use. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vilamoura, Portugal, October, 7–12 2012.

- Picklum, Mareike. To see what no robot has seen before - Recognizing objects based on natural-language descriptions. *Master's Thesis*, 2015.
- R. Polikar. Ensemble based systems in decision making. *IEEE Circuits and Systems Magazine*, pages 21–45, 2006.
- M. Pomarlan, D. Nyga, M. Picklum, S. Koralewski, and M. Beetz. Deeper Understanding of Vague Instructions through Simulated Execution (Extended Abstract). In *Proceedings of the 2017 International Conference on Autonomous Agents & Multiagent Systems, AAMAS '17*. International Foundation for Autonomous Agents and Multiagent Systems, 2017.
- H. Poon and P. Domingos. Sound and efficient inference with probabilistic and deterministic dependencies. In *AAAI*, volume 6, pages 458–463, 2006.
- G. Pratt and J. Manzo. The darpa robotics challenge [competitions]. *IEEE Robotics & Automation Magazine*, 20(2):10–12, 2013.
- A. Pronobis and P. Jensfelt. Large-scale semantic mapping and reasoning with heterogeneous modalities. In *Proceedings of the 2012 IEEE International Conference on Robotics and Automation (ICRA'12)*, Saint Paul, MN, USA, 2012. doi: 10.1109/ICRA.2012.6224637. URL <http://www.pronobis.pro/publications/pronobis2012icra>.
- M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.
- R. Reiter. A logic for default reasoning. *Artificial intelligence*, 13(1-2):81–132, 1980.
- M. Richardson and P. Domingos. Markov Logic Networks. *Machine Learning*, 62(1-2):107–136, 2006. ISSN 0885-6125. doi: <http://dx.doi.org/10.1007/s10994-006-5833-1>.
- A. Richtsfeld, T. Morwald, J. Prankl, M. Zillich, and M. Vincze. Segmentation of unknown objects in indoor environments. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 4791–4796, 2012. doi: 10.1109/IROS.2012.6385661.
- D. Roth. On the hardness of approximate reasoning. *Artificial Intelligence*, 82(1): 273–302, 1996.
- T. Rühr, J. Sturm, D. Pangercic, D. Cremers, and M. Beetz. A generalized framework for opening doors and drawers in kitchen environments. In *IEEE International*

-
- Conference on Robotics and Automation (ICRA)*, St. Paul, MN, USA, May 14–18 2012.
- S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*, volume 2. Prentice Hall, 2003.
- R. B. Rusu and S. Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–4, Shanghai, China, May 9-13 2011.
- J. Ryu, Y. Jung, K. Kim, and S. Myaeng. Automatic extraction of human activity knowledge from method-describing web articles. *Proceedings of the 1st Workshop on Automated Knowledge Base Construction*, page 16, 2010.
- S. Schaal. The new robotics—towards human-centered machines. *HFSP journal*, 1 (2):115–126, 2007.
- U. Schöning. *Logik für Informatiker*. BI Wissenschaftsverlag Mannheim, 1987. In German.
- D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587): 484–489, 2016.
- R. F. Simmons. *Synthetic language behavior*. System Development Corporation, 1963.
- H. Simon. Rational choice and the structure of the environment. *Psychological review*, 63(2):129, 1956.
- H. A. Simon. Models of man; social and rational. 1957.
- SPARC. Multi-annual Roadmap for Robotics in Europe 2014-2020, 2014a.
- SPARC. Strategic Research Agenda for Robotics in Europe 2014-2020, 2014b.
- S. Srivastava, S. Zilberstein, A. Gupta, P. Abbeel, and S. J. Russell. Tractability of planning with loops. In *AAAI*, 2015.
- Stanford University. Siri, a Virtual Personal Assistant.
<http://www-ksl.stanford.edu/projects/CALO/>, 2006. Accessed: 2017-02-22.
- Y. Su, M. Allan, and F. Jurie. Improving object classification using semantic attributes. In *BMVC*, pages 1–10, 2010.

- G. Sukthankar, C. Geib, H. H. Bui, D. Pynadath, and R. P. Goldman. *Plan, Activity, and Intent Recognition: Theory and Practice*. Newnes, 2014.
- Y. Sun, L. Bo, and D. Fox. Attribute based object identification. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 2096–2103. IEEE, 2013a.
- Y. Sun, L. Bo, and D. Fox. Attribute Based Object Identification. In *IEEE International Conference on Robotics and Automation*, 2013b.
- J. Sung, B. Selman, and A. Saxena. Synthesizing manipulation sequences for under-specified tasks using unrolled markov random fields. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2970–2977. IEEE, 2014.
- S. Tellex, T. Kollar, S. Dickerson, M. Walter, A. Banerjee, S. Teller, and N. Roy. Understanding natural language commands for robotic navigation and mobile manipulation. In *Proc. Nat'l Conf. on Artificial Intelligence (AAAI)*, 2011.
- S. Tellex, P. Thaker, J. Joseph, and N. Roy. Learning perceptually grounded word meanings from unaligned parallel data. *Machine Learning*, 94(2):151–167, 2014.
- J. B. Tenenbaum, C. Kemp, T. L. Griffiths, and N. D. Goodman. How to grow a mind: Statistics, structure, and abstraction. *science*, 331(6022):1279–1285, 2011.
- M. Tenorth. *Knowledge Processing for Autonomous Robots*. PhD thesis, Intelligent Autonomous Systems Group, Department of Informatics, Technische Universität München, 2011.
- M. Tenorth, D. Nyga, and M. Beetz. Understanding and Executing Instructions for Everyday Manipulation Tasks from the World Wide Web. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1486–1491, Anchorage, AK, USA, May 3–8 2010.
- D. Thau and B. Ludäscher. Reasoning about taxonomies in first-order logic. *Ecological Informatics*, 2(3):195–209, 2007.
- B. J. Thomas and O. C. Jenkins. Roboframenet: Verb-centric semantics for actions in robot middleware. In *2012 IEEE International Conference on Robotics and Automation*, pages 4750–4755, May 2012. doi: 10.1109/ICRA.2012.6225172.
- Tom Gruber. Siri, a Virtual Personal Assistant.
<http://tomgruber.org/technology/siri.htm>, 2015. Accessed: 2017-02-22.

- J. Uebersax. Genetic counseling and cancer risk modeling: An application of bayes nets. *Marbella, Spain: Ravenpack International*, 2004.
- A. C. Varzi. Vagueness, logic, and ontology. *The Dialogue. Yearbooks for Philosophical Hermeneutics*, 2001.
- D. Vernon. *Artificial cognitive systems: A primer*. MIT Press, 2014.
- S. Vosniadou and A. Ortony. *Similarity and analogical reasoning*. Cambridge University Press, 1989.
- J. Wang and P. Domingos. Hybrid Markov Logic Networks. In D. Fox and C. P. Gomes, editors, *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, pages 1106–1111. AAAI Press. ISBN 978-1-57735-368-3. URL <http://dblp.uni-trier.de/db/conf/aaai/aaai2008.html#WangD08>.
- J. Wang, K. Markert, and M. Everingham. Learning models for object recognition from natural language descriptions. 2009.
- Wikimedia Commons. Ein HSV-Farbwähler. https://de.wikipedia.org/wiki/HSV-Farbraum#/media/File:Triangulo_HSV.png, 2017. Accessed: 2017-02-22.
- Wikipedia. Uncertainty. <https://en.wikipedia.org/wiki/Uncertainty>, 2017. Accessed: 2017-02-22.
- K. Williams, E. Bilsland, A. Sparkes, W. Aubrey, M. Young, L. N. Soldatova, K. De Grave, J. Ramon, M. de Clare, W. Sirawaraporn, S. G. Oliver, and R. D. King. Cheaper faster drug development validated by the repositioning of drugs against neglected tropical diseases. *Journal of The Royal Society Interface*, 12(104), 2015. ISSN 1742-5689. doi: 10.1098/rsif.2014.1289. URL <http://rsif.royalsocietypublishing.org/content/12/104/20141289>.
- C. Wu and H. Aghajan. Recognizing objects in smart homes based on human interaction. In *Advanced Concepts in Vision Systems (ACIVS)*, volume LNCS 6475, pages 131–142, Sydney, Australia, December 2010. Springer Verlag Berlin. ISBN 978-3-642-17690-6.
- Z. Wu and M. S. Palmer. Verb semantics and lexical selection. In *ACL*, pages 133–138, 1994.
- P. R. Wurman and J. M. Romano. The amazon picking challenge 2015. *IEEE Robotics and Automation Magazine*, 22(3):10–12, 2015.

- F. X. Yu, L. Cao, R. S. Feris, J. R. Smith, and S.-F. Chang. Designing category-level attributes for discriminative visual recognition. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 771–778. IEEE, 2013.
- Z. Zhong and H. T. Ng. It makes sense: A wide-coverage word sense disambiguation system for free text. In *Proceedings of the ACL 2010 System Demonstrations, ACLDemos '10*, pages 78–83. Association for Computational Linguistics, 2010. URL <http://dl.acm.org/citation.cfm?id=1858933.1858947>.