TECHNISCHE UNIVERSITÄT MÜNCHEN

Lehrstuhl für Informatik IX
Intelligente Autonome Systeme

Probabilistic Cognition for Technical Systems

Statistical Relational Models for High-Level Knowledge Representation, Learning and Reasoning

Dominik Jain

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:	Univ.-Prof. Dr. Felix Brandt
Prüfer der Dissertation:	1. Univ.-Prof. Michael Beetz, Ph.D., Universität Bremen
	2. Univ.-Prof. Dr. Marc Toussaint, Freie Universität Berlin

Die Dissertation wurde am 31.01.2012 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 09.07.2012 angenommen.

Abstract

For the realisation of cognitive capabilities in technical systems such as autonomous robots, the integration of many distinct cognitive resources that support learning and reasoning mechanisms can help to overcome the many challenges posed by the real world. Since many real-world problems involve uncertainty, this work explores the potential of statistical relational models as a resource for probabilistic high-level learning and reasoning. Statistical relational models combine the principles of logical representations with the semantics of probabilistic graphical models and thus constitute a powerful framework for high-level probabilistic cognition in technical systems.

Two complementary representation languages for statistical relational models are considered: the established formalism of Markov logic networks (MLNs) and the novel formalism of Bayesian logic networks. Knowledge engineering issues that arise in statistical relational models are analysed, and important limitations regarding the generalisation of models across domains are identified. As novel extensions of MLNs that extend the scope of the generalised probability distributions that can be represented, this thesis introduces Markov logic networks with probability constraints as well as adaptive Markov logic networks, which dynamically adjust model parameters depending on the properties of the domain at hand. Bayesian logic networks (BLNs) are an alternative representation formalism which is based on the notion of mixed networks with probabilistic and deterministic dependencies, integrating local dependencies represented as conditional probability distributions with global constraints formulated in first-order logic. The combination of these two forms of representation can render both learning and reasoning problems more manageable in practice without substantially sacrificing expressiveness. Furthermore, as an important concept for the exchange of probabilistic information between components of a

technical cognitive system, this work addresses the treatment of uncertain evidence and presents an efficient Markov chain Monte Carlo method for soft evidential update.

Finally, concrete applications of statistical relational models in robot perception, cognition-enabled robot control and the assessment of production plans in a cognitive factory are considered. In perception, statistical relational models are used to categorise objects based on relational descriptions and for the data association task of spatio-temporal object identity resolution. In cognition-enabled robot control, models are tightly integrated with control structures, enabling robots to parametrise plans, to interpret and diagnose situations and to make predictions. In the cognitive factory, statistical relational models are used for the representation of factory station behaviour, taking the complex interactions between the production schedule, the products being produced and the state of the factory into consideration.

Kurzfassung

Für die Verwirklichung kognitiver Fähigkeiten in technischen Systemen wie autonomen Robotern kann die Verknüpfung verschiedenartiger kognitiver Ressourcen, die Lern- und Schlussfolgerungsmechanismen bereitstellen, ein adäquates Mittel sein, um den vielfältigen Herausforderungen der wirklichen Welt gerecht zu werden. Da eine Vielzahl von Problemen stark von Unsicherheit geprägt ist, untersucht diese Arbeit das Potenzial von statistischen relationalen Modellen als eine vielversprechende solche Ressource für probabilistische Lern- und Schlussfolgerungsprozesse auf einer abstrakten Ebene. Statistische relationale Modelle verbinden die Prinzipien logischer Repräsentationen mit der Semantik probabilistischer grafischer Modelle und stellen somit ein besonders ausdrucksstarkes Framework für probabilistische Kognition in technischen Systemen dar.

Zwei komplementäre Repräsentationssprachen für statistische relationale Modelle werden behandelt: der etablierte Formalismus der Markov-Logik-Netze sowie der in der Arbeit neu eingeführte Formalismus der Bayesschen Logik-Netze. Es werden Fragestellungen der Wissensmodellierung untersucht und insbesondere Limitierungen bezüglich der Generalisierung von Modellen über Domänengrenzen hinweg herausgearbeitet. Als neuartige Erweiterungen von Markov-Logik-Netzen, die den Anwendungsbereich der abbildbaren, generalisierten Wahrscheinlichkeitsverteilungen vergrößern, werden Markov-Logik-Netze mit Wahrscheinlichkeitsbeschränkungen sowie adaptive Markov-Logik-Netze, welche ihre Parameter den Gegebenheiten einer Domäne spezifisch anpassen, eingeführt. Bayessche Logik-Netze sind eine alternative Repräsentationssprache, die auf gemischten Netzen mit probabilistischen und deterministischen Abhängigkeiten basiert und lokale Abhängigkeiten, die über bedingte Wahrscheinlichkeitsverteilungen repräsentiert sind, mit globalen Beschränkungen, die in der Logik erster Stufe formuliert sind, verbindet. Durch diese Verknüpfung von

Repräsentationsformen kann die praktische Beherrschbarkeit von Schlussfolgerungs- und Lernproblemen verbessert werden, ohne jedoch die Ausdrucksstärke substanziell einzuschränken. Weiterhin wird die Behandlung von unsicheren Evidenzen als wichtiges Konzept für den Austausch unsicheren Wissens zwischen den Komponenten eines technischen Systems behandelt und ein effizientes Markov-Chain-Monte-Carlo-Verfahren für Belief-Updates mit weichen Evidenzen vorgestellt.

Es werden mehrere Anwendungen statistischer relationaler Modelle für technische Systeme betrachtet, inbesondere im Bereich Wahrnehmung für Roboter, in der kognitiven Robotersteuerung sowie für die Bewertung von Produktionsplänen in einer kognitiven Fabrik. Im Bereich Wahrnehmung werden statistische relationale Modelle für die Kategorisierung von Objekten basierend auf relationalen Beschreibungen eingesetzt, sowie für die Lösung des Datenassoziationsproblems der räumlich-zeitlichen Objektidentitätsauflösung. In der kognitiven Robotersteuerung werden Modelle gezielt mit den Kontrollstrukturen eines Roboters verknüpft, um die Parametrisierung von Plänen, die Beurteilung und Interpretation von Situationen sowie das Treffen von Vorhersagen zu unterstützen. In der kognitiven Fabrik finden statistische relationale Modelle für die Repräsentation des Produktionsstationsverhaltens Anwendung, wobei die Wechselwirkungen zwischen Produktionsplänen, den hergestellten Produkten und dem Zustand der Produktionsstätte berücksichtigt werden.

Acknowledgements

I could not have completed this thesis without the support of many people. The first thanks go to my advisor, Michael Beetz, whose ideas, insights and genuine enthusiasm have inspired my work in many ways. I am very grateful for his continued support throughout the years.

Some of the work presented in this thesis – especially the work pertaining to integrated applications – was done jointly with my fellow graduate students at TUM. I thank Nico Blodow, Paul Maier, Lorenz Mösenlechner, Moritz Tenorth, Dejan Pangercic, Bernhard Kirchlecher, Zoltan Marton, Ulrich Klank, Andreas Fedrizzi and the other members of the Intelligent Autonomous Systems Group for many fruitful collaborations. I also thank visiting researcher Enis Bayramoglu for his comments on parts of this thesis and for the many inspiring discussions we had about the meaning of degrees of belief, life, the universe, and everything. I am glad we happened to end up in the same place at the same time.

Furthermore, I have had the pleasure of working with many excellent students over the years, who temporarily joined our research group and who assisted with the implementation of some of the ideas presented in this work along the way. In particular, I thank Andreas M. Barthels, Stefan Waldherr, Klaus von Gleissenthall, Simon Grötzinger, Martin J. Schuster, Ralf Wernicke and Gregor Wylezich for their contributions to the ProbCog project.

Last but not least, I thank my family and friends for their endless support and encouragement, and especially Silvia Pidrmann for reminding me of the things that matter most in life.

My research was partly funded by the DFG research cluster of excellence *CoTeSys* (Cognition for Technical Systems).

Contents

List of Resources

Figures

Tables

Algorithms

Markov Logic Networks

Bayesian Logic Networks

Chapter 1

Introduction

A long-standing goal in artificial intelligence is the creation of artificial cognitive systems that would be capable of displaying competent problem-solving behaviour in the real world. Early attempts at realising artificial cognitive capabilities have emphasised the manipulation of abstract symbols through formal calculi, the corresponding notion of *cognitivism* being that the most central functions of the mind – in particular, thought – can indeed be adequately accounted for through the rule-based manipulation of symbols (Newell and Simon, 1976). The systematic representation of (abstract) knowledge about the world – in some symbolic form – is thus at the very heart of cognitivism.

By contrast, research on *embodied cognition* (Anderson, 2003) has largely eschewed symbols, abstract planning and logical calculi, emphasising instead the physical implementation of an agent and its interaction with the world it inhabits. Consequently, research in embodied cognition typically insists on working exclusively with symbols that could be straightforwardly related to the cognitive system's particular manifestations of percepts and actions, i.e. symbols that are necessarily physically grounded. The physical grounding hypothesis, coined by Brooks (1990), rejects the notion of symbols that assume a knowable objective truth and goes as far as stating that grounding is at the root of intelligence, the key proposition being that it is better to "use the world as its own model" (Brooks, 1991) rather than some abstract, symbolic representation of it. This view of cognition ultimately calls for a bottom-up, developmental approach, in which abstractions are somehow constructed from grounded low-level percepts and experience.

This work largely follows the cognitivist approach, conceding, however, that the realisation of a cognitive technical system that is to act in the real world clearly necessitates the physical grounding of symbols. Notably, the systems that will be considered in this work do not rely on a single well-defined formalism for the transformation of symbols in order to realise cognitive capabilities. Rather, a decidedly hybrid approach is advocated, in which heterogeneous symbolic and sub-symbolic reasoning and learning components are considered as *cognitive resources* and are integrated via an interface language that makes informed use of these resources in order to achieve the system's goals (Beetz et al., 2010a). Real-world problem solving involves many heterogeneous problems that arise at various levels of abstraction, and it seems futile to attempt to treat these problems in a single, unified manner (yet this does not preclude the possibility that the pursuit of artificial *general* intelligence, which aims at exactly this, may ultimately prove to have been a worthwhile endeavour). Some problems lend themselves to the abstract symbol processing of "good old-fashioned" AI, while others call for tightly coupled, reactive feedback loops and situated goal orientation. The form of the system should ultimately follow the structure of the problems it faces. A pragmatic approach towards the realisation of cognition in technical systems will thus seek to combine state-of-the-art cognitive resources as required by the applications at hand.

The specific focus of this work is on one particular type of cognitive resource that explicitly takes into consideration the fact that AI systems acting in the physical world are constantly faced with uncertainty. In real-world domains, uncertainty is typically due to the restriction that the environment is only partially observable and/or to the unavailability of complete domain knowledge. Probabilistic models have proved to be a robust, mathematically sound means of addressing the issue of uncertain knowledge and its representation within AI systems. The principles of probability theory and the relaxations introduced by *Bayesian probability theory*, where probabilities are not to be strictly interpreted as relative frequencies of occurrences of events but rather as potentially subjective *degrees of belief*, constitute a sound calculus for uncertain knowledge in AI, providing the very basis for cognitive capabilities in the presence of uncertainty.

In recent years, the notion of probabilistic models of cognition has also begun to be adopted by researchers in cognitive science. Traditionally, cognitive science views

the brain as an information processor, where information processing typically includes inferring new information from information that has been derived from the senses, from instructions, prior knowledge or other sources of information (Chater et al., 2006). As sensing, knowledge, and actuation are typically strongly affected by uncertainty (Körding and Wolpert, 2004), researchers have come to believe that the mathematical tools of probability theory might form a strong basis for building theories of cognition. Remarkable technical progress in mathematics and computer science has rendered practical the modelling of knowledge and beliefs of cognitive agents that employ probability distributions defined over highly structured systems of representation such as complex graphs, grammars and even predicate logic. As a consequence, *Bayesian cognition* (Griffiths et al., 2008) and the *Bayesian brain* (Knill and Pouget, 2004; Doya et al., 2007) have become a recent and emerging trend in cognitive science.

In technical systems, particularly in robotics, probabilistic models have been very successfully applied to many problems, predominantly at lower levels of abstraction (Thrun et al., 2005). For instance, probabilistic techniques have been applied to problems as diverse as simultaneous localisation and mapping (Leonard and Durrant-Whyte, 1991), navigation (Cassandra et al., 1996), imitation learning (Grimes et al., 2006) and low-level control (Nakanishi et al., 2005). Indeed, uncertainty is ubiquitous in robotics: As depicted in Figure 1.1, it permeates all aspects of robotic applications. Robot perception is subject to uncertainty owing to partial observability and unreliable, noisy sensor readings, necessitating an explicit representation of uncertainty with respect to both the robot's own state and its beliefs regarding the state of the world. Furthermore, the effects of actions carried out by robots typically cannot be considered as deterministic, and the possibility of intermittent failures has to be taken into consideration.

As many of the more fundamental problems in robotics (e.g. with regard to perception, navigation and motion planning) are being solved to a degree where we can consider the respective components to be sufficiently reliable to form the basis for complex tasks, the high-level reasoning and learning capabilities of robots will become increasingly important in the years to come. Uncertainty at the abstract task level will, therefore, more and more shift into the focus. At the task level, the notion of the expected utility of decisions can, for instance, provide a sound theory for ac-

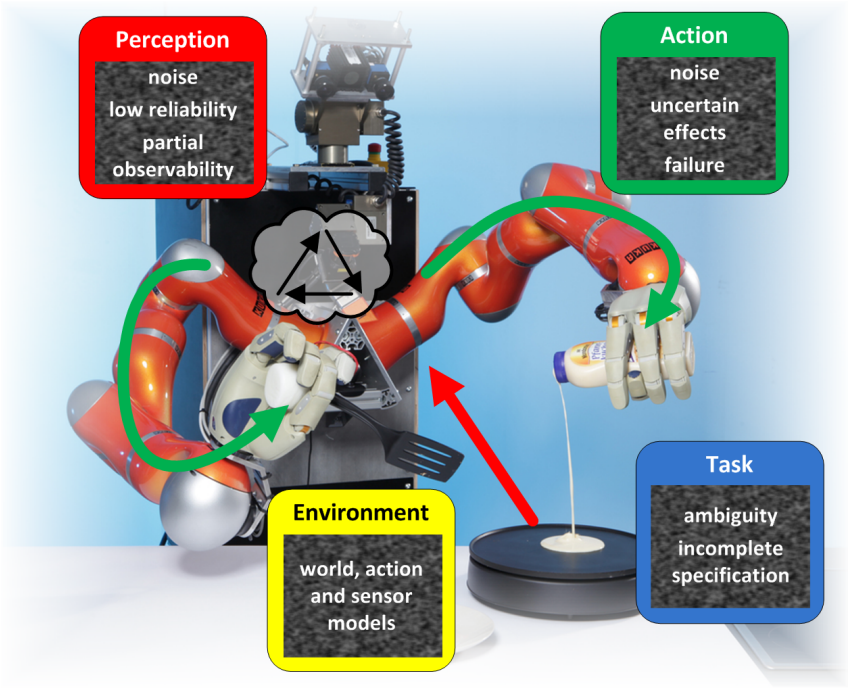


Figure 1.1: *The ubiquity of uncertainty in robotics*

tion selection under uncertainty (Shachter, 1988). Indeed, probabilistic inference can serve as a generic computational principle for the generation of planned behaviour by coupling representations of sensory inputs, actions and goals within a probabilistic setting (Toussaint, 2009). Probabilistic models that capture the complex interactions between the particular entities that are relevant to an agent's tasks – for the interpretation of observations, the selection of actions or the prediction of effects – can be considered as a most valuable resource in general.

Consider, for example, an autonomous household robot that is to assist in tasks such as setting the table, preparing food or cleaning up after a meal. One of the most impressive aspects of the way in which humans deal with such problems is their ability to handle uncertain information. When instructed to set the table for breakfast, people will infer who will participate in the meal, where the participants will sit, what they will want to eat given known personal preferences and what utensils they will consequently require as well as where to put them. Moreover, people will adequately adjust the way in which they set the table upon receiving new information, such as the piece of information that one of the participants prefers breakfast cereals to pancakes. People also have excellent heuristics on where to look for things to be

put on the table, and these heuristics typically take vast amounts of context information, such as recent activity in the kitchen and spatial relationships, into account, so as to determine a suitable order in which places should be searched. All the control decisions related to such tasks require reasoning in the light of uncertainty, and for autonomous household robots to achieve a similar level of competence in dealing with such tasks, it is vital that they be equipped with knowledge representation systems that fully support learning and reasoning at appropriate levels of abstraction.

In this work, I will consider precisely the kinds of high-level probabilistic models that can enable a technical system to reason about scenarios such as the one outlined above. Specifically, I will consider *statistical relational models* (Getoor and Taskar, 2007), which, by combining the principles underlying relational logic and probabilistic graphical models, can be used to probabilistically reason about propositions in a highly general manner. By abstracting away from a concrete set of propositions and representing instead general statistical interrelationships that apply to arbitrary sets of entities, these models alleviate many of the restrictions of standard probabilistic graphical models such as Bayesian networks and Markov random fields. The general statistical interrelationships that are represented essentially serve as templates for the construction of lower-level graphical models. For any given domain of discourse comprised of a particular set of relevant entities, a statistical relational model can be *instantiated* to obtain a concrete probabilistic model that can subsequently be reasoned about.

With statistical relational models as the fundamental representational paradigm, I investigate the promise of joint probability distributions as particularly flexible cognitive resources for technical systems: Once equipped with a statistical relational model representing a family of joint probability distributions over sets of relevant propositions, a technical system – be it an autonomous robot or a production plant – is put in a position where it is, in particular, able to derive arbitrary conditional probabilities that may be relevant to its tasks, i.e. it is able to compute the probability of arbitrary propositions that are of interest given observations corresponding to other propositions. A single model can be used to compute beliefs based on causal, diagnostic and mixed reasoning patterns alike, and the beliefs thus obtained can, for instance, be used by technical systems to make decisions, to parametrise action plans, to diagnose failures or to make predictions. Being based on the principles of probab-

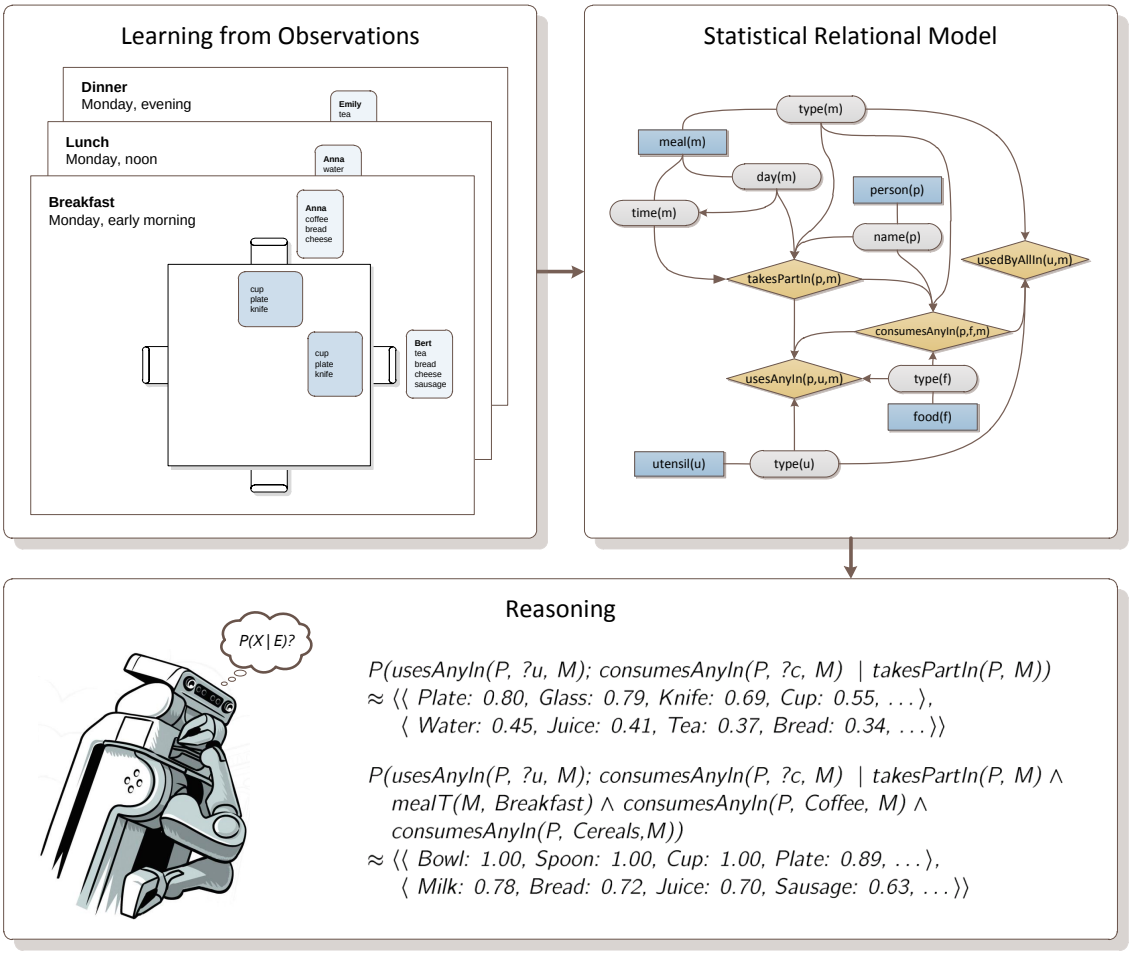
ility theory, statistical relational models combine knowledge representation, learning and reasoning within a single coherent framework.

Through the fusion of principles from first-order logic and probabilistic graphical models, representation formalisms for statistical relational models essentially attain an expressiveness comparable to that of natural language and therefore can equip a technical system with the indispensable capability of reasoning in the light of uncertainty – at higher levels of abstraction. Moreover, a robot system that is able to learn statistical relational models from data that represents the system's own experience in the world can gain substantial capabilities over time: Not only can it learn to appropriately quantify the influence of random variables on one another, using them to make predictions or assess situations, the learnt relationships can also, to a large degree, serve as representations for common sense. For instance, by observing activities such as meals in a kitchen, a robot can determine which types of food and drinks necessitate the use of which objects and utensils, and acquire common sense knowledge about meal habits in general (see Figure 1.2).

The application of statistical relational models is, however, not without its challenges. First of all, it is a prime requirement to find representations that facilitate knowledge engineering, enabling probabilistic knowledge to be captured in the intended way whilst retaining expressiveness and conciseness. Second, both probabilistic reasoning and learning problems are inherently hard, particularly in relational domains where the number of objects under consideration and therefore the number of relevant random variables is potentially large. One typically needs to make simplifying assumptions to render applications practical. Finding an appropriate trade-off between model complexity and faithfulness to real-world constraints is, therefore, a key problem.

1.1 Contributions & Reader's Guide

In this work, I analyse knowledge representation issues that arise in statistical relational models, introduce novel knowledge representation formalisms, present methods for learning and reasoning that make these formalisms practical, and describe several applications in the particular context of technical systems.



Robot illustration © Josh Ellington and Willow Garage

Figure 1.2: *An example application of statistical relational models, which is concerned with meals and corresponding table settings: From a large number of observations that contain information about the objects that were used during various types of meals and the food and drinks that were consumed by people, a statistical relational model can be learnt which captures the particular habits of people and the relationships between, for instance, tool use, food consumption and people's preferences. The learnt model can be used by a robot to infer, for example, the types of utensils and the food that is most likely to be used/consumed by a person P who participates in a meal M for various cases, enabling the robot to assist in setting the table. The two queries compare the results for a case where the robot knows nothing about the meal or the person to a case where the robot knows the type of the meal and has further information about P's consumptions, illustrating the way in which beliefs are affected as additional knowledge is provided. (The query syntax is explained in Section 1.2.)*

Chapter 2 reviews the fundamental knowledge representation formalisms in the realm of logical and graphical models and gives an overview of the key principles underlying statistical relational models, which essentially provide a unifying framework for the two former types of models.

Knowledge Engineering with Markov Logic Networks. Chapter 3 presents a detailed guide to Markov logic networks (MLNs) – a knowledge representation formalism introduced by Richardson and Domingos (2006), which is widely recognised as one of the most general first-order probabilistic languages to have been proposed. By attaching weights to formulas in first-order logic, MLNs combine a logical language with probabilistic semantics, subsuming both probabilistic graphical models and first-order logic for finite domains. However, knowledge engineering with MLNs is not a straightforward task, and one of the contributions of this thesis is an in-depth analysis of MLN semantics and their implications for knowledge engineering practice (Section 3.4). A number of potentially critical issues are identified – some of which are specific to MLNs while others are common to all statistical relational models that adopt template semantics similar to MLNs. In particular, I identify conditions under which the generalisation of MLN models across domains of relevant entities fails, because, notably, combinatorial effects render probabilistic properties of models dependent on the number of entities, such that invariant properties across domains often cannot be adequately maintained.

Markov Logic Networks with Probability Constraints. Section 3.5 presents a novel extension of the MLN formalism, which adds probability constraints on arbitrary formulas to the language, thus addressing the representational problem pertaining to invariant properties mentioned above. With IPFP-M, an algorithm for the integration of such probability constraints into MLN instantiations is presented.

Adaptive Markov Logic Networks. The problem of generalisation across domains can also be dealt with from the perspective of learning. The analysis of knowledge engineering issues suggests that for models to soundly generalise across domains, it can be necessary to adapt certain model parameters depending on the particular instantiation of the model. This is particularly evident in the presence of cardinality constraints that restrict the number of entities a given entity can be related to. Section 3.6 thus introduces adaptive Markov logic networks (AMLNs), an approach

that formalises the adaptation of model parameters to the instantiation at hand by phrasing the parameters of the model as functions over attributes of the instantiation. Learning methods that induce these functions from multiple instantiations are introduced, and the practical benefit of using AMLNs instead of standard MLNs is made clear in a series of experiments on a simple, exemplary scenario. The empirical analysis clearly illustrates the potential severity of the deviations that result from using static parameters on key tasks such as link prediction.

Reasoning and Learning in the Light of Uncertain Evidence. Probability theory can be understood as a calculus for uncertain inference. However, most approaches to probabilistic reasoning assume that the basis for belief updates are plain facts, yet the uncertain nature of, for instance, the perceptual apparatus of a technical system will typically cast inputs to probabilistic reasoning problems as uncertain pieces of information. As another key problem in probabilistic reasoning, this thesis explores the problem of updating beliefs in the light of uncertain evidence. The notion of such belief updates becomes all the more relevant if we assume probabilistic computations to be distributed across individual components of a technical system. The system might trigger probabilistic computations depending on probabilistic results previously obtained, thus requiring it to update beliefs based on uncertain beliefs. Past approaches to this problem have exclusively relied on computationally expensive iterative updating schemes. In Chapter 4, I introduce the first Markov chain Monte Carlo method for the problem of soft evidential update, which, by offering any-time approximations and avoiding time-consuming iterative updates, can be significantly more efficient in practice. Furthermore, I outline a number of methods that adapt the learning of parameters in Markov logic networks to the setting where the training data is subject to uncertainty.

Bayesian Logic Networks. Chapter 5 introduces Bayesian logic networks (BLNs), a novel first-order probabilistic language. BLNs are a type of statistical relational model that is based on the framework of mixed networks with probabilistic and deterministic constraints (Mateescu and Dechter, 2008). As such, BLNs combine local probabilistic dependencies expressed in the form of conditional distributions with the possibility of specifying global logical constraints. The use of directed models in the probabilistic part of the model can render knowledge engineering and learning problems more manageable. Furthermore, instantiations of BLNs can be compiled

into Bayesian networks, thus enabling the large pool of methods that have been developed for this class of models to be leveraged. In particular, the use of importance sampling methods is often a practically viable option. I thus give an overview of sampling-based inference and introduce novel methods that apply the principles of backward search and constraint learning to importance sampling. I furthermore describe a practical method for approximating the quality of the approximations that one can obtain with such schemes, which are particularly relevant in situations where approximations are necessary but solution quality must be appropriately bounded.

Applications. Chapter 6 presents a number of applications of statistical relational models in technical systems. The demonstration scenarios involve two types of technical systems: an autonomous robot that is to assist in kitchen tasks (Beetz et al., 2008) and a cognitive factory that automatically plans and diagnoses its production processes (Zäh et al., 2009). Section 6.1 presents two applications in robot perception: the categorisation of tabletop objects based on relational data from a composite sensor and a prototypical application that manages object-observation associations over time, taking partial observability and non-uniquely identifiable objects into consideration. In Section 6.2, I address the integration of statistical relational models into the control structures of autonomous robots. By transparently integrating probabilistic reasoning into the plan language that is used to control the robot and providing a tight coupling of knowledge processing, probabilistic reasoning and perception, statistical relational models become a valuable cognitive resource. Furthermore, the learning of probabilistic models based on experience data that has been collected by the robot's control program is considered, potentially enabling robots to reason probabilistically about their actions in order to improve their future problem solving behaviour. In Section 6.3, I present an application in the context of a factory that is to assess production plans based on observations made during production. In this context, statistical relational models can be used as a powerful representation formalism for the representation of factory station behaviour and the complex interactions between the production schedule, the products being produced and the state of the factory.

Open-Source Software. Implementations of the methods and applications presented in this work are largely made available to the community as open-source soft-

ware. They are available in the ProbCog toolbox for statistical relational learning and reasoning[1] and the TUM ROS repositories[2] respectively.

1.2 Notation

Random Variables and Assignments. Throughout this work, random variables or sets of random variables are denoted using capital letters such as X. Corresponding lower-case letters such as x are used to refer to (fixed but arbitrary) values within the domain of the respective (set of) random variable(s), denoted as $\mathrm{dom}(X)$. For an assignment $X = x$, x is frequently used as a shorthand, i.e. $P(X = x)$ and $P(x)$ are considered equivalent. Probabilities such as $P(x)$ are defined in terms of a particular model, e.g. M, which is usually clear from context. The more explicit notation $P(x \mid M)$ or $P_M(x)$ is used only if there is ambiguity with respect to the concrete model that applies. Since an assignment of a set of variables X can be viewed as a logical sentence (the conjunction of the true literals that the assignment implies), the entailment operator $\models$ is used to relate assignments to each other (e.g. $X = x \models Y = y$ if y is a projection of x) and to relate assignments to the formulas they satisfy ($X = x \models F$ if F is a logical formula defined over the random variables in X which is satisfied for the particular assignment $X = x$).

Queries. A special notation is used for queries for posterior marginal probabilities in statistical relational models. Queries take the general form $P(\text{query} \mid \text{evidence}) = \langle \text{results} \rangle$. A single query may ask for several marginals, e.g. $P(a(O); b(O) \mid c(O))$ asks for the posterior marginals of $a(O)$ and $b(O)$ separately (not the joint of these variables). For each posterior marginal that is queried, results are presented in a list $\langle v_1 : p_1, \ldots, v_n : p_n \rangle$, where v_i are the elements of the domain of the query random variable and p_i are the associated probabilities. Where not all results are reported, dots $(\ldots)$ are used to indicate omissions. Usually, results will be given in descending order of probability, and only values with lower associated probability will be omitted. A special syntax is used to query the probabilities of true ground instances of a predicate: A query such as $P(\mathit{pred}(X, ?y))$ returns a list $\langle E_1 : p_1, \ldots, E_m : p_m \rangle$, where E_i is an entity/term and $p_i = P(\mathit{pred}(X, E_i))$.

[1] http://ias.cs.tum.edu/software/probcog
[2] http://code.cs.tum.edu

Chapter 2

From Propositional Models to Statistical Relational Models

The pursuit of artificial intelligence calls for a language in which knowledge can be represented, for the adequate representation of knowledge could be considered as the very precondition for intelligence to manifest itself. Agency, i.e. the capacity of an agent to act in a world, depends on it. Deprived of the ability to represent, acquire and store knowledge, there can be no basis upon which to select actions that will serve to achieve an agent's goals (cf. the *knowledge representation hypothesis*, Smith, 1982). Percepts cannot be semantically stored, and past experience cannot be exploited in order to improve the capacity to act competently. Indeed, even the agent's fundamental motivations that cause it to act in the first place may call for an explicit representation.

The explicit representation of knowledge in a *knowledge base* (KB) has significant advantages over an implicit encoding of knowledge in procedures that might be used to control an agent. With a declarative specification, knowledge can flexibly be added or removed. Furthermore, the inner workings of processes that realise the agent's learning and reasoning capabilities can be detached from the data they depend on, facilitating the adaptability of an agent to new and changing circumstances.

2.1 Logic

Gottfried Wilhelm Leibniz pursued the creation of a *lingua universalis* – a formal knowledge representation language that would also be a calculus, an algebra of thought (Blanke, 1996). He also first conceived of the mathematical treatment of

logic, which, while not universal, does possess many desirable properties. As a mathematical formalism that has precise declarative semantics and is supported by algorithmic calculi that enable the automation of learning and reasoning processes, logic is often considered as a natural choice for knowledge representation in artificial intelligence. Indeed, logical approaches often provide a sound formal basis for the development of AI systems, as they can be adapted to many relevant problems. The diversity of AI applications has led to numerous specialised logics, including formalisms for spatio-temporal reasoning, default reasoning, abductive reasoning and reasoning about concepts. In the following, I will give a brief overview of two of the most elemental formalisms, propositional logic and first-order logic, which serve as a foundation for the more intricate probabilistic formalisms that will be considered further on.

2.1.1 Propositional Logic

Propositional logic is the perhaps simplest logical formalism (cf. Russell and Norvig, 2003, ch. 7). The key syntactic elements of propositional logical are *atomic sentences* that are represented by proposition symbols from an alphabet Σ. *Complex sentences* can be constructed from atomic sentences (*atoms*) using the logical connectives for *negation* ($\neg$), *disjunction* ($\vee$), *conjunction* ($\wedge$), *implication* ($\Rightarrow$) and *biimplication* ($\Leftrightarrow$). An atomic sentence or negated atomic sentence is called a *literal* or *unit clause*; a disjunction of literals is called a *clause*. Any atomic or complex sentence is also called a logical *formula*. In an implication $A \Rightarrow B$, A is called the *antecedent* and B is called the *consequent*.

The *semantics* of propositional logic are defined in terms of models. A *model* or *possible world* assigns a truth value (*True* or *False*) to each atomic sentence in Σ. If the set Σ is indexed and the set of truth values is denoted by $\mathbb{B} := \{True, False\}$, a model can be viewed as an assignment $\Sigma = \sigma$, where $\sigma \in \mathbb{B}^{|\Sigma|}$. Given patterns for the evaluation of the logical connectives, a model assigns a truth value to any complex sentence defined over Σ: Negation inverts the truth value; a conjunction is true iff all its parts (conjuncts) are true; a disjunction is true iff at least one of its parts (disjuncts) is true; an implication $A \Rightarrow B$ reduces to $\neg A \vee B$; a biimplication $A \Leftrightarrow B$ reduces to $(A \Rightarrow B) \wedge (B \Rightarrow A)$. If a sentence α evaluates to true in a model $\Sigma = \sigma$, then $\Sigma = \sigma$

is said to *satisfy* α. The (Boolean) *satisfiability problem* (SAT problem) – one of the best-known NP-complete problems – is the problem of determining whether there is at least one model in which a given sentence is satisfied.

As in natural language, there are many ways of making a statement with particular semantics in logic. Two sentences are considered *semantically equivalent* if the set of models in which they are satisfied is the same. To establish regularity in the representation of logical sentences, one can transform any sentence into a normal form. In *conjunctive normal form*, sentences are conjunctions of clauses; in *disjunctive normal form*, sentences are disjunctions of conjunctions of literals.

Logical *inference* involves the determination of *entailment* – the notion of a sentence following logically from another. In propositional logic, a knowledge base (KB) is simply a logical sentence that is known to hold. The problem of logical inference is therefore to determine whether or not other sentences are logically entailed by KB. By definition, a sentence KB entails another sentence α (written $KB \models \alpha$) iff in every model in which KB is true, α is also true. The resolution algorithm (Russell and Norvig, 2003, ch. 7) is a commonly used, complete method for the automated computation of entailment.

2.1.2 First-Order Logic

While propositional logic constitutes a sound knowledge representation language that enables automated reasoning, it is severely limited in its ability to represent knowledge in concise terms, as it essentially contains no notion of abstraction. The set of proposition symbols Σ – if physically grounded – represents a fixed set of statements about entities in the real world. The most important extension that is introduced in first-order logic (FOL) is the provision for generalisation by abstracting away from entity-specific propositions and introducing parametrised propositions, making entities and relations between them first-class citizens of the representation.

The basic syntactic elements of first-order logic (cf. Russell and Norvig, 2003, ch. 8) are *constant symbols* referring to entities, *predicate symbols* referring to relations and *function symbols* referring to (total) functions. A syntactic expression that refers to an entity is called a *term*. Every constant symbol is a term, and the application

of a function, which takes one or more terms as arguments, is also a term. In first-order logic, a *model* defines the *domain of discourse* (or simply *domain*), i.e. the set of entities that are considered, as well as the relations and functional mappings that hold between these entities. An *interpretation* defines how symbols are mapped to actual objects, relations and functions in the model.

An *atomic sentence* is constructed by applying a predicate symbol to a tuple of terms (e.g. $related(A, B)$). For a given model and interpretation, an atomic sentence is considered true/satisfied if the respective tuple of terms is contained in the relation that the predicate symbol refers to. *Complex sentences* can be constructed from atomic sentences using the logical connectives. Furthermore, first-order logic introduces *quantifiers* that enable generalised statements about the entire domain. By introducing *variables* which stand for any element of the domain, quantifiers enable such general statements to be made in a concise manner, lifting the limitations of propositional logic:

- Universal quantification can be used to state that a sentence is true for all elements of the domain (e.g. $\forall x.\ P(x) \rightarrow Q(x)$ states that $P(x) \rightarrow Q(x)$ holds for all substitutions of x with one of the elements of the domain).
- Existential quantification can be used to state that a sentence is true for at least one element in the domain (e.g. $\exists x.\ P(x)$).

Since the variables appearing in quantified statements stand for entities, they are considered terms. A term that contains no variables is a called a *ground term*. Likewise, an atomic sentence that contains only ground terms is called a *ground atom*, and a formula that contains only ground atoms is called a *ground formula*.

Entailment in first-order logic is only *semi-decidable* in the general case. This is due to the fact that the number of ground terms may be infinite. While every sentence that is indeed entailed by a FOL knowledge base will involve only a finite number of ground atoms that need to be considered and methods exist that are guaranteed to produce a finite proof (e.g. first-order resolution; Russell and Norvig, 2003, ch. 9), it can be shown that no algorithms exist that would be capable of proving non-entailment for all sentences that are indeed not entailed by the knowledge base (Turing, 1936).

Extensions of first-order logic that are frequently used in practice involve the use of typed entities and typed predicates, such that the consideration of irrelevant ground atoms can be avoided. Furthermore, the standard binary predicate "=" is frequently used to denote the equality of terms (using infix notation).

Propositionalisation. If the domain of discourse D is finite, a FOL knowledge base can be translated into a propositional knowledge base. For a finite domain D, the set of ground atomic sentences is also finite. Therefore, the set of ground atoms may serve as an alphabet Σ of propositions for a propositional knowledge base. Universal quantification simply translates to conjunction ($\forall x.\ P(x) \rightsquigarrow \bigwedge_{x \in D} P(x)$) and existential quantification translates to disjunction ($\exists x.\ P(x) \rightsquigarrow \bigvee_{x \in D} P(x)$).

2.2 Graphical Models

Although classical logics provide a flexible framework for the representation of knowledge, they essentially ignore the fact that, under real-world conditions, knowledge is, to a large degree, subject to uncertainty. For the representation of uncertain knowledge, the class of graphical models is frequently used as a representational paradigm.

Graphical models are convenient representations for knowledge about the dependencies between variables that typically correspond to some aspect of the real world. Indeed, most aspects of the world can – given a sufficient level of abstraction – adequately be modelled as interacting variables. Graphical representations of such interactions are highly desirable. For humans, they intuitively depict dependencies as edges between nodes in graphs that are easily visualised, facilitating comprehension and, in turn, the process of modelling the knowledge in question. For technical systems, they make the structure of knowledge explicit, consequently allowing algorithms to exploit this structure in order to gain efficiency.

Probabilistic graphical models, such as Markov random fields and Bayesian networks, are widely used representations for statistical knowledge, i.e., in a nutshell, models that represent probability distributions over random variables in a concise manner by exploiting independencies. In purely logical domains, graphical models called constraint networks are common representations for the structure of constraint satisfaction problems. At the intersection of logical and probabilistic domains, the notion of

mixed networks explicitly combining both approaches within a single model has appeared. Although probabilistic graphical models are capable of representing logical knowledge within the probabilistic framework, a more explicit representation that clearly separates statistical and logical knowledge can be desirable.

Graphical models provide a solid foundation for the more general and expressive formalisms that will be presented further on, for they are well-understood theoretically and are supported by a wide range of practically applicable methods. In the following, I therefore review the fundamentals underlying all of the aforementioned graphical representation formalisms – probabilistic, logical as well as mixed.

2.2.1 Markov Networks

As a type of probabilistic graphical model, a Markov network is a compact representation of a full-joint distribution over a set of random variables (Pearl, 1988). Compactness is achieved through an exploitation of conditional independencies, which, in turn, allow to independently consider mere subsets of the set of variables in order to define the distribution as a whole.

A particularly intuitive way of breaking down a large joint probability distribution into a set of smaller components is to consider, within local views, relative importance measures that concern only a subset of the random variables and to combine these in order to obtain a global probability measure. Locally, we can simply assign a (non-negative) numeric value to each particular configuration of the variables being considered. This value is to be proportional to the degree to which that configuration appears – with respect to the local view – and will thus be meaningful only in relation to other such values.

The variables we consider within a local view immediately imply a graphical structure: If two variables appear within a local view, we connect them through an edge. The framework of Markov networks formalises the way in which a full-joint probability distribution can be specified based on such local views and defines the relationship between conditional independencies and the graphical structure that is implied by the local views under consideration.

Definiton. Formally, a (discrete) *Markov network* or *Markov random field* (MRF) is a tuple $M = \langle X, D, G, \phi \rangle$ representing a joint probability distribution over a set of random variables $X = \{X_1, \ldots, X_N\}$ with corresponding (discrete) domains $D = \{D_1, \ldots, D_N\}$. The Markov network's undirected graph $G = \langle X, E \rangle$ contains one node for every random variable in X, and its set of edges $E \subseteq \{\{X_i, X_j\} \mid X_i, X_j \in X, i \neq j\}$ indicates dependencies between random variables. G is called the *structure* of the model. Since the graph is undirected, MRFs belong to the class of *undirected graphical models*.

The quantitative aspects of the distribution represented by M are given by a set of *potential functions* ϕ, where each potential function $\phi_k \in \phi$ maps from the configuration of a clique (fully connected subgraph) in G to the non-negative real numbers, i.e. if $C_k = \{X_{i_1}, \ldots, X_{i_{N_k}}\}$ is the k-th clique in G, then $\phi_k : D_{i_1} \times \cdots \times D_{i_{N_k}} \to \mathbb{R}_0^+$. The combination of all the ranges of the functions ϕ_k constitutes the model's set of *parameters*.

Probabilistic Semantics. Let the set of *possible worlds*, i.e. the set of possible assignments of random variables to values, be denoted by $\mathcal{X} := \mathrm{dom}(X) = \prod_{i=1}^{N} D_i$. M represents a distribution over $\mathcal{X}$ as follows,

$$P(X = x) = \frac{1}{Z} \prod_{k=1}^{|\phi|} \phi_k(x_{\{k\}}) \tag{2.1}$$

where $x_{\{k\}}$ is the projection of x that corresponds to the state of the k-th clique in G. Z is a normalisation constant and is given by $Z = \sum_{x \in \mathcal{X}} \prod_k \phi_k(x_{\{k\}})$.

Structural Properties. From this definition, the independence relationships that are imposed by a structure G can be derived (Pearl, 1988, ch. 3): Two random variables X_i and X_j are conditionally independent given $S \subset X \setminus \{X_i, X_j\}$ iff X_i and X_j are connected only via paths that lead through nodes in S. Hence conditional independence in a Markov network can be determined via the topological criterion of *graph separation*: If two sets of nodes become disconnected following the removal of S from the graph, then these sets of nodes are independent given S.

The independencies implied by a graph G determine the set of probability distributions that can potentially be represented by a Markov random field that uses the struc-

ture G (Pearl, 1988, ch. 3). Any probability distribution defines a *dependency model Dep* that determines the ternary predicate $I(A,S,B)$: $I(A,S,B)$ is true for disjoint subsets A,B,S of X iff A is independent of B given S. An undirected graph $G = \langle X, E \rangle$ is an *independency map* (I-Map) of *Dep* if for all disjoint subsets $A,B,S \subset X$, the following statement is true: If A and B are separated in G after removing the nodes in S, then A and B are conditionally independent given S, i.e. $I(A,S,B)$. A Markov random field M is capable of representing a probability distribution with dependency model *Dep* iff its graph G is an I-Map of *Dep*. Typically, we seek the most compact representation and require G to be a *minimal I-Map* of *Dep*, i.e. an I-Map out of which no further edges could be removed without giving up the I-Map property. Consequently, we can construct the topology of a Markov random field by connecting a node (random variable) to the nodes (random variables) that make up its Markov blanket, the *Markov blanket* of a random variable X_i being defined precisely as the set $MB(X_i)$ of random variables that must be given for X_i to be independent of all other random variables $X \setminus MB(X_i) \setminus \{X_i\}$.

Canonical Inference Problems. A probabilistic model that represents a full-joint distribution over a set of random variables can be flexibly used by an intelligent agent. As an agent acquires new pieces of information, its beliefs regarding the probabilities of assignments of particular random variables will change, necessitating a *belief update*. One of the most common inference tasks is, therefore, the computation of *posterior marginals*, i.e. for disjoint subsets $Q \subset X$ and $E \subset X$, the computation of the conditional probability distribution over $\mathrm{dom}(Q)$ (the query) given a known assignment $E = e$ (the evidence). With $U := X \setminus E \setminus Q$, the posterior probability of an assignment $Q = q$ can be computed as

$$\begin{aligned} & P(Q = q \mid E = e) \\ &= \frac{P(Q = q, E = e)}{P(E = e)} = \frac{\sum_{u \in \mathrm{dom}(U)} P(Q = q, E = e, U = u)}{\sum_{u \in \mathrm{dom}(U)} \sum_{q' \in \mathrm{dom}(Q)} P(Q = q', E = e, U = u)} \end{aligned} \tag{2.2}$$

For instance, the variables Q could correspond to the effects of an action and the assignment $E = e$ could correspond to the parametrisation of the action and the situation in which it is performed, such that the inference results would correspond to the prediction of the action's effects.

Inversely, an agent might use a model of action effects to determine which particular action parametrisation is most likely to bring about a particular desired effect by supplying the effect as evidence and asking for the most probable assignment of the variables that correspond to the action parametrisation. Therefore, another canonical problem is to determine the most probable assignment to a subset $Q \subset X \setminus E$ given an assignment $E = e$, i.e. the computation of the *maximum a posteriori (MAP) hypothesis*. With $U = X \setminus E \setminus Q$, it is given by

$$\arg\max_{q \in \mathrm{dom}(Q)} \sum_{u \in \mathrm{dom}(U)} P(Q = q, U = u, E = e). \tag{2.3}$$

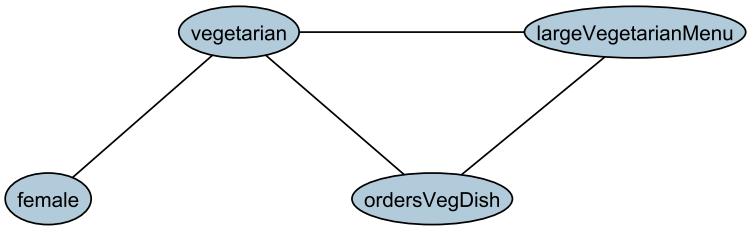
A related inference problem is the computation the possible world which, given the evidence $E = e$, has the highest probability, i.e. the determination of the *most probable explanation* (MPE) of the evidence:

$$\arg\max_{x \in \mathcal{X},\ x \models E = e} P(X = x) \tag{2.4}$$

MPE inference is the special case of MAP inference where $Q = X \setminus E$. In the general case, the MAP inference problem can, unfortunately, be hard even for cases where MPE inference and posterior marginals are easy to handle (Park and Darwiche, 2004).

Naive Inference. As indicated by Equations 2.2, 2.3 and 2.4, inference problems can be reduced to individual probabilities of possible worlds that are computable according to Equation 2.1. The naive inference algorithm that directly applies this principle, enumerating all (relevant) possible worlds as necessary, is called *Enumeration-Ask* in the literature (Russell and Norvig, 2003, ch. 14). More efficient and practical methods will be described in Chapter 3.

Example. As an example, consider the Markov random field in Figure 2.1, which deals with a simple scenario where a person goes to a restaurant and orders a dish. The model considers the property of being female and of being a vegetarian, and furthermore considers the variety of vegetarian dishes on offer in the restaurant. All random variables are Boolean. The model contains two clique potentials, one connecting *female* and *vegetarian*, the other connecting *vegetarian*, *largeVegetarianMenu* and *ordersVegDish*. From the topology, we can, for example, conclude that *female*



$\phi_1 : \text{dom}(\textit{vegetarian}) \times \text{dom}(\textit{female}) \to \mathbb{R}_0^+$
$\phi_1(\textit{True}, \textit{True}) = 2.333$
$\phi_1(\textit{True}, \textit{False}) = 1.000$
$\phi_1(\textit{False}, \textit{True}) = 8.783$
$\phi_1(\textit{False}, \textit{False}) = 10.106$

$\phi_2 : \text{dom}(\textit{vegetarian}) \times \text{dom}(\textit{largeVegetarianMenu}) \times \text{dom}(\textit{ordersVegDish}) \to \mathbb{R}_0^+$
$\phi_2(\textit{True}, \textit{True}, \textit{True}) = 10$
$\phi_2(\textit{True}, \textit{True}, \textit{False}) = 0$
$\phi_2(\textit{True}, \textit{False}, \textit{True}) = 90$
$\phi_2(\textit{True}, \textit{False}, \textit{False}) = 0$
$\phi_2(\textit{False}, \textit{True}, \textit{True}) = 2$
$\phi_2(\textit{False}, \textit{True}, \textit{False}) = 9$
$\phi_2(\textit{False}, \textit{False}, \textit{True}) = 9$
$\phi_2(\textit{False}, \textit{False}, \textit{False}) = 81$

Figure 2.1: *A Markov random field*

is independent of both *ordersVegDish* and *largeVegetarianMenu* given the value of *vegetarian*.[1]

The clique potentials characterise the quantitative aspects of the distributions. For example, the clique potential ϕ_1 on its own indicates that the case where a person is not a vegetarian and is male is approximately 10.1 times as likely as the case where the person is a vegetarian and is male. If *vegetarian* is observed, we can compute the degree to which we should believe a person to be female using ϕ_1 only: $P(\textit{female} = \textit{True} \mid \textit{vegetarian} = \textit{True}) = 2.333/(2.333 + 1.000) \approx 0.70$. Using both potentials, we can compute the values associated with individual possible worlds. For instance, consider the following two worlds:

	vegetarian	*female*	*largeVegMenu*	*ordersVegDish*	$\prod_{k=1}^{M} \phi_k(x_{\{k\}}^{(i)})$
$x^{(1)}$	*True*	*False*	*False*	*True*	$1.000 \cdot 90 = 90$
$x^{(2)}$	*True*	*False*	*True*	*True*	$1.000 \cdot 10 = 10$

The associated values indicate that world $x^{(1)}$ is nine times as probable as $x^{(2)}$. To determine the actual probability values, however, the normalisation constant Z would need to be computed by summing the values for all 16 possible worlds.

[1] One might question this independency: If, for instance, non-vegetarian females tend to select vegetarian dishes more frequently than non-vegetarian males, the model structure would need to be changed to reflect this dependency by adding additional edges.

In this tiny example model, an application of Enumeration-Ask is, of course, feasible. For example, the following probabilities can be derived from the model:

$$P(female = True) = 0.50$$
$$P(female = True \mid vegetarian = True) = 0.70$$
$$P(vegetarian = True) = 0.15$$
$$P(vegetarian = True \mid female = True) = 0.21$$
$$P(vegetarian = True \mid ordersVegDish = True) = 0.62$$
$$P(ordersVegDish = True) = 0.24$$
$$P(ordersVegDish = True \mid vegetarian = True) = 1.00$$
$$P(ordersVegDish = True \mid vegetarian = False) = 0.11$$
$$P(ordersVegDish = True \mid largeVegMenu = True) = 0.29$$
$$P(ordersVegDish = True \mid female = True, largeVegMenu = True) = 0.34$$

Factor Graphs. It is sometimes convenient to represent a Markov random field as a *factor graph*, i.e. as a bipartite graph in which the potentials (factors) are explicitly represented as nodes in the graph. Random variables are not connected to each other but to factor nodes which represent the potentials. Figure 2.2 shows the factor graph representation of the MRF in Figure 2.1 with two factor nodes (squares) – for ϕ_1 and ϕ_2 respectively.

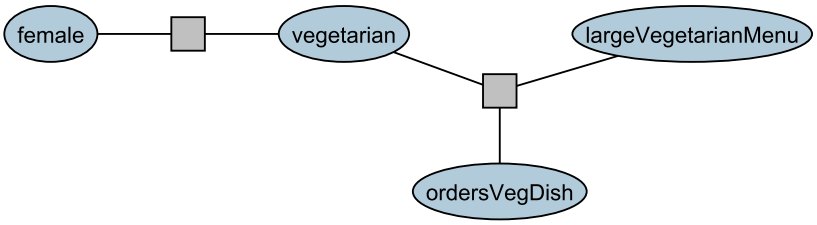


Figure 2.2: *The factor graph representation of the Markov random field in Figure 2.1*

Log-Linear Models. As an alternative to potential functions, a Markov network can be represented as a *log-linear model* $\langle X, D, G, f, w \rangle$, where X, D and G are defined as above and the potential functions ϕ_i are replaced by weighted features, a feature being a property that is either present or absent in a possible world $x \in \mathcal{X}$. Thus

f is a vector of *feature* functions $f_i : \mathscr{X} \to \{0,1\}$, and w is the associated vector of real-valued weights. The distribution over $\mathscr{X}$ is given by

$$P(X = x) = \frac{1}{Z} \prod_i \exp\left(w_i \cdot f_i(x)\right) = \frac{1}{Z} \exp\left(\sum_i w_i \cdot f_i(x)\right). \tag{2.5}$$

Note that any standard Markov network can be converted to an equivalent log-linear model provided that all $\phi_k(x_{\{k\}})$ are strictly positive: We only need to define one feature f_i with weight $w_i = \log(\phi_k(x_{\{k\}}))$ for each possible state $x_{\{k\}}$ of each clique such that $f_i(x) = 1$ if $X = x \models C_k = x_{\{k\}}$ and 0 otherwise.

The main appeal of this type of representation is due to computational reasons. Furthermore, the use of features may yield a very compact representation if the potential functions ϕ_k are not injective, since several cases can then be handled by a single feature.

2.2.2 Bayesian Networks

Bayesian networks are another commonly used representation formalism for uncertain knowledge. In contrast to a Markov random field, a Bayesian network represents a normalised, direct factorisation of the full-joint distribution over a set of random variables X as a product of conditional distributions. As in MRFs, conditional independencies are exploited to achieve compactness. To make conditioning explicit in the graphical representation, Bayesian networks use directed edges. This renders the representation of causal influence particularly intuitive, for a causal influence of a variable X_i on a variable X_j can simply be represented in a directed edge from X_i to X_j along with the conditional distribution $P(X_j \mid X_i)$.

Definition. Formally, a Bayesian network is a tuple $B = \langle X, D, G, P \rangle$, where $X = \{X_1, \ldots, X_n\}$ is an indexed set of random variables, $D = \{D_1, \ldots, D_n\}$ is the corresponding set of domains, $G = \langle X, E \rangle$ is a directed acyclic graph representing a dependency structure over the variables X, and $P = \{P_1, \ldots, P_n\}$ is a set of (conditional) probability distributions with $P_i = P(X_i \mid \mathrm{Par}_{X_i})$, where Par_{X_i} denotes the set of *parents* of X_i in G, i.e. the nodes from which X_i can be reached through a single directed edge. *Children*, *ancestors* and *descendants* are defined according to the same analogy.

The structure of a Bayesian network is given by the graph G, its parameters are given by the probability values in the set of conditional distributions P.

Probabilistic Semantics. Like a Markov random field, a Bayesian network represents a full-joint distribution over possible worlds. Specifically, B represents a probability distribution over $\mathscr{X} = \mathrm{dom}(X)$ as a product of entries in P: For all $x \in \mathscr{X}$,

$$P(X = x) = \prod_i P(X_i = x_i \mid \mathrm{Par}_{X_i} = \mathrm{par}_{X_i}^x) \tag{2.6}$$

where $\mathrm{par}_{X_i}^x$ is the assignment of X_i's parents in x.

Structural Properties. The independencies that are implied by the factorisation and therefore by the graphical structure are somewhat more difficult to determine in a Bayesian network than in a Markov random field. A topological criterion called *d-separation* (Russell and Norvig, 1995, ch. 15) characterises the conditions under which (sets of) variables are independent: Two disjoint sets of random variables A and B are conditionally independent given $S \subset X \setminus A \setminus B$ if all paths leading from a node in A to a node in B (ignoring the direction of edges) are d-separated. A path is d-separated if any of the nodes X_i along the path satisfies one of the following conditions:

(i) $X_i \in S$ and X_i has one ingoing and one outgoing edge.

(ii) $X_i \in S$ and X_i has two outgoing edges.

(iii) $X_i \notin S$, $\mathrm{descendants}(X_i) \cap S = \emptyset$ and X_i has two ingoing edges.

In particular, this implies that a node is independent of all its ancestors given its parents (according to condition (i)). Furthermore, nodes that do not have any parents are unconditionally independent (according to condition (iii)). Taking the implications of all three conditions into consideration, it follows that the Markov blanket of a node X_i in a Bayesian network is given by X_i's parents, X_i's children, and the parents of X_i's children: Given these nodes, X_i is independent of all other nodes in the network.

Example. As an example, consider the Bayesian network in Figure 2.3, which represents precisely the same distribution as the Markov random field in Figure 2.1.

Because the Bayesian network uses conditional and unconditional probabilities as parameters, it makes explicit some of the probabilities that demanded inference calculations in the Markov random field. It also reveals an independency that was not evident from the Markov network's structure alone (but could have been derived by inspecting its parameters): *vegetarian* and *largeVegetarianMenu* are unconditionally independent. Note that this might constitute a simplifying modelling assumption, for there may be reason to believe that vegetarians are more likely to select restaurants with large vegetarian menus. Furthermore, the model also contains an instance of so-called *context-specific independence*: Because the conditional distribution of *ordersVegDish* is the same for vegetarians regardless of the degree to which the menu is vegetarian, *ordersVegDish* is called context-specifically independent of *largeVegetarianMenu* given *vegetarian* $=$ *True* (the context).

The Complexity of Exact Inference. Bayesian networks are commonly used for the same types of probabilistic inference tasks as Markov random fields. Depending on the problem, however, the representation using directed edges may be beneficial. As in the case of Markov random fields, the naive enumeration of possible worlds is seldom practical. Since the number of possible worlds is exponential in the number of random variables, exact probabilistic inference is inherently hard, for in the worst case, the entire set of possible worlds will have to be considered explicitly. Efficient inference methods typically seek to exploit model structure or make use of approximations to render computations practical. The exact computation of posterior marginals is an NP-hard problem in general, for the NP-complete Boolean satisfiability problem can be reduced to it (the original proof by Cooper, 1990, uses a reduction from ver-

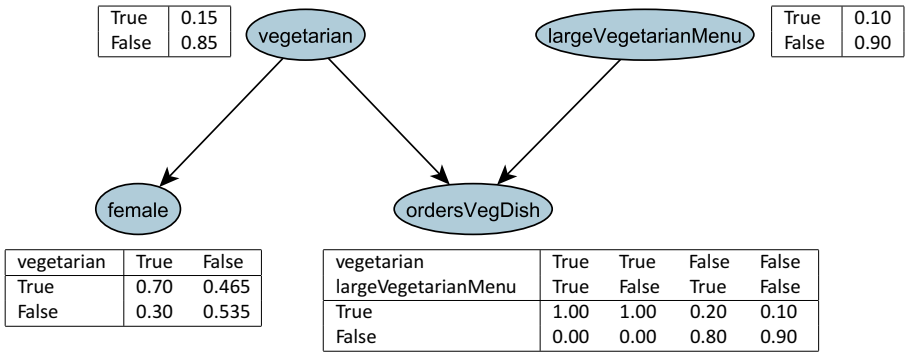


Figure 2.3: *A Bayesian network*

tex cover). In fact, the problem can even be shown to be #P-hard, for the problem of counting SAT solutions can also be reduced to probabilistic inference (Roth, 1996).

2.2.3 Constraint Networks

Graphical models can also be helpful for the representation deterministic knowledge in order to visualise dependencies and exploit problem structure. A *constraint network* is a tuple $R = \langle X, D, C \rangle$, where X is a set of variables, D is the corresponding set of variable domains, and $C = \{C_i\}$ is a set of constraints (Russell and Norvig, 2003, ch. 5). Each constraint $C_i \in C$ is a pair $\langle S_i, R_i \rangle$, where $S_i \subseteq X$ is the scope of the constraint, and $R_i \subseteq \prod_{X_j \in S_i} D_j$ is a relation denoting the allowed combinations of values. In its entirety, a constraint network R is a representation of the set of assignments to X satisfying all constraints. The graphical representation of a constraint network R is given by an undirected *constraint graph* $G = \langle X, E \rangle$, where $E = \{\{X_i, X_j\} \subseteq X \mid \exists \langle S_i, R_i \rangle \in C.\ \{X_i, X_j\} \subseteq S_i\}$, i.e. variables that co-occur in one of the constraints in C are connected through an edge.

In practice, the specification of the constraints' relations is typically implicit. Many languages for the concise definition of constraints (and of additional problem-specific knowledge) have been proposed in the field of constraint programming (Van Roy et al., 2003; Marriott et al., 2008). In general, the problems thus specified are referred to as *constraint satisfaction problems* (CSPs). Boolean satisfiability problems are an important special case in which the constraints are specified as formulas in propositional logic.

2.2.4 Mixed Networks

Although it is possible to represent deterministic/logical constraints within a probabilistic model, it can be desirable to separate probabilistic from deterministic knowledge. The framework of mixed networks (Mateescu and Dechter, 2008) offers a sound way of combining statistical knowledge with axiomatic, deterministic knowledge. A *mixed network* M is a pair $\langle B, R \rangle$, where $B = \langle X, D, G, P \rangle$ is a Bayesian network representing the joint probability distribution $P_B(X = x)$ and $R = \langle X, D, C \rangle$ is a

constraint network with set of solutions ρ. The joint probability distribution given by the mixed network M is

$$P_M(X = x) = \begin{cases} \frac{P_B(x)}{\sum_{x' \in \rho} P_B(x')} & \text{if } x \in \rho \\ 0 & \text{otherwise} \end{cases} \tag{2.7}$$

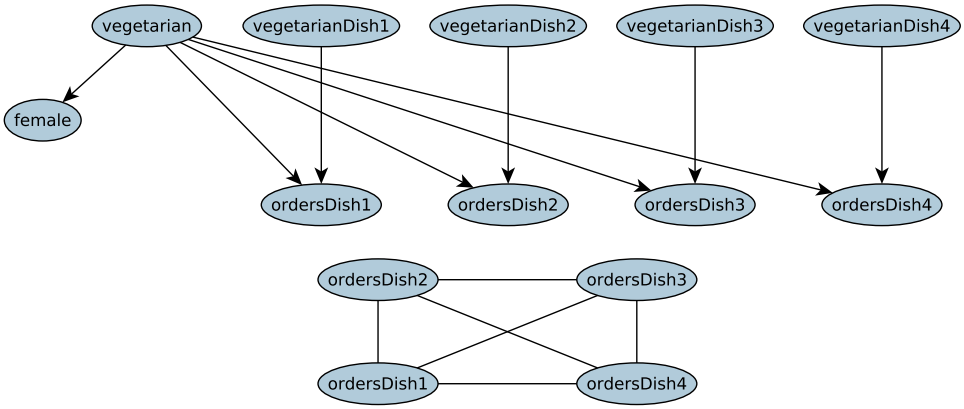
Assignments that do not satisfy the constraints in R are thus phased out and the remainder of the distribution represented by B is renormalised.

Auxiliary Bayesian networks. Any mixed network $M = \langle B, R \rangle$ can be translated into an equivalent Bayesian network B', called the *auxiliary Bayesian network* (Mateescu and Dechter, 2008). $B' = \langle X', D', G', P' \rangle$ is constructed from $B = \langle X, D, G, P \rangle$ by extending each of its elements as follows: For every constraint $C_i = \langle S_i, R_i \rangle \in C$, a Boolean auxiliary variable A_i is added to X' that represents the corresponding constraint and has as its parents in G' all the nodes that the constraint depends on (i.e. $\text{Par}_{A_i} = S_i$). The conditional probability table P'_{A_i} associated with A_i contains (as its entry for the value *True*) 1 for every configuration of parents contained in R_i and 0 for all other configurations. Since all constraints are required to be satisfied, the auxiliary variables are always conditioned on; all auxiliary variables are required to take on a value of *True*. For $|C| = k$, the following equivalence holds:

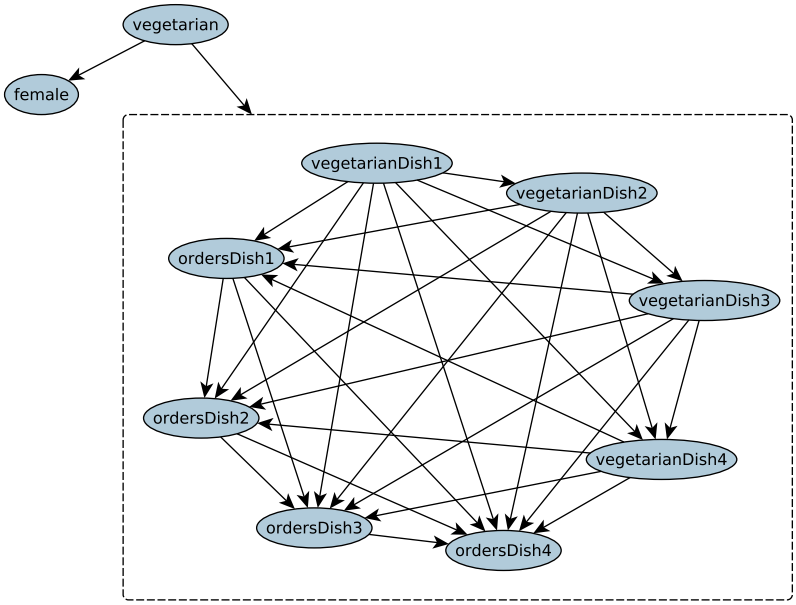
$$P_M(X = x) = P_{B'}(X = x \mid A_1 = \mathit{True}, \ldots, A_k = \mathit{True}) \tag{2.8}$$

Structural Properties. The independencies implied by a mixed network's structure can be determined via reduction to the auxiliary Bayesian network, in which d-separation can be applied. Notably, any constraint on a set of variables in a mixed network induces manifold dependencies among the ancestors and the descendants of the ancestors of the constrained variables (cf. d-separation condition (iii)).

Example. As an example, consider an adaptation of the restaurant scenario: Assume that the main course consists of two separate servings which patrons can freely select from a menu. The selection of a dish depends on the patron being a vegetarian and the dish being vegetarian. In the mixed network shown in Figure 2.4a, the conditional distributions that express the ordering preferences are the same for each of the *ordersDish* variables. In the Bayesian network part, the *ordersDish* variables are



(a) Mixed Network



(b) Dependency-Preserving Bayesian Network

Figure 2.4: *A mixed network and a corresponding Bayesian network, which preserves all the dependencies induced by the mixed network's constraints. The mixed network (a) is comprised of a Bayesian network (top) and a constraint network (bottom). In sub-figure (b), the edge pointing at the dotted box indicates that* vegetarian *is connected to all the nodes within the box.*

independent given *vegetarian*. However, the constraint network, which requires exactly two of the four *ordersDish* variables to be true, renders them dependent, and induces many further dependencies in the rest of the network. This is illustrated by Figure 2.4b, which shows a Bayesian network structure (without auxiliary nodes) that preserves all the dependencies implied by the mixed network. Note that this rep-

resentation is considerably less compact; it even contains a fully connected subgraph encompassing eight of the variables (dotted rectangle).[2] Furthermore, the mixed network's Bayesian network contained several conditional distributions that were the same for different variables and therefore could be easily generalised; yet the network in Figure 2.4b does not retain this property.

2.2.5 Sequence Models

The graphical models considered thus far support reasoning over a fixed set of random variables. For a given type of situation, these random variables can ideally be mapped to observable facts and queries that are relevant to the application. Many problems, however, involve a continuous stream of observations, which is to be interpreted in its entirety. For instance, a robot may want to estimate its state (e.g. its position in the environment) based on a stream of range sensor readings, or, in a dialogue with a human, it may want to discern the most likely phonemes being uttered from a stream of auditory features.

For such problems, the representation of a fixed set of propositions that can be reasoned about is insufficient; graphical models need to be extended to support streams of observations, yielding specifically adapted sequence models. Commonly used sequence models use discrete time. They typically represent prior beliefs about the beginning of the sequence, a probabilistic transition model from previous time steps to subsequent ones, and they include a model that associates observations with other random variables that correspond to states of the world. They thus generalise to state sequences of arbitrary length simply by reusing the same probabilistic templates for each time step.

Dynamic Bayesian networks (DBNs) were introduced as a general adaptation of Bayesian networks to dynamic, sequential reasoning problems (Dean and Kanazawa, 1989).

[2] The network topology in Figure 2.4b was created by considering only the graphical representation of the mixed network, not its parameters. If one takes the specific semantics of the constraint into consideration (as represented in the parameters), one could remove some of the directed edges. For instance, the variable *ordersDish4* unnecessarily depends on the four *vegetarianDish* variables. Given the truth values of the other *ordersDish* variables, its value is fully determined by the constraint: If two dishes have already been selected, dish four will not be ordered; otherwise, if only one of the other dishes has been selected, it will have to be ordered.

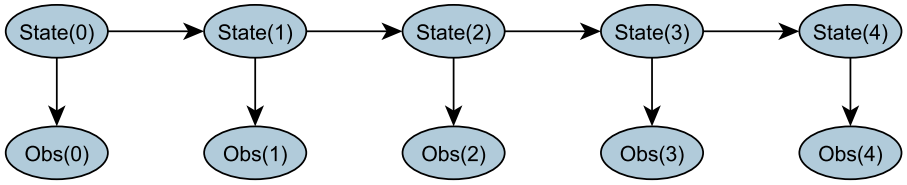


Figure 2.5: *Instantiation of a hidden Markov model for four time steps. The variable* State(0) *uses the model's distribution for the initial state, the other four state variables all share the same transition model. Similarly, all the observation variables share the same observation model, i.e. they use the same conditional distribution.*

Every instantiation of a DBN for a particular set of time steps corresponds to a Bayesian network. Specialisations of DBNs called *hidden Markov models* (HMMs), which historically precede DBNs (Baum and Petrie, 1966), are most frequently used in practice. A hidden Markov model is restricted to a single observation variable and a single hidden (unobservable) state variable per time step (see Figure 2.5). The term *Markov model* is due to the first-order *Markov assumption* being made in the sequence of hidden states, which renders a state conditionally independent of earlier states and observations given its immediate predecessor state. Because of this structural limitation, canonical inference problems in HMMs (e.g. finding the most likely sequence of hidden states given a sequence of observations) can be solved in time linear in the number of time steps via dynamic programming (Russell and Norvig, 2003, ch. 15).

Undirected models have also been extended for the sequential setting. The undirected counter-part of a hidden Markov model is a linear-chain *conditional random field* (CRF, Lafferty et al., 2001). In contrast to an HMM, a CRF does not represent a full-joint distribution, i.e. it does not represent a *generative model* that could be used to sample/generate possible worlds from. Instead, it represents only the conditional distribution of the hidden states given the observations, which is sensible, because in the corresponding application contexts, there is a strict separation between observable and unobservable variables. Training a model to represent only the conditional distribution that is indeed required by the application leaves learning methods free to focus on the relevant dependencies; dependencies among observed input variables can be disregarded. This also makes CRFs attractive for non-sequential tasks – in particular classification tasks, where the focus is on discrimination between classes

(Sutton and McCallum, 2007). For this reason, models that represent an application-specific conditional distribution are frequently referred to as *discriminative models*.

2.3 Statistical Relational Models

Both logic and graphical models have obvious limitations. The language of logic is too limited to deal with many real-world phenomena, for it lacks the ability to adequately represent uncertain beliefs. Indeed, the need for a logic that would take degrees of probability into consideration was already acknowledged by Leibniz (Hailperin, 1984). The main limitation of standard probabilistic graphical models is that the propositions over which probability distributions are represented is fixed. Sequence models introduce quantification over time steps to enable reasoning about dynamic processes, partly lifting this restriction. However, time is not the only domain over which a quantification is desirable. An AI system such as an autonomous robot acting in human environments is constantly faced with situations in which a variable number of objects may be relevant to its tasks, and these objects may be related in various ways. The influence of such relational structures involving variable sets of objects on the propositions relevant to the robot's tasks clearly cannot be accounted for through a model that involves a fixed set of propositions. What is needed is, therefore, a unification of the principles of first-order logic, which makes objects and relations first-class citizens of the representation, and probabilistic graphical models, which enable reasoning in the face of uncertainty, within a single representation formalism. By combining the respective semantics, we can obtain a language that possesses a level of expressiveness that is sufficient for robots to be equipped with the much-needed capability of reasoning about situations as they arise in real-world applications.

2.3.1 Early Probabilistic Logics

Nilsson's Probabilistic Logic. Early approaches towards the realisation of a probabilistic logic did not build upon the principles of graphical models. The first true generalisations of relational logic that took uncertainty into consideration date back to at least Hailperin (1984) and Nilsson (1986). Specifically, Nilsson presented a logic

in which the truth values of first-order sentences could range between zero and one. He addressed the problem of probabilistic entailment, i.e. the problem of determining the probability of a query sentence S given a knowledge base of sentences $\mathscr{S}$ with known associated probabilities, by framing it as a linear programming problem: Let Π be the column vector of known probabilities of sentences in $\mathscr{S}$. The probabilistic logic assumes possible worlds semantics, i.e. the individual sentence probabilities are to collectively determine a distribution over (equivalence classes of) possible worlds, where each possible world corresponds to a different assignment of truth values to the sentences in $\mathscr{S} \cup \{S\}$. Only possible worlds that are logically consistent are considered (e.g. if $\mathscr{S}$ contains the sentences A and $A \Rightarrow B$, then worlds in which both A and $A \Rightarrow B$ are false are considered impossible). Let X be the column vector of possible world probabilities, which is not known a priori, and let V be the $|\mathscr{S}| \times |X|$ matrix of truth values of sentences in $\mathscr{S}$ for the logically consistent possible worlds (*True* being represented as 1 and *False* as 0). The linear program is then given by $\Pi = VX$ subject to the constraint that entries of X range between 0 and 1. The constraint $\sum_i X_i = 1$ can be accounted for by including in $\mathscr{S}$ a tautology which is true in all possible worlds. For every solution X, the probability of the query sentence S can be computed as $V_S P$, where V_S is the row vector of truth values of the sentence S in all possible worlds. However, the equation system is typically underdetermined, permitting many solutions for X, in which case the solutions that provide lower and upper bounds on $P(S)$ are considered. The result of the query then is the corresponding interval. Unfortunately, as pointed out by Nilsson, inference in his probabilistic logic becomes intractable even for a small number of sentences under consideration.

Random Worlds and the Language $L^{\approx}$. Another language that integrates first-order logic and degrees of belief was developed by Bacchus, Grove, Halpern, and Koller (1996). In contrast to Nilsson's probabilistic logic, their language $L^{\approx}$ does not assume possible worlds semantics where worlds are assigned probabilities. Instead, it uses the *random worlds method*, which is based on the principle of indifference: all the worlds that are considered possible are treated as equally probable. Probabilities of sentences are therefore computed by determining relative frequencies of random worlds satisfying the respective sentences. As a fundamental language construct, the language combines standard first-order logic with proportion expressions that make statistical statements about the proportion of domain elements satisfying a given con-

dition. Such proportion expressions can be compared using *approximate* equality and inequality, which, in particular, enables the language to easily incorporate default reasoning (e.g. the conditional proportion expression $||flies(x) \mid bird(x)||_x \approx 1$ states that birds usually fly while still allowing for exceptions) and furthermore enables statistical statements to be made without imposing multiplicities (e.g. the statement $||bird(x)||_x \approx 1/4$ avoids the requirement of having a universe of objects in which the number of objects must be a multiple of four). For a given possible world, the truth of a proportion expression can be determined by systematically reducing conditional proportion expressions and approximate operators to computable expressions in a simpler language – specifically, a language that is analogous to the one described by Halpern (1990). The probability of a query sentence/expression ϕ given a knowledge base of sentences KB can then be computed as the ratio between the number of models that satisfy both ϕ and KB in relation to the models satisfying only KB in the limit of an infinite number of domain elements. As in the case of Nilsson's language, the practical applicability of $L^{\approx}$ is limited.

2.3.2 The Principles of Template Models

In the late 1990s, the trend shifted towards first-order probabilistic languages that integrate principles from first-order logic and probabilistic graphical models alike in order to render applications practical. A key idea that is common to most of these languages is to make the language explicitly take objects and relations into consideration and to rigorously apply the principle of universal quantification. The models represented using these languages do not correspond to probability distributions but should rather be viewed as meta-models which represent general principles that can be applied to obtain a probabilistic model – represented as a graphical model – for a given domain of discourse. Models can thus be viewed as blueprints or templates for the construction of graphical models. The process of generating a low-level graphical model from a higher-level language specification is (particularly in the context of directed models) frequently called *knowledge-based model construction* and dates back to Horsch and Poole (1990) and Breese (1992). The models thus generated are referred to as *ground models*, for they result from a propositionalisation of the first-order theory, yielding a grounded representation in which placeholders are substituted with concrete entities from the domain of discourse. Within the context of this work, the

class of first-order probabilistic models that adopt this principle is subsumed under the broad term *statistical relational models*.

Generalisation Through Universal Principles. Universal quantification, as a key concept, can be viewed from two related but distinct perspectives: From the knowledge representation perspective, it serves to abstract away from concrete entities, generalising over entire sets of domain elements having similar properties. From the technical perspective, it serves to state that for any given object or collection of objects satisfying a particular property (e.g. being a bird), the graphical model that is constructed for a given set of domain elements will have particular properties (e.g. that, for each domain element x, the model will contain a particular conditional distribution for $\mathit{flies}(x)$ given $\mathit{bird}(x)$, or, in the case of an undirected model, will contain a potential function whose domain extends over $\mathit{flies}(x)$ and $\mathit{bird}(x)$). As a consequence, many components in the graphical models constructed for a particular set of domain elements will share the same parameters (conditional distributions or potential functions). *Parameter sharing* is, therefore, a most prominent trait of first-order probabilistic languages.

Transfer. From the interpretation of statistical relational models as templates for the construction of graphical models, it follows that another key principle is the notion that the general statistical knowledge that is represented in meta-models can indeed be soundly transferred to domains of discourse involving an arbitrary number of relevant objects. This is referred to as the principle of *shallow transfer*. The corresponding notion of *deep transfer* (Davis and Domingos, 2009) goes far beyond this fundamental requirement of generalisation; it seeks to transfer statistical principles from one scenario to another, in which objects belonging to entirely different classes and predicates with entirely different interpretations are considered.

2.3.3 An Overview of First-Order Probabilistic Languages

The field of first-order probabilistic languages is utterly diverse. Countless languages have been developed since the late 1990s, and it is fairly easy to lose track of the key features that distinguish them. In the following, I will give an overview of some

of the main trends. Milch and Russell (2006) and de Salvo Braz et al. (2008) also provide descriptive surveys.

A particularly large number of first-order probabilistic languages have built upon directed graphical models, adopting semantics that can be reduced, for a given domain of discourse, to Bayesian networks. Koller and Pfeffer (1997) presented *object-oriented Bayesian networks* (OOBNs), which introduced objects into the language, and, most importantly, the notion of classes to which these objects belong, allowing models to concisely capture the properties that are common across objects belonging to the same class. This approach was fully implemented in the SPOOK system (Pfeffer et al., 1999), which also made relations first-class citizens of the representation, incorporating extensions to the OOBN language introduced by Koller and Pfeffer (1998). With *relational Bayesian networks* (RBNs), Jaeger (1997) introduced another first-order extension of Bayesian networks. In contrast to OOBNs, intantiations of RBNs do not reduce to Bayesian networks in which the random variables correspond to propositions about individual objects but to entire relations, an assignment of a random variable indicating the extension of a relation.

An important modelling issue in directed relational models is the representation of the influence of a variable number of parents on a variable: As the number of domain elements changes from instantiation to instantiation, the set of parents that is to have direct influence on a given variable may vary. Jaeger (1997) uses *combination functions* (also called *combining rules*), which map several separately modelled conditional distributions to a single distribution through an operation such as *max* or *noisy-or*. By contrast, the SPOOK system handles the issue of variable parent counts by computing an aggregate, such as the number of parents having particular properties, and modelling the conditional distribution given the aggregate rather than the parents themselves (Pfeffer et al., 1999).

Another language that is based on Bayesian networks is that of *probabilistic relational models* (PRMs, Friedman et al., 1999), which draws inspiration from entity relationship models and frame-based systems. In early incarnations, the formalism assumed known relations, concentrating on inference over attributes of the related objects – and thus focusing on the perhaps most practically relevant cases, which permit tractable inference and learning. Other notable languages that are based on the principles of directed models include the *BLOG* language (Milch et al., 2005), which

does not assume a known universe of objects and therefore accounts for the existence of unknown objects, *logical Bayesian networks* (Fierens et al., 2005), which differentiate random variables from logical variables whose values are known a priori, and *MEBN* (multi-entity Bayesian networks, Laskey, 2008), a rich language that adopts a network-centric approach towards the generalisation of Bayesian networks, using parametrised network fragments as the principal modelling construct.

Some approaches towards the unification of logical and probabilistic models are best understood as probabilistic extensions of logic programming formalisms rather than as generalisations of graphical models (De Raedt et al., 2008). Nevertheless, some of the respective languages still define their probabilistic semantics through a reduction to graphical models, particularly to Bayesian networks. The relationship between logic programming and Bayesian networks can be established by attaching probability values to implications (typically Horn clauses) appearing in logic programs and interpreting these values as the conditional probabilities of the consequent given the antecedent. Notably, *Bayesian logic programs* (BLPs) use a syntax inspired by logic programming to define templates for the construction of Bayesian networks, yielding a language that is conceptually simple but still retains the expressive power of approaches such as relational Bayesian networks or PRMs (Kersting and Raedt, 2007). Likewise, the *CLP(BN)* language integrates Bayesian networks with an established constraint logic programming framework, using a Prolog-based formalism as a basis (Costa et al., 2003). The *ProbLog* language (De Raedt et al., 2007) is a more recent development, which follows a similar tradition and has even been extended to continuous domains (Gutmann et al., 2010).

Some first-order probabilistic languages make use of conditional distributions to model probabilistic influence but do not exclusively adhere to the principles of Bayesian networks to do so. In particular, *relational dependency networks* (Neville and Jensen, 2007) use conditional distributions – represented, for instance, through probability trees that use relational features – to model uncertainty. Dependency networks, however, drop the acyclicity restriction of Bayesian networks. *Bayesian logic networks* (BLNs), which will be presented in detail in Chapter 5, are another language that makes use of conditional distributions but does not use a standard reduction to Bayesian networks. BLNs are the first language to use the notion of mixed networks with probabilistic and deterministic constraints as the underlying type of graphical

model. Regarding the representation of the directed part of the model, the BLN language adopts many of the principles of the other first-order probabilistic languages that were mentioned above. However, its use of logical formulas for the specification of constraints can significantly facilitate the representation of logical knowledge, which is often equally important in probabilistic knowledge representation and allows, in particular, the intuitive representation of constraints in cases where the requirement of acyclicity is otherwise challenging to maintain (e.g. the requirement of an a priori unknown relation being symmetric or transitive).

The use of undirected models for the representation of first-order probabilistic models can also lead to intuitive representations of statistical knowledge in cases where the requirement of acyclicity would be an inconvenience. The language of *Markov logic networks* (MLNs, Richardson and Domingos, 2006) is a particularly widely used formalism that builds upon Markov random fields rather than Bayesian networks. Like Nilsson's probabilistic logic, Markov logic associates formulas in a first-order logic knowledge base with probabilistic parameters. In Markov logic, however, these parameters are not probabilities but weights (representing factors) that reflect the hardness of the associated formulas as constraints on sets of possible worlds. For a given (finite) domain of discourse, an MLN can be instantiated to obtain a Markov random field whose structure and parameters are fully determined by the set of weighted formulas. The ground network specifies a distribution over a set of possible worlds, where the set of possible worlds is defined in terms of possible assignments to ground atoms (not first-order sentences as in Nilsson's logic). MLNs are a generalisation of first-order logic for finite domains: In the extreme case where all formula weights are considered infinite, MLN semantics reduce to logical semantics, and the ground model represents a uniform distribution over possible worlds that satisfy the formulas in the knowledge base. For infinite domains, a generalisation was presented by Singla and Domingos (2007). Markov logic networks – and extensions thereof – are presented in detail in Chapter 3.

Another noteworthy class of first-order probabilistic languages involves languages that depart from the notion of purely declarative semantics, adopting an approach that incorporates procedural and functional programming to attain a particularly high level of flexibility in the specification of probabilistic models. The *integrated Bayesian agent language* (IBAL) was an early such language, which not only sub-

sumed relational models based on Bayesian networks but also explicitly supported the representation of decision problems (Pfeffer, 2001): both influence diagrams and Markov decision processes could be represented in IBAL. Goodman et al. (2008) proposed the highly general language *Church* for the specification of generative stochastic process models, which is based on the Lisp model of lambda calculus. Generative processes modelled in Church are first-class objects that can be arbitrarily composed and abstracted, enabling Church programs to represent higher-order models. In contrast to Church, the *Factorie* system by McCallum et al. (2009) originally focused exclusively on discriminative models, providing a sophisticated Scala library for the imperative definition of factor graphs. Factorie was later extended to also support generative models. Whereas the motivation for the design of IBAL and Church was generality (their practical applicability being limited as a result), the mix of declarative and procedural domain knowledge in Factorie primarily aims at achieving gains in efficiency.

Lifted Inference. Indeed, efficiency is a key concern in probabilistic models, particularly with respect to inference, for inference typically needs to be performed online – while an AI system performs its tasks – whereas learning can often be performed in an offline phase where there are no hard time constraints. Koller and Pfeffer (1997) already pointed out that inference in the ground model may be inefficient and that regularities that stem from the repetition of template structures can and should be exploited to improve the efficiency of inference. Inference methods that exploit the regularities that result from the application of template models, essentially lifting the inference problem back to the first-order level that was used in the original representation, are called *lifted inference methods*. Poole (2003) presented the first lifted version of variable elimination, a popular exact inference method for Bayesian networks, and his work was later extended by de Salvo Braz et al. (2005) and Milch et al. (2008) to exploit further regularities in ground models. The first lifted version of another widely used method, belief propagation, was presented by Singla and Domingos (2008).

Chapter 3

Markov Logic Networks

Markov logic (Richardson and Domingos, 2006) is a representation formalism that has – owing to its generality and conceptual simplicity – garnered much attention in recent years. Because of its expressiveness, Markov logic is a prime candidate for the equipment of technical systems with high-level learning and reasoning capabilities. Essentially, Markov logic generalises both first-order logic and probabilistic graphical models by attaching weights to formulas in first-order logic. Collectively, the weighted formulas define a template for the construction of a graphical model that specifies a distribution over possible worlds.

As an example, consider the following Markov logic network (MLN), which represents the familiar scenario of people going to a restaurant to order a dish:

$\mathit{female}(\mathit{person})$
$\mathit{vegetarian}(\mathit{person})$
$\mathit{vegDish}(\mathit{dish})$
$\mathit{friends}(\mathit{person}, \mathit{person})$
$\mathit{orders}(\mathit{person}, \mathit{dish!})$

$w_1 = 0.000$	$\forall p.\ \mathit{female}(p)$
$w_2 = -1.735$	$\forall p.\ \mathit{vegetarian}(p)$
$w_3 = -1.099$	$\forall d.\ \mathit{vegDish}(d)$
$w_4 = 1.400$	$\forall p.\ \mathit{female}(p) \Rightarrow \mathit{vegetarian}(p)$
$w_5 = 1.300$	$\forall p_1, p_2.\ \mathit{friends}(p_1, p_2) \wedge \mathit{vegetarian}(p_1) \Rightarrow \mathit{vegetarian}(p_2)$
$w_6 = 2.300$	$\forall p, d.\ \mathit{orders}(p, d) \wedge \neg \mathit{vegetarian}(p) \Rightarrow \neg \mathit{vegDish}(d)$
$w_7 =$(hard)	$\forall p, d.\ \mathit{orders}(p, d) \wedge \mathit{vegetarian}(p) \Rightarrow \mathit{vegDish}(d)$

MLN 3.1: *Model of people ordering dishes in a restaurant*

As shown above, MLNs typically contain declarations that define the types of entities to which predicates can be applied in addition to the weighted formulas themselves. In our scenario, the predicates are applicable to *person* and *dish* entities. Furthermore, predicates/relations can be declared as being functional, as this is a particularly common requirement in relational domains. In MLN 3.1, *orders* is declared as functional, such that, in this variant of the scenario, every person is required to order exactly one dish.

The weights in the MLN determine the degree to which the formulas they are attached to represent a constraint that is believed to hold. In essence, the probability of a possible world is to be proportional to the exponentiated sum of weights of formulas that are satisfied in that world. By attaching a zero weight to the first formula, we essentially make no statement about whether being female or not being female is more probable. Most people, however, are not vegetarians, and most meals are not vegetarian; hence the negative weights w_2 and w_3. The positive weight w_4 is intended to express that females are more likely to be vegetarians. The fifth formula considers a social network structure defined by the *friends* relation and states that friends of vegetarians are more likely to themselves be vegetarians. The last two formulas express consumption preferences: Non-vegetarians are inclined to order non-vegetarian dishes; vegetarians always order vegetarian dishes. The latter constraint is hard, meaning that any world that does not satisfy the formula is to have zero probability.

3.1 Formalism

Formally, a *Markov logic network* L is given by a set of pairs $\langle F_i, w_i \rangle$, where F_i is a formula in first-order logic and w_i is a real-valued weight (Richardson and Domingos, 2006). For each finite domain of discourse D (set of constants), an MLN L defines a *ground Markov network* $M_{L,D} = \langle X, G \rangle$ as follows:

1. X is an indexed set of Boolean random variables. For each possible grounding of each predicate appearing in L, we add to X a Boolean variable (ground atom). We denote by $\mathcal{X} := \mathbb{B}^{|X|}$ the set of possible worlds, i.e. the set of possible assignments of truth values to the variables in X.

2. G is an indexed set of weighted ground formulas, i.e. a set of pairs $\langle \widehat{F}_j, \widehat{w}_j \rangle$, where $\widehat{F}_j$ is a ground formula and $\widehat{w}_j$ is a real-valued weight. For each possible grounding $\widehat{F}_j$ of each formula F_i in L, we add to G the pair $\langle \widehat{F}_j, \widehat{w}_j = w_i \rangle$. With each such pair, we associate a *feature* $\widehat{f}_j : \mathscr{X} \to \{0, 1\}$, whose value for $x \in \mathscr{X}$ is 1 if $\widehat{F}_j$ is satisfied in x and 0 otherwise, and whose weight is $\widehat{w}_j$.

The sets X, $\mathscr{X}$ and G are specific to D, and in cases where a distinction is necessary, this is made explicit by using a subscript (i.e. X_D, $\mathscr{X}_D$ and G_D will be used respectively).

Typed Markov Logic Networks. In practice, it is advisable to use a typed logic, as already indicated in MLN 3.1. The above definition is therefore augmented with declarations that define the types of entities a predicate applies to in order to avoid the instantiation of unnecessary ground atoms. Figure 3.1 schematically illustrates the grounding process in a typed Markov logic network, and Figure 3.2 shows the topology of two ground Markov random fields instantiated based on MLN 3.1.

Furthermore, a particularly common real-world restriction is to require relations to be functional and MLN implementations support the corresponding declarations as an extension. For instance, the declaration *orders*(*person*, *dish!*) in MLN 3.1 states that the predicate *orders* is applicable to *person* and *dish* entities and that for each

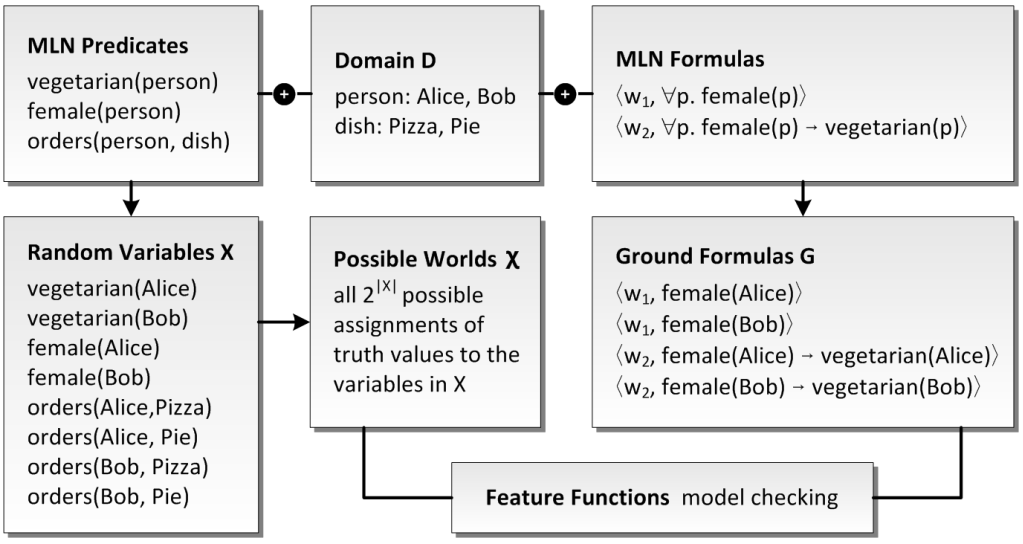
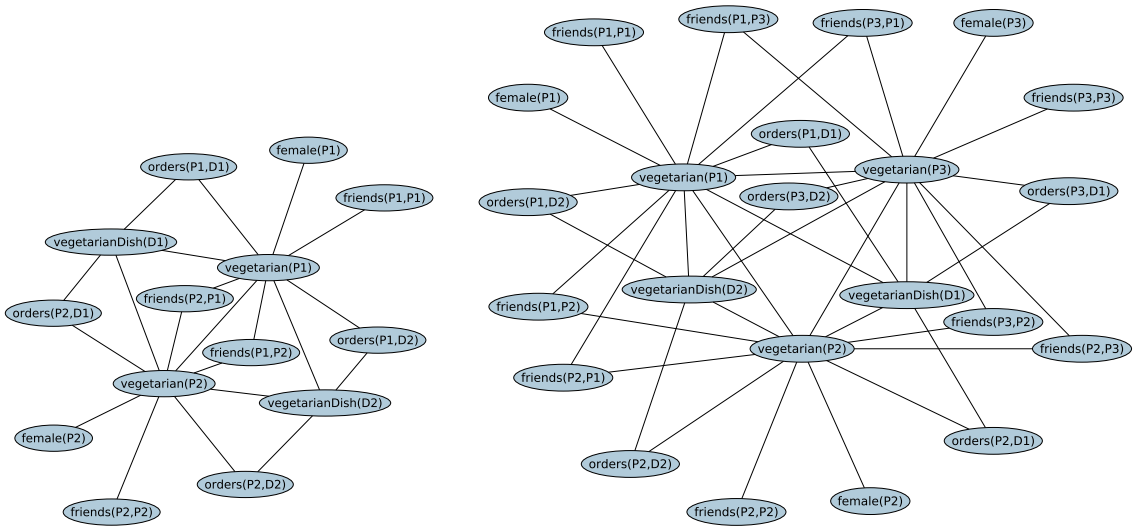


Figure 3.1: *Illustration of the grounding process (typed predicates)*



(a) two people, two dishes **(b)** three people, two dishes

Figure 3.2: *Ground Markov random fields generated from MLN 3.1 for a domain containing two dishes and two/three people. Every ground formula (feature) in the Markov random field induces a clique connecting the ground atoms appearing within the formula.*

person, there must be exactly one dish for which the predicate holds. Any possible world violating this functional property of the relation has zero probability.

Conventions. MLNs follow the convention of using lower-case identifiers for variables and upper-case identifiers for constants (domain elements). This syntactic property allows universal quantification to be implicit: Any free variable appearing in a formula is assumed to be universally quantified. Furthermore, MLNs use first-order logic with equality and make the *unique name assumption*: Every entity is denoted by a single unique name (constant symbol); different names refer to different entities. The equality relation therefore holds only for syntactically equivalent arguments.

3.1.1 Feature Granularity and Quantifier Semantics

A logical knowledge base (KB) is the conjunction of all the formulas that are known to hold. In an MLN, we can choose to partially soften the KB and attach weights to arbitrary parts of the full conjunction, weights indicating the degree to which the

individual formulas are required to hold. The level of granularity at which weighting is applied can be selected as necessary.

The Universal Quantifier for Parameter Sharing. In this context, the semantics of the universal quantifier are particularly relevant. In the logical case, a formula such as $\forall e.\ \mathit{apple}(e) \Rightarrow \mathit{fruit}(e)$ translates, for a finite domain of discourse D, to the conjunction $\bigwedge_{E \in D} \mathit{apple}(E) \Rightarrow \mathit{fruit}(E)$. In MLNs, however, the weighted formula $\langle w, \forall e.\ \mathit{apple}(e) \Rightarrow \mathit{fruit}(e) \rangle$ does not result in the weight w being attached to the conjunction but rather in the weight being attached to each conjunct separately, i.e. its semantics can be understood as $\forall E \in D.\ (\langle w, \mathit{apple}(E) \Rightarrow \mathit{fruit}(E) \rangle \in G)$. Thus, the universal quantifier actually serves to realise parameter sharing. As mentioned in Section 2.3.2, parameter sharing is a fundamental principle in relational models, which serves to achieve the generalisation across arbitrary domains, for it enables us to apply the same probabilistic patterns to any given set of entities (shallow transfer). In most implementations of MLNs, a universal quantifier at the outermost level can be omitted – and, given its semantics, its omission may even serve to improve clarity.

Existential Quantification. Notably, the existential quantifier does not behave analogously; it expands to a single disjunction with weight w (as does a universal quantifier at an inner level). Hence disjunctions are always evaluated strictly logically. Since KBs are "naturally" dissected into conjunctions, this should not be considered as a practically relevant restriction. Nevertheless, the formalism of *recursive random fields* (Lowd and Domingos, 2007b) was proposed to also allow disjunctions to be evaluated in a "softened" manner.

3.1.2 Probabilistic Semantics

The ground Markov network $M_{L,D}$ specifies a probability distribution over the set of possible worlds $\mathcal{X}$ as follows,

$$P(X = x) = \frac{1}{Z} \exp\left(\sum_{i=1}^{|L|} w_i n_i(x)\right) = \frac{1}{Z} \exp\left(\sum_{j=1}^{|G|} \widehat{w}_j \widehat{f}_j(x)\right) \tag{3.1}$$

where $Z = \sum_{x' \in \mathcal{X}} \exp\left(\sum_i w_i n_i(x')\right) = \sum_{x' \in \mathcal{X}} \exp\left(\sum_j \widehat{w}_j \widehat{f}_j(x')\right)$ is a normalisation constant and $n_i(x)$ denotes the number of true groundings of F_i in x.

As indicated by Equation 3.1, the probability of a possible world $x \in \mathcal{X}$ is proportional to the exponentiated sum of weights of formulas that are satisfied in x, i.e.

$$P(x) \propto \exp\left(\sum_j \widehat{w}_j \widehat{f}_j(x)\right) =: \omega(x). \tag{3.2}$$

Equation 3.1 uses the log-linear representation (see Equation 2.5). By definition, this representation does not allow worlds to be assigned zero probability. The inclusion of hard logical rules as in MLN 3.1, however, requires that worlds that are in violation of such rules be assigned zero probability. This is remedied by replacing hard weights with comparatively large real numbers, such that, for practical purposes, the ω value of a world that satisfies all hard formulas will be substantially larger than that of a world violating even one hard formula, rendering the probability of the latter negligible.

Since an MLN defines a full-joint distribution, we can use it to compute arbitrary conditional distributions. With $s(\widehat{F}) := \sum_{x \in \mathcal{X}, x \models \widehat{F}} \omega(x)$, we can compute the probability of any ground formula $\widehat{F}_q$ given that another ground formula $\widehat{F}_e$ holds as

$$P(\widehat{F}_q \mid \widehat{F}_e) = \frac{P(\widehat{F}_q, \widehat{F}_e)}{P(\widehat{F}_e)} = \frac{s(\widehat{F}_q \wedge \widehat{F}_e)}{s(\widehat{F}_e)}. \tag{3.3}$$

Similarly, the marginal probability of $\widehat{F}_q$ can be computed as

$$P(\widehat{F}_q) = \frac{s(\widehat{F}_q)}{s(True)} = \frac{s(\widehat{F}_q)}{s(\widehat{F}_q) + s(\neg \widehat{F}_q)}. \tag{3.4}$$

3.2 Inference

In this section, methods for two of the most common inference problems will be presented: the computation of (posterior) marginals as in Equations 3.3 and 3.4 and the problem of finding the most probable explanation of the evidence (MPE). Since

the probabilistic semantics of Markov logic networks are defined via ground Markov random fields, inference can essentially be handled by applying standard inference methods for MRFs. However, the particular properties of MRFs instantiated from MLNs can call for an explicit treatment.

3.2.1 Posterior Marginals

In practice, an inference problem as defined in Equation 3.3 typically cannot be solved by enumerating all the relevant possible worlds. Given that the number of worlds is exponential in the number of unknown ground atoms, this type of exact inference is infeasible in all but the smallest of instantiations. In the following, I will therefore give an overview of two inference schemes for Markov random fields that can be significantly more practical.

3.2.1.1 Message Passing

One frequently used class of inference methods is based on local message passing between nodes in the factor graph representation of the Markov rand field. These methods generally support the simultaneous computation of the posterior marginals of all the individual random variables in X. For MRFs that correspond to tree-structured factor graphs, the *belief propagation* algorithm (Pearl, 1988) is a particularly efficient method that is guaranteed to produce exact results. For general MRFs, one can use *loopy* belief propagation and iteratively apply the same message passing scheme. However, iterative belief propagation is not guaranteed to converge and may produce results that are far from the true posteriors. While it sometimes works rather well, theoretical results on the conditions under which this is the case are largely unavailable, unfortunately.

Another popular message-passing method is the join-tree algorithm (Lauritzen and Spiegelhalter, 1988), which transforms an arbitrary factor graph into a tree-structured factor graph by merging potentials and then performs standard belief propagation in order to obtain exact results. The transformations it involves can, unfortunately, be prohibitively expensive in practice. Dechter et al. (2002) proposed the iterative join-

graph propagation (IJGP) algorithm, which is more flexible, as it allows to control the degree to which potentials should be merged, thus enabling a practical trade-off between approximation quality and efficiency.

Singla and Domingos (2008) proposed a lifted version of belief propagation, which exploits the fact that in a model instantiated from a relational template, many of the messages being passed are the same owing to prototypical indifference, thus allowing the construction of a smaller lifted network from the original ground network. For MRFs that are not necessarily constructed from relational templates, Kersting et al. (2009) proposed a bottom-up construction of lifted networks, which does not depend on the availability of first-order information.

3.2.1.2 Markov Chain Monte Carlo

A second class of methods that are highly relevant in practice is based on the *Markov chain Monte Carlo* (MCMC) scheme (Koller and Friedman, 2009, ch. 12). MCMC methods are essentially approximate, sampling-based methods where the individual samples are not drawn independently but taken from a Markov chain, i.e. a model describing sequences of states.

Formally, a *Markov chain* over a state space $\mathscr{X}$ is given by an initial state distribution $\pi_0 : \mathscr{X} \to [0,1]$ and a probabilistic transition model that makes the first-order Markov assumption: For any given state $x \in \mathscr{X}$, the probability distribution over successor states x' is given by $\mathscr{T}(x \to x') := P(x' \mid x)$.

Let $\pi_t(x)$ be the probability that the Markov chain with initial distribution π_0 and transition model $\mathscr{T}$ is in state x after t time steps; π_t is obtained from π_{t-1} via $\pi_t(x) = \sum_{x' \in \mathscr{X}} \pi_{t-1}(x') \cdot \mathscr{T}(x' \to x)$. A Markov chain reaches a *stationary distribution* if $\pi_t = \pi_{t-1}$, i.e. if further applications of the transition model no longer alter the distribution:

$$\forall x \in \mathscr{X}. \quad \pi_t(x) = \sum_{x' \in \mathscr{X}} \pi_t(x') \cdot \mathscr{T}(x' \to x) \tag{3.5}$$

The stationary distribution is unique if the Markov chain is *irreducible*, i.e. if any state is reachable from any other state (in some number of steps).

A Markov chain is called *aperiodic* if, for all states $x \in \mathcal{X}$, the greatest common divisor of all path lengths leading from state x back to state x in the Markov chain is 1. A Markov chain that is both irreducible and aperiodic is called *ergodic*. Ergodicity is a particularly important property, for it can be shown that an ergodic Markov chain will reach its unique stationary distribution regardless of the initial distribution π_0.

The principle behind MCMC-based inference is to sample from a Markov chain, i.e. to obtain a particular *trajectory* (sequence of states) $\langle x^{(0)}, \ldots, x^{(N)} \rangle$ from the chain via *Monte Carlo simulation*: We iteratively sample a state from the transition model given an initial state $x^{(0)}$ sampled from $\pi_0(x)$. If the number of samples is large enough, the samples will be an approximate representation of the Markov chain's stationary distribution.

In order to use MCMC for the approximation of a particular distribution, the Markov chain's stationary distribution must, of course, correspond precisely to the distribution we are interested in. A relationship called detailed balance can be used to establish the link between a Markov chain's transition model and its stationary distribution: A Markov chain is said to be in *detailed balance* with a distribution π if its transition model $\mathcal{T}$ satisfies the following property:

$$\forall x, x' \in \mathcal{X}. \quad \pi(x) \cdot \mathcal{T}(x \to x') = \pi(x') \cdot \mathcal{T}(x' \to x) \tag{3.6}$$

Detailed balance immediately implies that π is the stationary distribution of the Markov chain, as for all $x \in \mathcal{X}$, we have:

$$\begin{aligned} \sum_{x'} \pi(x') \cdot \mathcal{T}(x' \to x) &\overset{(3.6)}{=} \sum_{x'} \pi(x) \cdot \mathcal{T}(x \to x') \\ &= \pi(x) \cdot \underbrace{\sum_{x'} \mathcal{T}(x \to x')}_{=1} = \pi(x) \end{aligned} \tag{3.7}$$

Pseudocode for an MCMC-based approximation of the probability of a query q with respect to a probability distribution $\pi(x)$ that is assumed to be the stationary distribution of the Markov chain is given in Algorithm 1. In the context of Markov logic, we are, as described earlier, typically interested in computing the probability of some ground formula $\widehat{F}_q$ given that some other ground formula $\widehat{F}_e$ holds for some instan-

tiation $M_{L,D}$ (Equation 3.3). The Markov chain's stationary distribution $\pi(x)$ should thus be the posterior distribution $P(x \mid \widehat{F}_e, M_{L,D})$ and the query q is the ground formula $\widehat{F}_q$. Since the Markov chain is required to be ergodic, we can, in principle, choose an arbitrary state within the support of $P(x \mid \widehat{F}_e, M_{L,D})$ as $x^{(0)}$ (step 2). Assuming that the set of N samples taken from the Markov chain are a sufficient approximation of the stationary distribution, the probability of the query can be approximated as the relative frequency of states in the sampled trajectory that satisfy the condition that corresponds to the query.

Algorithm 1 MCMC(π_0, $\mathscr{T}$, N, q)

1: $c \leftarrow 0$
2: sample $x^{(0)} \sim \pi_0(x)$
3: **for** $i \leftarrow 1$ to N **do**
4: sample $x^{(i)} \sim \mathscr{T}(x^{(i-1)} \rightarrow x^{(i)})$
5: **if** $x^{(i)} \models q$ **then** $c \leftarrow c + 1$
6: **return** $\pi(q) \approx c/N$

If the number of samples N we can afford to draw is low and the chain can, when starting from a low-probability state, be expected to only slowly reach the most relevant regions of the state space, i.e. the regions with high probability, then it can be sensible to use an initial *burn-in period*, in which the first $M < N$ samples are not used for the approximation. In this way, it can be more likely for the $N - M$ remaining samples to reflect the stationary distribution. Alternatively, the initial state $x^{(0)}$ could be selected such that it has high probability, which can, for instance, be achieved using a local hill-climbing search.

MC-SAT. In the following, I will describe a state-of-the-art MCMC inference method for Markov logic networks called MC-SAT, which was proposed by Poon and Domingos (2006). Unlike traditional MCMC methods such as Gibbs sampling (Geman and Geman, 1984), MC-SAT can soundly handle distributions with deterministic and near-deterministic dependencies, which, given the possibility of logical modelling, abound in Markov logic networks.

Consider a ground Markov random field $M_{L,D} = \langle X, G \rangle$ in which all formula weights are non-negative. Note that this is not a restriction, as any pair $\langle \widehat{F}_j, \widehat{w}_j \rangle \in G$ can be equivalently replaced by $\langle \neg \widehat{F}_j, -\widehat{w}_j \rangle$.[1]

MC-SAT is a slice sampler that makes use of auxiliary variables U in addition to the actual state variables X. It uses one auxiliary variable U_j per ground formula $\langle \widehat{F}_j, \widehat{w}_j \rangle \in G$. The joint distribution over the random variables X and the auxiliary variables U is defined as

$$P(X = x, U = u) = \frac{1}{Z} \prod_{j=1}^{|G|} I_{[0, \exp(\widehat{f}_j(x)\widehat{w}_j)]}(u_j), \tag{3.8}$$

where $I_{[a,b]}$ is an indicator function that returns 1 if the argument is within the interval $[a, b]$ and 0 otherwise. $P(X = x, U = u)$ is a uniform distribution over assignments where the assignment to the auxiliary variables, $U = u$, is in accordance with the intervals that are imposed by $X = x$. Notice that, for any x, the product of the sizes of the admissible intervals is precisely $\omega(x)$. Hence the fraction of Z that is covered by joint assignments where $X = x$ is proportional to $P(x)$. Therefore, to sample from $P(X)$, one can equivalently sample from $P(X, U)$ and ignore the auxiliary values.

The Markov chain is formed by alternatingly sampling the auxiliary variables and the actual state. For a given $x \in \mathcal{X}$, $P(U = u \mid X = x)$ is uniform in the assignments that are within the intervals imposed by x. For given auxiliary values, $P(X = x \mid U = u)$ is uniform in the slice (subset) of $\mathcal{X}$ where $u_j < \exp(\widehat{f}_j(x)\widehat{w}_j)$ for all j, which implies that if $u_j > 1$, then $\widehat{f}_j(x) = 1$ and $\widehat{F}_j$ must be satisfied. Therefore, one can proceed as follows: The initial state $x^{(0)} \in \mathcal{X}$ must be one in the support of $P(X \mid \widehat{F}_e, M_{L,D})$, i.e. one that satisfies the hard constraints in G and the evidence $\widehat{F}_e$. (Therefore, if evidence is given in the form of a ground formula $\widehat{F}_e$, we can simply add

[1] An MLN retains its semantics if any of its formulas are negated along with their weights, i.e. the probability distributions implied by an MLN L and an MLN L' derived from L by changing a pair $\langle F_k, w_k \rangle \in L$ to $\langle \neg F_k, -w_k \rangle \in L'$ are the same for all ground models. If L and L' are instantiated for domain D, for which F_k has N_k groundings, we have:

$$\begin{aligned} P(x \mid M_{L',D}) &= \frac{\exp\left(\sum_{i, i \neq k} w_i \cdot n_i(x) - w_k \cdot (N_k - n_k(x))\right)}{\sum_{x' \in \mathcal{X}} \exp\left(\sum_{i, i \neq k} w_i \cdot n_i(x') - w_k \cdot (N_k - n_k(x'))\right)} \\ &= \frac{\exp\left(\sum_i w_i \cdot n_i(x) - w_k \cdot N_k\right)}{\sum_{x' \in \mathcal{X}} \exp\left(\sum_i w_i \cdot n_i(x') - w_k \cdot N_k\right)} = \frac{\exp\left(\sum_i w_i \cdot n_i(x)\right)}{\sum_{x' \in \mathcal{X}} \exp\left(\sum_i w_i \cdot n_i(x')\right)} \cdot \frac{\cancel{\exp(-w_k \cdot N_k)}}{\cancel{\exp(-w_k \cdot N_k)}} = P(x \mid M_{L,D}) \end{aligned}$$

Algorithm 2 MC-SAT($M_{L,D} = \langle X, G \rangle$, N)

1: **for** $i \leftarrow 0$ to N **do**
2: **if** $i = 0$ **then**
3: $M \leftarrow \{\widehat{F}_k \mid \langle \widehat{F}_k, \widehat{w}_k \rangle \in G \wedge w_k \text{ is hard}\}$
4: **else**
5: $M \leftarrow \emptyset$
6: **for all** $\langle \widehat{F}_k, \widehat{w}_k \rangle \in G$ **do**
7: **if** $x^{(i-1)} \models \widehat{F}_k$ **then** with probability $1 - e^{-\widehat{w}_k}$ add $\widehat{F}_k$ to M
8: sample $x^{(i)} \sim \mathit{Uniform}(\mathrm{SAT}(M))$

$\widehat{F}_e$ to G as a hard formula.) Then, in step i, the auxiliary variables are first resampled given the previous state $x^{(i-1)}$. If $\widehat{F}_j$ was satisfied in $x^{(i-1)}$, then with probability $(\exp(\widehat{w}_j) - 1)/\exp(\widehat{w}_j) = 1 - \exp(-\widehat{w}_j)$, we sample $u_j > 1$ and therefore $\widehat{F}_j$ must also be satisfied in the current step. We thus obtain a set $M = \{\widehat{F}_j \mid u_j > 1\}$ of formulas that must be satisfied and use a SAT sampler to *uniformly* sample a state $x^{(i)} \in \mathcal{X}$ that satisfies the formulas in M. It can be shown that the Markov chain thus defined is ergodic and that its transition model is in detailed balance with the posterior distribution to be approximated (Poon and Domingos, 2006).

For the purpose of uniformly sampling SAT solutions, one can use the SampleSAT algorithm (Wei et al., 2004), which extends the WalkSAT algorithm (Selman et al., 1993) by injecting randomness via simulated annealing-type moves, allowing near-uniform samples to be obtained. Because SampleSAT requires formulas to be in conjunctive normal form, one must apply a conversion beforehand.

Pseudocode for MC-SAT is provided in Algorithm 2. Given a sampled sequence of states (possible worlds) $\langle x^{(0)}, \ldots, x^N \rangle$, the posterior marginal of a query ground formula $\widehat{F}_q$ can be computed as the relative frequency of worlds satisfying the formula (as in Algorithm 1). The only computationally expensive step of the algorithm is line 8, where a world satisfying a set of ground formulas M is to be sampled uniformly at random (using SampleSAT). The version of MC-SAT in Algorithm 2 also uses SampleSAT to select the initial state, but in ill-conditioned cases where mixing is slow, it can be preferable to start from a high-probability state, which can, for instance, be found by running the MaxWalkSAT algorithm (see below).

3.2.2 Most Probable Explanation

As already explained in Section 2.2, another common inference problem is to find the most probable explanation of the evidence, i.e. the world that is, among the worlds satisfying the evidence, the most probable: $\arg\max_{x \in \mathscr{X}} P(x \mid \widehat{F}_e, M_{L,D})$

This problem is closely related to the problem of *weighted maximum satisfiability*, where, given a set of weighted formulas, the goal is to find the state (assignment of truth values) with the greatest sum of weights of satisfied formulas. The relationship is evident from the following reformulation,

$$\arg\max_{x \in \mathscr{X}} P(x \mid \widehat{F}_e, M_{L,D}) = \arg\max_{x \in \mathscr{X},\, x \models \widehat{F}_e} \omega(x) = \arg\max_{x \in \mathscr{X}} \sum_{j=1}^{|G'|} \widehat{f}_j(x) \widehat{w}_j \tag{3.9}$$

where $G' = G \cup \{\langle \widehat{F}_{|G|+1} := \widehat{F}_e, \widehat{w}_{|G|+1} := (\text{hard}) \rangle\}$. A simple solution to the problem involves the use of a randomised local search scheme which, given a randomly chosen possible world as the initial state, successively alters the state in order to find possible worlds with larger sums of weights of satisfied ground formulas. The MaxWalkSAT algorithm (Kautz et al., 1996) is one particularly simple yet effective instance of this class of methods, which adapts the principles of the WalkSAT satisfiability solver (Selman et al., 1993) to the weighted setting.

Apart from methods for weighted maximum SAT, one can leverage the large body of methods that have been developed in the context of weighted constraint satisfaction problems (WCSPs). Ground MRFs instantiated from MLNs can easily be transformed to conform with the conventions commonly used in state-of-the-art WCSP solvers such as *toulbar2*[2] (Jain et al., 2009a). Riedel (2008) furthermore proposed an efficient inference method based on the cutting-plane method.

3.3 Learning

The learning of a statistical relational model involves the (partial) construction of a model from observed training data. The structure of the model can either be known a priori, leaving only the parameters to be determined, or can be part of the learning

[2] http://mulcyber.toulouse.inra.fr/projects/toulbar2

problem. One consequently differentiates *parameter learning* from the harder problem of *structure learning*. The first approach towards learning the structure of MLNs was presented by Kok and Domingos (2005). While structure learning is clearly important if AI systems are to build up probabilistic models with as little human assistance as possible, pragmatic approaches will, at present, still rely on engineered structures for the most part. Parameter learning is, therefore, a most important problem, for knowledge engineers can typically qualitatively assess the properties of a distribution and indicate the dependencies between variables but cannot quantitatively define the degree to which variables depend on one another.

In a Markov logic network, the goal of parameter learning is to set the weights of the model's formulas such that they reflect observations that have been made about the particular part of the world the model is concerned with. We thus assume that the observations that were made are, by and large, representative of the particular aspects of the world that are to be captured by the model, such that they allow us to extract precisely the general principles we are interested in.

The observations used for learning can be stored in a training database that uses the same language as our model. Since MLNs use logical predicates, the database should thus contain the truth values of a number of ground atoms. The entities appearing in the training database implicitly define a set X of ground atoms. We can obtain a fully specified possible world by making the *closed world assumption*: Any ground atoms in X whose truth values are not given in the training database are assumed to be false. Under this assumption, the training database thus specifies a full assignment $X = x$.

The Maximum Likelihood Principle. Probabilistically, the goal of parameter learning is to find the vector of weights w^* that satisfies $w^* = \arg\max_w P(w \mid x)$, i.e. we are interested in finding the parameters that are most probable given the data we have observed. The *likelihood principle* asserts that all the information in the data x is contained in the likelihood $P(x \mid w)$ (Cox et al., 2003). Applying this principle to our maximisation problem thus implies that we can reasonably find the desired vector of weights by maximising not $P(w \mid x)$ but rather $P(x \mid w)$, i.e.

$$w^* = \arg\max_w \; P(x \mid w). \tag{3.10}$$

Given the weights, the probability of the training database (possible world) x is given by Equation 3.1.

Let us briefly consider the implications of Equation 3.10. According to Bayes' rule, $P(w \mid x) \propto P(x \mid w) \cdot P(w)$, and therefore

$$\arg\max_w \; P(w \mid x) = \arg\max_w \; P(x \mid w) \cdot P(w). \tag{3.11}$$

If $\arg\max_w P(x \mid w) \cdot P(w) = \arg\max_w P(x \mid w)$ is to hold, it follows that the prior $P(w)$ must be a uniform distribution over the space $\mathbb{R}^{|L|}$ of weight vectors. This is the fundamental assumption that is made in maximum likelihood learning. In the absence of prior knowledge about the nature of weight vectors that might be suitable, it is, however, a rather reasonable assumption to make.

Maximum Likelihood Learning. An analytic computation of w^* as defined by Equation 3.10 is infeasible. However, since $P(x \mid w)$, viewed as a function of the weight vector w, is convex, one can use gradient-based optimisation methods in order to iteratively determine the global maximum (Richardson and Domingos, 2006): For a convex function, the global maximum can be found by moving – from a given current point (weight vector) – in the direction of the gradient until the maximum is reached (*gradient ascent*), as indicated by the first and second derivatives of the function. Quasi-Newton methods avoid the need to explicitly compute the Hessian matrix of second derivatives and are therefore convenient in practice. The limited-memory version of the Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) algorithm (Zhu et al., 1997) is a particularly efficient and popular method within this class of methods.

To apply L-BFGS, we require only methods that compute the function to maximise as well as its first derivative. Since the log function is strictly increasing, we can, in general, use the logarithm of the function for purposes of maximisation, which

often leads to computational simplifications. In the present case, one can use the log-likelihood, which is given by

$$\begin{aligned} L(X = x) &= \log P(X = x) \\ &= \sum_i w_i \cdot n_i(x) - \log(Z) \\ &= \sum_i w_i \cdot n_i(x) - \log\left(\sum_{x' \in \mathcal{X}} \exp\left(\sum_k w_k \cdot n_k(x')\right)\right). \end{aligned} \tag{3.12}$$

The i-th component of the gradient is given by

$$\begin{aligned} \frac{\delta}{\delta w_i} L(X = x) &= n_i(x) - \sum_{x' \in \mathcal{X}} n_i(x') \cdot \frac{\exp\left(\sum_k w_k \cdot n_k(x')\right)}{\sum_{x'' \in \mathcal{X}} \exp\left(\sum_k w_k \cdot n_k(x'')\right)} \\ &= n_i(x) - \sum_{x' \in \mathcal{X}} n_i(x') \cdot P(X = x'). \end{aligned} \tag{3.13}$$

This expression has an intuitive interpretation: It is the difference between the number of true groundings of the i-th formula in the training data and the expected number of true groundings that is indicated by the model given the current weight vector.

Unfortunately, the computations above rely on inference over the model in order to compute $P(X = x')$ for each $x' \in \mathcal{X}$. This problem being #P-hard, exact maximum likelihood learning is infeasible for all but the smallest of training databases.

Maximum Pseudo-Likelihood Learning. This problem can be addressed by maximising instead the *pseudo-likelihood*

$$P^*(X = x) = \prod_{k=1}^{|X|} P(X_k = x_k \mid \mathrm{MB}_x(X_k)) \tag{3.14}$$

where $X_k \in X$ is a ground atom, x_k is X_k's truth value in x, and $\mathrm{MB}_x(X_k)$ is the assignment of X_k's Markov blanket in x (Richardson and Domingos, 2006). The pseudo-likelihood approximates $P(X = x)$ by making strong independence assumptions, avoiding an expensive inference process.

Let $\overline{n}_{i,k}(x)$ be the number of true groundings of the i-th formula in a modification of x where the truth value of the k-th ground atom, X_k, has been inverted. Furthermore, let F_{X_i} be the set of indices of formulas for which there is at least one grounding that contains X_i. The pseudo-log-likelihood is then given by

$$\begin{aligned}
\log P^*(X = x) &= \log \prod_{k=1}^{N} P(X_k = x_k \mid MB_x(X_k)) \\
&= \sum_{k=1}^{N} \log \left(\frac{\exp\left(\sum_{i \in F_{X_k}} w_i \cdot n_i(x)\right)}{\exp\left(\sum_{i \in F_{X_k}} w_i \cdot n_i(x)\right) + \exp\left(\sum_{i \in F_{X_k}} w_i \cdot \overline{n}_{i,k}(x)\right)} \right) \\
&= \sum_{k=1}^{N} -\log \left(1 + \exp \left(\sum_{i \in F_{X_k}} w_i \cdot (\overline{n}_{i,k}(x) - n_i(x)) \right) \right) \qquad (3.15)
\end{aligned}$$

and the i-th component of the gradient can be computed as

$$\begin{aligned}
&\frac{\delta}{\delta w_i} \log P^*(X = x) \\
&= \sum_{k=1}^{N} \Big(n_i(x) - P(X_k = x_k \mid \mathrm{MB}_x(X_k)) \cdot n_i(x) - P(X_k = \neg x_k \mid \mathrm{MB}_x(X_k)) \cdot \overline{n}_{i,k}(x) \Big) \\
&= \sum_{k=1}^{N} \left(\overline{n}_{i,k}(x) - n_i(x)\right) \cdot \left(\frac{1}{1 + \exp\left(\sum_{j \in F_{X_k}} w_j \cdot (\overline{n}_{j,k}(x) - n_j(x))\right)} - 1 \right). \qquad (3.16)
\end{aligned}$$

The counts $n_i(x)$ and $\overline{n}_{i,k}(x)$ thus constitute a sufficient statistic for pseudo-log-likelihood learning. Computing the expressions in Equations 3.15 and 3.16 does not require inference. Therefore, once the counts have been computed (which may still be a time-consuming task), the pseudo-likelihood can be optimised efficiently. However, depending on the structure of the model and the problem at hand, the use of the pseudo-likelihood may constitute too great a simplification for the results to be an adequate approximation.

Learning from Multiple Databases. The above learning methods can easily be extended to take into consideration more than one training database, assuming independence between the individual databases. Learning from multiple databases is necessary if we have observed several instances of the scenario under consideration

between which there are no connections, e.g. several instances of the same type of social network or several instances of the same type of environment. If we added all the information to a single database, then the ground model would, notably, consider relations between entities that could never have been observed.

Let $\mathscr{D} = \{\langle D_1, x^{(1)}\rangle, \ldots, \langle D_n, x^{(n)}\rangle\}$ be the set of databases from which we intend to learn. Each pair consists of a domain of entities D_i and a possible world $x^{(i)} \in \mathscr{X}_{D_i}$ that represents the concrete observations that were made for that domain. Assuming independence between individual databases, the likelihood function to be maximised becomes $P(\mathscr{D}) = \prod_{i=1}^{n} P(x^{(i)} \mid M_{L,D_i})$. In log space, this reduces to the sum of the individual log-likelihoods, $\sum_{i=1}^{n} L(x^{(i)} \mid M_{L,D_i})$, and similarly for the gradient.

MAP Estimation. As an alternative to maximum likelihood learning, we can assume a prior distribution over weight vectors and perform *maximum a posteriori (MAP) estimation*, i.e. we can perform an optimisation based on Equation 3.11. If $L(x)$ was the objective function in maximum likelihood learning, the new objective function becomes $L(x)+\log P(w)$ and the i-th component of the gradient becomes $\frac{\delta}{\delta w_i}(L(x)+\log P(w))$. Richardson and Domingos (2006) proposed to use a zero-mean Gaussian for each of the weights, assuming mutual independence between weights. This will favour weights close to zero, biasing distributions constructed from the learnt model towards maximum entropy. Especially if the training data is not sufficiently large to capture the true frequencies of the scenario under consideration, the use of such a prior can effectively reduce the problem of overfitting.

Discriminative Learning. A most attractive feature of undirected models and therefore MLNs is that they can also be trained discriminatively, i.e. they can be trained to represent not a full-joint distribution but a conditional distribution. Assuming that there is a strict separation between observable and unobservable variables in the application at hand (e.g. a classification task, where only the classes are always unknown but everything else is given), discriminative training can yield models with superior performance. Methods for the discriminative training of MLNs were introduced by Singla and Domingos (2005).

3.4 Knowledge Engineering

For a probabilistic knowledge representation formalism such as Markov logic to be used in practice, it is imperative that the language allow knowledge to be represented as intended. In particular, it should be possible for a knowledge engineer to clearly and unambiguously define a template for the construction of probability distributions by characterising the key traits of the corresponding stochastic process. Following the principle of shallow transfer, models should generalise to arbitrary domains and arbitrary objects within them – in much the same way as universally quantified formulas in first-order logic are applicable to arbitrary objects. However, statistical relational models may, at times, produce counter-intuitive results. A thorough understanding of the semantics of models is necessary for models to represent what is intended. This section therefore addresses the task of knowledge engineering in more detail.

As an example, consider once more MLN 3.1 (page 41), which seems to be an intuitive representation of this very simple scenario. However, it displays some perhaps unexpected behaviour. First, we observe that, for any given domain of people and dishes, the probability of being female is not 0.5, yet *female* appears only in the first formula, which expresses no inclination in either direction, and in the antecedent of an implication (formula 3). Further, the probability is not only far from 0.5, it also varies depending on the number of people and dishes in the domain. For instance, if there are two dishes that can be ordered, then the probability of being female for cases where the number of people is one, two and three, we obtain approximately[3] 0.223, 0.211 and 0.204 respectively. For *vegetarian* and *vegDish*, we observe particularly wide variations: The probabilities for a dish being vegetarian are 0.182, 0.120 and 0.07 and the probabilities for a person being a vegetarian are 0.084, 0.044 and 0.020 respectively. While at least some of these observations are certainly obvious if one ponders the underlying mathematical definitions, they certainly suggest that a careful consideration of MLN semantics is the very foundation of any knowledge engineering endeavour.

The mathematical description of the semantics of Markov logic networks is deceivingly simple, yet the task of knowledge engineering in Markov logic networks is,

[3] All the inference results reported in this section were computed with exact methods. Where an approximation is mentioned, it refers to the result being rounded only.

as indicated by the example, not as straightforward as it may at first appear. The question of how one could manually define a Markov logic network remains largely unaddressed in the literature. Indeed, learning is often the only way to sensibly derive suitable parameters for an MLN. However, I believe that a thorough understanding of how one *could*, in principle, represent particular probabilistic properties in an MLN is imperative to selecting the formulas that are in fact required for a model to at all be able to reflect the desired properties and obtain a model that is, ideally, an adequate reflection of the real world. In this section, I will address the following knowledge engineering issues (Jain, 2011):

- I provide a discussion of how simple probabilistic properties can be represented in an MLN.
- I point out fallacious assumptions that result from a naive interpretation of the nature of the transition from logical to probabilistic knowledge. In particular, I discuss the fallacious use of probabilistic implications and describe how the intended pieces of knowledge can be represented by other means.
- I summarise practical issues pertaining to the way in which models in Markov logic – be they learnt from data or manually defined – generalise across domains.

3.4.1 Semantic Perspectives

We can differentiate at least two ways in which MLN semantics can be viewed:

(V1) the top-down view, in which we seek to realise a probabilistic logic by adding weights to formulas; Markov networks serve only as a formalism that provides the particular probabilistic semantics;

(V2) the bottom-up view, in which MLNs are an extension of Markov networks and weighted formulas are a particular way of compactly representing generalised features of Markov networks; Any ground formula defines a clique in the ground Markov network and the formula's set of models (satisfying assignments) determines the subset of the clique's configurations that is affected.

From the perspective of artificial intelligence, the top-down view is perhaps more attractive, because the generalisation of logic to probabilistic settings was a long-standing goal of the community. Probability theory is widely accepted as the preferred way of dealing with the inherent uncertainty that permeates real-world domains, and logics are well-understood formalisms for the representation of complex, structured knowledge. To seek a sound unification is, indeed, the merest step of logic.

However, it is the bottom-up view that is essential to a deep understanding of MLNs. As will be shown in the following, a thorough understanding of Markov networks and, in particular, the way in which weighted formulas translate to features in Markov networks is the very foundation for knowledge engineering in MLNs. Unfortunately, a modelling approach that is too firmly rooted in the principles of logic will often translate sub-optimally to probabilistic settings.

3.4.2 Formula Weights

Weights as logarithmised factors. To determine the precise effect of a weight in an MLN, we need only to consider the impact that weights have on the joint distribution over $\mathscr{X}$ (for an arbitrary but fixed grounding). Every non-zero weight w_i of every formula F_i affects the sum $\sum_i w_i n_i(x)$ for some subset of $\mathscr{X}$ and thus has bearing on the distribution. Because $P(X = x)$ is proportional to the exponentiated sum of weights that are true in world x, weights are best viewed as logarithmised factors. In the absence of formulas, every possible world's value $\omega(x)$ is $\exp(0) = 1$ and therefore all worlds are equally probable. If we now add the pair $\langle F_i, w_i = \log(\lambda_i)\rangle$ to L and some possible world x satisfies a ground instance $\widehat{F}_j$ of F_i, then $\omega(x)$ is simply multiplied by $\exp(w_i) =: \lambda_i = \widehat{\lambda}_j$, either increasing the probability of x (for $\lambda_i > 1 \Leftrightarrow w_i > 0$) or decreasing it (for $\lambda_i \in]0;1[\Leftrightarrow w_i < 0$). Correspondingly,

$$P(X = x) \propto \prod_j \exp(\widehat{f}_j(x)\widehat{w}_j) = \prod_{j, x \models \widehat{F}_j} \widehat{\lambda}_j. \tag{3.17}$$

When interpreting the impact of a particular weight w_i of a particular formula F_i, one should be aware of the factor it implies and the subset of $\mathscr{X}$ it will be applied to (for any imaginable instantiations of L). The impact of a particular formula is perhaps most clearly evident if the formula is translated to disjunctive normal form (DNF),

because the conjunctions therein immediately reflect partial assignments in possible worlds. For instance, all of the following are semantically equivalent,

$$
\begin{array}{ll}
w & \mathit{bird}(e) \Rightarrow \mathit{flies}(e) \\
w & \neg \mathit{bird}(e) \vee \mathit{flies}(e) \\
w & (\mathit{bird}(e) \wedge \mathit{flies}(e)) \vee (\neg \mathit{bird}(e) \wedge \mathit{flies}(e)) \vee (\neg \mathit{bird}(e) \wedge \neg \mathit{flies}(e)) \\
-w & \mathit{bird}(e) \wedge \neg \mathit{flies}(e)
\end{array}
$$

yet the latter two forms perhaps more clearly indicate the variable assignments that are affected. In the fourth case, the formula was negated along with its weight to obtain a more easily interpretable conjunction (see footnote 1 on page 51).

While the local view of the effect of a formula weight on the values $\omega(x)$ associated with possible worlds sufficiently describes the semantics of MLNs, it is, for the most part, not very helpful from a knowledge engineering perspective. In particular, it does not immediately answer the question of how we could define weights that will result in the model reflecting a particular probabilistic property.

Weights as log odds between world probabilities. To determine the effect of a particular weight in terms of probability, let us consider a single (ground) formula $\widehat{F}_j$ with an associated weight $w_i = \widehat{w}_j$ and let us assume that the MLN contains no further formulas. As mentioned by Richardson and Domingos (2006), we can, in such a case, interpret $\widehat{w}_j$ as the log odds between the probability of a world where $\widehat{F}_j$ is true and the probability of a world where $\widehat{F}_j$ is false. Thus, if $x_1 \models \widehat{F}_j$ and $x_2 \not\models \widehat{F}_j$ ($x_1, x_2 \in \mathcal{X}$), then $P(x_1) : P(x_2) = \widehat{\lambda}_j : 1$, since $\widehat{F}_j$ being true will result in a factor of $\widehat{\lambda}_j$ being applied to worlds satisfying $\widehat{F}_j$ while worlds not satisfying $\widehat{F}_j$ will keep the default factor of 1. Setting $\widehat{w}_j = \log(P(x_1)/P(x_2))$ will thus establish the proper ratio between worlds satisfying $\widehat{F}_j$ and worlds not satisfying $\widehat{F}_j$.

Weights that determine formula probabilities. Regarding the probabilistic properties that are to be represented in our model, the local view of ratios between probabilities of individual worlds is inadequate. We need to consider the global impact on all possible worlds in order to asses the properties of the distribution $P(x)$ that are induced by a particular choice of weights, which, in particular, necessitates taking model counts into consideration.

The perhaps most basic modelling task is to define a single formula F_i with an associated weight w_i such that $P(\widehat{F}_j) = p$ for some probability p and a ground instance $\widehat{F}_j$ of F_i – still assuming, for the time being, that F_i is the only formula in the network. If we set $w_i = \log(p_1/p_2)$, then, for $x_1 \models \widehat{F}_j$ and $x_2 \not\models \widehat{F}_j$, we have that $\omega(x_1) : \omega(x_2) = p_1/p_2 : 1 = p_1 : p_2$ (if x_1 and x_2 are equivalent as far as the truth values of other ground instances of F_i are concerned). To compute the probability $P(\widehat{F}_j)$, we need to consider $s(\widehat{F}_j)$ and $s(\neg\widehat{F}_j)$. With $\#_{\widehat{F}_j}$ as the number of models of the ground formula $\widehat{F}_j$, it follows that if $\#_{\widehat{F}_j} = \#_{\neg\widehat{F}_j}$, then $s(\widehat{F}_j) : s(\neg\widehat{F}_j) = p_1 : p_2$ and therefore $P(\widehat{F}_j) = p_1/(p_1 + p_2)$. A particularly relevant case where $\#_{\widehat{F}_j} = \#_{\neg\widehat{F}_j}$ holds is the case where F_i is an atomic formula such as $bird(e)$. Thus, the probability $P(bird(e))$ can be set to p for all entities e using

$$\log(p/(1-p)) \quad bird(e).$$

If, however, the weight w_i applies asymmetrically to the set of possible worlds, i.e. if $\#_{\widehat{F}_j} \neq \#_{\neg\widehat{F}_j}$, then this has to be taken into consideration. If, for example, the formula $\widehat{F}_j$ was satisfied in the majority of possible worlds, then using $\log(p/(1-p))$ as a weight would result in $P(\widehat{F}_j)$ being larger than p. To set $P(\widehat{F}_j) = p$, we need to find a factor $\widehat{\lambda}_j$ such that

$$\begin{aligned} P(\widehat{F}_j) &= \frac{s(\widehat{F}_j)}{s(\widehat{F}_j) + s(\neg\widehat{F}_j)} = \frac{\#_{\widehat{F}_j}\widehat{\lambda}_j}{\#_{\widehat{F}_j}\widehat{\lambda}_j + \#_{\neg\widehat{F}_j}1} = p \\ \Rightarrow\ \widehat{\lambda}_j &= \#_{\neg\widehat{F}_j} p/(\#_{\widehat{F}_j}(1-p)) \end{aligned} \tag{3.18}$$

For instance, if F_i is an implication such as $bird(e) \Rightarrow flies(e)$, then we can achieve $P(bird(e) \Rightarrow flies(e)) = p$ for all e using

$$\log(p/(3-3p)) \quad bird(e) \Rightarrow flies(e)$$

since $\#_{\widehat{F}_j} : \#_{\neg\widehat{F}_j} = 3 : 1$ in this case.

In all the above considerations, we have made the assumption that the sets of ground atoms appearing in any two ground instances of F_i is disjoint, for if they were to overlap, then there would be interactions between the formulas that might already be too difficult to manually foresee.

Weights of mutually exclusive formulas. There is one simple case where we can easily cope with ground atoms appearing in more than one ground formula – the case where the formulas are mutually exclusive, such that in any possible world, at most one of the formulas is true. For instance, if we have both the formula $bird(e)$ and the formula $\neg bird(e)$, then we can achieve $P(bird(e)) = p$ using

$$\begin{array}{ll} \log(p) & bird(e) \\ \log(1-p) & \neg bird(e). \end{array}$$

Because each of the cases is considered in an explicit factor, we need not use odds to establish the desired ratio; the above weighting will immediately ensure that $s(\widehat{F}_j) : s(\neg\widehat{F}_j) = p : (1-p)$ and therefore $P(\widehat{F}_j) = p$ for all instances $\widehat{F}_j$ of $bird(e)$.

This procedure straightforwardly translates to larger sets of mutually exclusive formulas. If, for instance, we have a predicate declared as $isa(entity, type!)$, i.e. an entity belongs to precisely one of several types, then if the set of types is e.g. $\{Mammal, Bird, Reptile\}$ and the corresponding probabilities are p_1, p_2, p_3, we could set:

$$\begin{array}{ll} \log(p_1) & isa(e, Mammal) \\ \log(p_2) & isa(e, Bird) \\ \log(p_3) & isa(e, Reptile) \end{array}$$

It is desirable for the set of formulas whose weights we set in this way to not only be mutually exclusive but also exhaustive, i.e. exactly one of the formulas should be true for every given world and every binding of the variables, because cases not mentioned in our set of formulas obtain an implicit weight of 0 by default, which in turn implies a higher probability for cases not mentioned, since $0 \geq \log(p_i)$.

It is also important to note that for a mutually exclusive and exhaustive set of formulas, weights are meaningful *only* relative to each other (as in the resulting MRF, the respective weights then essentially define a complete clique potential). In particular, for a probability value $p_i \in]0;1[$ the corresponding weight $\log(p_i)$ is smaller than 0, but this does not at all imply that the corresponding formula being true will lower the probability of worlds satisfying it. In fact, we can add arbitrary offsets $\delta \in \mathbb{R}$ to all of the formula weights without affecting the distribution $P(X = x)$, because if the formulas are mutually exclusive and exhaustive, then the offset will apply to each possible world in exactly the same way: If there are k ground instances of the mu-

tually exclusive and exhaustive set of formulas, then δ is summed k times for each possible world $x \in \mathcal{X}$ and we thus obtain:

$$P(x) = \frac{\exp\left(\sum_j \widehat{w}_j \widehat{f}_j(x) + k\delta\right)}{\sum_{x' \in \mathcal{X}} \exp\left(\sum_j \widehat{w}_j \widehat{f}_j(x') + k\delta\right)} = \frac{\exp\left(\sum_j \widehat{w}_j \widehat{f}_j(x)\right)}{\sum_{x' \in \mathcal{X}} \exp\left(\sum_j \widehat{w}_j \widehat{f}_j(x')\right)} \cdot \frac{\cancel{\exp(k\delta)}}{\cancel{\exp(k\delta)}} \quad (3.19)$$

3.4.3 The Probabilistic Implication Fallacy

A causal model is often an intuitive representation of a generative stochastic process. Causality dictates a temporal ordering in which values are assigned to variables: If A is the cause of B, then we first determine whether or not A is the case, and depending on the result, we determine whether B is the also case. In the strictly logical case, we can use the implication $A \Rightarrow B$ to represent logical causal influence. In the probabilistic case, the influence we need to model is "If A holds, then B holds with some probability", which corresponds to the conditional probability $P(B \mid A)$. The *probabilistic implication fallacy* lies in the assumption that a softened version of the implication $A \Rightarrow B$ is an adequate representation for probabilistic causal influence.

Fallacy 1: Probabilistic implications are equivalent to conditional probabilities. Most likely, one of the reasons for people to think of implications and conditional probabilities as being inextricably linked is that in the strictly logical case, there is indeed an obvious connection. Consider the classical example of modelling the probability with which birds are able to fly. If all birds are able to fly, $P(\mathit{flies} \mid \mathit{bird}) = 1$. In this case, the weighted formula

$$\text{(hard)} \quad \mathit{bird}(e) \Rightarrow \mathit{flies}(e),$$

which, by definition, implies $P(\mathit{bird} \Rightarrow \mathit{flies}) = 1$, has related semantics. If *bird* is true, we recover *flies* with probability 1.

In general, however, there is no immediate correspondence between the probability of the implication and the conditional probability. If we set the probability $P(\mathit{bird}(e) \Rightarrow \mathit{flies}(e))$ to $p \in]0;1]$, which we can achieve by setting the weight of the implication

to $\log(p/(3-3p))$ (Eq. 3.18) or, equivalently, by using the more explicit group of mutually exclusive formulas

$$\begin{array}{ll}
\log(p/3) & \mathit{flies}(e) \wedge \mathit{bird}(e) \\
\log(p/3) & \mathit{flies}(e) \wedge \neg \mathit{bird}(e) \\
\log(p/3) & \neg \mathit{flies}(e) \wedge \neg \mathit{bird}(e) \\
\log(1-p) & \neg \mathit{flies}(e) \wedge \mathit{bird}(e)
\end{array}$$

then we obtain $c := P(\mathit{flies}(e) \mid \mathit{bird}(e)) = (p/3)/(p/3 + 1 - p) = p/(3-2p)$. This expression is equal to p only for $p = 1$ and $p = 0$. For values in between, the conditional probability is lower than the probability of the implication. For example, if $p = 0.7$, the conditional probability is 0.4375.

Of course, it is possible to set the probability p of the implication such that it will result in any given conditional probability for c (specifically, $p/(3-2p) = c \Leftrightarrow p = 3c/(1+2c)$). However, as is evident from the above reformulation, the probabilistic implication has bearing not only on the conditional probability of *flies* given *bird* but also on the marginal distributions of both atoms. For instance, for $p = 1$ and $p = 0.7$, we obtain $P(\mathit{bird}(e)) = 1/3 = 0.\overline{3}$ and $P(\mathit{bird}(e)) = 7/15 = 0.4\overline{6}$ respectively. From the equivalence $\mathit{bird}(e) \Rightarrow \mathit{flies}(e) \equiv \neg \mathit{bird}(e) \vee \mathit{flies}(e)$, it should be clear that a greater probability of $\mathit{bird}(e) \Rightarrow \mathit{flies}(e)$ will result in a lower probability of $\mathit{bird}(e)$.

Therefore, a probabilistic implication is clearly inappropriate for the representation of a conditional probability.[4] The probability/weight of an implication does not correspond to a conditional probability in the general case, and implications have influence on the distribution beyond the conditional probability that was intended to be affected.

Fallacy 2: A single implication/formula is sufficient for the representation of probabilistic causal influence. When making the transition from logical to probabilistic knowledge, it might, motivated by the top-down view (V1), seem natural to replace the logical influence of *bird* on *flies*, which, in a logical model, might have been represented using the implication $\forall e.\ \mathit{bird}(e) \Rightarrow \mathit{flies}(e)$, with a weighted ver-

[4] As this essentially follows immediately from the very definition of the implication, similar observations have been previously made. In particular, Joseph Y. Halpern has remarked on a related issue in footnote 2 of his "Analysis of First-Order Logics of Probability" (Halpern, 1990).

sion of the same formula in an MLN. In other words, it might seem natural to make the transition from logical to probabilistic knowledge by simply adding weights to the formulas that are not universally true and thereby make room for exceptions. Unfortunately, this rather intriguing concept turns out to be insufficient if formerly logical dependencies are to be replaced by probabilistic dependencies.

In the strictly logical case, we want to conclude *flies* from *bird*, i.e. if $bird(e)$ holds, we expect to recover $flies(e)$ with probability 1. When we make the transition to the probabilistic realm, a causal model will need to represent the degree to which we can conclude *flies* from *bird* and thus represent the conditional probability $P(flies(e) \mid bird(e))$. As we saw above, the probabilistic implication is a poor candidate for the representation of this probability.

To determine how to best represent such conditional probabilities, a simple analysis suffices. By definition, $P(flies(e) \mid bird(e)) \propto P(flies(e) \wedge bird(e))$. The conditional probability is thus concerned with defining the degree to which *flies* and *bird* co-occur. If $bird(e)$ is given as evidence, we need not concern ourselves with how $P(bird(e))$ may be modelled, and by attaching a weight to $flies(e) \wedge bird(e)$, we can establish any ratio between $flies(e) \wedge bird(e)$ and $\neg flies(e) \wedge bird(e)$. In particular, the ratio $p : 1 - p$ can be established using

$$\log(p/(1-p)) \quad flies(e) \wedge bird(e),$$

setting the conditional probability to p. As long as $bird(e)$ is given, we could, notably, have used the formula $flies(e) \Rightarrow bird(e)$ to the same effect. Unfortunately, neither formulation leaves the marginal probability of $bird(e)$ unchanged, because both affect cases where $bird(e)$ holds and cases where $\neg bird(e)$ holds asymmetrically. The addition of a causal influence (via conditional probabilities) should, however, have no impact on the marginal probability of the cause. Therefore, we need to find weighted formulas that will represent the conditional probability we want to represent whilst guaranteeing that $P(bird(e))$ and therefore the ratio $s(bird(e)) : s(\neg bird(e))$ (for an arbitrary but fixed e) remains unaffected. If we consider all the configurations of the clique connecting $bird(e)$ and $flies(e)$ explicitly, we will have a maximum of flexibility to satisfy these restrictions.

$$
\begin{aligned}
w_1 &= \log(\lambda_1) \quad & \mathit{flies}(e) \wedge \quad & \mathit{bird}(e) \\
w_2 &= \log(\lambda_2) \quad & \neg\mathit{flies}(e) \wedge \quad & \mathit{bird}(e) \\
w_3 &= \log(\lambda_3) \quad & \mathit{flies}(e) \wedge & \neg\mathit{bird}(e) \\
w_4 &= \log(\lambda_4) \quad & \neg\mathit{flies}(e) \wedge & \neg\mathit{bird}(e)
\end{aligned}
$$

In order to represent the conditional probability $P(\mathit{flies}(e) \mid \mathit{bird}(e)) = p$ and thus establish the ratio between $s(\mathit{flies}(e) \wedge \mathit{bird}(e))$ and $s(\neg\mathit{flies}(e) \wedge \mathit{bird}(e))$ as $p : 1-p$, we can simply set $\lambda_1 = p$ and $\lambda_2 = 1-p$. If, prior to the addition of the four above formulas to our MLN, the model contains no information about the distribution of $\mathit{flies}(e)$ and therefore $S_B := s(\mathit{flies}(e) \wedge \mathit{bird}(e)) = s(\neg\mathit{flies}(e) \wedge \mathit{bird}(e))$ and $S_{\neg B} := s(\mathit{flies}(e) \wedge \neg\mathit{bird}(e)) = s(\neg\mathit{flies}(e) \wedge \neg\mathit{bird}(e))$, then $s(\mathit{bird}(e)) : s(\neg\mathit{bird}(e))$ will remain unaffected if $\lambda_3 + \lambda_4 = \lambda_1 + \lambda_2$, since

$$
\frac{s(\mathit{bird}(e))}{s(\neg\mathit{bird}(e))} = \frac{2 \cdot S_B}{2 \cdot S_{\neg B}} \overset{!}{=} \frac{S_B\lambda_1 + S_B\lambda_2}{S_{\neg B}\lambda_3 + S_{\neg B}\lambda_4} \Leftrightarrow \frac{\lambda_1 + \lambda_2}{\lambda_3 + \lambda_4} = 1. \tag{3.20}
$$

The use of complementary probability values for λ_3 and λ_4 trivially satisfies the equation, and the four formulas above will thus collectively represent the entire conditional distribution of *flies* given *bird*. Therefore, if unwanted side-effects are to be avoided, the representation of a full conditional distribution – using an exhaustive set of conjunctions as above – is perhaps the most obvious solution.

We can represent a direct factorisation of the full joint, i.e. $P(\mathit{bird}(e), \mathit{flies}(e)) = P(\mathit{flies}(e) \mid \mathit{bird}(e))P(\mathit{bird}(e))$, by adding the marginal probability $P(\mathit{bird}(e)) = p_b$ to the model using either $\langle \mathit{bird}(e), \log(p_b/(1-p_b)) \rangle$ or the mutually exclusive pair of $\mathit{bird}(e)$ and $\neg\mathit{bird}(e)$ with weights $\log(p_b)$ and $\log(1-p_b)$. In the latter case, we obtain a model that is completely analogous to a directed model in the sense that it represents precisely the same factors with $Z = 1$.

It has now been established that, in order to model causal influence, the use of implications is certainly not as helpful as it may, at first, have appeared. Indeed, the semantics of implications do not seem to translate well to the probabilistic setting. Nevertheless, it is worth noting that it would have been possible to use implications to achieve precisely the same effect as the four conjunctions above. Specifically, the following two transformations are semantically equivalent:

$-w_1$	$(w_2+w_3+w_4-2w_1)/3$	$\mathit{flies}(e) \Rightarrow \neg \mathit{bird}(e)$
$-w_2$	$(w_1+w_3+w_4-2w_2)/3$	$\neg\mathit{flies}(e) \Rightarrow \neg \mathit{bird}(e)$
$-w_3$	$(w_1+w_2+w_4-2w_3)/3$	$\mathit{flies}(e) \Rightarrow \mathit{bird}(e)$
$-w_4$	$(w_1+w_2+w_3-2w_4)/3$	$\neg\mathit{flies}(e) \Rightarrow \mathit{bird}(e)$

The second transformation even retains the normalisation constant. However, there is no good reason to prefer them from a knowledge representation perspective – to the contrary: Since implications are not mutually exclusive, their impact on possible worlds is considerably more difficult to interpret.

Syntactic Sugar. From the above considerations, we can draw an important conclusion: Simply adding weights to the logical formulas we might have used in the non-probabilistic case is unlikely to be sufficient if probabilistic dependencies are to be captured to their full extent. While Markov logic networks *are* a generalisation of logic that soundly integrates probabilistic semantics, thinking in logical terms is, for the most part, a hindrance as far as the probabilistic knowledge engineering process is concerned. In fact, one might suggest that logical formulations involving (bi)implications be used primarily for the purpose of specifying hard constraints. Indeed, with regard to probabilistic aspects, formulas are mere syntactic sugar for the specification of a subset of a clique's configurations, and whether or not the logical form fosters comprehensibility of the effect of a weighted formula is debatable. In particular, I largely discourage the use of implications in probabilistic contexts because implications are typically associated with a directedness which, in probabilistic settings, may not reflect the expected semantics. I argue that the probabilistic counterpart of an implication is not a probabilistic implication but rather a conditional probability (distribution). For the representation of non-deterministic features, an implication such as as $\mathit{bird}(e) \Rightarrow \mathit{flies}(e)$ should be used only if the negation $\mathit{bird}(e) \wedge \neg\mathit{flies}(e)$ is deemed equally appropriate for the task (where the negated form is perhaps the more readable alternative). The key question to ask is: Is the set of partial configurations affected by an implication truly the one we seek to affect or is it the co-occurrence of antecedent and consequent (as in a conditional probability statement) we are interested in?

3.4.4 Shallow Transfer

As mentioned in Section 2.3, one of the key ideas of first-order probabilistic languages such as MLNs is to represent models that generalise across domains, i.e. the principle of *shallow transfer*. For any particular domain of discourse (set of entities), we, ideally, want the model to generate a specific ground model that is "adequate". In this section, I discuss cases where the transition from one domain to another can affect the distribution in perhaps unexpected ways.

Invariant properties of distributions across domains. In many scenarios, it is reasonable to assume that the probability distributions represented by a statistical relational model will possess certain invariants that will hold regardless of the instantiation. For instance, in the introductory restaurant example, it might be reasonable to expect that the probability of being female should be approximately 0.5 for all people about whom we do not have any evidence. Indeed, if we assume that the model is to represent a (temporally ordered) generative stochastic process which first generates people along with their gender and subsequently generates further properties and links to other entities depending on the gender, then the probability of being female should remain constant in the absence of evidence. Similarly, an invariant might exist for the probability of a dish being vegetarian.

Consider the simplified version of the introductory example listed in MLN 3.2. As we already know from Section 3.4.3, the third formula will have influence on the marginal probability of a dish being vegetarian. Therefore, it is not surprising to learn the probability will be greater than the probability 0.25 that is indicated by the first formula. Perhaps more surprising is the fact that the influence of the hard formula varies with the number of entities in the domain. For example, the marginal probability of a given dish being vegetarian is approximately 0.28 for a domain where there is one person and one dish, whereas it is 0.39 for a domain where there are four people and one dish.

The variations are due to combinatorial effects (Jain et al., 2007). Without the third formula, if there are N_P people and N_D dishes, there would be $\alpha(N_P, N_D) := 2^{N_P+N_D} \cdot {N_D}^{N_P}$ possible worlds with non-zero probability (given that every person consumes exactly one dish). For any pair of possible worlds $\langle x, x' \rangle$, where x and x' differ only

vegetarian(*person*)
vegDish(*dish*)
orders(*person*, *dish!*)

$w_1 = \log(0.25/0.75)$	$vegDish(d)$
$w_2 = \log(0.15/0.85)$	$vegetarian(p)$
$w_3 =$(hard)	$orders(p,d) \wedge vegetarian(p) \Rightarrow vegDish(d)$

MLN 3.2: *Simplified model of people ordering dishes in a restaurant*

in the truth value of $vegDish(d)$ for some dish d, $\omega(x) : \omega(x') = 25 : 75$ if $x \models vegDish(d)$, $x' \models \neg vegDish(d)$. The inclusion of the hard constraint, however, causes the probability of

$$\delta(N_P, N_D) := \sum_{v=1}^{N_P} \binom{N_P}{v} \sum_{m=1}^{v} \binom{v}{m} \sum_{i=1}^{N_D} \binom{N_D}{i} i^m (N_D - i)^{v-m} {N_D}^{N_P - v} \tag{3.21}$$

worlds violating the constraint to be set to zero (where v is the number of vegetarians, m is the number of meat-eating vegetarians, and i is the number of non-vegetarian dishes), such that for some pairs $\langle x, x' \rangle$, the ratio $\omega(x) : \omega(x')$ becomes $25 : 0$, thus causing worlds with vegetarian dishes to become more probable. Since the fraction $\delta(N_P, N_D)/\alpha(N_P, N_D)$ increases as either N_P or N_D increases, $P(vegDish(d))$ varies with the number of people and dishes accordingly.[5]

There is, unfortunately, no way of adjusting the parameters of the model such that it will invariantly compute a fixed marginal probability for *vegDish*. Any adjustments made to the weight of the first formula will correct the probability only for a given number of entities.

Moreover, even if we replace the third constraint with the full conditional distribution of *orders* given *vegetarian* and *vegDish* (which could, in principle, leave the marginals unchanged as explained in Section 3.4.3), there will still be interactions with domain size owing to the constraint that requires exactly one dish to be consumed by each person. Therefore, in conclusion, there are probabilistic invariants that simply cannot be represented. Because the notion of invariants that are to hold in all instantiations is all but far-fetched – any model of a stochastic process that is causally ordered will

[5]Note that there is an analogous effect for the predicate *vegetarian*.

typically exhibit invariants – it is a key issue of knowledge engineering, and it is essential to be aware of the ramifications.

Parameter learning may not guarantee that shallow transfer will work as expected. From the analyses above, it should be clear that the manual engineering of MLNs is inherently difficult. As soon as we depart from the pattern of modelling marginal and conditional distributions using mutually exclusive and exhaustive formulas, the interactions between formulas will typically be impossible to predict intuitively. Learning parameters from data (or even the structure of the model) is often advisable. However, it is important to realise that learning may not guarantee that shallow transfer will work as expected – even for cases where a model that soundly generalises across domains exists. Consider MLN 3.3, which is capable of representing a factorisation analogous to a directed model: The first two formulas can represent the marginal distributions of *vegetarian* and *vegDish*, and the remaining formulas can represent the four conditional distributions of *orders* given *vegetarian* and *vegDish*. For a case where there are no constraints on the *orders* relation (such as the requirement that exactly one dish must be consumed by everyone), this set of of formulas will allow us to represent invariant marginals for both *vegetarian* and *vegDish* and, by assigning a hard negative weight to the fifth formula and a weight of $0 = \log(1.0)$ to the sixth, it also allows us to represent the "vegetarian constraint".

However, when learning the weights from data, we will not necessarily recover the invariants – regardless of the degree to which the empirical frequencies in the data match the true parameters (and regardless of the number of independent training

w_1	$w_1 + \delta$	$\mathit{vegetarian}(p)$
w_2	w_2	$\mathit{vegDish}(d)$
w_3	$w_3 - \delta / N_D$	$\mathit{orders}(p,d) \wedge \mathit{vegetarian}(p) \wedge \mathit{vegDish}(d)$
w_4	$w_4 - \delta / N_D$	$\neg\mathit{orders}(p,d) \wedge \mathit{vegetarian}(p) \wedge \mathit{vegDish}(d)$
w_5	$w_5 - \delta / N_D$	$\mathit{orders}(p,d) \wedge \mathit{vegetarian}(p) \wedge \neg\mathit{vegDish}(d)$
w_6	$w_6 - \delta / N_D$	$\neg\mathit{orders}(p,d) \wedge \mathit{vegetarian}(p) \wedge \neg\mathit{vegDish}(d)$
w_7	w_7	$\mathit{orders}(p,d) \wedge \neg\mathit{vegetarian}(p) \wedge \mathit{vegDish}(d)$
w_8	w_8	$\neg\mathit{orders}(p,d) \wedge \neg\mathit{vegetarian}(p) \wedge \mathit{vegDish}(d)$
w_9	w_9	$\mathit{orders}(p,d) \wedge \neg\mathit{vegetarian}(p) \wedge \neg\mathit{vegDish}(d)$
w_{10}	w_{10}	$\neg\mathit{orders}(p,d) \wedge \neg\mathit{vegetarian}(p) \wedge \neg\mathit{vegDish}(d)$

MLN 3.3: *Model structure capable of representing a factorisation as in a directed model*

databases we may use). This is due to the fact that the learning problem is often ill-posed (Jain et al., 2007). Although the learning problem is a convex optimisation problem, it may not be *strictly* convex. In the case of MLN 3.3, there is an infinite number of weight vectors that represent precisely the same distribution – for a given number of entities. In particular, if the number of dishes is N_D, then for any $\delta \in \mathscr{R}$, the two weight vectors listed above are equivalent and represent precisely the same distribution over possible worlds. The effect of modifying the weight of the unit clause $\mathit{vegetarian}(p)$ can be cancelled out by applying an inverse modification to a set M of mutually exclusive and exhaustive formulas within which that unit clause appears – scaled, however, using the ratio between the number of groundings of the unit clause and the number of groundings of each of the formulas in M. With N_D and N_P as the number of dishes and people in the domain respectively, the ratio is $N_P/(N_P \cdot N_D) = 1/N_D$. It should be clear, however, that for $\delta \neq 0$, if N_D changes as we move to another domain of discourse, then so does the implied probability distribution. In our example, the marginal probability of being a vegetarian will, for instance, vary widely.

A maximum likelihood learner will have no preference of one weight vector over the other. By applying a form of regularisation (e.g. by applying MAP estimation using Gaussian 0-mean priors on the weights as proposed by Richardson and Domingos (2006)), we ensure that the optimisation process returns a unique solution. However, that solution is not necessarily the one we expect. Indeed, the solution that results can, as far as generalisation across domains is concerned, be regarded as arbitrary. Hence the soundness of shallow transfer cannot be taken for granted. Introducing additional knowledge in the form of constraints may be necessary in order for the learning process to determine weights that will ultimately represent what is intended. From a knowledge engineering perspective, this is an important point, for it suggests that the task of knowledge engineering in MLNs may require considerations that go beyond the selection of the "right" formulas even when the parameters are learnt from data.

3.4.5 Discussion

In this section, I have scrutinised a number of knowledge engineering questions that arise in Markov logic networks. While the collection is by no means complete, many issues that are of high practical relevance were addressed. Fundamental semantic perspectives were discussed and the considerations that are necessary for the representation of simple probabilistic properties with MLNs were made clear. With the probabilistic implication fallacy, one of the most common sources of error was explicitly treated, explaining the inherent predicament in softening logic as well as its resolution. Finally, conditions under which generalisation across domains can be problematic even for cases where models are learnt from data were pointed out.

The issues pertaining to shallow transfer call for a more thorough discussion, for they suggest that a model with a fixed set of parameters may be inadequate for the representation of many real-world scenarios, as the requirements imposed in the "restaurant scenario" do not seem far-fetched. We can view any statistical relational model as a representation of a stochastic process that embodies some well-defined characteristics about a particular scenario. Whether or not the application of rigid templates containing a fixed set of parameters can be considered sound regardless of the nature of the stochastic process that is to be represented is highly dependent on the way in which we interpret the domains to which we apply the model. Since one generally seeks to minimise statistical errors that result from small sample sizes, models are typically trained using large training databases. If we now instantiate a trained model for a smaller set of objects, the probability distribution indicated by the ground model can certainly be considered sound (even if it contains parameters that are in fact specific to the training data[6]), if we view that set of objects as a subset of the set of objects in the training database, i.e. if we implicitly assume that the instantiation contains additional objects so as to reach the number of objects present in the training database. In this mode of application, the training database must be viewed as the single domain of discourse in its entirety: Any smaller instantiation is merely an excerpt of it, and larger instantiations should be considered as invalid. In some cases, this may indeed be sensible, yet clearly, it ultimately defeats the very

[6] It should be noted that the use of multiple training databases as described in Section 3.3 does not in any way circumvent the phenomenon of models being specific to particular domains. The use of multiple independent databases will simply result in the model being specific to the "average" domain.

purpose of statistical relational models to describe general principles that should be applicable to arbitrary instantiations, since what is really being captured is but a (propositional) model that describes the training database, and any relational aspects are reduced to mere syntactic sugar.

If, however, the training database is not to be viewed as the sole universe of objects, then the nature of the stochastic process that we seek to model may dictate that parameters must indeed change with domain size, and models must explicitly take this into consideration. In the following sections, two solutions to this predicament are introduced: The first augments MLN with constraints on formula probabilities, which, in an online adaptation step, can be used to compute formula weights that will satisfy these constraints. Because this type of online adjustment can be computationally inefficient, the second solution involves *adaptive* Markov logic networks that allow weights to be directly represented as functions of domain-specific parameters, which can be learnt from data.

3.5 Markov Logic Networks with Probability Constraints

As we have seen, it may be impossible to represent certain invariants of distributions as we change the number of objects in the domain that is used to instantiate the relational model. To enable Markov logic networks to be used in cases that are subject to the kinds of combinatorial effects that cause the model to be affected undesirably, this section introduces an extension of the language that involves constraints on formula probabilities (cf. Jain et al., 2007). Specifically, *formula probability constraints* of the form $P(F_{R_k}) = p_k$ are added to the formalism, where F_{R_k} is a formula and $p_k \in [0,1]$ is a probability value.

For instance, consider MLN 3.4, which extends MLN 3.1 (page 41) with three probability constraints. In the absence of evidence, the marginal probabilities of the constrained atoms are to be fixed. Note that the probabilities in MLN 3.4 correspond to the probabilities that would in fact be indicated by MLN 3.1 if formulas 4–7 did not exist, since $w_1 = \log(0.5/0.5)$, $w_2 = \log(0.15/0.85)$ and $w_3 = \log(0.25/0.75)$.

$female(person)$
$vegetarian(person)$
$vegDish(dish)$
$friends(person, person)$
$orders(person, dish!)$

$w_1 = 0.000$ $\quad \forall p.\ female(p)$
$w_2 = -1.735$ $\quad \forall p.\ vegetarian(p)$
$w_3 = -1.099$ $\quad \forall d.\ vegDish(d)$
$w_4 = 1.400$ $\quad \forall p.\ female(p) \Rightarrow vegetarian(p)$
$w_5 = 1.300$ $\quad \forall p_1, p_2.\ friends(p_1, p_2) \wedge vegetarian(p_1) \Rightarrow vegetarian(p_2)$
$w_6 = 2.300$ $\quad \forall p, d.\ orders(p, d) \wedge \neg vegetarian(p) \Rightarrow \neg vegDish(d)$
$w_7 =$(hard) $\quad \forall p, d.\ orders(p, d) \wedge vegetarian(p) \Rightarrow vegDish(d)$

$P(female(p)) = 0.5$
$P(vegetarian(p)) = 0.15$
$P(vegDish(d)) = 0.25$

MLN 3.4: *Extended version of MLN 3.1, which adds formula probability constraints that require fixed marginal probabilities for the three attributes.*

3.5.1 Iterative Proportional Fitting

For any instantiation $M_{L,D}$ of an MLN L, we can impose such constraints on the respective joint distribution $P(X)$ using a fixed point iteration scheme known as the *iterative proportional fitting procedure* (IPFP) (Csiszár, 1975). This procedure allows to modify a joint distribution such that it satisfies a set $\mathscr{R} = \{R_k\}_{k=1}^m$ of probability constraints, a *probability constraint* on a distribution $P(X)$ being a probability distribution $R_k(X_{R_k})$ over $X_{R_k} \subseteq X$. IPFP guarantees that the distribution $P(X)$ is modified in such a way that the resulting distribution satisfies all of the given probability constraints and is maximally similar to $P(X)$ with respect to Kullback-Leibler divergence. The *Kullback-Leibler divergence* is a (non-symmetric) distance measure defined on probability distributions: For a discrete distribution $Q(X)$, the distance to $P(X)$ is given by

$$D_{\mathrm{KL}}(Q \mid\mid P) = \sum_{x \in \mathscr{X},\, Q(x)>0} Q(x) \cdot \log \frac{Q(x)}{P(x)}. \tag{3.22}$$

provided that the support of P includes the support of Q; otherwise the distance is $+\infty$.

The iterative proportional fitting procedure works by iteratively adjusting the distribution to fit one of the probability constraints in a round robin fashion. The initial distribution is $Q_0(X) = P(X)$. Then, in the i-th step, IPFP adjusts the (support of the) distribution to satisfy the k-th constraint $R_k(X_{R_k})$ (where $k = ((i-1) \mod m) + 1$ if m is the number of probability constraints) as follows:

$$Q_i(X) = Q_{i-1}(X) \cdot \frac{R_k(X_{R_k})}{Q_{i-1}(X_{R_k})} \tag{3.23}$$

Iterative fitting continues until convergence, which can always be achieved provided that the probability constraints are consistent (Csiszár, 1975).[7]

The semantics of formula probability constraints translate directly to probability constraints as considered by IPFP. With respect to a ground Markov random field $M_{L,D}$, we can interpret a formula probability constraint $P(F_{R_k}) = p_k$ in terms of F_{R_k}'s groundings. With H as the set of groundings of F_{R_k} for domain D, the formula probability constraint can be interpreted as follows:

$$\forall \widehat{F}_{R_k} \in H.\ P(\widehat{F}_{R_k} \mid M_{L,D}) = p_k \tag{3.24}$$

A set of formula probability constraints thus induces a set of ground formula probability constraints $\mathscr{R}_F$. Each constraint in $\mathscr{R}_F$ can be written as a pair $\langle \widehat{F}_{R_k}, p_k \rangle$. For $\langle \widehat{F}_{R_k}, p_k \rangle \in \mathscr{R}_F$, let X_{R_k} be the set of ground atoms appearing in $\widehat{F}_{R_k}$ and let $\mathscr{X}_{\widehat{F}_{R_k}} \subseteq \mathrm{dom}(X_{R_k})$ be the set of assignments satisfying $\widehat{F}_{R_k}$. We can then translate the ground formula probability constraint into a regular probability constraint by distributing the probability p_k across the assignments that satisfy $\widehat{F}_{R_k}$ whilst leaving the relative importance of assignments unchanged otherwise (and similarly for the complementary probability $1 - p_k$):

[7] We can consider consistency among the constraints themselves and with respect to the original distribution $P(X)$. Intuitively, a set of probability constraints is consistent within itself if a distribution exists that simultaneously satisfies all the constraints. For instance, the two constraints $P(A) = 0.5$ and $P(A \wedge B) = 0.7$ are inconsistent. Furthermore, a constraint R_k is inconsistent with respect to $P(X)$ if R_k demands that the probability of some event must be non-zero while the prior $P(X)$ indicates that it is zero.

$$R_k(X_{R_k} = x_{R_k}) = \begin{cases} p_k \cdot \frac{P\left(x_{R_k}\right)}{\sum_{x \in \mathcal{X}_{\widehat{F}_{R_k}}} P(x)} & \text{if } x_{R_k} \models \widehat{F}_{R_k} \\ (1 - p_k) \cdot \frac{P\left(x_{R_k}\right)}{\sum_{x \in \mathcal{X}_{\neg \widehat{F}_{R_k}}} P(x)} & \text{if } x_{R_k} \models \neg \widehat{F}_{R_k} \end{cases} \tag{3.25}$$

We can thus, in theory, translate the set of ground formula probability constraints $\mathcal{R}_F$ into a set of probability constraints $\mathcal{R}$ and apply IPFP in order to adapt the model, substituting, for computations within IPFP, P in Equation 3.25 with Q_{i-1}.

3.5.2 Fitting at the Model Level

Unfortunately, a direct application of IPFP is generally infeasible, because it requires us to represent the full-joint distribution over $\mathcal{X}$ explicitly: Applying Equation 3.23 requires an update step for each element of $\mathcal{X}$. Therefore, it seems appropriate to apply iterative fitting not to the full-joint distribution itself but rather to a model representing the distribution more compactly.

Consider a ground Markov random field $M_{L,D} = \langle X, G \rangle$ and the distribution $P(X)$ it represents. For each constraint $\langle \widehat{F}_{R_k}, p_k \rangle \in \mathcal{R}_F$, we add to G the ground formula $\widehat{F}_{R_k}$ with weight $\widehat{w}_{R_k}$; Initially, we set $\widehat{w}_{R_k} \leftarrow 0$ for all constraints and thus obtain as the initial distribution $Q_0(X) = P(X)$.

I will now show that the transition described in Equation 3.23 (for the case where the constraint to fit takes the form 3.25) can be achieved by modifying only the respective weight $\widehat{w}_{R_k}$. Let the probability of $\widehat{F}_{R_k}$ that is indicated by the current model be

$$q := \frac{s(\widehat{F}_{R_k})}{s(\widehat{F}_{R_k}) + s(\neg \widehat{F}_{R_k})}, \tag{3.26}$$

while the constraint requires the probability to become p_k. By adding $\log(\lambda)$ to $\widehat{w}_{R_k}$, we scale the sum of exponentiated sums of weights of possible worlds satisfying $\widehat{F}_{R_k}$,

i.e. $s(\widehat{F}_{R_k})$, by a factor of λ whilst leaving the relative importance of affected possible worlds unchanged. Therefore, we need only to find λ such that

$$p_k = \frac{s(\widehat{F}_{R_k})\lambda}{s(\widehat{F}_{R_k})\lambda + s(\neg\widehat{F}_{R_k})}. \tag{3.27}$$

Combining Equations 3.26 and 3.27 yields

$$\lambda = \frac{p_k \cdot (1-q)}{(1-p_k)\cdot q}. \tag{3.28}$$

This factor indeed corresponds to the modification that results from inserting Equation 3.25 into Equation 3.23,

$$\begin{aligned} Q_i(x) &= Q_{i-1}(x) \cdot \frac{R_k(x_{R_k})}{Q_{i-1}(x_{R_k})} \\ &= Q_{i-1}(x) \begin{cases} \frac{p_k}{Q_{i-1}(x_{R_k})} \cdot \frac{Q_{i-1}(x_{R_k})}{\sum_{x' \in \mathscr{X}_{\widehat{F}_{R_k}}} Q_{i-1}(x')} & \text{if } x \models \widehat{F}_{R_k} \\ \frac{(1-p_k)}{Q_{i-1}(x_{R_k})} \cdot \frac{Q_{i-1}(x_{R_k})}{\sum_{x' \in \mathscr{X}_{\neg\widehat{F}_{R_k}}} Q_{i-1}(x')} & \text{if } x \models \neg\widehat{F}_{R_k} \end{cases} \\ &= Q_{i-1}(x) \begin{cases} \frac{p_k}{q} & \text{if } x \models \widehat{F}_{R_k} \\ \frac{1-p_k}{1-q} & \text{if } x \models \neg\widehat{F}_{R_k} \end{cases} \end{aligned} \tag{3.29}$$

since the combination of the factors p_k/q and $(1-p_k)/(1-q)$ for $x \models \widehat{F}_{R_k}$ and $x \models \neg\widehat{F}_{R_k}$ respectively can be equivalently replaced by a single factor $p_k/(1-p_k)\cdot(1-q)/q$ for $x \models \widehat{F}_{R_k}$. We can thus achieve the transition from Q_{i-1} to Q_i by applying the update

$$\widehat{w}_{R_k} \leftarrow \widehat{w}_{R_k} + \log\left(\frac{p_k \cdot (1-q)}{(1-p_k)\cdot q}\right) \tag{3.30}$$

and perform iterative fitting at the level of model parameters. For the case where $p_k = 1$ or $p_k = 0$, the update would be ill-defined, yet in this case, the constraint can simply be satisfied by making $\widehat{w}_{R_k}$ a hard (positive or negative) weight. For $q \in \{0,1\}$, the probability constraint is inconsistent with the model, rendering a fit impossible.

This leads to an algorithm that works as follows. In each step, we fit one of the ground formula probability constraints by applying the update rule 3.30. This involves running an inference algorithm in order to compute q, the current probability of the

ground formula that is to be fitted. Once the procedure has converged, i.e. all constraints are satisfied and no more weight changes are necessary, we have an adapted model that satisfies the desired constraints, which we can subsequently use to update beliefs.

In the following, the algorithm described above is referred to as *IPFP-M* (IPFP at the model level). Pseudocode is provided in Algorithm 3. First, the model is augmented with additional weighted formulas, then the actual fitting is performed based on the inference method *infer*, which takes a Markov random field as the first argument and a set of queries as the second. To determine convergence, some small maximum deviation ε is assumed: Once the absolute difference between the desired probability p_k and the actual probability that is indicated by the model (as computed via inference) is at most ε for all constraints, the procedure is stopped. This notion of convergence allows us, in particular, to use an approximate method such as MC-SAT as the inference method *infer*.

Algorithm 3 IPFP-M($M_{L,D} = \langle X, G \rangle$, $\mathscr{R}_F$, ε, *infer*)

for $\langle \widehat{F}_{R_k}, p_k \rangle \in \mathscr{R}_F$ **do**
  add $\langle \widehat{F}_{R_k}, \widehat{w}_{R_k} = 0 \rangle$ to G
$i \leftarrow 1$
repeat
  $k \leftarrow ((i-1) \mod |\mathscr{R}_F|) + 1$
  $\langle \widehat{F}_{R_k}, p_k \rangle \leftarrow k$-th element of $\mathscr{R}_F$
  results $\leftarrow$ *infer*$(M_{L,D}, \{\widehat{F}_{R_j} \mid \langle \widehat{F}_{R_j}, p_j \rangle \in \mathscr{R}_F\})$
  $q \leftarrow$ results$[\widehat{F}_{R_k}]$
  $\widehat{w}_{R_k} \leftarrow \widehat{w}_{R_k} + \log(p_k/(1-p_k) \cdot (1-q)/q)$
  $\delta_{\max} \leftarrow \max\{\text{abs}(\text{results}[\widehat{F}_{R_j}] - p_j) \mid \langle \widehat{F}_{R_j}, p_j \rangle \in \mathscr{R}_F\}$
  $i \leftarrow i + 1$
until $\delta_{\max} \leq \varepsilon$
return $M_{L,D}$

Adjusting Weights of Abstract Formulas. Instead of fitting every ground instance of every constrained formula separately, potentially introducing a large number of ground formula probability constraints, we can try to directly fit the weight of an abstract constraint, i.e. the weight of a formula that contains free/universally quantified variables. In many scenarios in which we might want to apply probability constraints, this would seem to be a most natural thing to do. For instance, to constrain

the probability of an attribute, it should suffice to adapt the weight of an abstract unit clause such $female(p)$, for the role of the weighted unit clause is precisely to influence the marginal distribution of the attribute. Indeed, assuming prototypical indifference between the ground instances of a formula in the absence of evidence, we can consider any grounding as a prototype – and adjusting its probability should simultaneously result in the same adjustment for all the other groundings. Specifically, we can compute the probability with which any given ground instance of the abstract formula holds and use this probability to adjust the weight of the abstract formula. However, we need to take into account the possibility of different ground instances of a formula influencing each other. Only if the individual groundings are independent will the update rule 3.30 result in the desired adjustment, for Equation 3.27 assumes that the impact of adding $\log(\lambda)$ – beyond its influence on the particular ground instance – is equal for the positive and negative case (i.e. $s(\widehat{F}_{R_k})$ and $s(\neg\widehat{F}_{R_k})$ should otherwise be affected in the same way). If the individual groundings of an abstract formula influence each other, then the update will result in an overshoot of the target probability p_k, which will have to be corrected in a reverse update of a subsequent iteration, which, however, will again be overshot, etc. The procedure will typically converge, but, depending on the degree of mutual influence, the number of steps required to reach a given threshold ε can be greatly increased.

Example. The effect of formula probability constraints in practice is illustrated in Table 3.1, where inference results obtained with MLN 3.1 are compared to the results obtained with MLN 3.4 on three instantiations. Exact inference was used to compute the results and was furthermore applied as the underlying inference method used in IPFP-M for MLN 3.4. As the example shows, the formula probability constraints affect the marginal probabilities of attributes as expected, and furthermore have significant bearing on conditional probabilities that depend upon them.

Related Work. Peng and Ding (2005) have previously proposed a scheme for modifying Bayesian networks by probability constraints without changing the structure of the model. Their approach is also based on the iterative proportional fitting procedure. The requirement that the structure remain unchanged constitutes an additional constraint, which, after having applied a full pass of standard IPFP over the actual probability constraints, can be handled by extracting the structure-specific parameters (i.e. marginal and conditional probabilities) from the explicitly represented full-

	$D^{1,2}$		$D^{2,2}$		$D^{3,2}$	
	3.1	3.4	3.1	3.4	3.1	3.4
$P(female(P_1))$	0.223	0.500	0.211	0.500	0.204	0.500
$P(female(P_1) \mid orders(P_1, D_1) \wedge vegDish(P_1))$	0.421	0.710	0.409	0.711	0.389	0.715
$P(vegetarian(P_1))$	0.084	0.150	0.044	0.150	0.020	0.150
$P(vegetarian(P_1) \mid orders(P_1, D_1) \wedge vegDish(P_1))$	0.738	0.803	0.700	0.814	0.633	0.827
$P(vegDish(D_1))$	0.182	0.250	0.119	0.250	0.071	0.250
$P(vegDish(D_1) \mid \neg vegetarian(P_1) \wedge orders(P_1, D_1))$	0.032	0.044	0.020	0.040	0.012	0.037
$P(orders(P_1, D_1))$	0.500	0.500	0.500	0.500	0.500	0.500
$P(orders(P_1, D_1) \mid vegetarian(P_1) \wedge vegDish(D_1))$	0.800	0.762	0.840	0.732	0.871	0.676
$P(friends(P_1, P_2))$			0.492	0.482	0.496	0.485
$P(friends(P_1, P_2) \mid vegDish(D_1) \wedge orders(P_1, D_1) \wedge orders(P_2, D_1))$			0.459	0.470	0.458	0.474

Table 3.1: *Comparison of inference results obtained with MLN 3.1 and MLN 3.4 on three instantiations. $D^{n,m}$ denotes a domain with n people, $\{P_1, \ldots, P_n\}$, and m dishes, $\{D_1, \ldots, D_m\}$.*

joint distribution that was computed by IPFP. In the particular context of MLNs, such structural constraints are not required, because every ground MRF is but a temporary model which serves only as a vehicle for inference. Other related work has explored first-order probabilistic languages that base their entire semantics on the notion of probability assertions, building upon the principle of maximum entropy (Loh et al., 2010; Fisseler, 2008).

3.6 Adaptive Markov Logic Networks

From the perspective of knowledge engineering, the problem of building a model that will represent particular probabilistic properties regardless of the concrete domain for which the model is instantiated can either be dealt with explicitly – through the declaration of probability constraints – or implicitly. In this section, the problem shall be addressed from the perspective of learning. Therefore, methods capable of deriving the relationship between model parameters and properties of an instantiation from data will be introduced. The resulting formalism, called *adaptive Markov logic networks* (Jain, Barthels, and Beetz, 2010), thus seeks to capture the way in which parameters are to be dynamically adapted to a particular instantiation – which includes, but is not limited to, the representation of adjustments that will bring about invariant properties across domains.

Statistical relational representation formalisms underlie the assumption that the parameters that were once learnt with a (sufficiently large) training database hold for every other domain, regardless of its size (or other properties). Unfortunately, this as-

sumption does not always hold. One particularly common trait of relational domains that typically results in a violation of the assumption is the presence of cardinality restrictions, i.e. restrictions that constrain the number of relational partners of an entity. Such restrictions are inherent properties of relational domains – as is evident from their abundance in e.g. entity-relationship models (Abrial, 1974) – and therefore need to be integrated into any statistical model that seeks to capture relational properties accurately. While, in principle, the cardinality constraints themselves can be integrated into some of the more expressive statistical relational models (including Markov logic networks), their complex interactions with probabilities pertaining to the constrained relations are largely neglected, severely limiting the models' capability of performing *link prediction* – one of the key tasks in statistical relational learning – within reasonable bounds of accuracy as domain size varies.

The impact of cardinality restrictions shall be demonstrated using a very simple scenario that deals with students taking courses at a university. Students can be enrolled in the Bachelor's programme or the Master's programme; beginner-level courses are differentiated from advanced-level courses. Students of either programme can pick courses of either level while, however, being inclined to pick courses that are appropriate with respect to their level of advancement, the probability of a student taking a particular course will depend on the type of the student and the level of the course. Assuming that every student is required to take a particular number of courses, one can typically observe that if a statistical relational model is trained on a database with a particular number of courses and students, the probabilities indicated for the relation linking students to the courses they take are specific to the cardinalities that were present in the training data. Therefore, link prediction tends to yield unsatisfactory results when the learned model is applied to domains that differ (in terms of size) from the training database.

3.6.1 The Impact of Cardinality Restrictions

For a model to be applicable to any particular domain of discourse, it is a prime requirement that all the principles being represented in the model are in fact *independent* of properties of the domain – in particular, its size. As soon as this is not truly the case, any model is bound to represent probability distributions that may diverge

arbitrarily from the true distribution as we apply it to domains that differ from the domain it was trained with.

We can reasonably view a statistical relational model as a representation of a stochastic process that embodies well-defined characteristics about a particular scenario. If instantiations containing variable numbers of entities are to be adequately covered by the relational model, then the nature of the stochastic process that we seek to model may dictate that parameters must change with domain size. A straightforward example of such a case is a process that contains relations that are subject to cardinality restrictions and where any world that is generated by it should be considered as *self-contained*. In other words, consider a process that demands that the number of objects to which particular objects are to be related is fixed or otherwise constrained, and, for each object, all the relational partners are always part of the universe of objects being generated. As we will see, this rather simple condition is sufficient for a model with fixed parameters to fail to generalise as expected.

The presence of cardinality restrictions in a self-contained domain implies that if the set of potential relational partners changes, then so must the probability of there being a relation. For instance, if we consider the relation $\mathit{isParentOf}(x, y)$, which holds if x is a parent of y, then, provided that every child's parents are in fact part of the instantiation, the probability of there being a relation between an arbitrary child-parent pair decreases as the number of parents in the domain increases (it is $2/n$ if n is the number of potential parents). Of course, without further restrictions, an explicit representation of the cardinality constraint that limits the number of parents to 2 is sufficient in order to correct the marginal probability of the *isParentOf* relation. Since Markov logic subsumes first-order logic, cardinality restrictions can easily be formulated: For an arbitrary binary relation $\mathit{rel}(x, y)$, we could state that, for each x, there should be exactly c objects to which x is to be related as follows:

$$\exists\, y_1, \ldots, y_c.\ \bigwedge_{i=1}^{c} \left(\mathit{rel}(x, y_i) \wedge \bigwedge_{j=1}^{i-1} y_i \neq y_j \right) \wedge \neg\exists\, y_{c+1}.\ \mathit{rel}(x, y_{c+1}) \wedge \bigwedge_{j=1}^{c} y_{c+1} \neq y_j \tag{3.31}$$

While such constraints succeed at correcting marginal probabilities of constrained relations, they do not suffice in cases where the relation is in fact dependent on, for example, attributes of the related objects (a quite common case, as we can imagine stochastic processes to first generate objects and then establish relations between them based on their properties). The reason is that the introduction of such dependencies requires that, for each configuration of the attributes the relation depends on, we are required to capture the proper ratio between the case where the relation occurs and where it does not occur. Modelling only the ratios between cases where the relation is to hold, which are indeed size-invariant, is, unfortunately, not sufficient.

To illustrate this, consider the simplest of cases, where we have two types of entities, children and parents, that (with probability $1/2$) have some Boolean property (e.g. being tall), which influences the likelihood of there being a relation. We know that each child must be related to exactly two parents. We could reasonably represent this scenario using the marginal distribution of the property for parents and children, the conditional distribution of the relation given the properties of both entities, plus the cardinality restriction. Assume that the probability p_1 with which a child is related to a parent given that both have the property or both do not is four times as high as the probability p_2 where one has the property and the other does not. Given a stochastic process that generates data reflecting the above, we can train models on databases of variable size and then instantiate the obtained models for a particular set of objects we are interested in. Exemplarily, assume that there is a particular child C and a parent P who both have the property as well as another child and three further parents about whom we do not know anything. We ask for the probability q of C being related to P in comparison to C being related to one of the three other parents (q'). Table 3.2 summarises results obtained for such an experiment. Note that even though the cardinality constraint (which implies $q + 3q' = 2$) was explicitly represented and the ratio $p_1 : p_2$ is indeed $4 : 1$ in each case, the probability q varies depending on the training database and, most importantly, diverges considerably from the true result – simply because the ratios $p_1 : (1 - p_1)$ and $p_2 : (1 - p_2)$ can be regarded as arbitrary with respect to the instantiations that were considered.

	M_{10}	M_{100}	M_{200}	*Correct/Extrapolation*
p_1	0.1600	0.0160	0.0080	0.8000
p_2	0.0400	0.0040	0.0020	0.2000
q	0.6237	0.6044	0.6034	0.8290
q'	0.4588	0.4652	0.4655	0.3903

Table 3.2: *Parameters obtained from training databases of various sizes and corresponding inference results, where M_N indicates a model derived from a database with N children and $2N$ parents.*

3.6.2 Definition

Let us now introduce a generalisation of Markov logic networks that addresses the issues described above. Since model parameters are to be dependent on attributes of an instantiation, the respective attributes must be formalised: Let $\{A_1, \ldots, A_k\}$ be the set of attributes that we consider. $\mathscr{A} := \mathrm{dom}(A_1) \times \cdots \times \mathrm{dom}(A_k)$ is thus the set of possible attribute configurations. An *adaptive Markov logic network* (AMLN) is defined as a set $\mathscr{L}$ of pairs $\langle F_i, W_i \rangle$, where F_i is a formula in first-order logic and $W_i : \mathscr{A} \to \mathbb{R}$ is a function that maps from the vector of attributes of the instantiation to the parameter domain. An AMLN can therefore be regarded as a template for the construction of an MLN: For any domain D with attribute vector $a_D \in \mathscr{A}$, we obtain a Markov logic network $L_{\mathscr{L},D} = \{\langle F_i, W_i(a_D) \rangle\}$ specific to D, which we can then apply in order to perform inference.

In the remainder of this work, the attributes A_i considered are strictly cardinalities of sets of objects as they appear in the instantiation, i.e. $\mathrm{dom}(A_i) = \mathbb{N}$, but it is conceivable that parameters could be dependent on other attributes. For instance, parameters could depend on the point in time for which we instantiate the model: We could learn from several historical records of, say, a social network, learn how dependencies changed over time and then instantiate the adaptive model for a point in time for which we do not (yet) have any data.

3.6.3 Parameter Learning

The key to the learning of AMLNs that will indeed generalise across domains with varying attribute configurations is to adjust the learning algorithms to learn weight

functions rather than scalar values. This naturally requires us to learn from multiple training databases; the combination of several training databases into a single, very large training database is clearly not adequate. We need to explicitly consider the effects of changing the domain of discourse.

3.6.3.1 Constrained Learning from Individual Databases

First of all, however, we need to take into consideration the fact that the parameter learning problem in MLNs may be ill-posed, as already explained in Section 3.4.4. Consider the model structure in MLN 3.5. The predicates *sT* and *cT* stand for *student type* and *course type* respectively. The structure is a basic model of our example scenario where the relation *takes* depends on the attributes of the related objects. As in the example from Section 3.4.4, the ill-posedness of parameter learning stems from the underdeterminism that allows us to obtain, for particular cardinalities of the sets of students and courses, precisely the same probability distribution using an infinite number of weight vectors, as the choice of δ in MLN 3.5 is arbitrary: If N_c is the number of courses, then for any $\delta \in \mathbb{R}$, the weight vectors w and w' are equivalent.

If unit clauses such as $sT(s, BSc)$ are indeed to be used to capture marginal probabilities, we must therefore ensure that, for a group of mutually exclusive and exhaustive unit clauses (such as formulas 1 and 2 in MLN 3.5), the ratio between any two exponentiated formula weights within the group matches the empirical relative frequencies in the training database. For instance, if the fraction of Bachelor students is 2/3, we should require that $\exp(w_1)/\exp(w_2) = (2/3)/(1/3) = 2$.

Therefore, in order to eliminate the underdeterminism, it is sensible to employ a controllable preprocessing stage during parameter learning, which predetermines the weights of, in particular, unit clauses. A straightforward choice of weights that guarantees the required ratios is to set the weight of a unit clause to the logarithm of its relative frequency in the data. Some weights being fixed a priori, the preprocessing stage thus reduces the weight vector to the projection that includes only the remaining weights. For the resulting lower-dimensional weight vector, we can simply apply the standard learning methods described in Section 3.3 in the second stage, the optimisation stage. Since the preprocessing stage already achieves that the probability

Predicate Declarations

$sT(student, studentType!)$
$cT(course, courseType!)$
$takes(student, course)$

Domain Definitions

$studentType = \{BSc, MSc\}$
$courseType = \{Beg, Adv\}$

Weighted Formulas

w	w'	
w_1	$w_1 + \delta$	$sT(s, BSc)$
w_2	w_2	$sT(s, MSc)$
w_3	w_3	$cT(c, Beg)$
w_4	w_4	$sT(c, Adv)$
w_5	$w_5 - \delta/N_c$	$takes(s,c) \wedge sT(s, BSc) \wedge cT(c, Adv)$
w_6	$w_6 - \delta/N_c$	$\neg takes(s,c) \wedge sT(s, BSc) \wedge cT(c, Adv)$
w_7	$w_7 - \delta/N_c$	$takes(s,c) \wedge sT(s, BSc) \wedge cT(c, Beg)$
w_8	$w_8 - \delta/N_c$	$\neg takes(s,c) \wedge sT(s, BSc) \wedge cT(c, Beg)$
w_9	w_9	$takes(s,c) \wedge sT(s, MSc) \wedge cT(c, Adv)$
w_{10}	w_{10}	$\neg takes(s,c) \wedge sT(s, MSc) \wedge cT(c, Adv)$
w_{11}	w_{11}	$takes(s,c) \wedge sT(s, MSc) \wedge cT(c, Beg)$
w_{12}	w_{12}	$\neg takes(s,c) \wedge sT(s, MSc) \wedge cT(c, Beg)$

MLN 3.5: *Model structure for students taking courses at a university, which exhibits underdeterminism*

of the respective unit clauses (formulas) is well-represented within the model, the actual optimisation stage is, intuitively, free to focus on the influence of these unit clauses on other atoms and must not concern itself with representing the respective aspects of the distribution.

Of course, it is necessary to choose the formulas to which the preprocessing stage is applied with care. In general, it is sensible to apply it to unit clauses pertaining to a priori independent attributes. As the intuition behind the second stage suggests, having a fixed weight for all potential unit clauses one could potentially include in the model is inappropriate. For instance, in our example, the learning process is to determine precisely how the relation *takes* is influenced by other variables and therefore it would not have been advisable to include a corresponding unit clause and have its weight determined by the preprocessing stage. The learner therefore uses explicit declarations in the model structure to determine the set of formulas that the preprocessing stage is to be applied to.

3.6.3.2 Learning from Multiple Databases

Let us now address the issue of learning dependencies of parameters on attributes of the instantiation from multiple databases, using the learning procedure outlined above as a subroutine. Let $\{D_1, \ldots, D_k\}$ be the set of training databases. The first step is to apply, to each training database D_j, the learning procedure described in the previous section. This yields a vector of weights $\vec{w}_i$ for each formula F_i appearing in the AMLN structure, where the entries are the weights obtained for the individual training databases:

$$\vec{w}_i = \begin{pmatrix} (\vec{w}_i)_1 & (\vec{w}_i)_2 & \cdots & (\vec{w}_i)_k \end{pmatrix} \tag{3.32}$$

Since we consider models where the objects are typed, each training database contains a set of objects for each of the types. Let $(D_j)_l$ be the set of objects belonging to the l-th type in the j-th training database. $(D_j)_l$ is called a *subdomain* of the domain that is given by D_j.

Reducing the set of domain attributes to cardinalities of subdomains, we consider the weight function W_i of the i-th formula to be a function of sizes of the subdomains referenced in F_i, i.e. its signature is given by $W_i : \mathbb{N}^{s_i} \to \mathbb{R}$, where s_i is the number of subdomains referenced in F_i. Every weight function can be expressed as a linear combination of a set of basis functions. First, let us consider the question of choosing these basis functions reasonably. Recall the factor representation of the full-joint distribution of a ground Markov random field:

$$P_w(X = x) = \frac{1}{Z} \exp\left(\sum_i w_i n_i(x)\right) = \frac{1}{Z} \prod_i \lambda_i^{n_i(x)} \tag{3.33}$$

where $\lambda_i = \exp(w_i)$. To capture the dependency on subdomain sizes, each λ_i is viewed a function over some size m (i.e. $m = |(D_j)_l|$ for some j, l). If we want the above product to be capable of representing a factorization that includes relative frequencies α/m (e.g. 6 out of 24), the basis functions need to be capable of representing $\lambda_i(m) := \alpha m^{-1}$. By choosing as basis functions

$$\phi_0 \equiv 1, \qquad \phi_1(m) := \log(m), \qquad \phi_2(m) := m \tag{3.34}$$

any monomial or exponential λ_i can be represented:

- $W_i(m) = \log(\lambda_i(m)) = \log(\alpha m^c) = \log\alpha + c\log m$ (3.35)
- $W_i(m) = \log(\lambda_i(m)) = \log(\alpha c^m) = \log\alpha + m\log c$ (3.36)

Similarly, should a formula depend on more than one subdomain, frequencies over products of subdomain sizes can be handled as follows:

$$\begin{aligned} W_i(m_1, m_2) &= \log(\lambda_i(m_1, m_2)) = \log(\alpha m_1^{c_1} m_2^{c_2}) \\ &= \log\alpha + c_1 \log m_1 + c_2 \log m_2 \end{aligned} \tag{3.37}$$

Including faster-growing classes of functions is unlikely to be advantageous. One might add more slow-growing basis functions, but the gain is questionable, especially since the use of too many basis functions may result in overfitting.

To learn the actual weight function W_i, we need to find coefficients $\alpha_{\cdot,\cdot}^{(i)}$ such that the combined basis functions approximate the learnt weights $\left(\vec{w}_i\right)_j$ as well as possible. For each formula F_i and training database D_j, we have

$$\left(\vec{w}_i\right)_j = \phi_0 \alpha_{0,\phi_0}^{(i)} + \sum_{l \in \mathscr{I}_i} \sum_{k=1}^{2} \phi_k(|(D_j)_l|)\alpha_{l,\phi_k}^{(i)} + \left(e_i\right)_j \tag{3.38}$$

where $\mathscr{I}_i$ is the set of indices of subdomains appearing in formula F_i and $(e_i)_j$ is an error term. The overall objective is to minimise the quadratic norm of the vector of errors $E_i = (\left(e_i\right)_j)$. Equation 3.38 is a system of linear equations. Let x_i be the vector of coefficients, i.e. $x_i = (\alpha_{\cdot,\cdot}^{(i)})$, and let A_i be the matrix of basis functions evaluated at the relevant subdomain sizes, sequentially taken from all training databases. Assuming canonical ordering of entries in the matrix and vector, Equation 3.38 can be rewritten as $A_i x_i \approx \vec{w}_i$ and one can use standard linear curve fitting methods to obtain an optimal approximation vector x_i that parametrises the weight function W_i.

3.6.4 Explicit Cardinality Constraints

In the examples considered in this section, relations are subject to cardinality restrictions. Since the use of existential quantification to express cardinality restrictions as

in Equation 3.31 is usually prohibitively expensive in practice – owing to the conjunctive normal form (CNF) conversion, which, unfortunately, results in an exponential blow-up of the representation[8] – the implementation of AMLNs was extended with a language construct that represents such restrictions more explicitly:

$$count(rel(x_1, \ldots, x_n) \mid x_{i_1}, \ldots, x_{i_m}) \in S \tag{3.39}$$

where $m \leq n$ and $S \subset \mathbb{N}$, the semantics being that if the parameters that are given by the index set $\{i_1, \ldots, i_m\}$ are bound to some fixed vector of constants, the number of bindings for the remaining parameters for which the relation holds true is required to be in the set S.

In the context of MLNs, a (weighted) *count* constraint defines, for each possible grounding of the fixed parameters, a feature in a ground Markov random field that corresponds to the clique connecting the variables (ground atoms) that one obtains by grounding the remaining parameters of the relation predicate; this is analogous to the clique implied by Equation 3.31.

The algorithm MC-SAT (see Section 3.2.1.2) is used for all inference tasks. Fortunately, the algorithm is easily extended to support the notion of explicit cardinality constraints. With respect to MC-SAT, a *count* constraint can be treated just like any regular logical constraint; it has an attached weight and we can easily check whether a given state satisfies it. Since MC-SAT requires weights to be positive and arbitrary weights could be attached to *count* constraints, we may need to negate *count* constraints beforehand. To negate a *count* constraint, we invert the set of counts S, i.e. we use as the new set of counts the complement $\mathbb{N} \setminus S$.

Furthermore, we need to make changes to the SampleSAT algorithm in order to enable it to find assignments that satisfy *count* constraints. The SampleSAT algorithm extends the WalkSAT algorithm by randomly performing simulated annealing-type moves which inject randomness, enabling SAT solutions to be sampled near-uniformly (Wei et al., 2004). Simulated annealing-type moves are largely unaffected by the introduction of *count* constraints; we only need to incorporate into the delta cost of the random move (which is used to probabilistically decide whether the move is actually

[8] Although a CNF conversion is not an unavoidable precondition for an implementation of Markov logic networks, it is generally necessary to support inference methods such as MC-SAT.

Cardinalities

$\text{count}(cT(c, Beg)) = 6$
$\text{count}(cT(c, Adv)) = |\text{course}| - 6$
$\text{count}(takes(s, c) \mid s) = 6$

Attribute Probabilities

$sT(s, BSc)$	2/3
$sT(s, MSc)$	1/3

Course Participation Preferences

	BSc	*MSc*
Beg	4/5	1/3
Adv	1/5	2/3

Table 3.3: *Properties of the stochastic process that generates students, courses and the course participation relation*

taken) the number of *count* constraints that would become unsatisfied or satisfied as a result of the move. For WalkSAT moves, we pick one of the yet unsatisfied constraints and then flip the truth values of one or more ground atoms that will satisfy the constraint and cause as few other constraints to be unsatisfied as possible. If an unsatisfied *count* constraint is selected, the current state will either satisfy too few atoms from the set Y of ground atoms appearing in the ground *count* constraint or too many, i.e. we must set one or more ground atoms that are currently false to true or vice versa. In either case, we greedily choose the ground atom(s) with the most favourable delta cost of newly satisfied minus newly unsatisfied constraints and flip their truth values. If the current number of true ground atoms from Y is between two values from the constraint's set of counts S, we consider the cost of both variants and choose the direction in which to move accordingly.

3.6.5 Experiments

In the following, the problems pertaining to shallow transfer – and, in particular, their influence on link prediction – are studied in more detail. Furthermore, we shall see how the novel methodology of AMLNs assists in solving them. To illustrate the gravity of these problems, it is sufficient to look at the very simple example scenario involving students taking courses at a university, which was introduced earlier. Specifically, consider the stochastic process characterised by Table 3.3 for this scenario. By clearly defining the properties of the stochastic process, we can easily evaluate any model's ability to reflect these properties – once it has been appropriately trained on training databases generated by the process.

The process indicates that 2/3 of all students are enrolled in the BSc programme, the remaining 1/3 in the MSc programme. Furthermore, every student is required to take exactly six courses out of the ones on offer: There are always six beginner-level courses; all other courses are advanced-level courses. The choice of courses taken by a student depends on both the student's status and the courses' levels. For example, if a course is taken by an MSc, it is four times as likely to be an advanced course than a beginner-level course. Since all students take exactly six courses, the probability of a course being taken by some student varies with domain size. It is important to note that the right part of Table 3.3 is *not* a conditional distribution that could simply be included in a directed model, because the probability of a course having a particular type is affected by all the students' relations to the course. In the following, the goal is to learn and represent the properties of this stochastic process in such a way that the effects resulting from an application of the model to instantiations of variable size are taken into account.

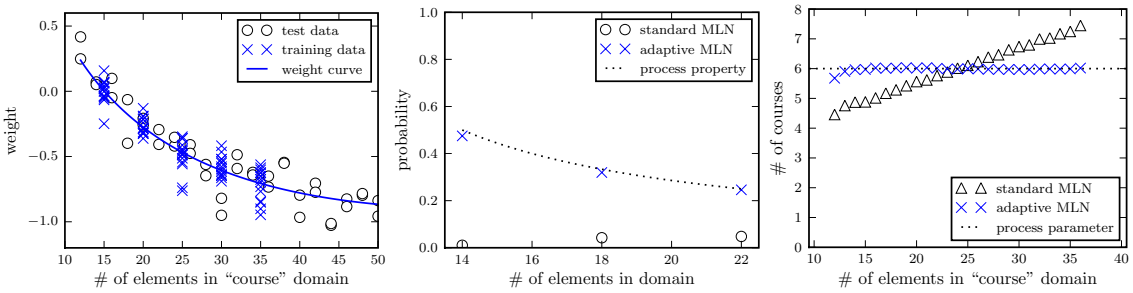
MLN 3.5 is used as the model structure, extended only with the following hard cardinality constraints, which correspond directly to the properties of the process:

$$\begin{aligned} &\mathit{count}(\mathit{takes}(s,c)\,|\,s) \in \{6\}. \\ &\mathit{count}(\mathit{cT}(c,\mathit{Beg})) \in \{6\}. \end{aligned} \tag{3.40}$$

Note that the structure in MLN 3.5 on its own would be ideal for a process where there is no cardinality restriction on the relation *takes*, for it is capable of representing precisely a direct factorization of the full-joint probability distribution as it would appear in a model derived via knowledge-based model construction.

Weight Functions. To evaluate the learning of weight functions, the AMLN was trained on a set of databases of five different sizes, using multiple databases for each size to compensate for the stochasticity of the process model. The number of courses in training databases ranged from 12 to 36, while the number of students was fixed at 45. In total, the training databases contained data on 3,375 students and 1,875 courses.

The appropriateness of the basis functions proposed in Section 3.6.3.2 is confirmed by the exemplary results shown in Figure 3.3a, which indicates that even for domain sizes that were not explicitly trained for, the weights are near-optimal. Thus, the



(a) **(b)** **(c)**

Figure 3.3: *Experiments with adaptive Markov logic networks: (a) weight curve of* $takes(s,c) \wedge sT(s,MSc) \wedge cT(c,Adv)$*; (b) probability of a particular Master student taking a particular advanced course given only the existence of further courses; (c) expected number of courses taken by a student depending on the number courses offered (Jain, Barthels, and Beetz, 2010)*

dynamics of the stochastic process appear to be adequately represented in the weight function that was learned.

Link Prediction. Having learnt an AMLN for the example domain, the next step is to evaluate it in terms of how well the model reflects the properties of the process for any given domain size. Results are compared to standard MLNs trained on a single database exhibiting *precisely* the expected values of the stochastic process (using a deterministic generator process to guarantee this), and its size was equivalent to that of the unification of all training databases that were used to induce the AMLN. All models used the same structure including cardinality constraints.

Since the task of link prediction is most critical within the context of self-contained worlds and constrained relations, the probability of a student taking a particular course was inferred, given the attributes of both the course and the student – for domains containing a variable number of additional courses (about which no further evidence was provided). The results are summarized in Figure 3.3b. The AMLN accurately represents the process property which demands that as the number of courses increases, the probability of an *MSc* student taking any given advanced-level course decreases appropriately (it is $\frac{2}{3} \cdot 6/(N_c - 6)$ if N_c is the number of courses in the domain). The standard MLN, on the other hand, does not capture this fundamental process property. Given what previous analyses have suggested, this was to be expected (since the MLN was trained on a database containing a number of courses that is

far greater than the number for which the model was instantiated), yet the degree to which the predictions diverge from the desired results is notable (the relative error exceeds 90%).

It is furthermore interesting to note that the AMLN learning approach is capable of capturing properties of the underlying stochastic process even if the model structure is incomplete. In particular, if we remove all cardinality constraints from the model, the AMLN still correctly estimates at least the expected values that these cardinality constraints imply: Applying link prediction to evidence databases containing ten students and a varying number of courses, providing information on all attribute values as they were generated by the stochastic process model, we observe that the expected number of courses taken by a student is predicted at almost exactly six by the AMLN for all domain sizes, as shown in Figure 3.3c. A standard MLN, which was induced using a (perfect) training database containing 24 students, predicts the desired outcome only for the domain size it was trained for.

Classification. Apart from probabilities pertaining to links, we can (as long as the *entire* relational skeleton is given) expect standard MLNs to accurately capture dependencies between class attributes and any predictor attributes. Therefore, the classification performance of MLNs and AMLNs – with known relations – is equivalent, and this was also empirically verified in additional experiments. As soon as, however, even a few links are left unknown, errors in link prediction propagate and classification results can, therefore, be greatly affected.

3.6.6 Discussion

This section introduced adaptive Markov logic networks (AMLNs) as a novel type of statistical relational model that explicitly represents dynamic parameter adjustments by phrasing the model's parameters as functions over domain-specific attributes. The dynamic adjustment of parameters is, as indicated by the experiments, an important concept in order to enable models to soundly generalise across domains with varying attributes (in particular, attributes pertaining to the number of objects under consideration). As has been shown, statistical models with a fixed vector of parameters may not succeed in capturing the general principles that in fact apply to even conceptu-

ally simple relational domains, potentially producing arbitrarily large errors in link prediction tasks. In contrast, the representation formalism of AMLNs and the associated learning methods are capable of handling the stochastic effects that result from varying attributes of the domain of discourse and thus enable a far greater class of stochastic processes to be accurately represented.

Related Work. The issue of learning models that specifically adapt themselves to render shallow transfer sensible was not previously addressed, although the particular conditions that cause models to fail to generalise are commonplace in relational models. Indeed, cardinality restrictions have found their way into most areas of computer science that deal with (natural) relations in one way or another, be it the modelling of entity relationships for database design (Abrial, 1974) or the logical modelling of real-world concepts and relations in description logics (Baader and Sattler, 1999). Soft constraints on cardinalities, i.e. distributions over counts, can, for instance, explicitly be expressed in the probabilistic description logic PClassic (Koller et al., 1997) and were also considered in the first-order probabilistic languages SPOOK (Pfeffer et al., 1999) and BLOG (Milch et al., 2005). Non-parametric approaches, e.g. NP-BLOG (Carbonetto et al., 2005), alleviate difficulties in model selection, which, in part, also result from having to handle a variable number of entities. Moreover, cardinality-based features of graphical models were previously defined by Gupta et al. (2007) in order to reduce the complexity of inference, albeit in an entirely different context. Rather than the shallow transfer problem treated in this section, Davis and Domingos (2009) and Mihalkova (2009) have studied the problem of deep transfer with Markov logic networks. On the knowledge representation side, Lippi and Frasconi (2009) have presented another extension of MLNs that uses functions to represent the weights of formulas. Their functions map specific groundings of a subset of the variables appearing in a formula to a weight, thus allowing a reduction of the size of models through concise representations of constant-specific dependencies.

Chapter 4

Uncertain Evidence and Probabilistic Information Interchange

A fundamental task in probabilistic reasoning is, as previously explained, to update beliefs as new information is obtained, i.e. to compute posterior probabilities $P(q \mid e)$ of relevant queries q given new evidence e. One typically assumes that the evidence e is a plain fact that could be represented as a logical conjunction. On many occasions, however, the evidence that we can supply as input to an inference problem does not correspond to firm, logical beliefs. Rather, evidence is often uncertain in the sense that it may represent a belief that either cannot be fully trusted (owing to unreliable mechanisms with which we obtain the evidence) or corresponds directly to a probability distribution that should be believed to hold.

For instance, we may regard the output of a probabilistic classifier as uncertain evidence. Surely it is more appropriate to make full use of the knowledge about the distribution over classes that such a classifier provides than to simply assume that the mode of the distribution is irrevocable fact. Moreover, it may be desirable to incorporate the (soft) beliefs that were computed by external sources of probabilistic information as input to an inference problem rather than to compute everything within a single model, given the fact that inference in all-encompassing probabilistic models is computationally infeasible. It may, therefore, be advisable to find ways of suitably factoring domains into loosely coupled probabilistic fragments and combine them as necessary. Whenever we do combine them, we will usually have to pass on probabilistic information rather than logical facts. The problem of updating beliefs in the light of uncertain evidence can thus be viewed as a subproblem of probabilistic information interchange. Probabilistic information interchange is particularly relevant within the context of multi-agent systems. Multiple agents that autonomously explore sim-

ilar environments may want to share their probabilistic findings with other agents, and these agents will have to incorporate the received findings into their own probabilistic knowledge bases and update their beliefs accordingly (Kim and Valtorta, 2000). The transfer of probabilistic findings may even be relevant when using just a single model, e.g. a sequence model, which is applied iteratively by issuing the probabilistic results computed in previous iterations as evidence for subsequent iterations.

In the following, precisely the issues pertaining to belief update that are fundamental for applications such as the ones named above will be addressed. I will first give an overview of semantically distinct ways in which evidential uncertainty can be interpreted, reviewing the notions of *virtual evidence* and *soft evidence* put forth in the literature (Section 4.1). Because the former is usually straightforward to handle, I concentrate, for the most part, on the harder problem of *soft evidential update*, i.e. the integration of firm beliefs corresponding to probability distributions. In Section 4.2, I present two solutions to this problem, one of which is similar to previous state-of-the-art approaches, while the other is fundamentally new in that it extends a Markov chain Monte Carlo method to directly incorporate probability constraints into the inference process, which can be significantly more efficient. The performance of the newly proposed MCMC method is thoroughly evaluated in a series of experiments based on both synthetic and real-world applications. Finally, the problem of taking soft evidence into consideration during learning is addressed in Section 4.3.

4.1 On the Semantics of Uncertain Evidence

Whenever evidence on some set of random variables X_E is uncertain, we associate with each assignment $X_E = x_E$ a degree of belief $p_{x_E} \in [0, 1]$. For every assignment $X_E = x_E$, there are (at least) two ways in which we might interpret the evidence and the corresponding degree of belief p_{x_E}:

(1) We have made an observation indicating $X_E = x_E$ and the degree to which this observation is reliable is p_{x_E}.

(2) The degree to which $X_E = x_E$ should be believed is p_{x_E}.

In either case, p_{x_E} is a well-defined degree of belief that should appropriately alter our posterior beliefs on unobserved variables, yet the semantics are fundamentally different. In the former case, we speak of *virtual* (or *intangible*) *evidence* (as the evidence impinges not immediately upon random variables in X but virtual observations thereof), while the latter is referred to as *soft evidence*. In the following, I will explain the differences between these two semantics in more detail.

Consider a full joint distribution over a set of random variables $X = \{X_1, \ldots, X_{|X|}\}$, which is partitioned into observed variables X_E (evidence variables) and unobserved variables X_U, i.e. $X = X_E \uplus X_U$. As usual, $\mathcal{X}$ denotes the set of possible worlds, i.e. the set of possible assignments of values to each of the variables in X. Without loss of generality, assume that all random variables under consideration are Boolean.

4.1.1 Virtual Evidence

Virtual evidence can be thought of as evidence concerning (undisclosed) observations that are outside of the set of variables X that we consider in our joint probability distribution but that have bearing on a subset X_E of these variables (Pearl, 1988). Technically, virtual evidence on X_E is realised by adding to the model auxiliary evidence variables V_E (corresponding to the observations that were made) and adding the likelihood of an assignment $X_E = x_E$ under these observations as an additional potential, i.e. we add $L(x_E) := P(V_E = v_E \mid X_E = x_E)$ for all $x_E \in \mathcal{X}_E$, where v_E is an arbitrarily chosen value assigned to the auxiliary evidence variables.

Adding virtual evidence on a variable X_i with beliefs $\langle p, q \rangle$ for *True* and *False* respectively can be viewed as adding an observation with a corresponding observation model in which the probability with which we observe $X_i = \textit{True}$ (*False*) if X_i is indeed *True* (*False*) is p (q).[1]

[1] Note that the values appearing in a virtual evidence vector (here p and q) need not sum to 1 (only the ratio between any two values matters), but we could equivalently represent them as a distribution (and Section 4.1.3 further on assumes that we do).

By applying the principle of virtual evidence, we essentially weight each possible world with the likelihood of the evidence variables X_E:

$$\begin{aligned} P(X \mid v_E) &= P(X_U, X_E \mid v_E) = P(X_U \mid X_E) \cdot P(X_E \mid v_E) \\ &\propto P(X_U \mid X_E) \cdot P(v_E \mid X_E) \cdot P(X_E) = P(X_U, X_E) \cdot P(v_E \mid X_E) \\ &= P(X) \cdot L(X_E) \end{aligned} \tag{4.1}$$

4.1.2 Soft Evidence

The concept of *soft evidence* is both simpler and more difficult. It is simpler, because its interpretation is straightforward: Any piece of soft evidence is but a constraint on the probability distribution over assignments to (a subset of) X_E. Soft evidence on a single variable X_i with distribution $\langle p_i, 1 - p_i \rangle$ simply requires the probability of $X_i = \mathit{True}$ to be p_i. However, as we will see, soft evidence is also the more difficult concept, because its treatment cannot simply be reduced to the addition of auxiliary variables for which potentials can straightforwardly be derived.

Let s_E represent the observations that have led us to believe in the soft evidence that we are given. In its entirety, soft evidence on the set of variables X_E can then be understood as defining the distribution $P(X_E \mid s_E)$. *Jeffrey's rule* (Pearl, 1988) provides a way of updating our posterior beliefs on the unobserved variables X_U in the light of our observations:

$$P(X_U \mid s_E) = \sum_{x_E \in \mathrm{dom}(X_E)} P(X_U \mid x_E) \cdot P(x_E \mid s_E) \tag{4.2}$$

As is evident from this equation, Jeffrey's rule is applicable only if the observations s_E have no impact on the conditional distribution $P(X_U \mid x_E)$, which is the case if the observations that were made have no bearing on the variables X_U beyond the mediated bearing through the evidence variables X_E (see Pearl, 1988, for a detailed discussion).

The resulting posterior distribution over $\mathcal{X}$ is:

$$\begin{aligned} P(X \mid s_E) &= P(X_U, X_E \mid s_E) = P(X_U \mid X_E) \cdot P(X_E \mid s_E) \\ &= \frac{P(X_U, X_E)}{P(X_E)} \cdot P(X_E \mid s_E) = P(X) \cdot \frac{P(X_E \mid s_E)}{P(X_E)} \end{aligned} \tag{4.3}$$

4.1.3 Discussion

Both virtual and soft evidence can be seen as generalisations of hard evidence. In the special cases where values of 1.0 or 0.0 are specified, both equivalently reduce to hard evidence. For values in between, however, the semantics differ greatly. Most importantly, only soft evidence retains the property of hard evidence that any evidence values are directly reflected in posterior beliefs.

From a computational perspective, virtual evidence does not pose any challenges, as it imposes only strictly local dependencies by introducing additional likelihoods, which can be handled by adding appropriate potentials to a probabilistic model (Pearl, 1988; Li, 2009). Its use is appropriate only if the uncertain information originates from a source that indeed computes mere likelihoods. The effect that a piece of virtual evidence has on the posterior of the respective variable(s) is highly dependent on the probability that is indicated by the model without the respective piece of evidence (and given any further evidence).[2] Soft evidence is concerned precisely with defining that effect, and its treatment therefore needs to take into consideration the fact that several pieces of uncertain evidence cannot usually be treated independently but influence each other.

It is perhaps worth noting that there is a loose connection between uncertain evidence and *fuzzy logic*, as inference in fuzzy logic also deals with updates involving input values between 0 and 1. However, these inputs have fundamentally different semantics from either virtual or soft evidence, for they do not correspond to degrees of belief but rather to degrees of truth. A language called *probabilistic soft logic*

[2] For example, if the probability of $X_i = \mathit{True}$ given the other evidence is 0.1, adding virtual evidence on X_i with values $\langle 0.8, 0.2 \rangle$ will cause the probability of $X_i = \mathit{True}$ to become $0.1 \cdot 0.8 / (0.1 \cdot 0.8 + 0.9 \cdot 0.2) \approx 0.31$, not 0.8.

(formerly probabilistic similarity logic; Bröcheler et al., 2010) integrates principles of fuzzy logic and probabilistic reasoning in a relational setting.

4.2 Soft Evidential Update

In the following, I address the question of how beliefs can efficiently be updated in the light of soft evidence. I present two methods for the problem of *soft evidential update* (see Figure 4.1) that are considerably more practical than the naive approach involving $2^{|X_E|}$ inference runs that Equation 4.2 gives rise to, discussing related work along the way. Markov logic networks will be used as the probabilistic framework.

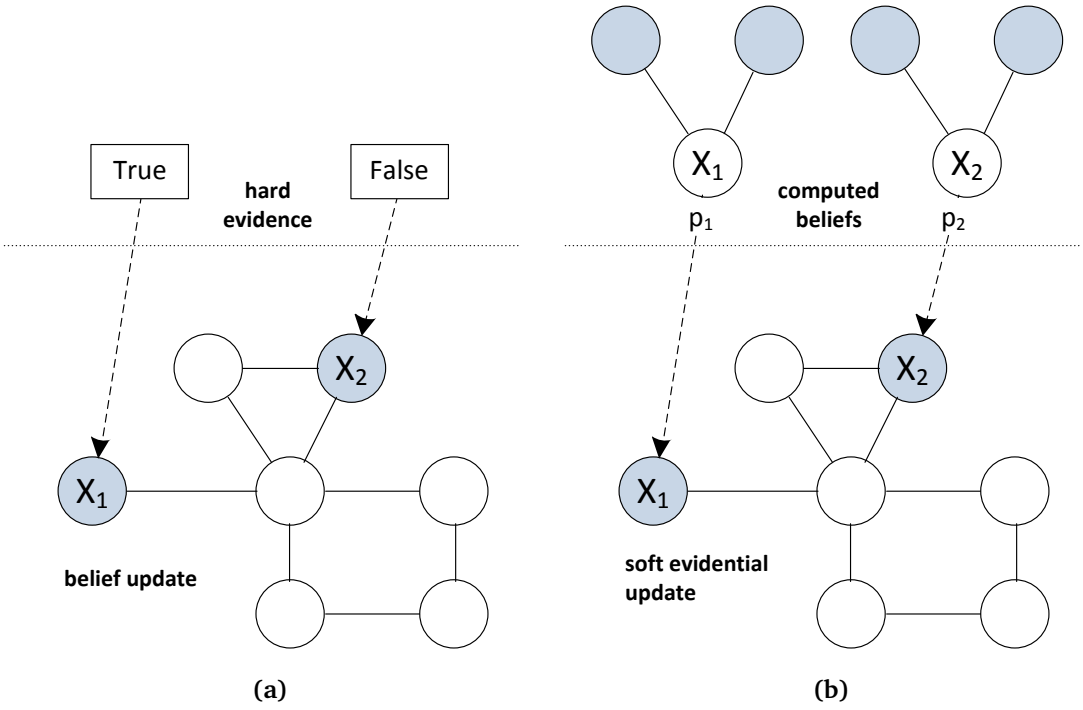


Figure 4.1: *Standard belief update (a) versus soft evidential update (b)*

4.2.1 Probability Constraints and Iterative Fitting

The problem of soft evidential update is closely related to the concept of imposing probability constraints on a distribution. To make this explicit, recall the iterative

proportional fitting procedure (IPFP) from Section 3.5.1, which allows to modify a joint distribution such that it satisfies a set $\mathscr{R} = \{R_k\}$ of probability constraints, where a probability constraint is a probability distribution over a subset of X. Notice that the update rule of IPFP, i.e. Equation 3.23 (page 77), is analogous to Equation 4.3. Hence soft evidence can be handled by applying IPFP with a single constraint $R_1(X_E) := P(X_E \mid s_E)$.

In practice, however, we will rarely be given a joint distribution $P(X_E \mid s_E)$ as evidence, for such distributions are exponential in the size of X_E and therefore impractical to obtain and represent explicitly. Rather, from now on, consider the case where we are given a number of soft evidences, each concerning a ground formula. This subsumes the perhaps most practically relevant case where every piece of soft evidence impinges upon an single ground atom. We thus consider a set of ground formula constraints $\mathscr{R}_F$ as in Section 3.5.1, where each constraint takes the form $P(\widehat{F}_k) = p_k$. Note that the factorisation of soft evidence into separate constraints means that the joint distribution $P(X_E \mid s_E)$ is no longer given – and Jeffrey's rule is inapplicable unless we make additional assumptions. We should, of course, not simply assume independence of the individual pieces of soft evidence, since we do have information about dependencies in the form of the prior $P(X)$. An application of IPFP-M (see Section 3.5.2) with the set of constraints $\mathscr{R}_F$ will ensure that this information is appropriately considered. Therefore, IPFP-M can be viewed as a method for soft evidential update. In contrast to the application of IPFP-M considered in Section 3.5, where it served to adjust the parameters of a model which could subsequently be used to update beliefs based on standard inference with the actual evidence, IPFP-M now serves to directly compute the posterior beliefs for soft evidential update – simply by phrasing all soft evidence as ground formula probability constraints. If there is hard evidence in addition to soft evidence, it can either be translated to soft evidence or be treated explicitly by passing it as an additional parameter to IPFP-M and passing it on to each inference call that is made within IPFP-M (line 7 of Algorithm 3, page 80). The latter approach is preferable, because it reduces the number of probability constraints.

Related Work. To solve the problem of soft evidential update in the context of Bayesian networks, Pan et al. (2006) have previously proposed a method that is similar to IPFP-M. It extends a Bayesian network model with virtual evidence nodes whose conditional distributions are iteratively adjusted to fit the probability constraints (Pan

et al., 2006, Algorithm 1). Indeed, virtually all related work on the subject of soft evidential update has been centred around variations of IPFP (Kim and Valtorta, 2000; Pan et al., 2006; Langevin and Valtorta, 2008).

Since repeated inference over the full model can be expensive, Pan et al. (2006) have suggested another method that can be more efficient in cases where the set of soft evidence nodes X_E is very small. They suggest an explicit application of standard IPFP to the full-joint distribution $P(X_E)$ by first precomputing the prior $P(X_E)$ based on the full model of $P(X)$, then applying standard IPFP to $P(X_E)$, and finally running an inference algorithm to update the beliefs on unobserved variables using a model extended with virtual evidence on X_E corresponding to the results of IPFP (Pan et al., 2006, Algorithm 2). For this method, the complexity of every IPFP step is exponential in the size of X_E, which renders the algorithm inapplicable as X_E grows larger. It is, however, advantageous if X_E is very small. The same is true for the Big Clique algorithm (Kim and Valtorta, 2000), which takes a similar approach, albeit within a more confined setting (it is formulated strictly as an extension to the junction tree algorithm).

For larger sets of evidence variables, the model-based approach to IPFP is the better choice. IPFP-M's performance is, of course, largely dependent on the efficiency of the underlying inference method that is applied to the entire model in each step. Notably, IPFP-M enables us to use methods that do not require an explicit representation of a cumbersome full-joint distribution and, in particular, approximate inference methods.

4.2.2 A Markov Chain Monte Carlo Method

I now introduce a Markov chain Monte Carlo (MCMC) method, which allows to directly integrate soft evidence into the inference procedure – without generating substantial overhead. Specifically, I present an extension of the algorithm MC-SAT by Poon and Domingos (2006) (see Section 3.2.1.2), which, unlike other MCMC inference algorithms, can soundly handle deterministic and near-deterministic dependencies.

Consider a ground Markov random field $M_{L,D} = \langle X, G \rangle$ in which all formula weights are non-negative. Recall that MC-SAT is a slice sampler that uses one auxiliary variable U_j per ground formula $\langle \widehat{F}_j, \widehat{w}_j \rangle \in G$. The Markov chain is formed by alternatingly sampling the auxiliary variables and the actual state as follows. The initial state $x^{(0)} \in \mathscr{X}$ must be one that satisfies all hard constraints in G. Then, in step i, the auxiliary variables are first resampled given the previous state $x^{(i-1)}$. If $\widehat{F}_j$ was satisfied in $x^{(i-1)}$, then with probability $1 - \exp(-\widehat{w}_j)$ we sample $u_j > 1$ and therefore $\widehat{F}_j$ must also be satisfied in the current step. We thus obtain a set $M = \{\widehat{F}_j \mid u_j > 1\}$ of formulas that must be satisfied and use a SAT sampler to uniformly sample a state $x^{(i)} \in \mathscr{X}$ that satisfies the formulas in M.

This algorithm is now extended to explicitly consider the probability constraints $\mathscr{R}_F$ corresponding to a set of soft evidences. Intuitively, we would like to achieve the effect of applying MC-SAT to the model we would have obtained by applying IPFP-M to $M_{L,D}$ and $\mathscr{R}_F$, yet without having to actually compute the weights $\widehat{w}_{R_k}$ that IPFP-M indicates upon convergence. Consider a constraint $P(\widehat{F}_{R_k}) = p_k$ and the pair $\langle \widehat{F}_{R_k}, \widehat{w}_{R_k} \rangle$ that IPFP-M would have added to G. Observe that without $\langle \widehat{F}_{R_k}, \widehat{w}_{R_k} \rangle$, there is some relative frequency with which states in which $\widehat{F}_{R_k}$ is *True* would have appeared in MC-SAT's Markov chain. The weight $\widehat{w}_{R_k}$ computed by IPFP-M would achieve that this relative frequency becomes approximately p_k (given that the number of samples is large enough). The weight achieves this simply by specifying the probability with which the Markov chain should remain within the subspace of the support of $P(X)$ where $\widehat{F}_{R_k}$ (or, if $\widehat{w}_{R_k}$ is negative, $\neg\widehat{F}_{R_k}$) is satisfied once it reaches a state where $\widehat{F}_{R_k}$ ($\neg\widehat{F}_{R_k}$) is satisfied. Technically, if $\widehat{F}_{R_k}$ ($\neg\widehat{F}_{R_k}$) was satisfied in the previous step, it is *reselected* for addition to the set M of formulas to satisfy in the current step with probability $1 - e^{-\widehat{w}_{R_k}}$ (respectively $1 - e^{\widehat{w}_{R_k}}$).

We can essentially simulate this (without knowing the actual probability with which reselection should occur) by keeping track of the relative frequency with which $\widehat{F}_{R_k}$ was indeed true in the Markov chain and, if $\widehat{F}_{R_k}$ was satisfied in the previous step, adding $\widehat{F}_{R_k}$ to M whenever the relative frequency is, thus far, lower than p_k. Inversely, if $\widehat{F}_{R_k}$ was not satisfied in the previous step, we add $\neg\widehat{F}_{R_k}$ to M if the relative frequency of $\widehat{F}_{R_k}$ is greater than p_k. We therefore simulate the effect of $\widehat{w}_{R_k}$ (as it would have

Algorithm 4 MC-SAT-PC$(M_{L,D} = \langle X, G\rangle, \mathscr{R}_F, numSamples)$

$n_k \leftarrow 0$ **for all** $\langle \widehat{F}_{R_k}, p_k\rangle \in \mathscr{R}_F$
for $i \leftarrow 0$ to $numSamples - 1$ **do**
  if $i = 0$ **then**
    $M \leftarrow \{\widehat{F}_k \mid \langle \widehat{F}_k, \widehat{w}_k\rangle \in G \wedge w_k$ is hard$\}$
  else
    $M \leftarrow \emptyset$
    for all $\langle \widehat{F}_k, \widehat{w}_k\rangle \in G$ **do**
      if $x^{(i-1)} \models \widehat{F}_k$ **then** with probability $1 - e^{-\widehat{w}_k}$ add $\widehat{F}_k$ to M
    for all $\langle \widehat{F}_{R_k}, p_k\rangle \in \mathscr{R}_F$ **do**
      if $x^{(i-1)} \models \widehat{F}_{R_k}$ **then**
        if $n_k/i < p_k$ **then** add $\widehat{F}_{R_k}$ to M
      else
        if $1 - n_k/i < 1 - p_k$ **then** add $\neg\widehat{F}_{R_k}$ to M
  sample $x^{(i)} \sim Uniform(\mathrm{SAT}(M))$
  for all $\langle \widehat{F}_{R_k}, p_k\rangle \in \mathscr{R}_F$ **do**
    if $x^{(i)} \models \widehat{F}_{R_k}$ **then** $n_k \leftarrow n_k + 1$

been computed by IPFP-M) by splitting its impact across the formulas $\widehat{F}_{R_k}$ and $\neg\widehat{F}_{R_k}$.[3] Because the resulting algorithm effectively integrates posterior probability constraints into MC-SAT, I refer to it as *MC-SAT-PC* (see Algorithm 4).

I now provide a more detailed intuitive argument why the method is sound. Given that MC-SAT is sound, it suffices to consider MC-SAT-PC's behaviour in relation to an application of MC-SAT to a model extended with the weights computed by IPFP-M. For a particular inference problem, denote by $M_{L,D}$ the original model and denote by $M^*_{L,D}$ the extended model that is computed by IPFP-M. Assume, for the time being, that there is a single probability constraint to be fitted: $\langle \widehat{F}_{R_k}, p_k\rangle$. The pair $\langle \widehat{F}_{R_k}, \widehat{w}_{R_k}\rangle$ that is added to G for this constraint can be equivalently replaced by two pairs $\langle \widehat{F}_{R_k}, \widehat{w}_{R_k,1}\rangle$ and $\langle \neg\widehat{F}_{R_k}, \widehat{w}_{R_k,2}\rangle$ with $\widehat{w}_{R_k,1} \geq 0$ and $\widehat{w}_{R_k,2} \geq 0$.[4] Without the newly added weighted formulas, the model would indicate some relative frequency p with which $\widehat{F}_{R_k}$ is true in the Markov chain, which implies a corresponding probability of remaining within the subspace of $\mathscr{X}$ where $\widehat{F}_{R_k}$ is satisfied once a state is reached

[3] As explained in Section 3.4.2, F with weight $\log(p/(1-p))$ for $p \in]0,1[$ is equivalent to the combination of F with weight $\log(p)$ and $\neg F$ with weight $\log(1-p)$.

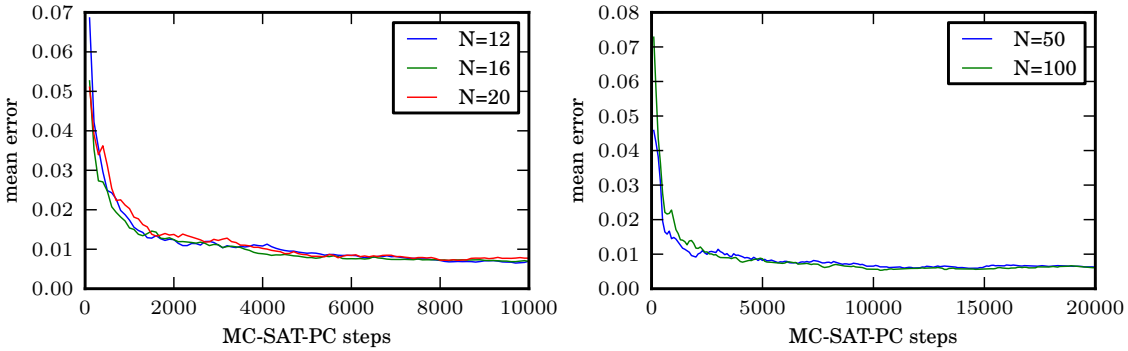
[4] See footnote 3. Since p and $1-p$ are both in $]0,1[$, the resulting weights would be negative. Negating both formulas along with their weights results in weights for $\widehat{F}_{R_k}$ and $\neg\widehat{F}_{R_k}$ that are both positive and therefore are suitable for application within MC-SAT.

in which $\widehat{F}_{R_k}$ is satisfied. This probability is effectively increased by the weight $\widehat{w}_{R_k,1}$. Similarly, the probability of remaining within the subspace of $\mathscr{X}$ where $\neg\widehat{F}_{R_k}$ is satisfied is increased by $\widehat{w}_{R_k,2}$. Therefore, if the combination of both weights is to achieve a shift from p to the desired probability p_k, then if $p < p_k$, $\widehat{w}_{R_k,1}$ will be larger than $\widehat{w}_{R_k,2}$, and if $p_k < p$, $\widehat{w}_{R_k,2}$ will be larger than $\widehat{w}_{R_k,1}$, such that the probability of reselecting either $\widehat{F}_{R_k}$ or $\neg\widehat{F}_{R_k}$ is rendered proportionately larger as required by the discrepancy between p and p_k. While MC-SAT applied to $M^*_{L,D}$ will make the decision to reselect the formulas $\widehat{F}_{R_k}$ and $\neg\widehat{F}_{R_k}$ memorylessly (the desired relative frequency p_k being the result on expectation), MC-SAT-PC ensures that the formula is neither reselected too frequently nor too infrequently by stopping reselection precisely at the point at which the intended effect of the weight is reached. Notably, for a case where the formula that was satisfied in the previous state is *not* reselected, both MC-SAT applied to $M^*_{L,D}$ and MC-SAT-PC applied to $M_{L,D}$ behave in exactly the same way: The respective formula may or may not be satisfied in the next step, and the probability with which satisfaction results in the next step is the same in either case (it depends exclusively on the number of solutions of the SAT problem induced by the formulas that *are* reselected for addition to M which satisfy the formula in question). Therefore, MC-SAT-PC should be capable of computing the same results as MC-SAT applied to $M^*_{L,D}$ in an equivalent number of steps. Through an inductive argument, the above generalises to an arbitrary number of formula probability constraints.

4.2.3 Experiments

In the following, the performance of MC-SAT-PC is evaluated in a series of experiments. In particular, it is compared to IPFP-M, noting that, as mentioned previously, IPFP-M belongs to a class of methods that could hitherto be considered as the methods of choice whenever the set X_E is not extremely small. In all the experiments, the soft evidence is factored such that every piece of evidence constrains the probability of a single atom in X_E.

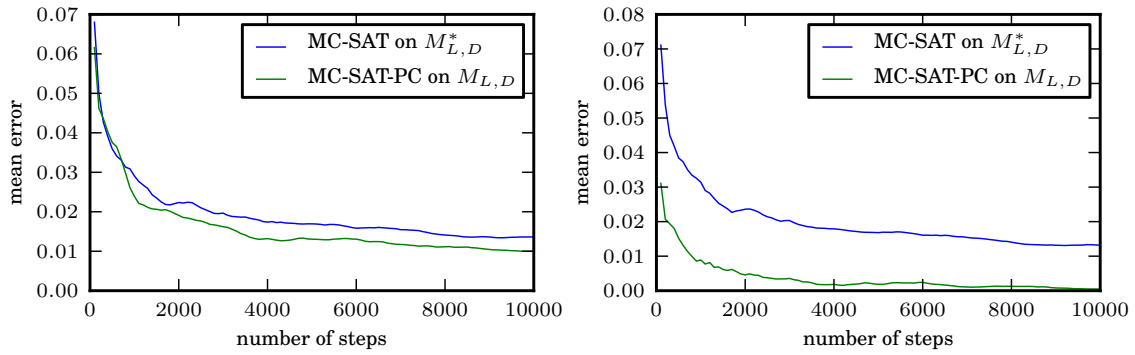
In the first experiment, results obtained using IPFP-M with exact inference as the underlying inference algorithm (IPFP-M[exact]) are compared to results computed by MC-SAT-PC. Markov random fields (MRFs) of three different sizes ($N \in \{12, 16, 20\}$) were randomly generated, ten of each size. For an MRF with N Boolean variables, N



(a) Experiment 1 **(b)** Experiment 2

Figure 4.2: *Mean error in posterior marginals of variables in X_U computed by MC-SAT-PC with respect to results computed by (a) IPFP-M[exact], averaged across all 10 experiments, and (b) IPFP-M[MC-SAT]. N is the number of random variables considered in the respective (series of) experiments.*

random features were created by generating clauses with a random number of literals (uniformly sampled from $\{1,\ldots,10\}$), drawing weights from $\mathcal{N}(0,(\log(50)/2)^2)$. For each MRF, $N/2$ variables were selected as the set X_E of (soft) evidence variables, sampling beliefs uniformly from $[0,1]$. IPFP-M[exact] was run on the inference problems thus obtained. The procedure was terminated as soon as the maximum absolute error across all soft evidence variables dropped below $\varepsilon = 0.001$. Typically, IPFP-M[exact] terminated after at most 5 rounds over all constraints (i.e. $5N/2$ iterations and therefore $5N/2$ inference runs). In each iteration, not only the probability of the constraint to be fitted was inferred but also the probabilities of all constraints in order to determine convergence as well as the posterior marginal probabilities of all unobserved variables, $P(X_i \mid s_E)$ for all $X_i \in X_U$, which served as queries. MC-SAT-PC was run with the same queries, comparing the results to the ones obtained by IPFP-M[exact] every 100 steps. As shown in Figure 4.2a, MC-SAT-PC quickly produced results that are, on average, within 0.01 of the true results. After 10000 steps, the maximum error in any query variable across all 30 experiments was 0.035. (The error margins are equivalent to the ones typically obtained when applying MC-SAT on regular belief update problems.) These results indicate that, MC-SAT-PC is, in general, a suitable method for the approximation of posterior beliefs in the presence of soft evidence.



(a) Unobserved variables **(b)** Soft evidence variables

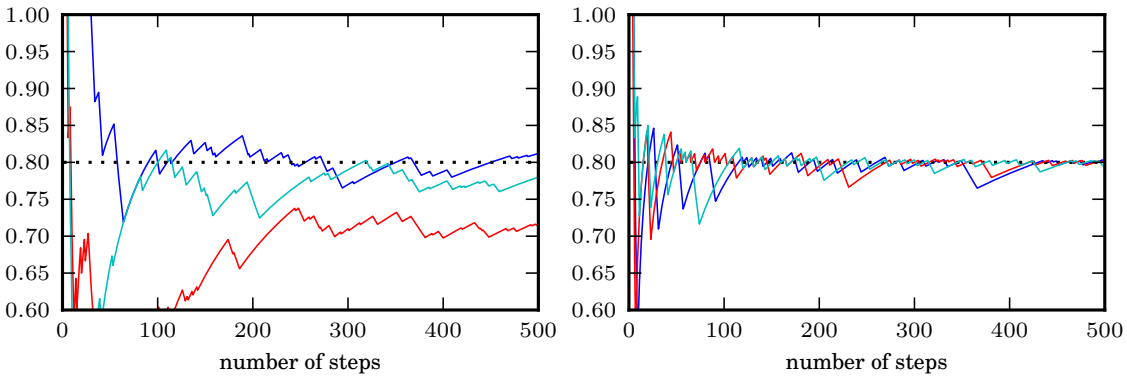
Figure 4.3: *Experiment 3: Comparison between the approximations obtained with MC-SAT-PC applied to the original model and MC-SAT applied to a previously fitted model that satisfies the probability constraints corresponding to the soft evidence. The plots show the mean errors in the posterior marginals of (a) query variables and (b) soft evidence variables, averaged across 50 runs (5 random models, 10 runs each).*

Of course, using exact inference as the underlying inference mechanism in IPFP-M is infeasible in all but the smallest domains (IPFP-M[exact] took over 80 minutes to converge for $N = 20$). Since MC-SAT has proved to be a most robust method, which is also applicable in the presence of deterministic constraints, the second experiment compares MC-SAT-PC to IPFP-M with MC-SAT as the underlying inference algorithm (IPFP-M[MC-SAT]) in order to evaluate potential time savings. MRFs of sizes $N = 100$ and $N = 50$ were randomly generated as above. For each experiment, 20 soft evidence variables were randomly selected, and, as before, the remaining variables were used as query variables. Owing to the approximate nature of MC-SAT, IPFP-M's termination criterion had to be relaxed: The fitting procedure was stopped as soon as the mean absolute error across all soft evidence variables dropped below $\varepsilon_1 = 0.01$ and the the maximum error dropped below $\varepsilon_2 = 0.05$. In each iteration of IPFP-M[MC-SAT], MC-SAT was run for 10000 steps. Figure 4.2b summarises the main result: MC-SAT-PC produces roughly the same results, yet it does so in a fraction of the time. While IPFP-M[MC-SAT] ran for approximately 9.5 and 76.6 minutes (3 and 7 rounds) for $N = 50$ and $N = 100$ respectively, MC-SAT-PC drew 20000 samples in less than 35 seconds in each case. The maximum deviation from the results computed by IPFP-M was 0.024 and therefore within the expected range.

A third experiment serves to empirically validate the claim that MC-SAT-PC is capable of computing the same results (in an equivalent number of steps) as MC-SAT applied to a ground MRF which has been specifically adapted to satisfy the soft evidence. For any given ground MRF $M_{L,D}$, the adapted MRF $M^*_{L,D}$ can be obtained by extending the original $M_{L,D}$ with the additional weighted formulas computed by IPFP-M[exact]. Five ground Markov random fields with 16 variables were randomly generated (as in Experiment 1), and IPFP-M[exact] was used to compute the adapted model $M^*_{L,D}$ and the exact posterior marginals of all unobserved variables in X_U. In each case, MC-SAT was applied to the adapted model and MC-SAT-PC was applied to the original model ten times. Figure 4.3a shows the mean error in the posterior marginals of the query variables, averaged across all runs. The results indicate that MC-SAT-PC even displays superior convergence on this particular set of experiments. Figure 4.3b, which plots the averaged mean errors in the posterior marginals of the soft evidence variables X_E, provides an explanation: When applying MC-SAT, the satisfaction of the probability constraints implied by the soft evidence is subject to approximation errors to a larger degree, because MC-SAT lacks explicit knowledge about the probabilities that are to be brought about. These approximation errors propagate to the unobserved variables. MC-SAT-PC, however, eliminates some of the stochasticity by enforcing the satisfaction of the probability constraints directly (see Figure 4.4).

MC-SAT-PC was also tested on a complex real-world problem (which in fact provided the initial motivation for the design of MC-SAT-PC): spatio-temporal object identity resolution, where the goal is to manage associations between observations and (not necessarily uniquely identifiable) objects over time – based on the similarity in appearance between observed entities and models of the objects and based on information on previous observations. A Markov logic network for this problem was created, which is applied iteratively as new observations come in. Previously computed beliefs are provided as soft evidence for a subsequent application of the model (as are beliefs indicating degrees of similarity). The application is described in more detail in Section 6.1.2.

An exemplary ground instance of the problem contained 279 random variables and 2688 features; 42 variables were given as soft evidence, 36 as hard evidence. An application of IPFP-M[MC-SAT] to a problem of this size is essentially infeasible, as far too many iterations are required, each of which involves only insignificantly less



(a) MC-SAT on $M^*_{L,D}$ **(b)** MC-SAT-PC on $M_{L,D}$

Figure 4.4: *Approximations of the posterior marginal probability of a single soft evidence variable obtained with (a) MC-SAT applied to a previously fitted model and (b) MC-SAT-PC applied to the original model. The original model indicated a probability of 0.4 for the variable, whereas the soft evidence required the probability to become 0.8. Three random trials are shown for each method.*

effort than a full application of MC-SAT-PC. Therefore, because we do not have any results on the unobserved variables to compare against, we plot in Figure 4.5 the number of steps taken by MC-SAT-PC against the mean and maximum errors we obtain for the soft evidence variables X_E, which (if $|X_E|$ is large enough) are fairly good indicators of convergence. The errors drop fairly quickly, and the results obtained for X_U are consistent with common sense expectations.

As with any MCMC method, there can be cases where MC-SAT-PC may take a long time to converge. If, for example, the probability indicated by the model for some $\widehat{F}_{R_k}$ is very low (given the remaining evidence), e.g. 10^{-4}, but R_k requires it to be substantially higher, then a large number of steps may be required for the Markov chain to enter and appropriately explore the subspace where $\widehat{F}_{R_k}$ is true. Of course, in such cases, IPFP-M will also have to use a rather large number of steps in the underlying inference algorithm, as we are required to compute an approximately correct non-zero probability for $\widehat{F}_{R_k}$ if Equation 3.23 is to be applicable and have the desired effect.

In conclusion, soft evidential update is a practically important problem which extends probabilistic inference to evidence that corresponds to (subjective) degrees of

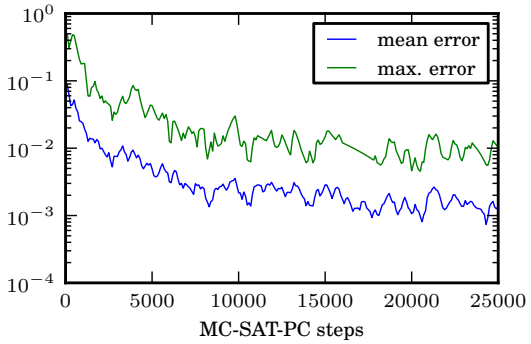


Figure 4.5: *Experiment 4: Mean and maximum errors across all soft evidence variables for an instance of an MLN for object identity resolution (logarithmic scale); final errors are 0.0014 and 0.0103 respectively.*

belief. It is particularly important in the context of probabilistic information interchange and can soundly enable AI systems to distribute probabilistic computations across several loosely coupled probabilistic models. With MC-SAT-PC, a novel MCMC method for soft evidential update was presented which effectively integrates (posterior) probability constraints into the inference procedure. I described its relation to IPFP-based methods and demonstrated that, particularly for hard problems involving a large number of evidence variables, it can improve performance by orders of magnitude without sacrificing accuracy.

4.3 Learning with Soft Evidence

If soft evidence is given as input to a learning problem, the learning methods need to be adapted to take the particular semantics into consideration. We may need to learn the parameters of a model in a situation where some or all of the data is subject to uncertainty, i.e. training databases may include not only logical facts pertaining to the truth or falsity of atoms but also degrees of belief. This problem is again treated using Markov logic networks as the representation formalism, and the soft evidence that is given as input to the learning problem is, as before, denoted by s_E.

When making the closed-world assumption during learning, the truth value of every atom that can be constructed from the MLN L's set of predicates is known, and every training database thus corresponds to a single possible world x. The task in standard maximum likelihood learning is to maximize the probability of this world x relative to

all other worlds in the respective set of worlds $\mathcal{X}$. As degrees of belief are introduced, the training data no longer corresponds to a single world in $\mathcal{X}$. Rather, the soft evidence s_E immediately gives rise to a distribution $P(x \mid s_E, L)$, which takes into consideration both the soft evidence and the model L (including the dependencies it implies). Thus, the goal of maximum likelihood learning can be rewritten as

$$\arg\max_{w} \sum_{x \in \mathcal{X}} P(x \mid L) \cdot P(x \mid s_E, L), \tag{4.4}$$

i.e. the estimate is obtained by weighting the probability that is assigned to each possible world by the model L with the probability that can be allocated to that world according to the soft evidence s_E.

In the following, I present several approaches that adopt different interpretations of the soft evidence and hence differ in their approximation of the above. I first present methods that rely on approximations in order to reduce the optimisation problem to a single world in which formulas are evaluated non-logically and then present an alternative that makes use of soft evidential update techniques in order to directly maximise the weighted sum in Equation 4.4.

4.3.1 Learning with Soft Features

In this section, soft evidence learning methods that make two simplifying assumptions will be considered. First, a factorisation of the soft evidence where every piece of soft evidence concerns a single atom is assumed, i.e. s_E specifies, for each atom $X_i \in X$ a belief $P(X_i \mid s_E)$. Second, the interpretation of the evidence is simplified by making independence assumptions.

Recall that the numbers of true groundings of formulas are decidedly important figures in the context of parameter learning. In particular, the numbers of true groundings of each formula in each possible world collectively constitute a sufficient statistic for maximum likelihood parameter learning (see Section 3.3). Since, in the presence of soft evidence, the training database does not correspond to a single possible world in which the formulas' feature functions could be evaluated, the number of true groundings of any given formula cannot straightforwardly be computed. Although

the immediate interpretation of soft evidence involves a distribution over possible worlds, we shall, in the following, assume a single "soft world" in which formulas are not required to be either true or false but instead can be true to some degree, depending on the values present in the soft evidence s_E. As will be shown, this notion leads to particularly simple generalisations of the learning methods for hard evidence that were described in Section 3.3.

The deterministic features $\widehat{f}_j : \mathscr{X} \to \{0,1\}$ are replaced by *soft features* $\tilde{f}(\widehat{F}_j)$ that map to $[0,1]$ within the specific context of the soft evidence database s_E, yielding a continuous belief. The evaluation assumes mutual independence between any two pieces of soft evidence pertaining to individual ground atoms and therefore uses noisy-and and noisy-or to compute degrees of belief for complex formulas. Specifically, the evaluation of soft features can be described using structural induction as follows:

- For an atom $X_i \in X$, the feature value is the degree of belief that is given in the (soft) evidence, i.e. $\tilde{f}(X_i) = P(X_i \mid s_E)$.
- For a negated formula $\neg\widehat{F}$, the feature value is computed as $\tilde{f}(\neg\widehat{F}) = 1 - \tilde{f}(\widehat{F})$.
- For a conjunction $\widehat{F} \equiv \widehat{F}_1 \wedge \cdots \wedge \widehat{F}_n$, the feature value is computed as $\tilde{f}(\widehat{F}) = \prod_{i=1}^{n} \tilde{f}(\widehat{F}_i)$.
- For a disjunction $\widehat{F} \equiv \widehat{F}_1 \vee \cdots \vee \widehat{F}_n$, the feature value is computed as $\tilde{f}(\widehat{F}) = 1 - \prod_{i=1}^{n}(1 - \tilde{f}(\widehat{F}_i))$.
- Implications and biimplications are reduced to negations, disjunctions and conjunctions and are handled accordingly.

It should be noted that this scheme does not guarantee that for two semantically equivalent ground formulas $\widehat{F}_i$ and $\widehat{F}_j$, $\tilde{f}(\widehat{F}_i) = \tilde{f}(\widehat{F}_j)$.[5] To establish regularity, formulas are required to be normalised and fully factorised. In the ProbCog system's implementation, formulas are converted to (factorised) conjunctive normal form.

In the case where all beliefs on atoms are either 0 or 1, i.e. $P(X_i \mid s_E) \in \{0,1\}$ for all $X_i \in X$, the soft feature is equivalent to the hard feature. Noisy-and and noisy-or reduce to logical conjunction and logical disjunction respectively.

[5] As a simple example, consider the formulas A and $A \wedge A$, which are clearly semantically equivalent, yet the soft feature evaluates to $\tilde{f}(A)$ for the former and $\tilde{f}(A)^2$ for the latter.

Let $M_{L,D} = \langle X, G \rangle$ be the ground Markov random field that is induced by the training database, and let $G_{F_i} \subseteq G$ be the set of ground formulas that were instantiated from the first-order formula F_i in L. Based on the above notion of soft features, we can then replace the aggregated counts $n_i(x) = \sum_{\langle \widehat{F}_j, \widehat{w}_j \rangle \in G_{F_i}} \widehat{f}_j(x)$ that are computed for the training database by soft counts that are defined as follows:

$$\tilde{n}_i := \sum_{\langle \widehat{F}_j, \widehat{w}_j \rangle \in G_{F_i}} \tilde{f}(\widehat{F}_j) \tag{4.5}$$

We denote by x_{s_E} the fictitious "soft world" to which these soft counts apply and whose probability is to be maximised ($x_{s_E} \notin \mathcal{X}$).

Maximum Likelihood with Soft Features (MLSF). The likelihood, i.e. the probability of x_{s_E} given the current vector of weights, can be computed as

$$P_{\text{MLSF}}(x_{s_E}) = \frac{\exp\left(\sum_i w_i \tilde{n}\right)}{\sum_{x' \in \mathcal{X}} \exp\left(\sum_k w_k n_k(x')\right)} = \frac{1}{Z} \exp\left(\sum_i w_i \tilde{n}_i\right). \tag{4.6}$$

Note that the numerator term is not explicitly included in the normalisation constant Z. This is, however, reasonable because we imagine the soft world to be an approximation of the weighting of (actual) possible worlds that the soft evidence gives rise to.

In practice, it is convenient to maximise the log-likelihood

$$L_{\text{MLSF}}(x_{s_E}) = \log P_{\text{MLSF}}(x_{s_E}) = \sum_i w_i \tilde{n}_i - \log(Z) \tag{4.7}$$

whose gradient is given by

$$\begin{aligned} \frac{\delta}{\delta w_i} L_{\text{MLSF}}(x_{s_E}) &= \tilde{n}_i - \sum_{x' \in \mathcal{X}} n_i(x') \cdot P(X = x') \\ &= \tilde{n}_i - \sum_{x' \in \mathcal{X}} n_i(x') \cdot \frac{\exp\left(\sum_k w_k n_k(x')\right)}{\sum_{x'' \in \mathcal{X}} \exp\left(\sum_k w_k n_k(x'')\right)}. \end{aligned} \tag{4.8}$$

As explained in Section 3.3, the log-likelihood and its gradient are all we need in order to perform parameter learning. For the gradient-based optimisation, we can again employ quasi-Newton methods.

MLSF is a true generalisation of maximum likelihood learning with hard evidence. In the case where all soft evidence probabilities of atoms are either 0 or 1, x_{s_E} can be interpreted as a regular possible world and $\tilde{n}_i = n_i(x_{s_E})$. Equations 4.7 and 4.8 reduce to Equations 3.12 and 3.13 respectively.

Maximum Pseudo-Likelihood with Soft Features (MPLSF). Maximum likelihood learning being inapplicable in all but the smallest of instantiations, we can again draw upon the pseudo-likelihood approximation. The principles described above can essentially be transferred directly to pseudo-likelihood learning. With $\tilde{\bar{n}}_{i,k}$ as the soft count of the i-th formula when evaluated on a modification of x_{s_E} in which the belief for X_k is inverted, we can compute the pseudo-log-likelihood as

$$L_{\text{MPLSF}}(x_{s_E}) = \sum_{k=1}^{N} -\log\left(1 + \exp\left(\sum_{i \in F_{X_k}} w_i \cdot (\tilde{\bar{n}}_{i,k} - \tilde{n}_i)\right)\right) \tag{4.9}$$

and its gradient as

$$\frac{\delta}{\delta w_i} L_{\text{MPLSF}}(x_{s_E}) = \sum_{k=1}^{N} \left(\tilde{\bar{n}}_{i,k} - \tilde{n}_i\right)\left(\frac{1}{1 + \exp\left(\sum_{j \in F_{X_k}} w_j \cdot (\tilde{\bar{n}}_{j,k} - \tilde{n}_j)\right)} - 1\right). \tag{4.10}$$

4.3.2 Sampling-Based Learning

The soft features used above and the assumption of independence their computation is based upon may constitute too great a simplification for sufficiently accurate learning results to be obtained. In this section, we therefore treat the soft evidence explicitly, making fewer simplifying assumptions and specifically taking into consideration the distribution over possible worlds that is induced by the soft evidence s_E. In contrast to the methods proposed above, we do not assume that the individual pieces of soft evidence are independent. In particular, we do not assume that the distribution implied by s_E is given by $\prod_i P(X_i \mid s_E)$ – even for cases where the evidence is given as individual beliefs on ground atoms. Rather, if the starting point for the optimisation is the vector of zero weights, then this assumption is made only in the very first step of the optimisation where no further knowledge about dependencies

has yet been derived. In subsequent steps, however, the weights that are assigned to formulas already indicate dependencies between the random variables that were discovered in the data and that may demand an entirely different interpretation. This should be taken into consideration. Hence the use of the distribution $P(x \mid s_E, L)$, which takes the current weights in L into account and which can be approximated using the techniques of soft evidential update.

We can use MC-SAT-PC to draw a multiset S_{s_E} of samples from $P(x \mid s_E, L)$. If the number of samples drawn is large enough, the relative frequency of a possible world x in S_{s_E} will be approximately equal to $P(x \mid s_E, L)$. Let $n(x, S)$ be the number of times the assignment $X = x$ appears within the set of samples S. We can then compute the optimisation target in Equation 4.4 as

$$\begin{aligned} \sum_{x \in \mathscr{X}} P(X = x \mid L) \cdot P(X = x \mid s_E, L) &\approx \sum_{x \in \mathscr{X}} \frac{\omega(x)}{Z} \cdot \frac{n(x, S_{s_E})}{|S_{s_E}|} \\ &= \sum_{x \in S_{s_E}} \frac{\omega(x)}{Z} \cdot \frac{1}{|S_{s_E}|}. \end{aligned} \tag{4.11}$$

Unfortunately, we still require the normalisation constant Z in order to compute $P(x \mid L)$, whose computation is infeasible in most practical situations. Z could, however, be replaced by a sample-based approximation $\tilde{Z}$. A multiset of samples S that is to be used for the approximation of Z should satisfy

$$Z = \sum_{x \in \mathscr{X}} \omega(x) \overset{!}{\approx} \frac{|\mathscr{X}|}{|S|} \sum_{x \in S} \omega(x) = \sum_{x \in \mathscr{X}} \omega(x) \cdot \tilde{P}_S(x) \cdot |\mathscr{X}| =: \tilde{Z} \tag{4.12}$$

where $\tilde{P}_S(x) = n(x, S)/|S|$ is the sample-based approximation of the sampling distribution $P_S(x)$ from which S was drawn. The factor $\frac{|\mathscr{X}|}{|S|}$ compensates for the sample size. Equation 4.12 suggests that for $\tilde{Z}$ to be an approximation of Z, the sampling distribution P_S should be a uniform distribution over $\mathscr{X}$, i.e. $P_S(x) = 1/|\mathscr{X}|$ for all $x \in \mathscr{X}$. Based on the two sets of samples S_{s_E} and S, we could thus, in theory, perform the optimisation with

$$L_{\text{USMLSE}}(s_E) = \log\left(\frac{1}{|S_{s_E}|} \sum_{x \in S_{s_E}} \omega(x)\right) - \log\left(\frac{|\mathscr{X}|}{|S|} \sum_{x \in S} \omega(x)\right) \tag{4.13}$$

as the objective function and

$$\frac{\delta}{\delta w_i} L_{\text{USMLSE}}(s_E) = \sum_{x \in S_{s_E}} n_i(x) \frac{\omega(x)}{\sum_{x' \in S_{s_E}} \omega(x')} - \sum_{x \in S} n_i(x) \frac{\omega(x)}{\sum_{x' \in S} \omega(x')} \quad (4.14)$$

as the i-th component of the gradient.

Unfortunately, if the set of samples S is not very large, sampling from the uniform distribution is likely to result in $\tilde{Z}$ being a poor approximation of Z, since the worlds with the most significant contribution to Z may be missing from S, especially if the distribution $P(x \mid L)$ is highly non-uniform. We might, instead, sample from the prior ($P_S(x) = P(x \mid L)$), using MC-SAT sampling without any evidence to generate the samples. Using the approximation above, this will, however, lead to an estimate that is clearly not proportional to Z:

$$\tilde{Z} = \sum_{x \in \mathcal{X}} \omega(x) \cdot \tilde{P}_S(x) \cdot |\mathcal{X}| \approx \sum_{x \in \mathcal{X}} \omega(x) \cdot \frac{\omega(x)}{Z} \cdot |\mathcal{X}| \propto \sum_{x \in \mathcal{X}} \omega(x)^2 \quad (4.15)$$

$\tilde{Z}$ will, for example, overestimate Z in the presence of hard constraints, since worlds with zero probability are not sampled by MC-SAT.

A multiset of samples S generated with MC-SAT can, however, be used to approximate the expected number of true groundings of the formulas, thus giving rise to the following gradient computation,

$$\frac{1}{|S_{s_E}|} \sum_{x \in S_{s_E}} n_i(x) - \frac{1}{|S|} \sum_{x \in S} n_i(x), \quad (4.16)$$

which is based on the insight that the i-th component of the gradient should correspond to the difference between the number of true groundings of the i-th formula in the training data and its expectation as indicated by the model (see Section 3.3). In practice, it is best to avoid an error-prone approximation of Z and apply an optimisation technique that uses only the gradient and second-order information, without requiring an explicit computation of the objective function, such as the diagonal Newton method (cf. Lowd and Domingos, 2007a). To apply this method, we require only the gradient 4.16 and the Hessian matrix of second-order partial derivatives. As an approximation for the Hessian, one can use the negative covariance matrix of the

counts n_i in the sample set S, analogous to the hard evidence case described by Lowd and Domingos (2007a), i.e.

$$H_{ij} = \left(\frac{1}{|S|}\sum_{x\in S} n_i(x)\right) \cdot \left(\frac{1}{|S|}\sum_{x\in S} n_j(x)\right) - \frac{1}{|S|}\sum_{x\in S} n_i(x)\cdot n_j(x). \tag{4.17}$$

While sampling-based methods will typically be less efficient than the pseudo-likelihood-based method presented in the previous section, they make fewer simplifying assumptions regarding the interpretation of the soft evidence and its factorisation, and they do not rely on a scheme that is, in general, a crude approximation.

Related Work. So far, the problem of learning in the presence of soft evidence has not been given due treatment in the literature. Nevertheless, Xiao et al. (2009) previously presented a learning method for Bayesian networks which handles soft evidence that is given for individual random variables, assuming mutual independence between soft evidence probability values. When assuming independence, a data case containing degrees of belief for all the networks' variables straightforwardly defines a distribution over possible worlds, such that for each conditional distribution $P(X_i \mid \mathrm{Par}_{X_i})$ in the Bayesian network, one can consider the joint distribution over assignments to X_i and Par_{X_i} and use the probability of each assignment to build up the sufficient statistics (which, in standard Bayesian network maximum likelihood learning, are given by the number of times each joint assignment to X_i and Par_{X_i} appears in the data; cf. Heckerman (1995)). Since the methods presented in Section 4.3.1 make many of the same assumptions, they are similar in spirit to Xiao et al.'s method – in contrast to the sampling-based method presented in this section, which is somewhat less restrictive.

Chapter 5

Bayesian Logic Networks

With Markov logic networks, a very expressive representation formalism that elegantly combines logical and statistical knowledge was described. However, the practical applicability of MLNs is often limited due to the expensive nature of both the learning and the inference methods that support them. As we have seen, parameter learning in MLNs is an ill-posed problem and coarse approximations involving the use of pseudo-likelihood are often necessary to render applications practical. Furthermore, approximate inference (using, for example, MC-SAT) can be expensive even for conceptually simple queries. Other approaches integrating relational and statistical knowledge are loosely based on Bayesian networks (see Section 2.3.3), which are often easier to deal with – both in terms of learning and inference. However, these formalisms partly sacrifice expressiveness[1] for tractability. Typically, one can easily express only local probabilistic constraints, while even simple relational properties required on a global level, such as the transitivity or symmetry of an uncertain relation, cannot easily be modelled. While, in theory, we *are* able to represent most such properties using most representation formalisms (Jaeger, 2008), we cannot, unfortunately, do so in practice without substantially increasing the complexity of the first-order model (which, notably, does not necessarily coincide with the complexity of ground models instantiated from it, however). From a knowledge engineering perspective, simplicity and conciseness are key.

The representation formalism proposed in the following, Bayesian logic networks (BLNs) (Jain et al., 2009c, 2011), is a reasonable compromise in this regard. Its probabilistic components are based on conditional probability distribution templates

[1] Expressiveness is, in this context, not merely to be understood as a reflection of the ability to express but rather as the ability to express *in concise terms*.

for the construction of a Bayesian network. Global constraints are supported by a second model component, a template for the construction of a constraint network, in which deterministic constraints are represented using first-order logic.

5.1 Formalism

A *Bayesian logic network* can be viewed as a template for the construction of mixed networks with probabilistic and deterministic dependencies. Formally, a Bayesian logic network (BLN) $\mathscr{B}$ is a tuple $\langle \mathscr{D}, \mathscr{F}, \mathscr{L} \rangle$, where

- $\mathscr{D} = \langle \mathscr{T}, S, E, t \rangle$ comprises the model's fundamental *declarations*. $\mathscr{T}$ is a taxonomy of types, which is represented as a directed forest $\langle T, I \rangle$, where T is the actual set of types and $I \subset T \times T$ is the *generalises* relation (inverse is-a), i.e. $(T_i, T_j) \in I$ iff T_i is a generalisation of T_j. S is a set of signatures of functions (used to define the model's set of random variables), and E is a set of (abstract) entities that are to exist in all instantiations, whose types are given by the function

  $$t : E \to 2^T \setminus \{\emptyset\}$$

  which maps every entity to the non-empty subset of types $T = \left\{T_1, \ldots, T_{|T|}\right\}$ it belongs to. The function t thus induces a cover of the set of entities with sets $E_{T_i} = \left\{e \in E \mid T_i \in t(e)\right\}$. (The function t is assumed to be consistent with $\mathscr{T}$, i.e. if $(T_i, T_j) \in I$, then $e \in E_{T_j}$ implies $e \in E_{T_i}$.)

  The set S contains the signature of every function f, defining the domain and the range of the function in terms of types, i.e.

  $$(f, (T_{i_1}, \ldots, T_{i_n}), T_r) \in S \iff f : E_{T_{i_1}} \times \cdots \times E_{T_{i_n}} \to E_{T_r}.$$

  Logical predicates are simply boolean functions, i.e. functions that map to $E_{T_r} = \mathbb{B}$, and we implicitly assume that the corresponding type symbol *Boolean* is always contained in T and that $\mathbb{B} = \{\mathit{True}, \mathit{False}\} \subseteq E$.

  A set E_{T_r} that corresponds to a type $T_r \in T$ which appears as the return type of a function is regarded as a (fixed) *domain*, i.e. as a fixed set of entities that must be fully contained in E, whereas the extensions of other types may vary from

instantiation to instantiation (and the corresponding subsets of E may even be empty in the model).

- $\mathscr{F}$ is a set of *fragments* of conditional probability distributions. Every fragment $F_i \in \mathscr{F}$ defines a dependency of an abstract random variable $f(p_1, \ldots, p_n)$ (the *fragment variable* V_{F_i}) on a set of other abstract random variables (the *parents* Par_{F_i}), where f is one of the functions defined in S, and the parameters $p_1, \ldots, p_n$ are either variables (called *meta-variables*) typed according to f's signature or entities in E belonging to the respective type. The dependencies are encoded in a *conditional probability function* (CPF) $P_{F_i} : \mathrm{dom}(V_{F_i}) \times \mathrm{dom}(\mathrm{Par}_{F_i}) \to [0,1]$, which defines, for every setting of the parent variables (i.e. every element in the domain product of the parent variables), a probability distribution over the elements in the domain of the fragment variable (i.e. the range of f): $\forall u \in \mathrm{dom}(\mathrm{Par}_{F_i}). \sum_{v \in \mathrm{dom}(V_{F_i})} P_{F_i}(v,u) = 1$

  Additionally, a fragment may define preconditions for its applicability, which may involve arbitrary logical statements about the parameters $p_1, \ldots, p_n$ (or parameters that can be functionally determined from these).

- The set $\mathscr{L}$ consists of formulas in first-order logic (with equality) over the functions/predicates defined in S, which represent hard deterministic dependencies.

Instantiation. For any given set of entities E', whose types are given by a function $t' : E' \to 2^T \setminus \emptyset$, a Bayesian logic network $\mathscr{B}$ defines a ground mixed network $M_{\mathscr{B},E'} = \langle\langle X, D, G, P\rangle, \langle X, D, C\rangle\rangle$ as follows:

- E and t in $\mathscr{B}$ are augmented to include E', t'. Typically, an instantiation is considered valid only if in the resulting set E, all type-specific subsets are non-empty, i.e. $\forall T_i \in T.\ E_{T_i} \neq \emptyset$.

- The set of random variables X contains, for each function $(f, (T_{i_1}, \ldots, T_{i_n}), T_r) \in S$ and each tuple of applicable entities $(e_1, \ldots, e_n) \in E_{T_{i_1}} \times \cdots \times E_{T_{i_n}}$, an element $X_i = f(e_1, \ldots, e_n)$. The corresponding domain $D_i \in D$ is E_{T_r}.

- The conditional probability function $P_i \in P$ that is applicable to a random variable $X_i = f(e_1, \ldots, e_n)$ is determined by $\mathscr{F}$, which must either contain exactly

one fragment for f whose preconditions are met given the actual parameters or must specify a *combining rule* that defines how to combine several fragments into a single conditional distribution. The connectivity of the directed graph G is such that there is a directed edge from every parent to the fragment variable – as indicated by the applicable fragments. An instantiation is valid only if the resulting graph is acyclic.

- For every grounding of every formula in $\mathscr{L}$ (obtained by substituting quantified variables by the applicable elements of E accordingly), the set C contains a constraint $C_i = \langle S_i, R_i \rangle$, where S_i, the scope of the constraint, is the set of random variables mentioned in the ground formula, and R_i is the relation indicating the combinations of values for which the formula is satisfied.

In the special case where $\mathscr{L} = \emptyset$, the mixed network contains no constraints and the ground network is a Bayesian network $\langle X, D, G, P \rangle$.

For purposes of inference, we may generally convert any mixed network $M_{\mathscr{B},E'}$ into an *auxiliary Bayesian network* $B_{\mathscr{B},E'}$, allowing us to leverage the large body of approximate inference algorithms available for Bayesian networks. As explained in Section 2.2.4, $B_{\mathscr{B},E'}$ is constructed from $M_{\mathscr{B},E'}$ by adding to X for every constraint $C_i = \langle S_i, R_i \rangle \in C$ a Boolean auxiliary variable A_i that represents the corresponding constraint and has as its parents in G all the nodes that the constraint depends on (as indicated by its scope S_i), such that for $|C| = k$, the following equivalence holds:

$$P_{M_{\mathscr{B},E'}}(X = x) = P_{B_{\mathscr{B},E'}}(X = x \mid A_1 = \mathit{True}, \ldots, A_k = \mathit{True}) \tag{5.1}$$

Notice that conditioning on the auxiliary variables is equivalent to adding virtual evidence on the constrained random variables with observation probabilities 1.0 and 0.0.

Example. As a simple example, let us consider a relational version of the mixed network from Section 2.2.4 (page 29) which generalises to arbitrary numbers of dishes and people. The model uses the types $T = \{\mathit{dish}, \mathit{person}\}$, whose extensions in the model are empty. A person has two Boolean properties (being female and being a

vegetarian) and a relation connects people with the dishes they order. Therefore, the set of signatures is given by

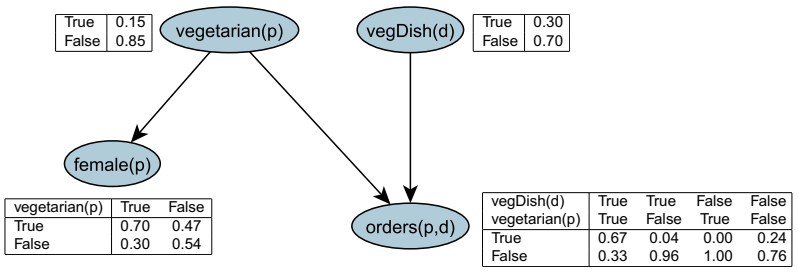
$$\begin{aligned} S = \{&(\mathit{female}, \mathit{person}, \mathit{Boolean}),\\ &(\mathit{vegetarian}, \mathit{person}, \mathit{Boolean}),\\ &(\mathit{vegDish}, \mathit{dish}, \mathit{Boolean}),\\ &(\mathit{orders}, (\mathit{person}, \mathit{dish}), \mathit{Boolean})\} \end{aligned}$$

Because all functions are *Boolean*, the set E contains no elements beyond the set $\mathbb{B}$ of truth values. The set of logical formulas $\mathscr{L}$ contains the constraint that restricts the number of dishes that must be ordered by each person to two:

$$\begin{aligned} \forall p.\, \exists d_1, d_2.\, \mathit{orders}(p, d_1) \wedge \mathit{orders}(p, d_2) \wedge \neg(d_1 = d_2) \wedge\\ \neg\exists d_3.\, \mathit{orders}(p, d_3) \wedge \neg(d_1 = d_3) \wedge \neg(d_2 = d_3) \end{aligned}$$

The set of fragments $\mathscr{F}$ associates a (conditional) probability distribution with each of the functions in S. These distributions, along with declarations that define the above, are shown in BLN 5.1.

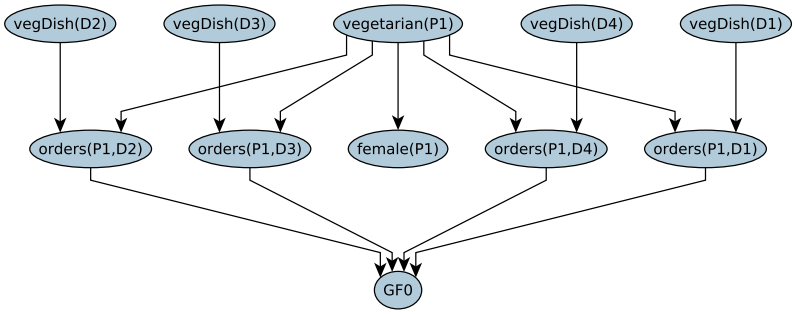
For given extensions of the types *dish* and *person*, this model can be instantiated to obtain either a ground mixed network or a ground auxiliary Bayesian network.



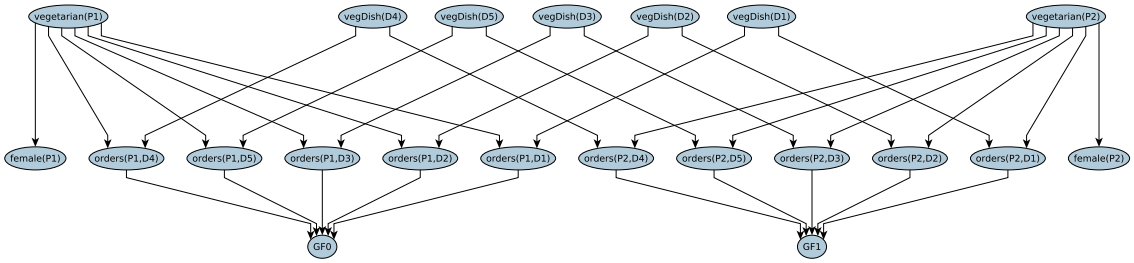

```
1  type person, dish;
2  random boolean vegetarian(person);
3  random boolean female(person);
4  random boolean orders(person,dish);
5  random boolean vegDish(dish);
6  constraint EXIST d1,d2 (orders(p,d1) ^ orders(p,d2) ^ !(d1=d2) ^ !(EXIST d3 (orders(p,d3) ^ !(d1=d3)
       ^ !(d2=d3))));
```

BLN 5.1: *A relational adaptation of the "restaurant" mixed network from Section 2.2.4: The set of fragments is defined using a graphical representation (top) and plain-text definitions are used for the set of declarations $\mathscr{D}$ and the set of logical formulas $\mathscr{L}$.*

Two ground auxiliary Bayesian networks are shown in Figure 5.1. Note that, in the given model, the number of dishes that are permissible for an instantiation is not entirely arbitrary: Domains with fewer than two dishes are not permissible because of the constraint in $\mathscr{L}$; neither are instantiations paired with evidence that demands that there is a vegetarian but at the same time rules out the presence of at least two vegetarian dishes.



(a) $E_{\text{person}} = \{P1\}, E_{\text{dish}} = \{D1, D2, D3, D4\}$



(b) $E_{\text{person}} = \{P1, P2\}, E_{\text{dish}} = \{D1, D2, D3, D4, D5\}$

Figure 5.1: *Ground auxiliary Bayesian networks constructed from BLN 5.1 for domains with (a) one person and four dishes (the network is equivalent to the mixed network from Section 2.2.4) and (b) two people and five dishes. The nodes GF0 and GF1 are the auxiliary nodes that represent groundings of the logical formula. In case (b), the constraint is split into two separate ground formulas, because p is not explicitly universally quantified. As in MLNs, p being a free variable translates to separate formula groundings, the simultaneous truth of which is semantically equivalent to universal quantification.*

5.2 Representation in Practice

This section provides detailed information about the representation of Bayesian logic networks in practice. It describes the syntax that is used for the declarations $\mathscr{D}$ which specify types, entities belonging to these types and the signatures of functions. For the declaration of the set of fragments $\mathscr{F}$, a graphical representation is used, while the set $\mathscr{L}$ of logical formulas is again defined based on textual declarations.

5.2.1 Fundamental Declarations

Let us first address the fundamental declarations that make up a Bayesian logic network model. BLNs follow some of the same conventions as MLNs: They make the unique names assumption and use the same identifier naming convention for constants and variables.

5.2.1.1 Types, Entities and Signatures

The set of types T, the set E of known entities (and, implicitly, the function t assigning types to these entities), as well as the set of signatures S can be defined as follows:[2]

```
1 type person, student isa person, professor isa person;
2 type domGrade, course;
3 guaranteed domGrade None, A, B, C, D, F;
4 logical boolean takes(student, course);
5 random domGrade grade(student, course);
6 random boolean likes(person, person);
```

- The keyword type is used to list elements of the set of types T, and isa is used to build up the taxonomy.
- The keyword guaranteed is used to list constant symbols referring to entities that belong to a particular type and are guaranteed to be contained in all instantiations of the model (which is mandatory for fixed domains, i.e. types that

[2]The syntax of BLN declarations is, in part, equivalent to the one used by BLOG models (Milch et al., 2005). The background is that early implementations of BLN learning methods could be used to train restricted BLOG models.

appear as function ranges, such as *domGrade*). We may also define guaranteed domain elements for non-fixed domains; in such cases, the declaration does not prohibit further elements from being added to the type's extension during instantiation.

- The keyword random precedes a function signature declaration, which serves as the basis for the instantiation of random variables. Line 5 defines a function that maps from a student-course pair to the domain of grades (*domGrade*), and line 6 defines a predicate applicable to pairs of people.
- The keyword logical is used to declare predicates that are determined logically. Such predicates are either given as evidence or are determined based on logical reasoning (from atoms that are given as evidence), and we consequently make the closed-world assumption, i.e. all atoms that are neither given as evidence nor are logically entailed are assumed to be false. Because logical variables are not random variables (their truth values are known a priori), we need not define probabilistic fragments for them. In the example above, the courses taken by students are assumed to be known a priori and the predicate *takes* is declared as logical accordingly.

5.2.1.2 References to Other Model Components

BLN declarations may reference external files. In particular, references to files containing the BLN's network of fragments $\mathscr{F}$ and the set of logical constraints $\mathscr{L}$ can be specified as follows:

```
1 fragments grades_network.pmml;
2 constraints grades_constraints.blnl;
```

5.2.1.3 Functional Relations

In addition to the fundamental declarations above, we may want to make statements about functional dependencies between the entities connected by a logical relation. The declaration of such dependencies makes it possible to perform a functional lookup, provided that the predicate itself is an evidence predicate, i.e. its full

extension is always given as evidence. Functional relations thus replace functions in first-order logic. A functional relation can even replace several functions, for a relation can be used to simultaneously define several mappings.

For example, imagine a model of a sequence of actions as given by a relation such as next(a1,a2), which states that an action a2 happens after a1. In such a model, if an action is to probabilistically depend on its predecessor, we need to be able to functionally determine the predecessor from the successor during the instantiation of the ground network. In BLNs, functional relations can be declared using *relation keys*: Any (sub-)tuple of the relation's arguments can be defined as a key for a functional lookup. A relation key definition thus takes a Boolean function f and declares that some of its arguments form a key. The syntax is as follows:

```
1 type action;
2 logical boolean next(action, action);
3 relationKey next(a1, _);
4 relationKey next(_, a2);
```

An underscore indicates a functionally determined argument while the remaining arguments (which can be arbitrarily named in the declaration) make up the key. The above example declares two functional dependencies for the next relation. Line 3 states that for an action a1, we can functionally determine the action that succeeds it, i.e. we can map any a1 to its successor. Line 4 declares the inverse mapping. Such functional dependencies can, in particular, be used to look up variables whose values cannot otherwise be determined during the instantiation of a fragment.

5.2.2 Conditional Probability Fragments

The second defining part of a BLN, the set $\mathscr{F}$, specifies generalized conditional probability fragments for the functions/predicates defined in $\mathscr{D}$, indicating the dependencies between abstract random variables (typically function terms where at least some of the arguments are variables). A graphical representation method is used to define $\mathscr{F}$ in a *fragment network*, which facilitates the intuitive modelling of abstract random variables and their dependencies: As shown in BLN 5.1, dependencies are modelled similar to standard Bayesian networks, arcs indicating dependencies.

The template structure given in the fragment network is the foundation for the materialisation of the probabilistic part of ground mixed network $M_{\mathscr{B},E'}$: For every function and every tuple of applicable entities, we instantiate a node and determine its parents in the ground network according to the sub-structure of the fragment network that is applicable.

5.2.2.1 Auxiliary Nodes

By default, every node defines a fragment and thus represents a template for the construction of a random variable's conditional or marginal distribution in the ground network. The parents of a node may be fragment variables themselves, interconnecting different fragments in the network. However, to keep the fragment network clearly arranged, a fragment may also be completely detached from other fragments. We then define the parents of the fragment variable using auxiliary nodes that do not represent fragments themselves and therefore are not associated with a conditional distribution. The # operator allows us to declare such auxiliary parent nodes without simultaneously declaring a fragment, as shown in Figure 5.2. In Figure 5.2a, the three fragments for a, b and c are defined in three detached parts, using auxiliary parent nodes in the specification of the fragments for b and c, whereas in Figure 5.2b, the three fragments are defined in a single connected network without auxiliary parent nodes. In general, the parents in any fragment are strictly interpreted at the syntactic level; the conditional distributions attached to them or their ancestors are irrelevant for the specification of the fragment.

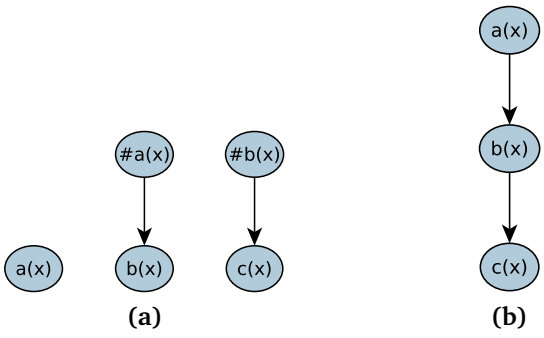


Figure 5.2: *Two equivalent fragment networks demonstrating the use of auxiliary nodes*

The use of auxiliary variables can improve clarity, as it can reduce the number of arcs in crowded regions of the fragment network. Furthermore, auxiliary nodes can be used in cases where a fragment variable is to depend on two or more variables referring to the same function, or to allow renaming of the meta-variables appearing as parameters of fragment variables.

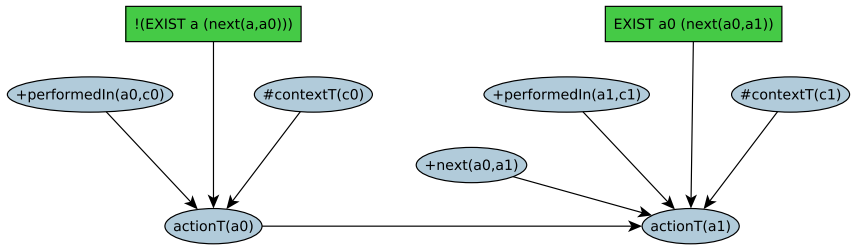
5.2.2.2 Preconditions

The fragment network may contain more than one fragment for any function declared in the set of signatures S. To decide which fragment is applicable, fragments can define preconditions. The definition of preconditions is, for example, useful if we want to generate different structures in the ground network depending on the values of given evidence variables. Since preconditions are evaluated during the grounding/instantiation step, preconditions are necessarily limited to checks of evidence variables.

Preconditions are defined in special parent nodes of fragment variables. There are two ways of specifying a precondition:

- The most general way of specifying preconditions is through logical *precondition nodes* (see the green/rectangular nodes in Figure 5.3), which check for the truth of an arbitrary logical formula (in which only evidence predicates appear). Only if the formula evaluates to true do we use the respective fragment to instantiate the conditional probability distribution of the random variable in question. Precondition nodes thus allow to create completely different substructures in the ground network depending on the evaluation of formulas (the formula syntax is described in Section 5.2.3). Because preconditions frequently refer to mutually exclusive conditions, a precondition node may represent the negation of another precondition: the precondition can then simply bear the text "neg" and have the precondition that is to be negated as its parent.

  When a precondition formula is evaluated during instantiation, the meta-variables that appear in the fragment variable are always bound to their respective values and can be used in the formula. Any other variables appearing in the formula must be either universally or existentially quantified.




```
1 logical boolean performedIn(action, context);
2 relationKey performedIn(a, _);
3 logical boolean next(action, action);
4 relationKey next(_, a);
```

Figure 5.3: *Excerpt of a fragment network for a sequence of actions and declarations that are most relevant to this particular excerpt. The two fragments for* actionT *that are shown make use of preconditions specified using both logical formulas (green/rectangular nodes) and the + operator.*

In Figure 5.3, an excerpt of a fragment network for a sequence of actions is shown; it contains two fragments for the function actionT, which maps an action to its type. Actions depend on the context in which they are performed. Decision nodes are used to differentiate between initial actions at the beginning of a sequence (left fragment) and all the other actions (right fragment), where the types of the latter are determined not only by the context they are performed in but also by the type of the preceding action. Notice that in the precondition formulas, the variables that would otherwise be free are bound by the respective fragment variable (i.e. a0 in the left fragment and a1 in the other). When checking the applicability of a fragment during instantiation, these variables are bound to constants.

- Simpler preconditions that only demand that a particular evidence predicate be true (for a particular binding of the variables) can be handled using the + operator: By prefixing a Boolean parent (i.e. a parent referring to a Boolean evidence function) with a + character, we state that the fragment in question is to apply only if (for the variable assignment in question) the parent evaluates to true.

  Usage examples are shown in Figure 5.3. The fragment on the right hand side can be applied only if next(a0,a1) and performedIn(a1,c1) hold in addition to the

precondition formula specified in the green rectangular node.[3] Note that the preconditions refer to meta-variables that do not appear in the fragment variable (a0 and c1 are not bound by actionT(a1)). For any binding of a1, we can, however, determine which entities we must consider because of the functional dependencies that were declared for next and performedIn, which allow us to functionally determine a0 and c1 from a1.

Since preconditions are always required to be true, the corresponding nodes never appear in the ground network that is instantiated. In the case of Figure 5.3, we will therefore obtain random variables with one and two parents for the left and right fragment respectively.

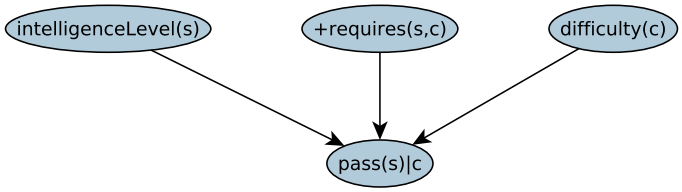
5.2.2.3 Combining Rules

So far, we assumed that precisely one fragment was applicable for each random variable that is instantiated. If more than one fragment is applicable, we can use *combining rules* to compute a single conditional distribution from the individual distributions associated with the applicable fragments and use as the set of parents the union of all the sets of parents that are indicated by the applicable fragments. For a particular Boolean random variable X_i, let there be K applicable fragments. For the j-th applicable fragment, let the set of ground parents be $\mathrm{Par}_{X_i}^{(j)}$, i.e. let the fragment define the conditional distribution $P(X_i \mid \mathrm{Par}_{X_i}^{(j)})$. Combining the K conditional distributions using a combining rule *cr*, which maps an arbitrary list of probability values to a new probability value, the conditional distribution of X_i is instantiated as

$$P(X_i = \mathit{True} \mid \cup_{j=1}^{K} \mathrm{Par}_{X_i}^{(j)}) = \mathit{cr}(P(X_i = \mathit{True} \mid \mathrm{Par}_{X_i}^{(1)}), \ldots, P(X_i = \mathit{True} \mid \mathrm{Par}_{X_i}^{(K)})). \quad (5.2)$$

For $X_i = \mathit{False}$, we use the complementary probabilities. In the case of a non-Boolean variable X_i, we can use the combining rule to compute a value for each element in the domain of the variable and then apply a normalisation if necessary. Not all combining rules can sensibly be applied to non-Boolean variables, however.

[3] The green rectangular node is actually superfluous for the right hand fragment, because the condition it requires is already covered by the precondition +next(a0,a1).




```
1 logical boolean requires(student, course);
2 random boolean pass(student);
3 combining−rule pass noisy−and;
```

Figure 5.4: *Excerpt of a model that is concerned with students at a university that are required to take certain courses. For each required course, the model defines the conditional probability of the student passing, which is influenced by the student's intelligence level and the course's level of difficulty. The overall probability of passing is determined by combining the influence of all courses using the combining rule* noisy-and*, as declared in line 3. The suffix |c in the fragment variable for* pass *explicitly states that the variable* c *is free (as it cannot be functionally determined from* s*) and therefore the fragment can potentially be applied multiple times depending on the entities that* c *can be bound to given the precondition* requires(s,c)*.*

The following combining rules are supported by the ProbCog system:

$$\textit{noisy-or}(p_1,\ldots,p_K) = 1-\prod_{j=1}^{K}(1-p_j) \tag{5.3}$$

$$\textit{noisy-and}(p_1,\ldots,p_K) = \prod_{j=1}^{K} p_j \tag{5.4}$$

$$\textit{min}(p_1,\ldots,p_K) = \min\{p_j\}_{j=1}^{K} \tag{5.5}$$

$$\textit{max}(p_1,\ldots,p_K) = \max\{p_j\}_{j=1}^{K} \tag{5.6}$$

$$\textit{average}(p_1,\ldots,p_K) = \frac{1}{K}\sum_{j=1}^{K} p_j \tag{5.7}$$

The combining rule that is to be used for a particular predicate/function is defined by adding a statement to the model's declarations using the keyword combining-rule. For instance, consider the excerpt of the university model shown in Figure 5.4, where the probability of a student passing all his required courses is modelled using the combining rule *noisy-and*, thus stating that the overall probability of passing is the result of the student passing all of the individual courses, assuming that the events of passing any two courses are mutually independent. In this example, the influence of several instances of the same fragment was combined. Of course, the conditional

distributions that we combine with a combining rule can also originate from entirely different fragments pertaining to the same function/predicate.

5.2.2.4 Functional Value Definitions

In some cases, it can be useful to define the value of a random variable as a function of other random variables. For instance, we might want to define a Boolean random variable whose value corresponds to the disjunction of sentences involving other random variables. To support this, the BLN language includes the =OR prefix operator (see Figure 5.5). For a Boolean random variable X_i whose fragment uses this operator, let there be K parent instantiations $\{\text{Par}_{X_i}^{(j)}\}_{j=1}^{K}$. With $\text{Par}_{X_i} := \cup_{j=1}^{K}\text{Par}_{X_i}^{(j)}$, the conditional probability distribution of X_i is given by

$$P(X_i = \mathit{True} \mid \text{Par}_{X_i} = \text{par}_{X_i}) = \begin{cases} 1.0 & \text{if } \text{Par}_{X_i} = \text{par}_{X_i} \models \bigvee_{j=1}^{K} \bigwedge_{X_l \in \text{Par}_{X_i}^{(j)}} X_l = \mathit{True} \\ 0.0 & \text{otherwise} \end{cases} \tag{5.8}$$

The =AND prefix operator is defined analogously.

Figure 5.5 shows a model that is concerned with the habits of people participating in meals. Several types of utensils and goods may be used and consumed respectively. People use particular utensils in order to consume particular goods in meals, as indicated by the predicate usesAnyForIn(p,u,g,m). If, in applications of the model, we sometimes are not concerned with what particular types of utensils are used

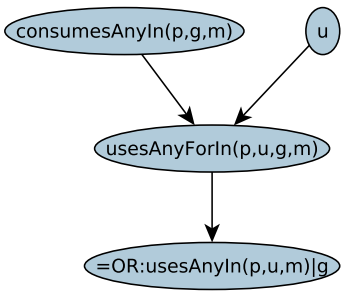


Figure 5.5: *Excerpt of a fragment network on meal habits that uses a functional value definition and domain nodes*

for, then it may be sensible to additionally define usesAnyIn(p,u,m). The =OR operator in Figure 5.5 achieves that the truth value of usesAnyIn(p,u,m) is equivalent to $\exists g.\ usesAnyIn(p,u,g,m)$. We could alternatively define this type of equivalence using a logical formula in the set $\mathscr{L}$, but the use of the prefix operator may be preferable for computational reasons.

5.2.2.5 Domain Nodes (Per-Constant Dependencies)

In previous examples, a fragment's conditional distribution was exclusively dependent on values of abstract random variables declared in S. There are many situations in which it can be sensible to have conditional distributions that depend on elements of a (fixed) domain, i.e. in effect, to have a separate conditional distribution per constant belonging to the domain. To support this, the parent of a fragment variable may simply bear the name of one of the meta-variables appearing in the fragment variable. The parent, called a *domain node*, then stands for the entities/constants in the domain that corresponds to the type the meta-variable belongs to. In Figure 5.5, the domain node u is used for utensil types, such that, for each type of utensil, the model contains a separate conditional probability function for usesAnyForIn.

5.2.3 Logical Formulas

The set $\mathscr{L}$ in a Bayesian logic network defines a set of logical formulas constraining the set of possible worlds. Formulas are either declared in a separate text file, which contains one formula per line and is included from the declarations via the keyword constraints, or can be defined directly in the declarations using the keyword constraint followed by the actual constraint to be included. The following plain-text syntax is used for the logical operators:

negation	!
conjunction	^
disjunction	v
implication	=>
biimplication	<=>

Quantifiers are expressed as follows,

existential quantification	EXIST vars (formula)
universal quantification	FORALL vars (formula)

where vars is a comma-separated list of variables (identifiers beginning with lowercase letters) and formula is any complex formula (in which these variables appear) expressed using the logical connectives. Universal quantification may be implicit, i.e. if a formula contains free variables, these variables are assumed to be universally quantified. For use in logical formulas, non-Boolean functions in the BLN are converted to predicates with the value of the function as an additional parameter, i.e. an assignment such as $f(A,B) = C$ of function $(f,(T_1,T_2),T_3) \in S$ becomes $f(A,B,C)$.

5.2.4 Evidence

To instantiate a ground model, we typically require evidence on some set of entities E'. The evidence may either state the mere existence of the entities or may make arbitrary statements about the entities using the predicates/functions defined in S. The syntax is as follows,

```
1  takes(John, ArtificialIntelligence) = True
2  student = {Mary, Alice, Bob}
```

where line 1 makes a concrete statement about a student taking a particular course, and line 2 states only that three particular entities of type student exist. Sometimes, the use of integer constants as entities may be appropriate – particularly in sequence models for the definition of the time domain. An integer range may be specified using the syntax

```
1  time = {0..10}
```

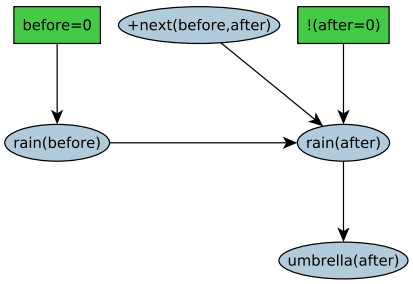
Prolog Rules. The use of a logical model can be helpful for the specification of evidence, because there may be patterns according to which additional pieces of evidence can be computed from given seed evidence. For instance, an evidence predicate may represent a transitive relation, and we may want the transitive closure to be computed automatically. To this end, we can make use of Prolog rules that we can add to

the model's declarations using the keyword prolog. For instance, we can compute the reflexive, symmetric, transitive closure similarRST of the relation similar as follows,

```
1 prolog similarRST(X,X)
2 prolog similarRST(X,Y) :- similar(X,Y)
3 prolog similarRST(X,Y) :- similar(Y,X)
4 prolog similarRST(X,Z) :- similar(X,Y), similarRST(Y,Z)
```

and use in our model either similar or similarRST as required. Arbitrary auxiliary Prolog predicates that never actually appear in the probabilistic model may be defined.

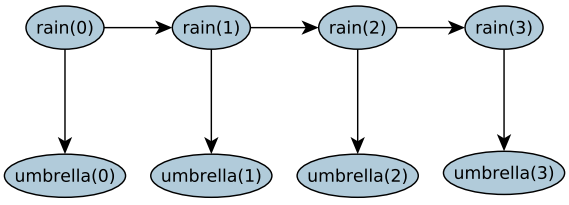
As a particularly useful pattern, consider the sequence model defined in BLN 5.2. Prolog is used to fully define the temporal succession relation (line 6). To instantiate the model for a particular number of time steps, we therefore only need to specify a concrete range of time steps.



(a) Fragment Network

```
1 type time;
2 random boolean rain(time);
3 random boolean umbrella(time);
4 logical boolean next(time, time);
5 relationKey next(_,y);
6 prolog next(X,Y) :- V is X+1, Y=V.
```

(b) Declarations



(c) Ground Model

BLN 5.2: *Representation of the "umbrella world" hidden Markov model (Russell and Norvig, 2003, ch. 15) as a Bayesian logic network (a, b) and a ground model generated from it for time = {0..3} (c)*

5.2.5 Discussion

As this section has shown, Bayesian logic networks integrate many practical features that facilitate the construction of models in practice. The use of mixed networks, which combine conditional distributions with logical constraints, sets them apart from other formalisms that strictly adhere to the principles of Bayesian networks. The parts of the BLN language that are concerned with the representation of the directed model, however, bear similarities to other formalisms. For instance, the notion of a network-centric, fragment-based approach is also a prominent feature of multi-entity Bayesian networks (Laskey, 2008), the separation of random variables from logical variables also exists in logical Bayesian networks (Fierens et al., 2005), and the use of combining rules also appears in relational Bayesian networks (Jaeger, 1997) and Bayesian logic programs (Kersting and Raedt, 2007). The motivation for the development of BLNs – beyond the construction of a flexible framework on top of mixed networks – was to provide a practical toolbox that integrates the features that were most needed for many high-level applications in technical systems, and to make this toolbox freely available in the hope that it will foster the development of cognition-enabled applications.

5.3 Approaches to Learning and Inference

This section gives an overview of learning and inference methods for Bayesian logic networks. The current implementation of BLN learning in the PROBCOG system takes a very pragmatic approach, which makes many simplifying assumptions. For problems where these assumptions would be inappropriate, applications can make use of the more advanced – and more computationally expensive methods – that are available for (adaptive) Markov logic networks in the PROBCOG system. Regarding inference, however, the PROBCOG system supports a wide range of state-of-the-art methods.

5.3.1 Learning

One of the main advantages of directed models over undirected models is that exact maximum likelihood parameter learning can be performed without requiring inference over the model. Specifically, maximum likelihood learning in directed models can be performed based purely on counting relative frequencies in the data. Assuming independence of the parameters (Spiegelhalter and Lauritzen, 1990), the maximum likelihood estimate of the conditional probability of a variable assignment $X_i = x_i$ given an assignment to X_i's parents, $\mathrm{Par}_{X_i} = \mathrm{par}_{X_i}$, is simply the relative frequency of $X_i = x_i$ among all data instances where $\mathrm{Par}_{X_i} = \mathrm{par}_{X_i}$ (Heckerman, 1995).

In directed relational models, parameter learning can make use of the same principles, the only difference being that many of the parameters in a relational model are tied. If the relational data is given in the form of a relational database management system that can be queried efficiently, one can even compute the sufficient statistics (i.e. the counts) using, for instance, SQL queries (Getoor, 2001). The PROBCOG system supports the computation of the frequency of occurrence of relational structures in the data by translating the respective conditions to queries in the graphical query language QGraph (Blau et al., 2002), which can be efficiently answered through the use of a vertical database system as a backend.[4]

[4]The implementation is based on the Proximity software suite, http://kdl.cs.umass.edu/proximity. Because Proximity supports only binary relations, higher-arity predicates and functions are reified for use with BLN databases, and queries are automatically translated to comply with the modified structure of the data.

Since Bayesian logic networks are partly based on directed models, a pragmatic learning method is to apply standard learning methods of directed relational models in order to learn the parameters in the fragment network. For a BLN $\mathscr{B} = \langle \mathscr{D}, \mathscr{F}, \mathscr{L} \rangle$, consider the problem of learning the parameters from a completely specified training database d. For a fragment $F_i \in \mathscr{F}$, let $N_i(v,u)$ be the number of times a ground variable to which the fragment F_i applies takes on the value v while the parents take on the value u (in the database d). Let the parameters of F_i be given by θ_i and denote by $\theta_{i,v|u}$ the conditional probability for the case where $V_{F_i} = v$ and $\mathrm{Par}_{F_i} = u$, i.e. $\theta_{i,v|u} = P_{F_i}(v,u)$. Assuming independence of the parameters, the log-likelihood of the fragment parameters can be computed as

$$\begin{aligned} L(d \mid \theta) &= \log \prod_{F_i \in \mathscr{F}} \prod_{u \in \mathrm{dom}(\mathrm{Par}_{F_i})} \prod_{v \in \mathrm{dom}(V_{F_i})} \theta_{i,v|u}^{N_i(v,u)} \\ &= \sum_{F_i \in \mathscr{F}} \sum_{u \in \mathrm{dom}(\mathrm{Par}_{F_i})} \sum_{v \in \mathrm{dom}(V_{F_i})} N_i(v,u) \cdot \log(\theta_{i,v|u}) \end{aligned} \tag{5.9}$$

Analogous to pure Bayesian networks, the parameters that maximise this expression are the relative frequencies in the data, i.e. $\theta_{i,v|u} = N_i(v,u) / \sum_{v' \in \mathrm{dom}(V_{F_i})} N_i(v',u)$. The counts $N_i(v,u)$ thus constitute sufficient statistics for parameter learning.

If data availability is limited, using these relative frequencies may, however, result in overfitting, such that models may be unnecessarily restrictive, as many assignments that could in fact occur in the real world may be assigned zero probability. A most ad hoc solution to this problem is to add small *pseudo-counts* to the counts $N_i(v,u)$.

If the training data is not completely specified, i.e. the values of some variables are unknown, one can use the *expectation maximisation* algorithm: Starting from an initial parameter estimate, the algorithm alternates between an expectation step, which computes the expected sufficient statistics via inference over the model (conditioning on the data that *is* given), and a maximisation step, which computes the maximum likelihood or MAP parameter estimate based on these statistics, until convergence (Dempster et al., 1977; Heckerman, 1995).

The notion of disregarding the set of constraints $\mathscr{L}$ during parameter learning is sensible if one interprets the hard constraints only as a post-filter on the distributions that are given by the directed model; the relative importance of possible worlds that is indicated by the directed model is maintained. Also, combining rules are not ex-

plicitly considered; the fragments that would be combined by a rule are considered independently, the effect of the rule itself is disregarded. Depending on the reasoning behind the use of a combining rule, learning the fragments independently may, however, be sensible.

If both the effect of hard constraints and combining rules are to be considered to their full extent, learning will necessarily involve inference over the model, thus leading to hard learning problems. A potential problem in ignoring the effect of hard constraints is that constraints will typically change the marginals of the variables that appear within the constraints (and in turn the beliefs pertaining to other variables influenced by these). While we could, in theory, restore the marginals of the constrained atoms to the frequencies that indeed appear in the training database whilst maintaining the satisfaction of the constraints – as long as the constraints were indeed satisfied in the training database – by adding virtual evidence on the constrained atoms (using likelihoods that achieve precisely the desired correction), we cannot, unfortunately, do so in a way that will generalise across domains. If there are multiple constraints (or multiple instances of the same constraint) and the atoms appearing in different constraints are not independent, then the local corrections made to modify a marginal distribution will influence each other, rendering our efforts futile as the number of constraints varies across domains. We might address this problem in two ways. First, we could enforce the marginals that are to be maintained by applying an IPFP-based method during inference. Because the marginals can, however, be domain-dependent, a distribution satisfying both the hard constraints and the probability constraints that maintain the marginals indicated by the directed model may not exist. For such cases, Peng and Zhang (2010) have introduced a smoothing algorithm which computes a reasonable compromise. Second, we could use learning methods analogous to those of adaptive Markov logic networks and explicitly take the constraints into consideration. Since such methods are not currently implemented for BLNs in ProbCog, one should fall back to using AMLNs or MLNs if an application of such learning methods is absolutely necessary.

5.3.2 Inference

There are several approaches to inference in Bayesian logic networks. A straightforward approach is to instantiate a ground network, i.e. either a ground auxiliary Bayesian network or a ground mixed network, and then apply one of many standard inference algorithms. Ground mixed networks may also be translated into Markov random fields with weighted features, which effectively enables MLN inference algorithms to be applied to BLNs. Lifted inference methods will seek to abstract away from ground networks, exploiting the repeated structures that result from the application of the template model.

Inference in the Ground Auxiliary Bayesian Network. By converting a ground mixed network to an auxiliary Bayesian network, we can leverage the large body of inference algorithms devised for standard Bayesian networks, including

- exact inference methods, such as variable elimination (Zhang and Poole, 1996) or arithmetic circuits evaluation (ACE, Chavira and Darwiche, 2005),
- sampling-based methods, such as likelihood weighting (Fung and Chang, 1990), backward simulation (Fung and Del Favero, 1994) or importance sampling based on evidence-prepropagation (Yuan and Druzdzel, 2003),
- Markov chain Monte Carlo methods, such as Gibbs sampling (Geman and Geman, 1984),
- message-passing methods, such as loopy belief propagation (Pearl, 1988) or iterative join-graph propagation (Dechter et al., 2002).

All of the methods named above (and many more) are supported by the ProbCog toolbox for BLNs. Sampling-based methods, which are particularly attractive, as they are conceptually simple yet can often provide reliable any-time approximations, will be discussed in more detail in Section 5.4.

Inference in the Ground Mixed Network. The use of the mixed network can allow algorithms to explicitly take the constraints represented in the constraint network into consideration. Mateescu and Dechter (2008) propose two exact inference algorithms for mixed networks – one based on bucket elimination, the other based

on AND/OR search. Furthermore, an approximate inference algorithm that adapts importance sampling to the specific requirements of mixed networks, SampleSearch, was introduced in Gogate and Dechter (2007). MC-SAT, though originally formulated for Markov logic networks (see Section 3.2.1.2), is another inference algorithm that is applicable to mixed networks, as any ground mixed network can straightforwardly be translated into a set of weighted ground formulas.

5.4 Sampling-Based Inference

This section takes a more detailed look at sampling-based inference methods – specifically, at sampling methods that are particularly well-suited to applications in probabilistic models with determinism as we obtain them by instantiating a BLN. I first give an overview of fundamental sampling techniques (Section 5.4.1) and then present novel approaches in Sections 5.4.2 and 5.4.3 (cf. Jain et al., 2011).

We shall consider the inference task of computing the probability of an assignment to query variables $Q \subset X$ given an assignment to evidence variables $E \subset X$, i.e. the task of computing the posterior distribution $P(Q = q \mid E = e)$. Notice that, in any model with determinism (such as a mixed network whose constraint network is non-empty), the inference problem of approximating $P(Q = q \mid E = e)$ via sampling includes a constraint satisfaction problem (CSP) as a sub-problem: In each sampling step, any assignment that is sampled must be a solution to the CSP that is given by the constraint network. Furthermore, additional constraints may be implicitly defined in the network's set of conditional probability distributions: If the conditional probability of an evidence node can be zero under certain conditions, then the avoidance of these conditions constitutes additional constraints.

5.4.1 Fundamental Sampling Techniques

Rejection Sampling. In Bayesian networks, sampling from the prior $P(x)$ is a particularly simple task: If random variables are ordered topologically with respect to G, we can sequentially sample from each (marginal or conditional) distribution in order to generate a full assignment to all variables in X. Let $S = \{x^{(1)}, \ldots, x^{(N)}\}$ be a multiset

of samples drawn from $P(x)$. For sufficiently large N, the relative frequency with which an assignment was sampled will be an approximation of its probability, i.e. for all $x \in \mathscr{X}$, $n(S,x)/N \approx P(x)$, where $n(S,x)$ is the number of samples in S satisfying the assignment x. We can therefore compute a query as $P(q \mid e) \approx n(S, q \wedge e)/n(S,e)$. Since all samples that do not satisfy $E = e$ are irrelevant to the query, they can be ignored and are consequently rejected by the sampler. If the probability of the evidence, $P(E = e)$, is low a priori, the sampling process may need to reject the majority of samples, rendering the sampling process inefficient. This is known as the *rejection problem*.

Importance Sampling. Importance sampling (Rubinstein, 1981) is a general technique that allows to reduce the problem of rejections by sampling from some *importance function* $Q(x)$ instead of the prior $P(x)$, the idea being that $Q(x)$ should somehow use the evidence and the structure of the model to avoid sampling irrelevant assignments. If $Q(x)$ is a probability distribution that is to be used for the approximation of a query $P(q \mid e)$, the only additional constraint that needs to be placed on $Q(x)$ is that it must not assign zero probability to any $x \in \mathscr{X}$ for which $P(x \mid e) \neq 0$, i.e. the support of the importance function must include the support of the true posterior $P(x \mid e)$. Given a set of samples S (as above but now taken from $Q(x)$), we have that for all $x \in \mathscr{X}$, $n(S,x)/N \approx Q(x)$, and therefore $n(S,x)/N \cdot P(x)/Q(x) \approx P(x)$ for all x satisfying $E = e$. Thus a query can be computed as

$$\begin{aligned} P(q \mid e) &= \frac{\sum_{x \in \mathscr{X}, x \models q \wedge e} P(x)}{\sum_{x \in \mathscr{X}, x \models e} P(x)} \approx \frac{\sum_{x \in \mathscr{X}, x \models q \wedge e} n(S,x)/N \cdot P(x)/Q(x)}{\sum_{x \in \mathscr{X}, x \models e} n(S,x)/N \cdot P(x)/Q(x)} \\ &= \frac{\sum_{x \in \mathscr{X}, x \models q \wedge e} n(S,x) \cdot w(x)}{\sum_{x \in \mathscr{X}, x \models e} n(S,x) \cdot w(x)} = \frac{\sum_{x^{(i)} \in S, x^{(i)} \models q \wedge e} w(x^{(i)})}{\sum_{x^{(i)} \in S, x^{(i)} \models e} w(x^{(i)})} \end{aligned} \tag{5.10}$$

where $w(x) := P(x)/Q(x)$. Therefore, by assigning a *weight* of $P(x)/Q(x)$ to each sample, we compensate having sampled from $Q(x)$ instead of $P(x)$ and can approximate the desired posterior.

Note that importance sampling generalises rejection sampling: For $Q(x) = P(x)$, we recover the standard rejection sampler with $w(x) = 1$ for all $x \in S$ satisfying e.

Likelihood Weighting. The perhaps most straightforward choice for the importance function $Q(x)$ is used in *likelihood weighting* (Shachter and Peot, 1989; Fung and

Chang, 1990): Evidence nodes are not sampled but simply assigned the value that is given in the evidence; otherwise the sampling process is completely analogous to rejection sampling. Hence $Q(x)$ is given by

$$Q(x) = \prod_{X_i \in X} \begin{cases} P(X_i = x_i \mid \text{par}_{X_i}^x) & \text{if } X_i \notin E \\ 1 & \text{if } E = e \models X_i = x_i \\ 0 & \text{otherwise} \end{cases} \tag{5.11}$$

and sample weights are computed as

$$w(x) = \frac{P(x)}{Q(x)} = \prod_{X_i \in E} P(X_i = x_i \mid \text{par}_{X_i}^x). \tag{5.12}$$

Of course, if the conditional probability of the evidence is zero at any evidence node (i.e. $P(X_i = x_i \mid \text{par}_{X_i}^x) = 0$ for some $X_i \in E$), the sample is still effectively rejected, since samples with zero weight do not contribute to the estimation.

Backward Simulation. Fung and Del Favero (1994) presented another way of focusing the importance function $Q(x)$ on the evidence. Instead of sampling forward in topological order, we can sample backward from the evidence, i.e. we can assign the parents of a node X_i whose value is already given by sampling an assignment to Par_{X_i} according to the conditional distribution of X_i: We sample $\text{Par}_{X_i} = \text{par}_{X_i}$ with probability proportional to $P(x_i \mid \text{par}_{X_i})$. The backward sampling process is repeated until all ancestors of evidence nodes have been instantiated. Any remaining nodes to which no values have been assigned are forward sampled to obtain a complete assignment to all variables. In Figure 5.6, this process is illustrated for a simple Bayesian network.

The algorithm first determines a sampling order $\mathcal{O}$. This step partitions the set of nodes into three sets: backward sampled nodes $\mathcal{O}_B$, forward sampled nodes $\mathcal{O}_F$ and unsampled nodes $\mathcal{O}_O$ that are outside the sampling order. To compute an ordering, we manage a list of backward sampling candidates – initially the list of evidence nodes. For each candidate X_i, we determine whether any of its parents are yet uninstantiated and if so, we add it to $\mathcal{O}$ as a backward sampled node and add its parents to the list of candidates (as during sampling, X_i's value will be used to *instantiate* the previously uninstantiated parents). Otherwise the node is outside the actual sampling order and

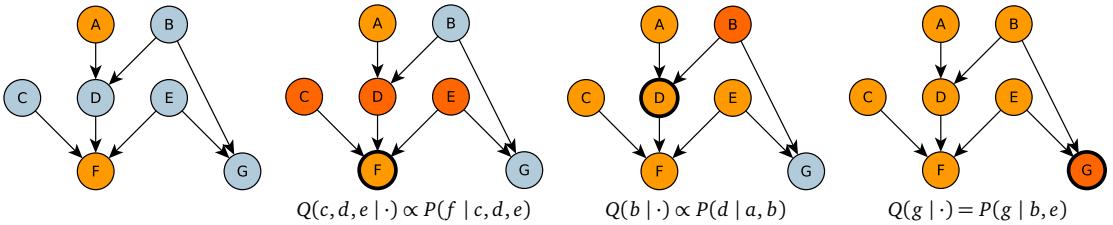


Figure 5.6: *Illustration of backward simulation. Initially, $A = a$ and and $F = f$ are given as evidence. In the first sampling step, we backward sample from F, assigning values to its parents C, D and E according to the conditional distribution $P(f \mid c,d,e)$. In the second step, we backward sample from the now instantiated variable D, assigning its yet uninstantiated parent B according to the conditional distribution $P(d \mid a,b)$. In the last step, no further nodes can be backward sampled, and we thus forward sample the only yet uninstantiated node G according to its conditional distribution $P(g \mid b,e)$.*

is part of $\mathcal{O}_O$. All remaining uninstantiated nodes are forward sampled and are added to the sampling order in topological order (a forward sampled node instantiates itself).

Given the semantics of forward and backward sampling, the importance distribution of backward simulation is

$$Q(x) = \prod_{X_i \in \mathcal{O}_B} \frac{P(X_i = x_i \mid \mathrm{par}^x_{X_i})}{Z_i} \cdot \prod_{X_i \in \mathcal{O}_F} P(X_i = x_i \mid \mathrm{par}^x_{X_i}) \tag{5.13}$$

where the Z_i are normalisation constants. Thus sample weights are computed as

$$w(x) = \frac{P(x)}{Q(x)} = \prod_{X_i \in \mathcal{O}_B} Z_i \prod_{X_i \in \mathcal{O}_O} P(X_i = x_i \mid \mathrm{par}^x_{X_i}). \tag{5.14}$$

The SampleSearch Scheme. Even though importance sampling techniques such as backward simulation can considerably reduce the rejection problem, there may still be too many rejections to obtain a sufficient number of samples within a reasonable time frame. To address this problem, Gogate and Dechter (2007) proposed the SampleSearch scheme, which systematically searches for samples. Rather than discarding a sample as soon as its weight becomes zero and restarting, we can go back and reassign variables until we find an assignment that has a non-zero weight.

The simplest version of SampleSearch samples from the prior $P(x)$, consecutively assigning values to each random variable in topological order. If it encounters an evidence variable where the probability of the evidence is 0 given the values previously assigned to the variable's parents, it backtracks to the previous variable in the topological ordering, excluding the value that was previously assigned, renormalizing the conditional distribution and choosing a different value. Should all the values within a domain have been excluded, we backtrack further – and so on, until all variables have ultimately been assigned a value that is compatible with the evidence. Of course, more elaborate versions of SampleSearch can make use of advanced techniques known from satisfiability testing (Gogate and Dechter, 2007) or constraint satisfaction problems (e.g. arc consistency and conflict-driven backjumping (Dechter and Frost, 2002)), and, instead of the prior $P(x)$, we could use an importance function computed via a message-passing algorithm such as loopy belief propagation.

It is important to note that the probability $Q(x)$ with which a sample is selected in SampleSearch cannot directly be used to compute weights that will yield an asymptotically unbiased estimator for the desired posterior, for we may select the same sample with different probabilities depending on the exclusions we had to make in order to obtain it. In effect, the algorithm samples from the backtrack-free distribution and hence the highest possible probability for each sampling step (which occurs if the maximum number of exclusions has been made) will yield an unbiased estimator (Gogate and Dechter, 2007). This probability can be approximated using the set of samples collected.

5.4.2 Backward SampleSearch

In the following, the notion of backward simulation is combined with a sample searching scheme to obtain *Backward SampleSearch* (BSS), i.e. we perform backward simulation and apply backtracking whenever the sample weight becomes zero – until we find an assignment that represents a solution to the induced constraint satisfaction problem.

Since chronological backtracking is typically highly inefficient, *conflict-directed backjumping* (Dechter and Frost, 2002) is used in the more efficient variant BSS-BJ:

Whenever we encounter a partial assignment that cannot be completed to a full assignment, we jump back to the node with the highest index in the sampling order that has bearing on the conflict that we encountered. With Algorithm 5, pseudocode for the generation of a sample with BSS-BJ is provided (some of the symbols used in the algorithm are not introduced until further on). Line 11 performs, for $X_i \in \mathcal{O}_B \cup \mathcal{O}_F$, either backward sampling or forward sampling (using the given exclusions and returning a new state), and, for $X_i \in \mathcal{O}_O$, it checks whether the current assignment is valid given the previously sampled parents of X_i. In sampledIndices$[i]$, we store the index of the value (node assignment) that was sampled from the respective distribution. Lines 14-22 realize conflict-directed backjumping for the specific semantics of backward simulation. The set B_i is used to collect order indices of candidate nodes for a backjump from X_i.

As an additional optimisation, the distribution used for backward sampling makes use of a more informed estimate of the probability of parent assignments. The original backward simulation algorithm goes backward "blindly" in the sense that it does not explicitly consider any information from evidence that is further up – beyond the parents of the variable X_i that is backward sampled. As a computationally simple improvement, one can use a crude approximate belief $\tilde{P}(X_j = x_j)$ for parents X_j of X_i in addition to the conditional distribution $P(x_i \mid \mathrm{par}_{X_i})$. $\tilde{P}$ propagates evidence only forward. If $X_i \in E$, then $\tilde{P}(X_i = x_i) = 1$ if $E = e \models X_i = x_i$ and 0 otherwise. If $X_i \notin E$, it is computed as

$$\tilde{P}(X_i = x_i) := \sum_{\overline{x} \in \mathrm{dom}(\overline{X})} P(X_i = x_i \mid \overline{x}, \mathrm{par}^e_{X_i}) \cdot \prod_{X_j \in \overline{X}} \tilde{P}(X_j = \overline{x}_j) \tag{5.15}$$

where $\overline{X} = \mathrm{Par}_{X_i} \setminus E$ and $\mathrm{par}^e_{X_i}$ is the assignment of $\mathrm{Par}_{X_i} \cap E$ in e.

When backward sampling from X_i, some of X_i's parents are yet uninstantiated while others may already be instantiated (either because they are evidence variables or because they were instantiated by a previous backward sampling step, as determined by the sampling order). Par_{X_i} is partitioned into two sets U^{X_i} and I^{X_i} accordingly. We sample the yet uninstantiated parents U^{X_i} of X_i given previous assignments via

$$Q(U^{X_i} = u^{X_i} \mid \cdot) = \frac{1}{Z_i} P(X_i = x_i \mid U^{X_i} = u^{X_i}, I^{X_i} = i^{X_i}) \prod_{X_j \in U^{X_i}} \tilde{P}(X_j = u^{X_i}_j) \tag{5.16}$$

where Z_i is a normalization constant and $u_j^{X_i}$ is X_j's value in u^{X_i}. Forward sampled nodes are treated as in backward simulation. Therefore, we can compute sample weights as follows:

$$w(x) = \prod_{X_i \in \mathscr{O}_B} \frac{Z_i}{\prod_{X_j \in U^{X_i}} \tilde{P}(X_j = x_j)} \prod_{X_i \in \mathscr{O}_O} P(X_i = x_i \mid \mathrm{par}_{X_i}^x) \tag{5.17}$$

As in SampleSearch, directly using these weights will yield a biased estimator. An unbiased estimated can be obtained by keeping track of the highest probability values with which each particular partial assignment was sampled, which we obtain for the smallest Z_is in Equation 5.16 that were observed during the sampling process.

Algorithm 5 SAMPLE-BSS-BJ

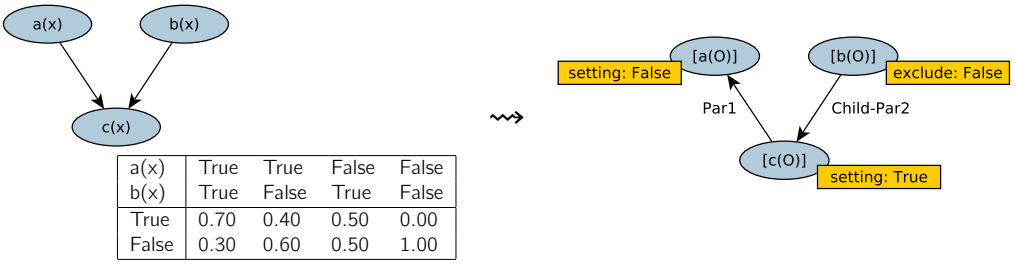
```
 1: state ← empty assignment
 2: i ← 0
 3: backtracking ← false
 4: while i < |X| do
 5:    X_i ← nodes[samplingOrder[i]]
 6:    if backtracking then
 7:       exclusions[i].append(sampledIndices[i])
 8:    else
 9:       exclusions[i] ← empty list
10:       B_i ← ∅
11:    sampledIndices[i], state ← sample(state, X_i, exclusions[i])
12:    if sampledIndices[i] is null then
13:       backtracking ← true
14:       if X_i ∈ O_B then
15:          B_i ← B_i ∪ {j | The j-th node in the order instantiated X_i or a node in I^{X_i}}
16:       else
17:          B_i ← B_i ∪ {j | The j-th node in the order instantiated a node in Par_{X_i}}
18:          if X_i ∈ O_O then B_i ← B_i ∪ {j | The j-th node in the order instantiated X_i}
19:       if B_i = ∅ then return "inconsistent evidence"
20:       i_prev ← i
21:       i ← max B_i
22:       B_i ← B_i ∪ B_{i_prev} \ {i}
23:    else
24:       backtracking ← false
25:       i ← i + 1
26: return state
```

5.4.3 SampleSearch with Abstract Constraint Learning

When sampling, we are repeatedly faced with the same problem – generating a sample from a distribution – and when applying SampleSearch, we thus repeatedly encounter the same dead-ends and are forced to backtrack in precisely the same way. Even during the generation of a single sample, we may encounter the same local dead-end several times. In the constraint solving community, this has given rise to the notion of constraint (or *nogood*) learning (Dechter and Frost, 2002) – the natural principle of learning from one's past mistakes. The combination of SampleSearch with constraint learning is essentially straightforward: Whenever we backtrack/backjump, we record a constraint that forbids the configuration that caused the dead-end, and whenever an assignment that is relevant to the constraint is made in the future, we check whether the constraint is applicable and if so, we add a domain exclusion. Should all domain values have been exhausted, we backtrack, recursively generating further constraints.

In instances of relational models such as BLNs, we not only encounter specific dead-ends repeatedly as described above but also dead-ends that pertain to different variables but are otherwise completely analogous. This is due to the repeated substructures found within instantiations of a relational model. Thus, the recording of abstract constraints that allow to represent such analogous constraints in a single form suggests itself. The use of abstract constraints potentially has several advantages:

- There are fewer abstract constraints and fewer constraints are more quickly found. Therefore, the search for samples can potentially be speeded up.
- In the best case, the recording of constraints successfully trades speed for memory usage. With fewer constraints, an unrestricted recording of abstract constraints may still be feasible, while the number of propositional constraints may already be too large to fit in memory.
- Because they are not specific to a particular instantiation, abstract constraints can be saved and stored for future use in order to apply them across different instantiations of the same relational model.



a(x)	True	True	False	False
b(x)	True	False	True	False
True	0.70	0.40	0.50	0.00
False	0.30	0.60	0.50	1.00

Figure 5.7: *Construction of a simple abstract constraint: If, for the grounding where x is substituted with O, $c(O) = \mathrm{True}$ is given as evidence, the constraint shown on the right may be discovered. In contrast to a propositional constraint, which would only store the setting of $a(O)$, the abstract constraint explicitly stores the information on the evidence that must be present for the constraint to apply.*

For purposes of abstraction, we consider an equivalence relation over the random variables X. In BLNs, we can uniquely identify the CPF that was generated from the fragments for any variable X_i and use this information – perhaps along with information on evidence – to define the relation. To compute an equivalence relation that considers the evidence, one might, for instance, use a node colouring scheme similar to the one described by Kersting et al. (2009) to obtain an equivalence relation that will be sound with respect to the domain exclusions it will later result in. In practice, even very simple equivalence relations that consider only the CPF identifiers of all the variables in a variable's Markov blanket can suffice if information on relevant evidence is represented within the constraints themselves. This notion of equivalence is the one presently implemented.

An abstract constraint is represented as a graph whose nodes are equivalence classes and whose edges indicate the relationship that is to exist between the two nodes against which the edge is matched. The relationships are based on the topology of the ground network, and we thus consider the i-th-parent relation and its inverse. To represent the actual constraint, any node in the graph may involve a check for a particular assignment of the node it is matched against. An example is shown in Figure 5.7. SampleSearch with abstract learning (SS-AL) was implemented based on this definition of abstract constraints.

It should be clear that abstraction comes at a price. Abstract constraints are larger, because the graph representation requires additional nodes that would not appear in

a propositional constraint which does not need to concern itself with relationships between variables. Moreover, checking an abstract constraint incurs an unavoidable overhead, since it requires matching a relational structure. To evaluate whether the positive or the negative aspects of abstraction prevail, SampleSearch with propositional learning (SS-L) was implemented as a point of reference.

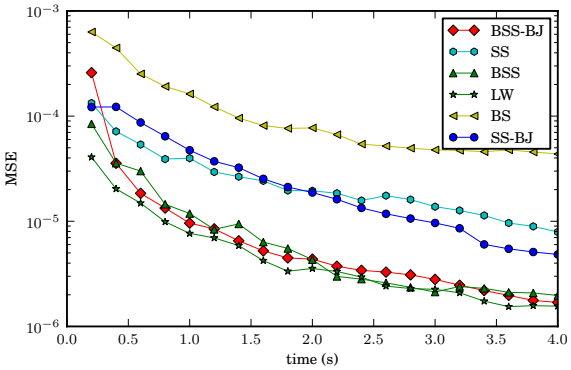
5.4.4 Experiments

In a series of experiments, the inference methods described above are compared on instantiations of five BLN models (see Figure 5.8): The "grades" models are adaptations of the university model that was described by Getoor et al. (2007); the "meal habits" model captures the consumption and utensil usage habits of people during their daily meals; the "kitchen layouts" model associates the spatial configuration of storage locations and devices in a kitchen with consumable objects and tools; and the "sequential activities" model is concerned with object locations and states changing as a result of activities of daily life being performed over time. The three latter models are described in more detail in Section 6.2.1.2. For each model, an inference task was selected at random.

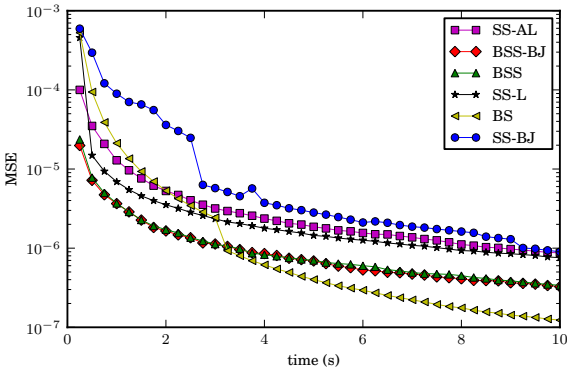
In Figures 5.8a-5.8e, the mean squared error in the posterior marginals of non-evidence variables is plotted against runtime, averaged across five trials. The algorithms used are: likelihood weighting (LW), SampleSearch with chronological backtracking (SS) and backjumping (SS-BJ), backward simulation (BS), backward SampleSearch (BSS, BSS-BJ), and learning-based search methods with backjumping (SS-L, SS-AL).[5]

We observe that, in three of our scenarios, BSS methods are among the best, clearly outperforming their forward SS counterparts in two scenarios. In the two other scenarios, the roles are reversed. However, it is the learning-based methods that perform best in these scenarios. In "sequential activities", this is due to the sequential nature of this particular model, which causes frequent backtracking that can be prevented by learning constraints. The abstract learning method SS-AL never surpasses its propositional counterpart SS-L. Apparently, the overhead incurred by abstraction cannot

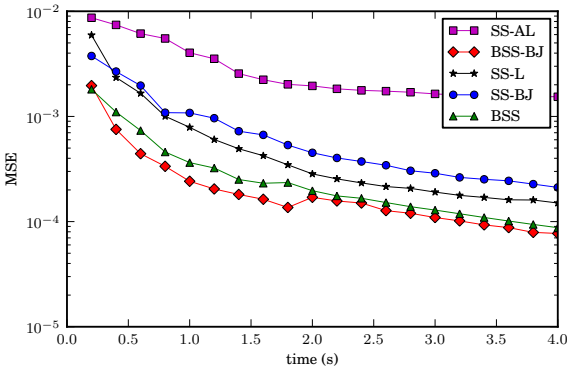
[5]All methods were implemented in Java. For all search-based methods, asymptotically unbiased estimators were used.



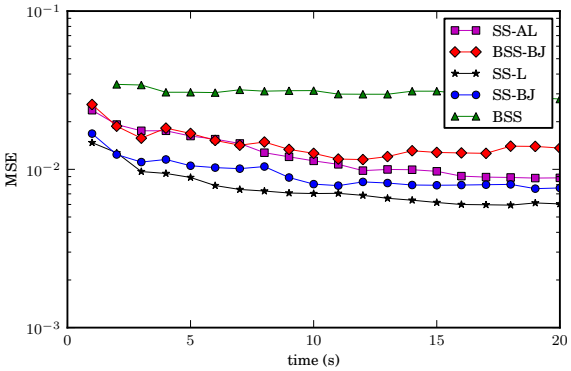
(a) grades



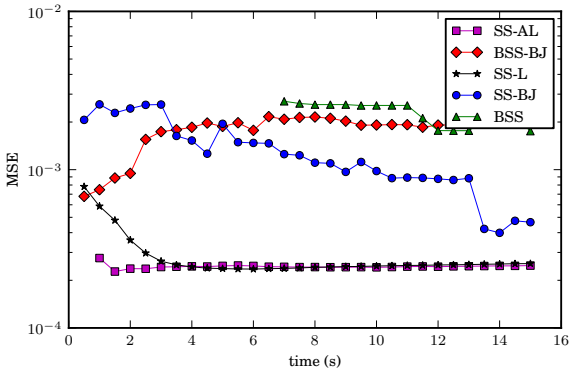
(b) grades extended



(c) meal habits



(d) kitchen layouts



(e) sequential activities

	grades	grades ext.	meal habits	kitchen layouts	seq. act.
#nodes	144	1200	562	125	153
N(LW)	14603	0	0	0	0
N(SS)	49760	0	0	0	0
N(SS-BJ)	56341	5085	2634	1063	6947
N(BS)	1379	31004	6	0	0
N(BSS)	7170	6883	4502	13	7
N(BSS-BJ)	9429	6586	4502	102	1292
N(SS-L)	9210	3469	2910	32648	39910
N(SS-AL)	7615	2702	827	704	10977

(f) problem sizes and #samples drawn

Figure 5.8: *Evaluation of sampling methods on instances of five different BLNs. Mean squared errors in posterior marginals of non-evidence variables are plotted in subfigures (a)–(e). The table (f) provides data on problem sizes and the number of samples drawn by the end of the time limit.*

be amortised by the reduced number of constraints in the instances that were tested. Because constraint learning constitutes additional overhead in general, it is a disad-

vantage in cases where it is essentially unnecessary. As expected, conceptually simple methods such as LW and BS are unable to compute approximations for any of the more challenging problems. The quality of the samples generated by the algorithms does not seem to vary widely, as approximation quality appears to be strongly correlated with the number of samples drawn (cf. Figure 5.8f).

As a point of reference, a Markov chain Monte Carlo method, namely MC-SAT, was also applied to each of the scenarios. While MC-SAT could generate approximately 100 samples for the simple "grades" model, no more than five samples could typically drawn for any of the other models due to the sheer size of the SAT problems that resulted from the large number of weighted formulas in each of the problem instances. Although MC-SAT can presently be considered as the perhaps most practical inference method for Markov logic networks, its performance was not comparable to the sampling-based methods on this particular set of problem instances.

In conclusion, sampling-based methods can provide efficient approximations, yet there can, of course, be no single method that should be preferred in all cases. The experimental results indicate that both backward simulation and constraint learning are viable options for the search for samples under the right circumstances. The recording of abstract constraints, however, incurs too much of an overhead, which cannot be compensated by gains in compactness – at least in the problem instances that were investigated.

5.4.5 Estimating Approximation Quality

Sampling methods allow us to obtain any-time approximations, i.e. they enable us to obtain an approximation at any point in time during the inference process by returning an estimate based only on the set of samples that has thus far been collected. The question that arises in this context is: How good is the approximation, and how many samples are required for the approximation to be "approximately correct"? Especially in scenarios where the quality of the approximations or the decisions that will be based upon them can have serious consequences, we desire guarantees on the quality of the approximations. Firm guarantees are, of course, difficult to obtain,

yet we can attempt to provide probabilistic confidence bounds on the quality of the approximations obtained with a sampling-based inference method.

In the sampling methods described above, samples are independently drawn from the same distribution. Hence the statistical tools applicable to independent and identically distributed (i.i.d.) samples can directly be applied to this form of inference. In particular, the well-known concepts underlying Bayesian parameter estimation (Russell and Norvig, 2003, ch. 20) can be transferred to the inference problem.

5.4.5.1 Bayesian Parameter Estimation

In Bayesian parameter estimation, we treat the parameter to be estimated as a random variable Θ whose domain encompasses all potential parameter values. This allows us to explicitly take into consideration the uncertainty with regard to the true value of the parameter. We assume some initial prior $P(\Theta = \theta)$ and update our beliefs on the distribution of Θ whenever a new sample s is observed using Bayes' rule:

$$P(\Theta = \theta \mid s) \propto P(s \mid \Theta = \theta)P(\Theta = \theta) \tag{5.18}$$

We can incrementally update our beliefs in this way for each sample s in order to recover a new posterior (taking, for subsequent steps, the previous posterior as the new prior for the update).

If the posterior $P(\theta \mid s)$ and the prior $P(\theta)$ are within the same family of distributions, then the two distributions are called *conjugates* and the prior $P(\theta)$ is called a *conjugate prior* for the likelihood $P(s \mid \theta)$. Given the form of the likelihood, any conjugate prior and its associated family of distributions can potentially be used to compute posterior beliefs on the parameter θ using successive Bayesian updates.

In the context of inference, we can reasonably view the posterior marginal distribution $P(X_i \mid e)$ of some Boolean random variable X_i as a Bernoulli distribution whose parameter p_i is to be estimated based on the set of samples S that is drawn by the sampling procedure and that is relevant to our query (i.e. S does not contain rejected samples not satisfying $E = e$). S can be partitioned into two sets $S^i_{True} := \{s \in S \mid s \models X_i = True\}$ and $S^i_{False} := S \setminus S^i_{True}$ accordingly. The likelihood $P(s \mid p_i = p)$,

i.e. the probability of a sample $s \in S$ given the success probability of the Bernoulli distribution, is then given by

$$P(s \mid p_i = p) = \begin{cases} p & \text{if } s \in S^i_{True} \\ 1-p & \text{otherwise.} \end{cases} \tag{5.19}$$

The *beta distribution*, whose probability density function is given by

$$\text{beta}[\alpha,\beta](p) = \frac{p^{\alpha-1}(1-p)^{\beta-1}}{\int_0^1 p'^{\alpha-1}(1-p')^{\beta-1}dp'}, \tag{5.20}$$

is a conjugate prior of the Bernoulli-distributed likelihood 5.19. As the initial prior, we can use $\text{beta}[1,1]$ and thus assume that p_i is uniform in $[0;1]$. Given that the evidence can potentially affect the posterior of X_i arbitrarily, this is a reasonable assumption to make.

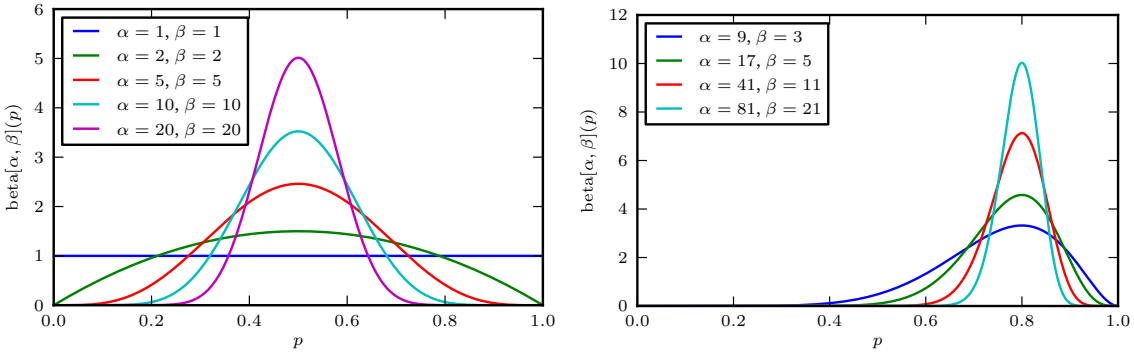
To estimate the parameter p_i based on the set of samples S, we perform a Bayesian update for each sample $s \in S$. For $s \in S^i_{True}$, the update results in an increment of α,

$$\begin{aligned} P(p_i = p \mid s) &\propto P(s \mid p_i = p)P(p_i = p) \\ &= p \cdot \text{beta}[\alpha,\beta](p) \\ &\propto p \cdot p^{\alpha-1}(1-p)^{\beta-1} \\ &\propto \text{beta}[\alpha+1,\beta](p) \end{aligned} \tag{5.21}$$

and for $s \in S^i_{False}$, the update results in an increment of β. Hence p_i can, having considered the entire set of samples S, be assumed to be beta-distributed according to the frequencies $N^i_{True} := |S^i_{True}|$ and $N^i_{False} := |S^i_{False}|$:

$$p_i \sim \text{beta}[N^i_{True}+1, N^i_{False}+1] \tag{5.22}$$

Figure 5.9 illustrates how the posterior beliefs on the distribution of the parameter p_i change with the number of samples in S^i_{True} and S^i_{False}. Notably, the mode of the beta distribution, which is given by $(\alpha-1)/(\alpha+\beta-2)$, is equivalent to the estimate of the (rejection) sampler, $N^i_{True}/|S|$, such that an application of Bayesian parameter estimation can, in principle, be soundly integrated into the existing sampling formalism.



(a) **(b)**

Figure 5.9: *Plots of the beta distribution's probability density function for various parametrisations. Figure (a) shows the initial prior distribution (*beta$[1,1]$*) and four densities for cases where the number of positive and negative samples is equal; the mode of these distributions is 0.5 accordingly. Figure (b) shows plots for cases where the ratio* $N^i_{True} : N^i_{False} = \alpha - 1 : \beta - 1$ *of positive to negative samples is not* $1:1$ *but* $8:2$*, thus implying modes at 0.8. In all cases, we observe that as the number of samples increases, distributions become increasingly peaked.*

In the case of a categorical, non-Boolean variable, the Dirichlet distribution, a multivariate generalisation of the beta distribution, can be used as the conjugate prior. If we are only interested in assessing the approximation of one particular value of a categorical variable, we may also treat the other values as a single event and apply the beta distribution.

Adaptation for Importance Sampling. When applying importance sampling, we cannot directly use the counts N^i_{True} and N^i_{False} to parametrise the beta distribution, since we must account for having sampled from the importance distribution rather than the prior. In particular, we must take into consideration the variance of the importance sampler's estimator, which decreases linearly with the number of samples $N := |S|$ and depends on the sample variance $\mathrm{Var}_Q[w(x)]$ of the weights that are obtained using $Q(x)$. The estimator variance of a standard sampler, which samples from $P(x)$ rather than $Q(x)$, also decreases linearly with the number of samples N. Since the ratio between the estimator variance of a standard sampler and that of an importance sampler can be approximated as $1/(1 + \mathrm{Var}_Q[w(x)])$ (Koller and

Friedman, 2009), we can expect N samples drawn using the importance sampler to be roughly equivalent to

$$N_{\text{eff}} := \frac{N}{1 + \text{Var}_Q[w(x)]} \tag{5.23}$$

samples drawn using the standard sampler. N_{eff} is called the *effective sample size*. The variance term can be computed as

$$\text{Var}_Q[w(x)] = E_Q[w(x)^2] - E_Q[w(x)]^2 = \frac{1}{N}\sum_{x \in S} w(x)^2 - \left(\frac{1}{N}\sum_{x \in S} w(x)\right)^2. \tag{5.24}$$

For importance sampling, we thus set the hyperparameters α and β by taking the sample weights into consideration, scaling the effective sample size with the current estimate of the importance sampler:

$$\alpha = 1 + N_{\text{eff}} \frac{\sum_{x \in S,\, x \models X_i = \mathit{True}} w(x)}{\sum_{x \in S} w(x)} \tag{5.25}$$

$$\beta = 1 + N_{\text{eff}} \frac{\sum_{x \in S,\, x \models X_i = \mathit{False}} w(x)}{\sum_{x \in S} w(x)} \tag{5.26}$$

With this choice of hyperparameters, the mode of the beta distribution will match the estimate of the importance sampler, and the effective sample size is reflected in $\alpha + \beta - 2$.

5.4.5.2 Confidence Intervals and Termination Criteria

Within our sampling-based inference process, we can explicitly consider the uncertainty with respect to the posterior marginal probability – as represented in the distributions obtained via Bayesian parameter estimation – in order to derive a notion of confidence for the approximate posterior beliefs that are computed. As we gather more samples, distributions get increasingly peaked and therefore the probability with which the true result lies within a small interval around the mode of the distribution increases. We consider the probability γ with which, according to the distribution obtained via Bayesian estimation, the true parameter is within an interval I around the mode $\tilde{p}$ of the distribution; we call γ the *coverage probability* or *confidence level*. For any confidence level $\gamma \in]0, 1]$, we can compute an interval I around

$\tilde{p}$ such that $P(p_i \in I) = \gamma$. As the confidence level is raised, the size of the interval increases.

Depending on the application, the confidence interval I may be interesting in itself, yet we can also use it to derive a termination criterion. Given a specific confidence level, the quality of the approximation that has been reached is reflected in the size of the interval I: The smaller the interval, the better the approximation. Thus, a reasonable termination criterion that is based on an estimation of the quality of the approximation is to assume some confidence level γ and a maximum interval size $s_{\max} \in]0;1]$ and to subsequently run sampling-based inference until the size of an interval $I = [p_l, p_u]$ that satisfies $P(p_i \in I) = \gamma$ is at most $s_{\max}$, i.e. until $p_u - p_l \leq s_{\max}$.

To compute the confidence interval's bounds, we split the interval around the mode $\tilde{p}$ into the part that extends from p_l to $\tilde{p}$ and the part that extends from $\tilde{p}$ to p_u. We require that the probability mass that is covered by these parts of the interval be proportional to $P(p_i \leq \tilde{p}) =: C$ and $P(p_i > \tilde{p}) = 1 - C$ respectively, such that the actual mass covered will be equal to $\gamma_l := \gamma C$ and $\gamma_u := \gamma(1 - C)$ respectively. This ensures that $\tilde{p}$ is proportionately enclosed by the interval.[6] Let $\text{Beta}[\alpha, \beta]$ be the beta distribution's cumulative density function, i.e. $\text{Beta}[\alpha, \beta](p) = P(p_i \leq p)$, and let $\text{Beta}^{-1}[\alpha, \beta]$ be its inverse. With $C = \text{Beta}[\alpha, \beta](\tilde{p})$, the actual bounds p_l, p_u can then be computed as follows:

$$p_l = \text{Beta}^{-1}[\alpha, \beta](C - \gamma_l) \tag{5.27}$$

$$p_u = \text{Beta}^{-1}[\alpha, \beta](C + \gamma_u) \tag{5.28}$$

An example is shown in Figure 5.10. As is evident from the illustration, the scheme introduced above may not provide the most optimistic estimation of confidence, for the interval I that results is not necessarily the smallest that satisfies $P(p_i \in I) = \gamma$.

During the sampling procedure, we can use the technique described above in order to continually estimate the confidence of the results that are obtained, by computing, for

[6] This proportionate placement of the interval is reasonable regardless of the value of the estimate. In particular, it can soundly be applied in cases where the estimate is very close to either 0 or 1, for which the seemingly intuitive choice of a symmetric interval around $\tilde{p}$ is likely to partially extend beyond the bounds of the parameter domain $[0, 1]$.

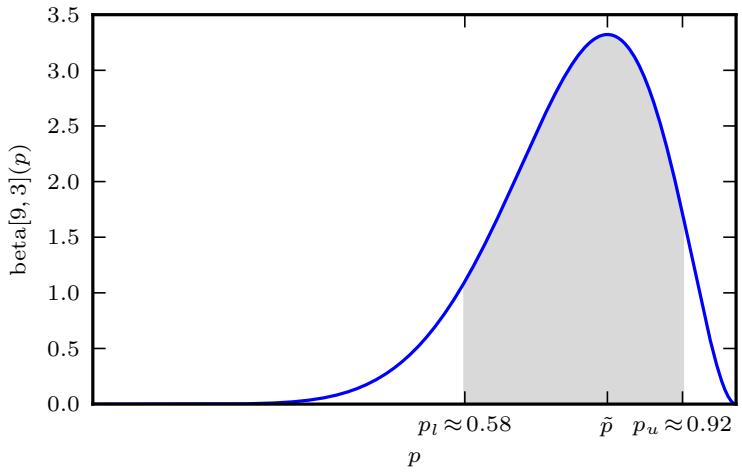


Figure 5.10: *Confidence interval for* beta[9, 3] *with confidence level* $\gamma = 0.85$*: Having seen* 8 *positive and* 2 *negative samples, the probability value to be estimated is within the interval* $[0.58; 0.92]$ *with probability* $\gamma = 0.85$*. The shaded area covers the probability* γ*.*

each query, the size of the respective confidence interval, and running the sampling process until all interval sizes are at most $s_{\max}$.

A simplified version of this scheme was originally developed in collaboration with Paul Maier (Maier, Jain, Waldherr, and Sachenbacher, 2010) for an application in plan assessment for manufacturing – a domain where reliable inference results are of utmost importance. More details on the application are provided in Section 6.3.

The technique described in this section should be viewed as a mere heuristic, for there can, unfortunately, be no firm guarantees. Yet it can certainly provide reasonable pointers for the approximation of posterior marginals in practice. For approximations of unconditional probabilities via importance sampling, a rigorous theoretical analysis for the estimation of approximation quality exists, which may also be applied to posterior marginals if conditional probabilities are computed based on two separate estimates (Koller and Friedman, 2009, ch. 12).

Chapter 6

Applications

All the methods presented thus far are implemented in the ProbCog system for statistical relational learning and reasoning, which is designed to facilitate integration into complex applications for technical systems. This chapter will present applications in robot perception (Section 6.1), cognition-enabled robot control (Section 6.2) and the probabilistic assessment of production plans in a factory (Section 6.3).

6.1 Robot Perception

Technical systems such as robots operating in human environments have strong demands on their perception systems. They must be able to deal with manipulable objects and, in particular, must be able to categorise and identify objects relevant to their tasks. Furthermore, they must maintain beliefs about the locations of objects over time, such that relevant objects can quickly be retrieved.

6.1.1 Categorisation of Kitchen Objects

Household robots are required to manipulate a variety of everyday objects such as cups, glasses, plates, silverware, bottles and boxes. Appropriately recognising the respective types of objects is a precondition for manipulation to take place. This section presents a statistical relational model for the task of object categorisation, which is integrated in a system that was developed in joint work (Marton, Rusu, Jain, Klank, and Beetz, 2009c).

Specifically, the perceptual task of localising and categorising such objects based on views of flat surfaces on which they are positioned is considered. Fortunately, flat surfaces abound in artificial environments such as kitchens. However, both the materials objects are made of and the typical dimensions of objects pose challenges to state-of-the-art sensing technology. Glasses and bottles are typically translucent and knives and forks are shiny, causing 3D sensor readings to be improperly reflected. Plates and silverware are rather flat and may therefore be difficult to separate from the supporting plane in noisy point cloud data. Indeed, one could argue that the perception problem is too difficult for any individual sensor: Pure 3D sensors necessarily ignore many visual features, stereo cameras cannot provide the necessary depth information for untextured objects, and thermal cameras can provide meaningful information only in the presence of temperature variations. The combination of complementary perceptual evidence provided by such sensors may constitute a richer source of information. A *sensor fusion* approach is thus proposed. Specifically, the composite sensor shown in Figure 6.1 is used, which includes a time-of-flight (TOF) camera, a colour stereo camera and a thermal camera.

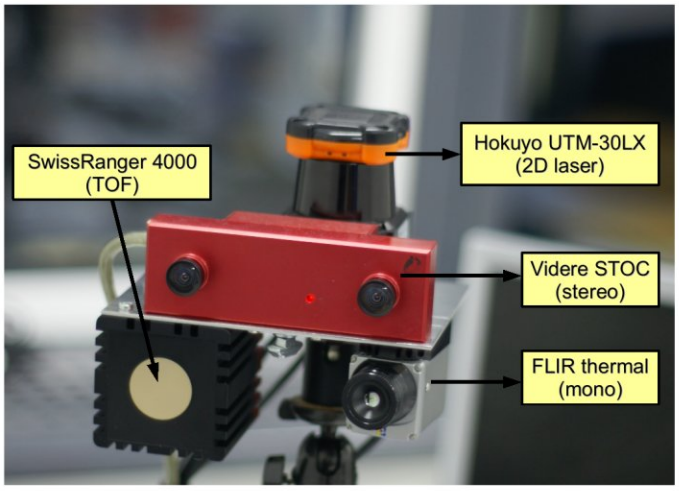


Figure 6.1: *A composite sensor consisting of a time-of-flight camera, a stereo colour camera and a thermal camera*

6.1.1.1 Data Processing Pipeline

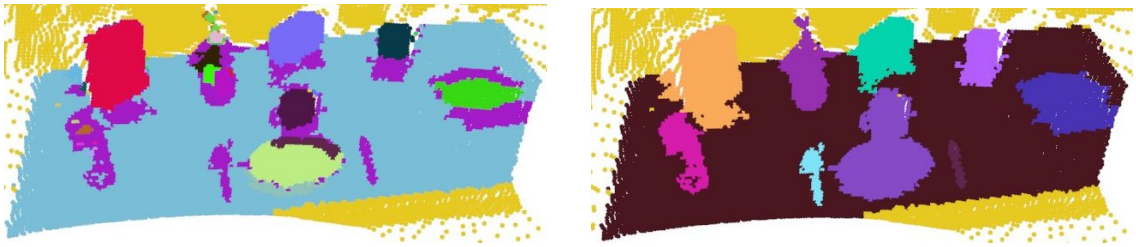
In the statistical relational model that will ultimately perform the categorisation, abstract features generated from the low-level sensor data are considered. In the following, a brief overview of the entire data processing pipeline is given. The pipeline generates the high-level data that the learning and inference processes will be based upon, performs the actual probabilistic categorisation and may apply further processing depending on the result (Marton et al., 2009c).

Registration of Sensor Modalities. In the first step, the data from the individual sensors are combined into a single point cloud, and each point in the cloud is annotated with the data from all sensors.

View Registration. Point clouds obtained through separate scans from different positions are combined into a global point cloud for the scene. This step is supported by visual odometry, which involves an estimation of the pose of the sensor based on the stereo image stream obtained.

Extraction of 3D Regions of Interest and Segmentation. Since objects are assumed to be supported by a planar surface – specifically, a table – the surface needs to be segmented and the objects atop need to be extracted as separate regions. The object segmentation step thus separates points belonging to the table from points belonging to the objects (see Figure 6.2). For the resulting sets of points (clusters), features are computed based on a set of specific inference mechanisms.

Probabilistic Object Categorisation. In the categorisation step, the features computed in the previous step are used to compute a distribution over possible categories for any perceived cluster/object based on a Bayesian logic network model, which is initially trained on a set of labelled training objects, where the features as well as the categories are known. The model is to capture the probabilistic dependencies between object categories and features of objects, some of which are relations between sub-components of an object.



(a) Regions of Interest **(b)** Segmentation: Object Candidate Clusters

Figure 6.2: *Regions of interest and segmentation: (a) regions identified in the depth image: points on the supporting plane (light blue), discarded points (yellow) and various other regions; (b) combination of neighbouring regions into object candidate clusters, random colours (Marton, Rusu, Jain, Klank, and Beetz, 2009c)*

Informed Object Reconstruction. Based on the information from the categorisation step, object surfaces can be reconstructed into more compact models, taking the particular shape implied by the category into consideration.

6.1.1.2 Features

3D Features. Based on an analysis of the points in each cluster obtained from the measurements of the time of flight camera, attributes reflecting mostly geometric properties and relations reflecting parameters of the clusters' components are extracted:

- *average normal angle*: the average angle of the estimated point normals (relative to the table's normal)
- *maximum and average height*: maximum and average distance of the points from the supporting plane
- *base area*: estimated area of the cluster's footprint on the supporting plane
- *volume*: estimated volume of the cluster (based on the maximum heights above the cells in the base area)
- *average point density*: number of points per unit volume
- *thickness, length, width*: proportions of the cluster along its principal directions

- *points above and below the table*: percentage of points with positive/negative distance to the table's plane;
- *shadow and near points*: percentage of points marked with the *shadow* and *near* label[1]
- *zero and low confidence*: percentage of points which have zero and respectively very low (< 1% of the maximum) confidence values assigned by the sensor[2]
- *number of components*: the number of *outside*, *near* and *shadow* components

If there is more than one component within a cluster, the following relations between the components are included:

- *above/below/same level*: comparison of the component centroids' relative heights above the table
- *front/behind*: comparison of the components' distances from the viewpoint

The computation of the above features was performed through the 3D data processing system developed by Zoltan Marton and Radu Bogdan Rusu (Marton et al., 2009c).

2D Colour Camera Features. 2D features are extracted by first determining the set of pixels in the camera images that correspond to the clusters identified in the 3D data, yielding a set of regions of interest in the images (see Figure 6.2a). Because circular shapes are particularly relevant for the categories under consideration, the image regions obtained were searched for occurrences of ellipses, which in turn were characterised by a quantisation describing their size and their orientation relative to the table. The ellipses are assumed to be projections of circles, the specific features considered being:

- *radius-distance ratio*: the (unique) ratio between the radius of the circle the ellipse is assumed to be a projection of and the distance from the camera, indicating the size of the underlying circle
- *orientation (x, y, z)*: the rotation of the circle against the camera

[1]Points with the *near* label belong to a region of an object that is close to the table, and *shadow* belong to a small region with distinctive amplitude value near the table, as they typically occur in shadows.
[2]The SwissRanger 4000 evaluates the accuracy of each measurement with a value, 0 being the lowest confidence, which is typically assigned to points on metallic surfaces.

The computation of these features was implemented by Ulrich Klank (Marton et al., 2009c).

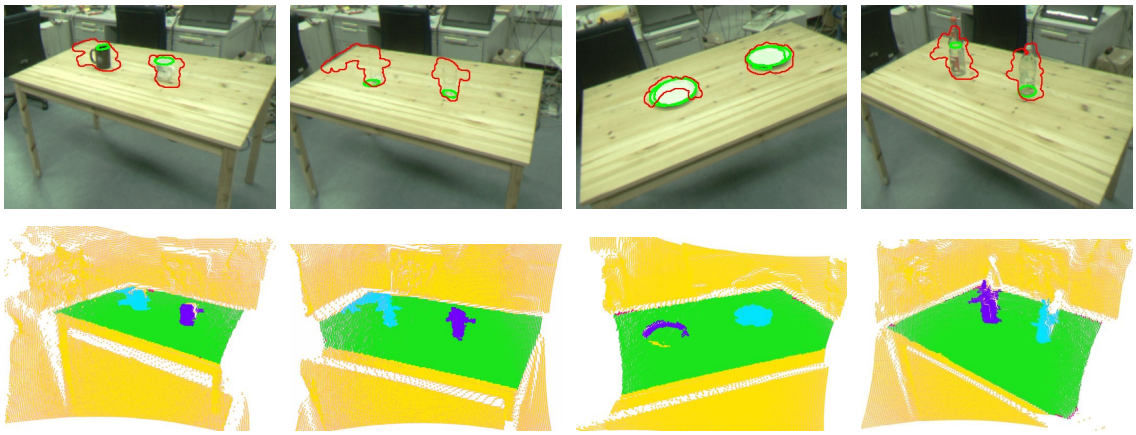


Figure 6.3: *Training examples for the classes mug, glass, plate and bottle. The upper row shows the RGB images with the 3D segmentation (shwon in the bottom row) projected; detected ellipses are shown in green. (Marton, Rusu, Jain, Klank, and Beetz, 2009c)*

Temperature Information. The thermal sensing device allows to collect data about the temperature of the objects, which can be indicative of a substance that is in immediate contact with the objects. Given that no particular action context is assumed in which temperature would be indicative of the categories to which the objects belong, the respective data is not considered in the categorisation task. If a particular action context were given, however, the temperature data could potentially provide relevant additional pointers.

6.1.1.3 Object Categorisation

A statistical relational model is, in general, well-suited to the categorisation task at hand. An object may consist of a variable number of spatially related components, and the relational model can potentially consider an arbitrarily complex configuration of components, sensibly adjusting its prediction with each new component and relationship it is told about. Moreover, an object may or may not be associated with visual features such as ellipses, and the relational models can deal with the presence of an arbitrary number of such features. For the classification task, the model thus

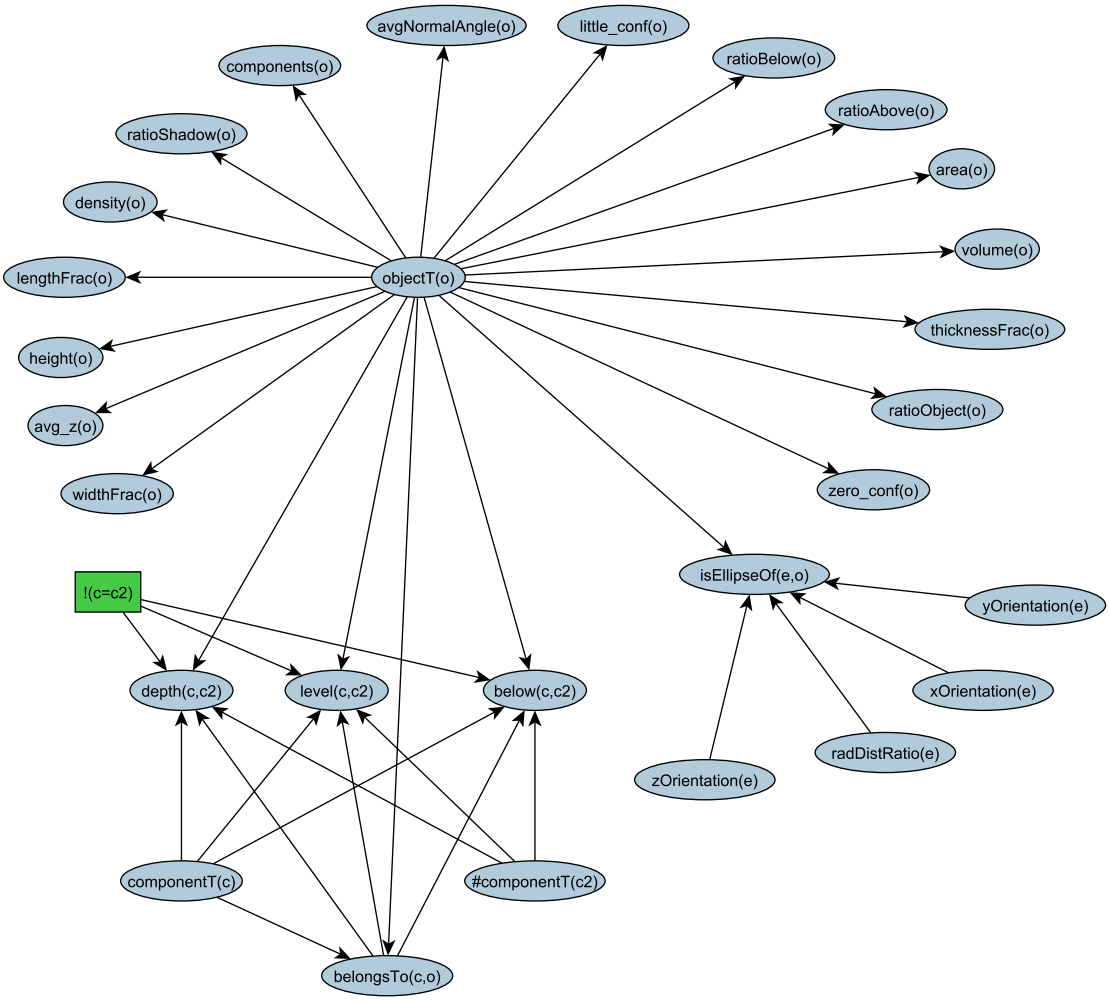


Figure 6.4: *Fragment network of the Bayesian logic network used for object categorisation*

considers probabilistic dependencies between an object's class/category, its immediate attributes, the typed components that belong to it and the spatial relationships between them, as well as ellipses that were detected and parameters of these ellipses.

A Bayesian logic network $\mathscr{B} = \langle \mathscr{D}, \mathscr{F}, \mathscr{L} \rangle$ was created for the task. The network fragment structure representing the set of fragments $\mathscr{F}$ is shown in Figure 6.4. All the features mentioned in the previous section are appropriately handled in the model. At its core, the model follows the architecture of a naive Bayesian classifier, making the (occasionally naive) assumption that individual features are independent given

the categorisation of the object. The continuous 2D and 3D features mentioned above were discretised using expectation maximisation clustering as implemented in Weka (Witten and Frank, 2005), determining a suitable number of clusters automatically. This resulted in domain sizes ranging from two to ten, depending on the variability present in the training set. Since the domain lacks global logical constraints, the set $\mathscr{L}$ of the Bayesian logic network is empty.

In the application, we are interested in conditional probability queries of the form

$$P(\text{objectT}(O) \mid e, B_E) \tag{6.1}$$

where O is one of the segmented objects we wish to classify, B_E is the ground Bayesian network that was instantiated for the relevant set of entities E, and e is the conjunction of observations that were made about O, i.e. a conjunction such as

$$\begin{aligned} &\mathit{height}(O) = \mathit{Height2} \wedge \mathit{belongsTo}(C_1, O) \wedge \\ &\mathit{componentT}(C_1) = \mathit{Shadow} \wedge \mathit{isEllipseOf}(E_1, O) \wedge \ldots \end{aligned}$$

In a classification problem such as this, the only unknown variable is the category; all other variables are given as evidence. With respect to inference, a full enumeration of all hypotheses is therefore feasible.

6.1.1.4 Experiments

The approach outlined above was applied to the problem of object categorisation in tabletop scenes containing everyday kitchen object such as boxes of cereals, different types of glasses and mugs, bottles, plates and two types of silverware. The training database for the statistical relational model was constructed by recording scans of scenes in which one or two objects belonging to the same category were placed on a tabletop. For each scene, a series of scans was made from different perspectives, moving the sensor on a semicircle around the table. For two-object configurations, this resulted in occlusions for some perspectives, such that clusters could not always be properly segmented. In the annotation of the data, such instances were labelled as *ambiguous*. In total, 50 scans were made for each category, such that approximately 80 objects of each category were present in the dataset. The relational training database contained 19224 positive literals on 570 objects that were comprised of 1851

Class	Correct	Total	Ratio
1 - box	23	27	0.85
2 - plate	52	67	0.78
3 - glass	7	31	0.23
4 - mug	39	40	0.98
5 - bottle	4	5	0.80
6 - silver	21	43	0.49
7 - ambiguous	17	87	0.20
overall (1-7)	163	300	0.54
properly segmented clusters (1-6)	146	213	0.69
Comparisons			
BLN without relations between components	143	300	0.48
BLN without 2D data	151	300	0.50
BLN without 3D data	62	300	0.21
C4.5 classification tree	148	300	0.49

Table 6.1: *Summary of categorisation results for all experiments performed*

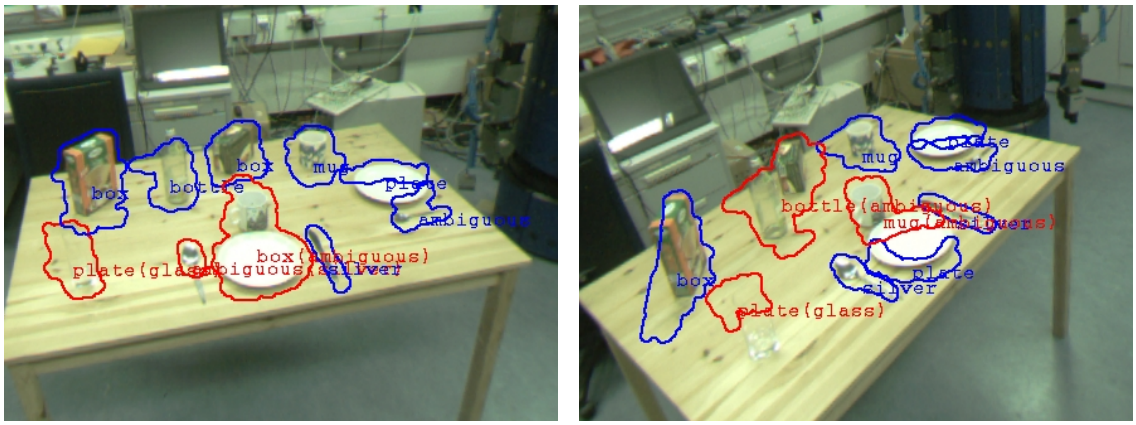
spatially related components and featured 1308 ellipses. Maximum likelihood training (with small pseudo-counts) of the BLN model was completed in approximately 10 minutes.

For testing, the model was applied to a more complex, partially cluttered scene, which contained both previously unseen objects and objects from the training set. A full run of the processing pipeline, including feature estimation and probabilistic categorisation, took but a few seconds. The categorisation results are summarised in Table 6.1. They indicate an overall classification rate of approximately 54%. However, the accuracy on properly segmented objects is almost 70%.

As a baseline, a C4.5 classification tree was used, which was trained only on object attributes, disregarding relations. While a simplified Bayesian logic network working on the same features does not surpass the performance of the classification tree, the full statistical relational model, which takes into consideration all the information that is provided by the features, performs best overall. From the comparisons with BLNs that omit parts of the attribute space, once can draw the conclusion that the fusion of sensory information indeed helps to attain a superior result. The removal of any sensor modality resulted in inferior categorisation performance.

ground truth \ categorisation	box	plate	glass	mug	bottle	silver	ambiguous
box	23	0	0	2	1	0	1
plate	0	52	0	0	0	5	10
glass	0	14	7	0	0	2	8
mug	0	0	0	39	1	0	0
bottle	1	0	0	0	4	0	0
silver	0	0	7	2	0	21	13
ambiguous	20	8	1	5	28	8	17

Table 6.2: *Confusion matrix for the results in Table 6.1*



(a) **(b)**

Figure 6.5: *Categorisation results for two views of the same scene. Correctly classified clusters are marked blue, incorrect classifications are marked red, ground truth labels are given in parentheses. (Marton, Rusu, Jain, Klank, and Beetz, 2009c)*

Given the nature of the training data, the model could not properly be trained for the ambiguous class, yet the category was still recognised correctly approximately 20% of the time. Notably, the system often categorised ambiguous clusters as belonging to one of the object classes that did indeed appear within the respective cluster. Such a case is, for instance, shown in Figure 6.5b, where the cluster consisting of a mug and part of a plate is categorised as a plate and the cluster consisting of part of a bottle and a box is categorised as a bottle.

In conclusion, the results presented above indicate that the fusion of sensory information is indeed helpful. The statistical relational learning approach is suitable for this type of problem and particularly suggests itself in the presence of variable relations

between entities. In domains where the objects to be classified have meaningful arrangements (i.e. their positions relative to each other are not arbitrary), one could even consider spatial relations between them and consequently perform *collective classification*, i.e. one could collectively determine the most likely classification of an entire group of objects, taking interactions between neighbouring objects into consideration. Collective classification was, for instance, done based on the associative Markov network framework by Triebel et al. (2006) to identify walls, windows and gutters in laser data. In the present domain, the inclusion of inter-object relations would in principle be feasible. However, the nature of the training data was such that no meaningful dependencies could have been extracted, since no particular action context was assumed.

Related Work. Segmentation and object categorisation are problems that are frequently addressed in perception, typically using data from a single sensor. Many approaches consider large sets of training cases from multiple views and use abstract features to categorise objects (Quack et al., 2007; Yan et al., 2007; Savarese and Li, 2007). Posner et al. (2008) propose a two-stage process, where, at the local level, classification is based on appearance descriptors, and at the global scene level, Markov random fields are applied to model relationships. An approach that considers segmentation and classification at the same time – by defining a Markov random field directly over 3D scan points – has, for instance, been proposed by Anguelov et al. (2005). The approach by Triebel et al. (2006) even draws upon statistical relational learning methods, yet it uses laser data exclusively.

6.1.2 Dynamic World State Logging

Beyond the categorisation of objects, an autonomous robot acting in human environments is faced with many additional challenges pertaining to perception. In particular, the dynamic nature of human environments and the uncertainty that stems from objects having similar or even identical appearances – in addition to the inherent noisiness of sensory inputs – calls for an explicit probabilistic treatment. Moreover, the current beliefs a robot may have about the environment should ideally take into consideration not only its present sensory inputs but also its past beliefs. In the following, I present a prototype of a system that takes these factors into consideration

(Blodow, Jain, Marton, and Beetz, 2010). The problem that is addressed is referred to as *spatio-temporal object identity resolution*, for we are interested in obtaining beliefs about the identities of objects that are currently observed or were observed in the past (observation-object associations) and in pinpointing the most likely locations of objects in the environment.

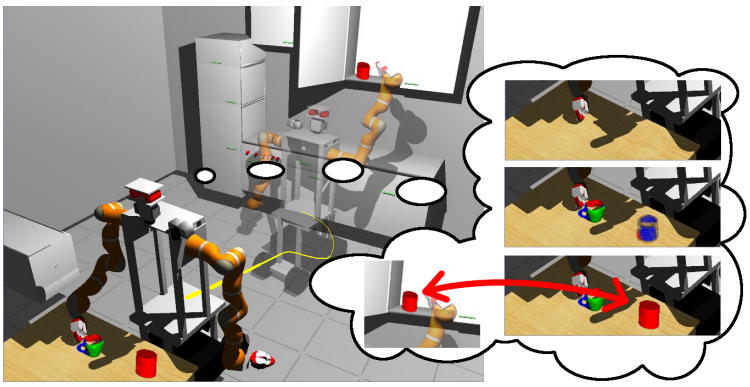


Image courtesy of Nico Blodow

Figure 6.6: *Keeping track of objects over time: A robotic household assistant must know about its environment and the objects within – both in the temporal and spatial domain.*

Maintaining beliefs about objects over time is potentially beneficial for a large number of robotic tasks, for such beliefs are clearly relevant to both basic actions such as pick-and-place in order to grasp (where we need to know the location and 3D pose of an object in order to navigate and determine a reaching motion) and high-level reasoning tasks (e.g. reasoning about everyday activities from observed locations of objects as addressed in Section 6.2.1).

For robots acting in human environments, a key requirement is that non-uniquely identifiable objects must be considered. There is an abundance of highly standardised objects in kitchen environments: We typically encounter sets of identical cups, glasses and plates. However, depending on the context, objects cannot be regarded as interchangeable. For instance, a certain cup might, over the course of an activity, be associated with a particular person and therefore cannot be equated with other cups.

For mobile robots that are not confined to a particular workspace and operate in extensive environments, another important aspect is *partial observability*. If the robot

relies on sensors that are attached to its body, it will, at any given point in time, be able to observe but a tiny fraction of the environment. Moreover, certain objects may be occluded by other objects or may be hidden behind the doors of containers such as cupboards. The robot should thus be able to recall past observations and appropriately integrate them into current beliefs.

In the following, a system is proposed that takes into account the above points whilst making the reasonable assumption of steadiness: Unless there is evidence to the contrary, it assumes that the state of (temporarily) unobserved entities remains unchanged. A simple example is shown in Figure 6.7.

The system processes streams of geometrical data that are appropriately interpreted by a 3D processing pipeline in order to compute beliefs about observation-object associations. These beliefs can in turn provide important inputs to higher-level reasoning problems such as planning and action recognition. The system thus mediates between low-level sensor data processing and high-level reasoning layers of an autonomous robot system. In the following, I first give an overview of the 3D data processing aspects (developed by Nico Blodow) and then provide details on the statistical relational model and the associated mode of application that allows us to handle the problem of spatio-temporal object identity resolution.

6.1.2.1 3D Processing Pipeline

The input to the system is a stream of 3D data from a tilting laser scanner. The passive perception pipeline (which is similar but not identical to the one described in Section 6.1.1) continuously collects sensor data and computes information about clusters that can be considered as separate objects in the data.

In the first step, point cloud processing, the data from the laser scanner is assembled into a 3D scan, removing spurious measurements and estimating surface normals for each point. In the second step, geometric and functional reasoning, the principal planar regions (walls, ceilings, floors and large surfaces on pieces of furniture such as tabletops) are detected. The final step reconstructs object models and computes descriptors that can later be used for identity resolution. Several refinement algorithms are applied, including box reconstruction (Marton et al., 2009a), rotational

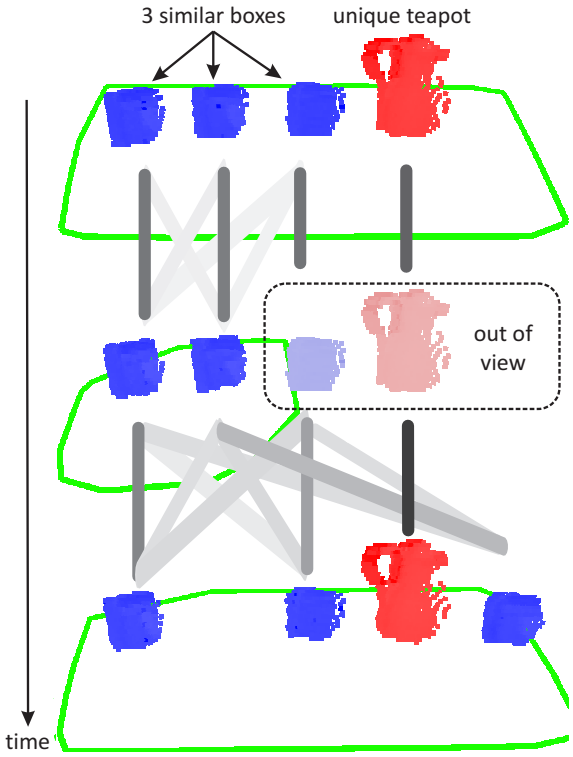


Figure 6.7: *Illustration of the problem of spatio-temporal object identity resolution. An environment containing three boxes and one uniquely identifiable teapot is observed for three time steps. The coloured objects represent scans taken by the 3D perception system, and the connecting grey lines indicate the associations computed by the system, the strength of associations being reflected in the blackness of connections. (Blodow, Jain, Marton, and Beetz, 2010)*

reconstruction (Blodow et al., 2009), primitive shape reconstruction for the fitting of geometric shapes such as cylinders, cones and spheres (cf. Schnabel et al., 2007), and, in the absence of a better reconstruction method, triangulation, which connects points to form a surface model without specific assumptions about the nature of the underlying geometry (Marton et al., 2009b).

Measurements that have been processed by the above pipeline are stored in a *dynamic object store* on a per-surface basis, i.e. the store maintains, a for each tabletop or cupboard surface, a list of *clusters* that have been identified at each time step. From the information that was gathered in the refinement stage, we can compute distance and similarity measures that indicate the degree to which a cluster is similar to the prototypical appearance of a particular object or to another cluster. Further details are provided by Blodow, Jain, Marton, and Beetz (2010).

6.1.2.2 Object Identity Resolution

The key reasoning task of the system is the task of spatio-temporal object identity resolution. Whenever new observations arrive in the dynamic object store, we are interested in determining which of the current observations correspond to which past observations and which real-world entities can be associated with them. Only by solving this problem can a robot establish the notion of an *object*, i.e. a certain entity that is known to exist in the real world.

Consider Figure 6.8, which schematically depicts a number of measurements over time ($t_{k-2} < t_{k-1} < t_k$) for three tables. There are instances of two different object categories (represented as squares and circles). These may get replaced, moved, taken away or added by an agent not perceived by the system. The most interesting questions are: At any point in time, which object is each of the observations associated with, which objects have been moved where, and which objects have remained at the same position? At time t_{k-1}, the robot changed its orientation and is therefore unable to observe tables 2 and 3. In order to be able to answer queries about objects that are temporarily out of view (faded area in Figure 6.8), we need to maintain a sensible belief state which assumes that objects have not been moved unless we perceive evidence to the contrary (e.g. an object has appeared elsewhere).

A statistical relational model is applied in order to tackle this problem, enabling us to consider all the observations and facts that are relevant simultaneously, without making strong independence assumptions. Consequently, we can obtain posterior be-

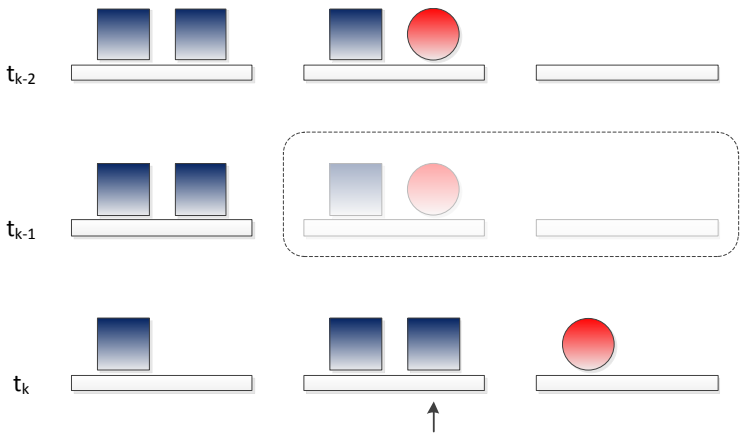


Figure 6.8: *Exemplary object identity resolution problem: Where was this entity before?*

liefs on the associations between objects and observations that are guaranteed to be *globally consistent*.

A Markov Logic Network for Spatio-Temporal Object Identity Resolution. The concrete model that is used to solve the problem of object identity resolution is given by MLN 6.1. As entity types, the model considers objects that can potentially be identified (*objects*), observations pertaining to these objects (*clusters*) as well as abstract entities representing time and basic object shapes. The time domain differentiates only between the most recent point in time (*Now*) and the point in time that preceded it (*Past*).

As soon as new observations come in, the model is applied to update beliefs, making the first-order Markov assumption. In the present system, a new scan arrives approximately every 10 seconds; time is discretised accordingly.

Markov logic networks are particularly well-suited for the representation of such a model, as they allow us to specify the various hard constraints that any valid entity-observation association must satisfy in first-order logic. At the same time, probabilistic rules (which either increase or decrease the probability of associations) can be expressed as soft constraints. Any beliefs computed by the model are guaranteed to be probabilistically sound and globally consistent with respect to the constraints that were specified.

Domain Declarations

domTime = {Now, Past}
domShape = {Cylinder, Box}

Predicate Declarations

observed(cluster, domTime)
same(cluster, cluster)
similarExt(cluster, object)
similarPos(cluster, cluster)
is(cluster, object!)
shape(object, domShape!)
cshape(cluster, domShape!)
areaObserved(cluster)
persists(cluster)

Weighted Formulas

i	w_i	F_i
1	(hard)	observed(c,t1) ∧ observed(c,t2) ⇒ (t1=t2).
2	(hard)	is(c,o) ∧ is(c2,o) ∧ observed(c,t) ∧ observed(c2,t) ⇒ (c=c2).
3	(hard)	is(c,o) ∧ is(c2,o) ∧ observed(c,Now) ∧ observed(c2,Past) ∧ persists(c2) ⇒ (c=c2).
4	(hard)	same(c,c2) ⇔ ∃ o. (is(c,o) ∧ is(c2,o)).
5	(hard)	same(a,a).
6	(hard)	same(a,b) ⇒ same(b,a).
7	(hard)	same(x,a) ∧ same(y,a) ∧ observed(x,t) ∧ observed(y,t) ⇒ (x=y).
8	(hard)	areaObserved(c) ⇒ ¬persists(c).
9	(hard)	is(c,o) ∧ is(pc,o) ∧ areaObserved(c) ∧ ¬areaObserved(pc) ⇒ ¬persists(pc).
10	log(0.90/0.10)	persists(c)
11	log(0.95/0.05)	cshape(c,cl) ∧ shape(o,cl) ⇒ is(c,o)
12	log(0.83/0.17)	similarExt(c,o) ⇒ is(c,o)
13	log(0.83/0.17)	similarPos(c,c2) ∧ observed(c,Past) ∧ observed(c2,Now) ⇒ same(c,c2)

MLN 6.1: *Model for spatio-temporal object identity resolution*

Regarding our application, the key predicate we are interested in is the association between observations/clusters and objects (*is(cluster, object)*). We require that, in any possible world, every object can be associated with at most one cluster for any point in time (formula 2). Furthermore, similarity along any dimension should increase the degree to which we believe there to be an association (e.g. similarity with respect to spatial extents, *similarExt(cluster, object)*). Each similarity feature influences our beliefs about cluster-object associations (e.g. formula 12; other similarity features can be handled equivalently). Because the basic shape that is detected in a cluster is a particularly discriminative feature, shape is considered explicitly (formula 11).

Apart from cluster-object associations, we need to be able to determine which clusters are equivalent (*same(cluster, cluster)*). By definition, two clusters are equivalent (in any possible world) if and only if there is some object with which both are associated (formula 4). Furthermore, since clusters have positions in space, a similarity feature over pairs of clusters is added (*similarPos(cluster, cluster)*, formula 13). As basic axioms, we require cluster equivalence to be reflexive and symmetric (formulas 5 and 6), and we require that, for any point in time, there can be at most one cluster that is equivalent to any other cluster (formula 7).

Moreover, we need to cope with partial observability: The model needs to be able to handle the fact that areas in which we previously made observations can currently be unobserved (*areaObserved(cluster)*) and maintain an appropriate belief about these areas (and the cluster-object associations therein). By default, we assume that the observations in unobserved areas are likely to carry over into the present (*persists(cluster)*, formulas 8 and 10), unless there is evidence to the contrary (which is the case if another cluster whose area *is* currently observed refers to the same object, formula 9). With respect to the consistency of cluster-object associations, persisting clusters are treated as if they were currently observed (formula 3).

Application. The MLN is applied by instantiating it with the universe of objects that is considered (providing *shape* for each object) and the set of relevant observations that were made, i.e. the set of clusters that are currently being observed and the most recent past clusters in all areas of the environment (including, in particular, the

most recent clusters in areas that are currently unobserved, for which we need to determine whether or not we should believe they persist).

As evidence, we also need to provide the degree to which an observation/cluster matches the prototypical appearance of any of the objects under consideration, i.e. the degree to which similarity features hold. For each similarity feature, we calculate a distance measure (e.g. the Mahalanobis distance in three dimensions for spatial extents) and map the (normalised) distances $d \in \mathbb{R}_0^+$ we obtain to similarity beliefs in $[0,1]$ using $d \mapsto 2 - 2/(1+\exp(-d))$ – a function that decays approximately linearly at first and asymptotically approaches 0. In this way, we obtain a similarity belief for each cluster-object pair and each similarity feature. Each belief constitutes *soft evidence* (pertaining to ground atoms such as $\mathit{similarExt}(\mathit{Cluster}_i, \mathit{Object}_j)$).

The model is applied whenever new scans are obtained that need to be interpreted. Inference results from a previous iteration thus become soft evidence for the current iteration, i.e. results from the *Now* time step become evidence for the *Past* time step in the subsequent iteration. Specifically, soft evidence on the beliefs regarding cluster-object associations (i.e. *is* ground atoms) is added. Moreover, in order to account for areas that are currently unobserved, evidence on the predicate *observed* is artificially injected: Whenever the model previously inferred that an unobserved cluster persists (with probability p), the cluster is assumed to be observed in the next application of the model (with degree of belief p).

To perform inference in the light of both soft and hard evidence, the algorithm MC-SAT-PC, which was introduced in Section 4.2.2, is used. Note that one inference run (application of the model) can essentially be viewed as a filtering step (combined prediction and update).

6.1.2.3 Experiments

Experiments were conducted to show the general feasibility of the system prototype. Two instances of the example scenario shown in Figure 6.8 are considered in order to illustrate that the approach is generally sound. As ground truth, assume that in the first time step, none of the objects moved but the robot was repositioned such that

it could only perceive table 1; and in the second time step, the second box (square) from table 1 was moved to table 2 and the plate (circle) was moved to table 3.

Figure 6.9a depicts the results for a case where the initial association of clusters to objects is known (e.g. the first box is the object with identifier *Box1*, etc.) and the universe of objects comprises three boxes and one plate (all of which were observed in the first step). The image indicates that both the association of clusters with objects and the equivalence of pairs of clusters over time is consistent with common sense (and the modes of the respective distributions indeed correspond to ground truth). The positional similarity beliefs were high (0.95) for clusters that are drawn at the same horizontal coordinate in Figure 6.9a, and appropriately low otherwise (0.05). The extents for all objects having the same shape were considered as equally similar. Note that the model purposely uses a fairly low weight in formulas indicating similarity (e.g. formula 13) in order to amplify the decay of beliefs over time, facilitating visualisation of low probabilities. Beliefs thus drop substantially in a single time step.

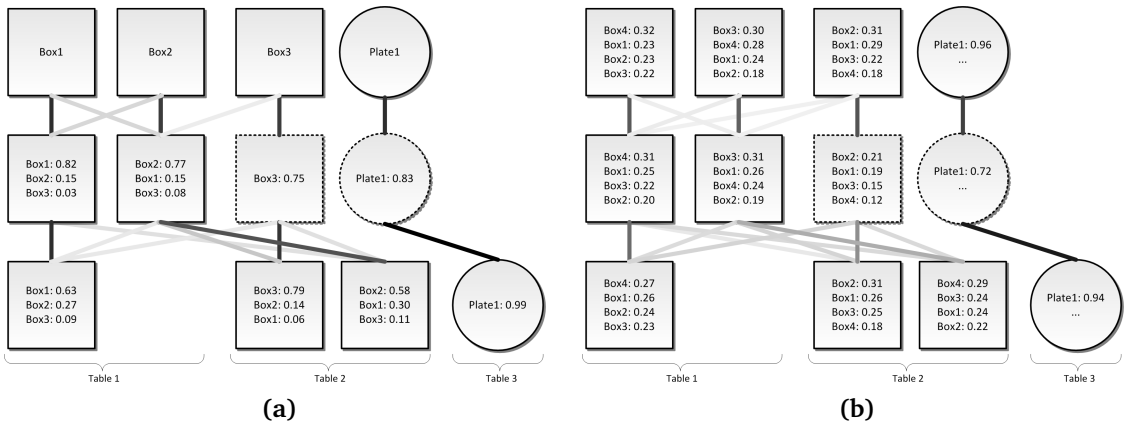


Figure 6.9: *Illustration of inference results. Every square/circle represents an observation that was made (cluster). Dotted squares/circles are in fact not observed but are shown in order to illustrate beliefs on unobserved areas (observation persistence). Cluster-object associations are listed in cluster labels, cluster equivalence is reflected in the blackness of lines connecting clusters (black = 1.0, white = 0.0). In case (a), the universe comprises three boxes and one plate, and the cluster-object associations for time step 1 are known. In case (b), the universe comprises four boxes and one plate, and we have no evidence on initial associations. (Blodow, Jain, Marton, and Beetz, 2010)*

Higher weights should be chosen for real-world applications. (Given enough data, it would be most sensible to obtain them using machine learning techniques.)

In Figure 6.9b, no prior knowledge on cluster associations was assumed, and an additional box was added to the universe of objects (such that all boxes are never observed simultaneously). We observe that, even though beliefs on associations are essentially uniform (discrepancies are due only to approximate inference), cluster equivalence is still inferred correctly.

6.1.2.4 Discussion

The prototype presented above shows that the methods of statistical relational reasoning can be applied to the hard problem of spatio-temporal object identity resolution as defined above, taking into consideration the inherent uncertainty in sensor readings, partial observability of the environment and the presence of both unique and non-unique objects. The approach builds upon a sophisticated 3D data processing system and, by establishing the important "semantic" link between observations and entities in the real world, enables that system to lay the foundation for many high-level reasoning problems.

Naturally, the approach that was presented will not scale arbitrarily in practice. The computation of belief updates using approximate soft evidential update is expensive and, for the example scenario described above, required about one minute for sufficiently accurate results to be obtained using a Java implementation of MLNs and MC-SAT-PC running on a single core of an Intel Core i7. It is, however, reasonable to assume that a parallelised, more optimised implementation in a language such as C++ could sufficiently speed up belief updates for the system to be applied under real-world conditions.

Related Work. If one considers belief updates at higher frequencies, the problem of object identity resolution becomes partly related to (multi-hypothesis) tracking (Cox and Hingorani, 1996; Schmitt et al., 2002) and statistical data association (Schulz et al., 2001). Especially with sensing devices such as the Kinect enabling high-frequency streams of 3D data, an application of tracking techniques could be sensible for objects that are continuously observed. In fact, a combination of tracking

and spatio-temporal object identity resolution is conceivable, where tracking is used as long as the robot's view does not change significantly, and identity resolution is used to update global beliefs whenever the robot perceives a different part of the scene as it moves around the environment. The problem of spatio-temporal object identity resolution is furthermore related to information integration problems, where matching entries from different sources have to be identified. While early work has often phrased this problem as a classification problem, where a pair of data records would be independently classified as either matching or non-matching (Fellegi and Sunter, 1969), recent approaches have phrased the problem as link prediction in statistical relational models (Singla and Domingos, 2006).

Markov logic networks – in combination with appropriate belief update and perhaps learning mechanisms – can serve as a highly general framework for the specification of data association and information integration problems. By changing the structure of the model, MLNs can quickly be adapted to a new set of assumptions without requiring implementation changes. Of course, algorithms that specifically exploit a particular set of assumptions and are specifically adapted to well-defined, restricted problem classes can at times be more efficient than the general-purpose inference mechanisms of statistical relational models.

6.2 Cognition-Enabled Control of Autonomous Robots

Let us now reprise the notion of enabling cognition in technical systems through an approach that integrates manifold cognitive mechanisms and resources, as put forth in the introduction. A key challenge in achieving cognition-enabled control for autonomous robots is the sound integration of robot programming, learning and reasoning. Therefore, a unifying methodology that provides access to mechanisms and resources that provide the respective functionality is necessary. A specific architecture that seeks to implement such a methodology is CRAM, a *cognitive robot abstract machine* (Beetz et al., 2010b).

The concrete realisation of cognition-enabled robot control through CRAM hinges upon three core principles (Beetz et al., 2012):

1. the specification of robot behaviour through a concurrent, reactive, high-level programming language
2. the provision of reasoning mechanisms for deciding on the course of actions and their parametrisations
3. the use of semantically interpretable action plans that enable control programs to be modularly designed

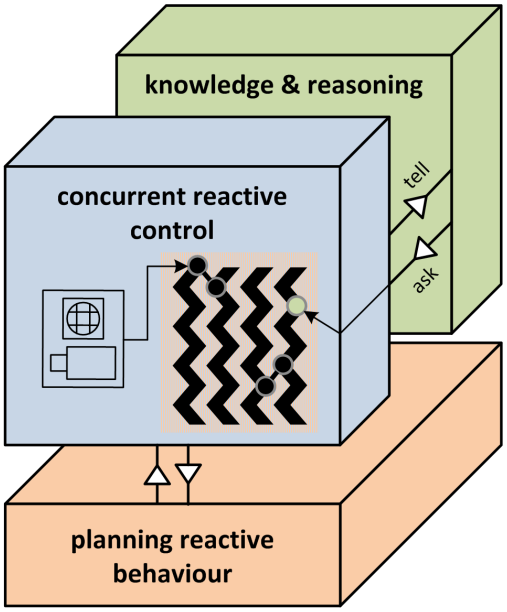


Figure 6.10: *An architecture realising cognition-enabled control*

Figure 6.10 illustrates the interplay of the three corresponding components of a technical systems that adopts these principles in its design. The control component that closes the cognitive perception-action loop by choosing actions based on perceived data is tightly coupled with knowledge processing and reasoning systems, which partly realise the interpretation of the perceived data, implicitly representing part

of the robot's belief state, and provide mechanisms that allow the controller to infer information upon which control decisions can be based.

The use of a concurrent, reactive, high-level language suggests itself due to the requirements of the application domain: The complexity of acting in the real world demands that the intrinsic parallelism and the structure of actions and behaviours be adequately reflected in the control structures of the language that is used to describe them. By using a high-level reactive language such as CRAM-PL (Beetz et al., 2010b), which explicitly supports parallelism, one can ensure that complex real-world behaviours can be represented in the control structures of the program whilst keeping the program transparent and modular through a sufficiently high level of abstraction.

Systems that employ abstract planning at high levels of abstraction (cf. Fikes and Nilsson, 1971; McDermott, 2000) and, at the same time, intend to act in the real world, where the effects of actions often cannot be ascertained and the validity of many action preconditions is neither observable nor guaranteed to be maintained due to the dynamics of the environment, must somehow bridge the gap between high-level abstractions and low-level action execution. Three-tier (3T) architectures are frequently proposed as solutions to this problem. These architectures introduce a sequencing layer that links high-level action planning with a low-level layer that provides reactive skills (Firby, 1987; Bonasso et al., 1997; Gat, 1998). For the concrete implementation of the sequencing layer, which plays a crucial role as mediator between the abstract behaviour specifications in the top layer and low-level behaviour, so-called *reactive plan languages* were proposed, e.g. RAP (Reactive Action Packages, Firby, 1987), ESL (Execute Support Language, Gat, 1996) and PRS (Procedural Reasoning System, Ingrand et al., 1996). While these languages were specifically designed for the flexible and robust execution of sequences of abstract actions, they largely disregard the intrinsic parallelism and the structure of actions required by many real-world applications.

While CRAM-PL essentially follows the tradition of these reactive plan languages, it differs in its design by explicitly taking into consideration of the structure of real-world problems. Consequently, plans represented as CRAM-PL programs are not restricted to sequences of high-level actions as they would be computed though classical planning but can be explicitly designed with the structure of problems in mind. In particular, CRAM-PL uses control structures for parallel execution and hierarch-

ically decomposes goals into manageable sub-plans. At each level in the hierarchy, plans can be carefully designed to monitor their own execution and to recover from intermittent failures. Most importantly, CRAM-PL programs satisfy the third principle described above, for they represent semantically interpretable action plans. Semantic interpretability facilitates the automated generation of plans that achieve particular goals or satisfy given conditions, the transformation of plans as well as introspection. In turn, introspection allows the logging of semantically interpretable data pertaining to the execution of control programs, enabling the robot's logged experience data to be leveraged as a resource for learning tasks.

The CRAM framework is highly extensible and serves to integrate a variety of heterogeneous modules that plans written in CRAM-PL may draw upon during execution. In particular, it transparently integrates key robotic modules pertaining to perception, navigation and manipulation, and can furthermore support access to various reasoning and learning components. The CRAM plan language serves as a unifying language with which all of these modules can be conveniently accessed. CRAM thus becomes an interface layer for robotic applications, as illustrated in Figure 6.11. Robotic applications implemented on top of CRAM can make use of learning and

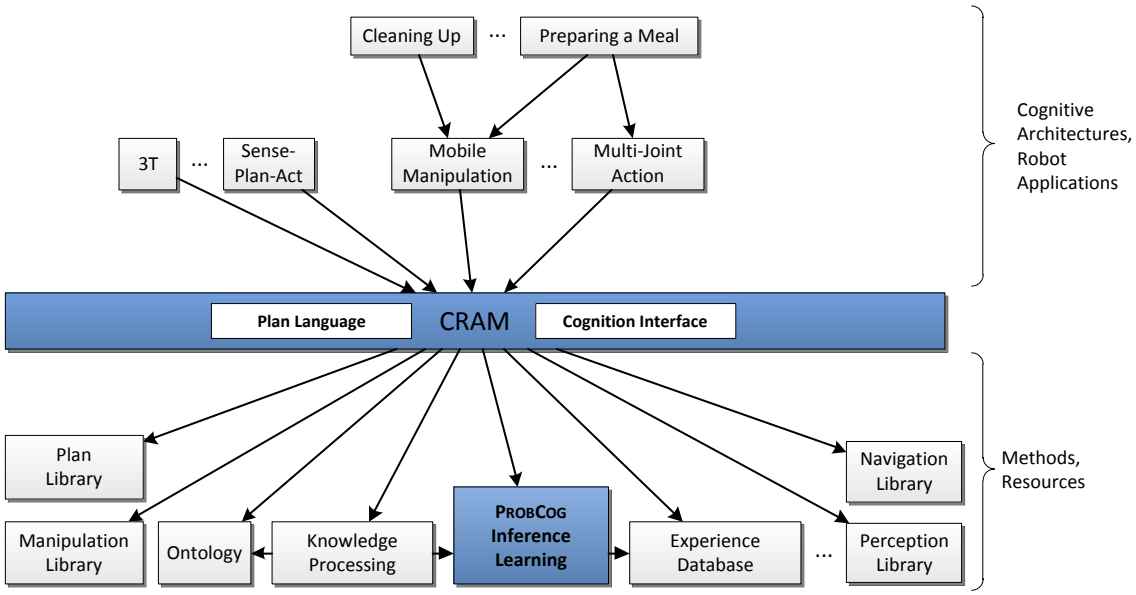


Figure 6.11: *The cognitive robot abstract machine (*CRAM*) serves as an interface layer for cognitive robotics, providing a unified interface to methods, algorithms and data for arbitrary robotic applications.*

reasoning methods and draw upon the corresponding sources of knowledge without concerning themselves with method-specific interfaces or the implementations of the underlying algorithms.

In the following, I will the describe the role of the ProbCog system for statistical relational learning and reasoning in this context. The ProbCog system equips CRAM-PL programs with the much-needed ability to deal with uncertainty at high levels of abstraction.

There are many situations in which an autonomous robot can potentially benefit from probabilistic reasoning capabilities. For example, many of the tasks of a service robot are under-specified in the sense that not all the information that is potentially relevant to the successful handling of tasks is provided when commands are given. In such situations, a robot can fill in the missing parts of task specifications by inferring the most likely specification and adjusting its control program appropriately. The task of setting the table is a good example of such a case: How the table is to be set depends on a number of factors, such as the type of the meal and, in particular, the participants and their individual requirements and habits. Setting the table for an unknown group of four people is an entirely different task than is setting the table for four known members of the family, about whom we know quite a bit from experience. In the former case, we should probably be conservative and provide all the things that could potentially be needed, while in the latter case, we should probably take any knowledge we may have about the seating order, consumption habits of the individuals and perhaps even preferred utensils into account. If we do not know who the people participating in the meal are going to be, it could make sense to infer who is most likely to take part given the type of the meal (e.g. breakfast), the day of the week and the current time. Moreover, social structures that may affect the participation of certain individuals can be accurately captured in a relational model: If we know, for instance, that Steve will take part and that Pete is Steve's friend, the model could indicate an increased probability for Pete's participation. Parametrising a plan for table setting with the knowledge represented by and retrieved from a statistical relational model can thus be highly desirable, for it allows to adapt default plans to the concrete situation at hand, achieving highly individualised behaviour that is inspired by past experience.

Another possible application is the process of context-specific decision-making and plan selection. A robot that is observing humans as they take actions or is given initial commands to carry out certain actions can conclude from context information what the most likely intentions of the people it interacts with are, allowing it to anticipate necessary future actions, which can subsequently be carried out without the need for further instructions – provided that the robot has a statistical relational model that captures precisely the connection between sequences of actions and the respective context. Furthermore, the plan to select for a particular task may depend on certain facts that are not known to the robot. Inferring the degree to which the corresponding facts should be believed to hold can thus help to select appropriate plans. For instance, a robot that is supposed to deliver an object to somebody whose location or even whose presence in the house is not known can use a model of people's habits to infer the most likely location given the information that is known (e.g. the current time, day of the week or known facts about others) in order to select a plan that is appropriate for delivery.

Similarly, we can use probabilistic models to select heuristics, e.g. search heuristics. A robot that initially takes on its role as a household assistant can use models about common layouts of kitchen environments to direct its search for utensils whose locations are yet unknown by inferring the most likely locations. For instance, it might be likely for a wooden spoon to be in a drawer next to the stove if there is such a drawer. Even a robot that is already familiar with the environment may not know at all times where the objects it may require are going to be located. Given a model of how family members generally move objects around the house as events take place, the robot can, however, infer the most likely positions in order to determine a suitable search order.

6.2.1 Models of Human Everyday Activities and Environments

Comprehensive knowledge about everyday activities is essential for robotic household assistants – not only for the purpose of recognising what is going on in their surroundings but also for the challenging task of deriving suitable actions to be taken. In the following, we thus consider models of everyday activities that could serve as cog-

nitive resources of an AI system that is to learn about, interpret and even participate in these activities.

For high-level knowledge about everyday activities to which a variable number of entities could be relevant, the use of statistical relational models immediately suggests itself. The representation of (generalised) full-joint probability distributions over relevant propositions is particularly attractive, for it offers the greatest degree of flexibility: We attain the ability of inferring the probabilities of any propositions we may be interested in, given any set of evidences; arbitrary queries are computable, be they causal (from causes to effects), diagnostic (from effects to causes) or mixed in nature.

Suppose we had a probabilistic model of common everyday activities such as preparing meals, setting a table or cleaning up, which was trained on a sufficiently large body of training data encompassing the activities within a kitchen over the course of, say, half a year. Such a model would enable us to answer instructive queries, including:

- What is the effect of carrying out an action or sequence of actions on the objects involved: How is the state of objects altered as actions take place?
- Which objects are likely to be involved in an activity (depending on the circumstances)? How is the set of relevant objects affected by individual preferences? Which objects are usually used in combination?
- Given the states and locations of a number of objects (at various points in time), what is the activity that is taking place or has recently taken place? Which subactivities are involved?

In the following, probabilistic models capable of answering such queries are presented. Of course, learning models from data is not an easy task, especially since sufficiently large amounts of data are largely unavailable. Ways of acquiring data as well as existing data sources will be described in the following.

6.2.1.1 Data Acquisition

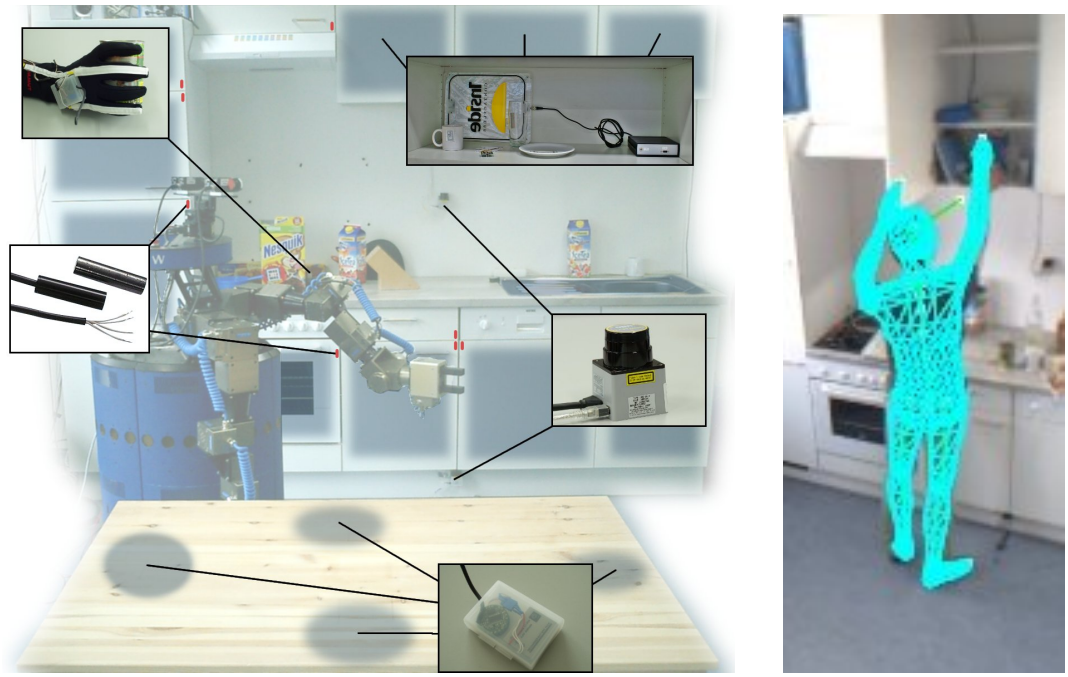
While system designers or knowledge engineers can typically indicate dependencies that might exist in a domain, they generally cannot quantitatively define these dependencies. Therefore, the probabilistic parameters of models should ideally be learnt from data. In the case of an autonomous robot acting in everyday environments, the observations that have been made by the robot itself or have been gathered by a sensor-enabled environment and made accessible to the robot can be used as training data for learning.

In the experimental scenario of the Intelligent Kitchen at TUM, everyday activities are observed within a sensor-equipped environment. As indicated in Figure 6.12, various sensors are deployed throughout the environment that allow everyday activities to be monitored. In particular, contact sensors allow to detect whether cupboards or drawers have been opened, and RFID readers allow to identify the objects being manipulated (provided that RFIDs tags have previously been attached to them).

To obtain a training database, we need to collect data in a (logical) form that matches that of the models. A very simple approach can involve the recording of data based exclusively on RFID readings. Consider, for example, a scenario in which we want to record training data for a model that is concerned with pick and place actions. The following atoms representing a sequence of actions and situations can be added to a training database based on observations of RFID sensors alone:

Data	Sensor	Time	Generated atoms
ID_Cup_3	RFID:Cupboard_1	T_1	
ID_Cup_3	RFID:Glove_{P_1}	$T_2 > T_1$	$\text{performed}(P_1, A_1, S_1)$ $\text{actionT}(A_1, \text{Pickup})$ $\text{place}(A_1, \text{Cupboard}_1)$ $\text{involves}(A_1, \text{Cup}_3)$
ID_Cup_3	RFID:Table	$T_3 > T_2$	$\text{succ}(S_1, S_2)$ $\text{performed}(P_1, A_2, S_2)$ $\text{actionT}(A_2, \text{Putdown})$ $\text{place}(A_2, \text{Table})$ $\text{involves}(A_2, \text{Cup}_3)$

From first recognizing the RFID of Cup_3 in the cupboard and then recognizing it in person P_1's glove, we can deduce that there was some situation S_1 in which the person performed a pick up action at the containing cupboard that involved Cup_3, and this information can be represented using appropriate logical atoms that match the representation in the model.



(a) (b)

Figure 6.12: *The sensor-equipped kitchen environment (a) and markerless human motion tracking (b)*

A more sophisticated data collection scheme can incorporate many additional sources of information. In particular, the Intelligent Kitchen supports the observation of human body motions through a markerless full-body tracking system based on multiple camera views (Bandouch et al., 2008). The system was, for instance, used to collect data on table setting activities: Through the tracking system, a number of table setting sequences were recorded, human motion tracking was applied and the individual low-level actions within them were fully labelled (Tenorth et al., 2009).

Since our statistical relational models should mostly consider activities at higher levels of abstraction, a scheme of hierarchical abstractions can be used to generate more abstract data. On unlabelled data, the first step in the abstraction process is the identification of semantic motion segments (such as reaching for something with the hand, retracting the hand or carrying something to another place) from a sequence of frames indicating body configurations. As proposed by Beetz et al. (2009), linear-chain conditional random fields (CRFs) can be used to label the sequence of frames and thus obtain a semantic segmentation automatically – using both obser-

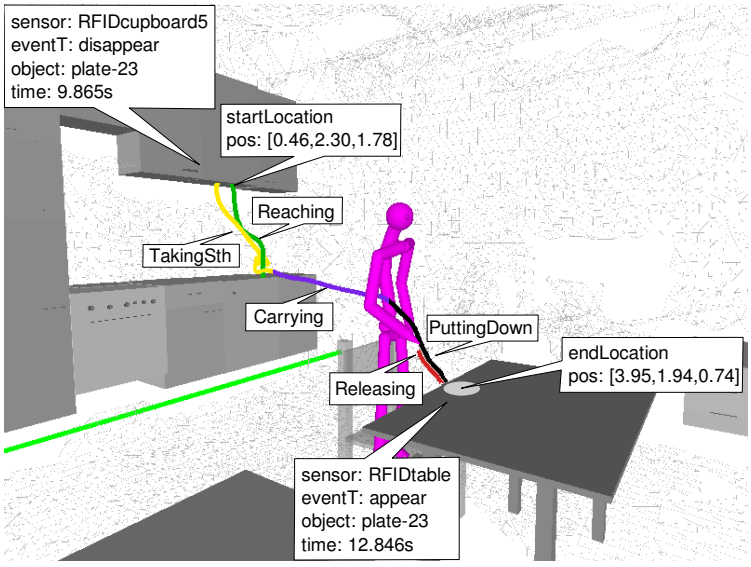


Figure 6.13: *Observations from which a physically grounded instance of PuttingSomethingSomewhere can be derived: The system uses a combination of RFID events and human motion data to identify the action and its parameters. RFID readings alone can be insufficient, since RFID readers usually cover fairly large areas (such that exact placement parameters can be obtained only from human motion data) and because erroneous RFID readings do occur quite frequently, which can be corrected only by explicitly constraining the sequence of possible motion segments in order to avoid incorrect conclusions. (Beetz, Tenorth, Jain, and Bandouch, 2010c)*

vation features derived from the body pose itself (e.g. discretised angles, positions of limbs and end-effectors) and external features derived from the environment map and the sensor network (e.g. distances to handles and knobs of containers and RFID readings). Once the basic low-level segmentation has been performed, one can recursively generate more and more abstract descriptions of the actions, such that one can obtain, for example, from a motion sequence and additional sensor readings, the abstract information that a particular object was taken from some container and placed at another location (see Figure 6.13).

Appropriately abstracted information that is semantically interpretable and thus easily represented as logical facts provides the basis for the learning of statistical relational models of everyday activities. For example, an instance of the action *PuttingSomethingSomewhere*, parametrised with the object involved and the locations from which and to which the object was moved, provides a data instance with which a

model could be trained which is concerned with the evolution of object locations as a result of human actions – or a model that is more concretely concerned with how tables are set.

Using the methods proposed by Tenorth and Beetz (2009), the relational data obtained from sensory data is stored in the KnowRob knowledge processing system, which, by integrating computational methods based on a logic programming interface, can be used to flexibly process the data. In a table setting scenario, we are interested in the utensils that are put at particular places on the table in order for people to consume food and drinks – and we may even want to capture individual preferences of the people participating in the meals we observed. In common households, table settings will be highly dependent on the type of the meal, which in turn influences the types of food and drinks that are likely to be consumed and therefore the utensils that will be used. For a model to be trained, we thus need to collect data on, for example, the utensils that are positioned at the various places on the table. Exemplarily, consider the predicate *usesAnyIn(person, utensilType, meal)*, which indicates the use of a utensil of a particular type in a particular meal by a person and which could reasonably appear within a model concerned with meals and table settings. When observing a table setting activity, we assume that the objects put at the abstract place associated with a person taking part in the meal (indicated by a subspace of the space occupied by the table) are precisely the objects that are related to an action of type *PuttingSomethingSomewhere* via the *objectActedOn* relation and whose *toLocation* attribute (given as a point in 3D space) is within the subspace of the place in question. Thus, the relational data for *usesAnyIn* can be obtained from lower-level information by using the KnowRob knowledge processing system as follows:

```
holdsOnce(atLocation(Obj, Pos), [StartTime, EndTime]) :-
    type(A, PuttingSomethingSomewhere),
    occurs(A, [AStartTime, AEndTime]),
    temporallySubsumes([StartTime, EndTime], AEndTime),
    objectActedOn(A, Obj),
    toLocation(A, Pos).
usesAnyIn(Person, ObjType, Meal) :-
    placeOfPersonInMeal(Place, Person, Meal),
    withinPlace(Pos, Place),
    type(Obj, ObjType),
    occurs(Meal, [StartTime, EndTime]),
    holdsOnce(atLocation(Obj, Pos), [StartTime, EndTime]).
```

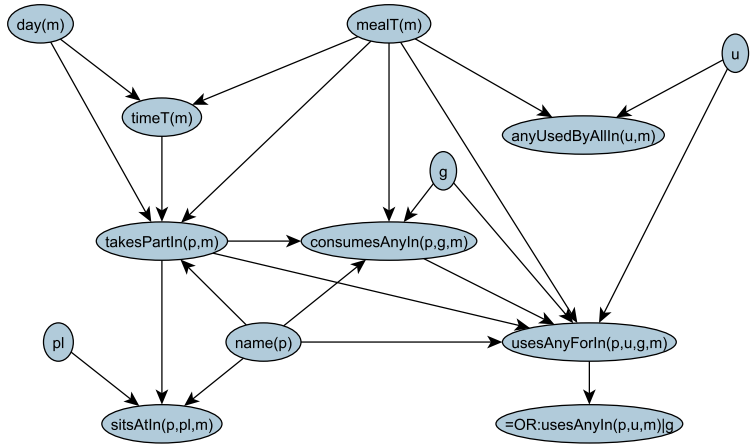
Unfortunately, learning truly expressive models of everyday activities would require substantially more data than has so far been observed and labelled by the members of the Intelligent Kitchen research group or any other researchers the world over. (Nevertheless, one of the models presented further is trained solely on real data.) Given enough time and test subjects willing to undergo continuous supervision, the setup described above would, in principle, enable the collection of large amounts of relational data.

Scripted Stochastic Process Models. The problem of limited data availability can be overcome by augmenting real-world data with synthetic data generated from stochastic processes which are modelled in a scripting language. In the case of pick and place actions, these processes essentially model entire daily schedules (considering the needs of six people with different personalities) at the level of abstraction that is required. The generators were realised as Python scripts using ProbCog's data generation library. These scripts first create a universe of objects and then, according to the activity context, sequentially generate actions that move the objects around the kitchen, altering their states, choosing the objects that are involved in actions by sampling from the set of available objects depending on action requirements, individual preferences and the current state of the world.

6.2.1.2 Modelling Everyday Activities

In the following, I describe several BLN models of everyday activities that could be of immediate use to an autonomous robotic kitchen assistant: a model of meal habits, a simple model of two cooking activities, a hierarchical model of activity sequences and a model of kitchen layouts are presented. The integration of such models into concrete robotic systems is discussed in Sections 6.2.2 and 6.2.3.

Because structure learning is often prohibitively expensive, the structure of models, i.e. the specification of possible dependencies, is assumed to be given by a knowledge engineer. All the models presented in the following were trained using standard maximum likelihood parameter learning techniques. The cooking model was exclusively trained on real data, while the others made use of synthetic data. Training database sizes ranged from 5493 positive literals in the cooking model to almost 100,000 in the hierarchical sequence model.




```
1  type person, meal, place;
2  type domMealT, domUtensilT, domGoods, domName;
3  type domDay, domTime;
4  guaranteed domGoods Bread, Cake, Cereals, Cheese, Coffee, Juice, Milk, Noodles, Pizza, Salad, Sausage,
       Soup, Tea, Water;
5  guaranteed domName Anna, Bert, Charly, Dorothy, Emily, Frank;
6  guaranteed domMealT Breakfast, Lunch, Dinner;
7  guaranteed domDay Saturday, Sunday, Monday, Tuesday, Wednesday, Thursday, Friday;
8  guaranteed domTime EarlyMorning, LateMorning, Noon, EarlyAfternoon, LateAfternoon, Evening;
9  guaranteed domUtensilT Bowl, Cup, Fork, Glass, Knife, Napkin, Pitcher, Plate, Platter, Spoon;
10 guaranteed place Seat0, Seat1, Seat2, Seat3;
11 random boolean usesAnyForIn(person, domUtensilT, domGoods, meal);
12 random boolean consumesAnyIn(person, domGoods, meal);
13 random domName name(person);
14 random domMealT mealT(meal);
15 random domDay day(meal);
16 random domTime timeT(meal);
17 random boolean sitsAtIn(person, place, meal);
18 random boolean takesPartIn(person, meal);
19 random boolean anyUsedByAllIn(domUtensilT, meal);
20 random boolean usesAnyIn(person, domUtensilT, meal);
21 constraint sitsAtIn(p,pl,m) ^ sitsAtIn(p2,pl,m) => p = p2;
22 constraint takesPartIn(p,m) => (EXIST pl (sitsAtIn(p,pl,m) ^ !(EXIST pl2 (sitsAtIn(p,pl2,m) ^ !(pl =
       pl2)))));
```

BLN 6.1: *Model of meal habits*

A Model of Meal Habits. One type of activity that shall be considered in more detail is that of having meals and setting the table for meals. Usually, there is no standard recipe with which the task of setting the table could be carried out. How a table is to be set depends on a number of factors, including the type of the meal, what is going to be consumed, the number of participants and even the identities of the participants (assuming that we know about their individual preferences).

The BLN model 6.1 links many relevant pieces information, including the type of the meal (*mealT*), the participation of a (known or unknown) person (*takesPartIn,*

name), the types of utensils used by a particular person (*usesAnyIn*), the food/drinks consumed (*consumesAnyIn*), the utensils shared among the participants (*anyUsedBy-AllIn*) and the places at which people sit (*sitsAtIn*). As logical constraints, the model requires completeness and functional properties of the *sitsAtIn* relation.

The model is specific to a particular kitchen and its users (i.e. the set of potential participant identities is a fixed domain and so is the set of places around the table). The extension of the domain of people and meals, however, can vary across instantiations of the model. To render the application practical, the model makes some simplifying independence assumptions. For instance, the dependency between any two propositions pertaining to consumption is only indirect. Furthermore, the model does not represent individual instances of, for instance, cups; it considers only the classes of utensils, which is appropriate in many application contexts – including table setting. In fact, differentiating different utensils of the same type would only be helpful if a distinction was required (e.g. to provide a particular person with his/her favourite cup); otherwise it would only render individual usage probabilities low, distributing probability mass across identical objects. The model also considers a perhaps arbitrary selection of food and drinks. In order to make such a model scale to a wide variety of common foodstuffs, we can make use of the principle of abstraction: In scenarios where, for instance, the difference between two pasta dishes is largely immaterial, we can simply subsume them under an abstract concept such as *PastaDish*. An ontology such as the one represented in KNOWROB can provide the necessary domain knowledge to perform such abstractions (Tenorth and Beetz, 2009).

One application of the model is to infer sensible table settings given a number of preconditions. For instance, imagine that a person has partially set the table for breakfast and then asks a kitchen robot to complete the task. Given that at one of the table places, a cup and coffee is already present, the query the robot issues might be

$$
\begin{aligned}
&P(usesAnyIn(P, ?u, M); consumesAnyIn(P, ?f, M) \mid \\
&\qquad mealT(M) = Breakfast \wedge usesAnyIn(P, Cup, M) \wedge consumesAnyIn(P, Coffee, M)) \\
&\approx \langle\langle\ Plate: 0.94, Knife: 0.94, Spoon: 0.57, Bowl: 0.55, Glass: 0.48, Fork: 0.26, \ldots\rangle, \\
&\quad \langle\ Bread: 0.83, Cheese: 0.74, Cereals: 0.46, Milk: 0.45, Juice: 0.44, \ldots\rangle\rangle
\end{aligned}
$$

and the robot could subsequently bring to the table the objects for which a sufficiently high probability is indicated by the model.

As additional evidence becomes known, e.g. the information that in addition to the cup of coffee, a tablespoon is present, the probability distribution will change appropriately,

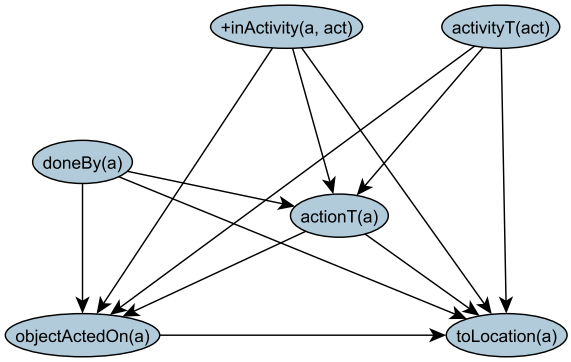
$$
\begin{aligned}
&P(usesAnyIn(P, ?u, M); consumesAnyIn(P, ?f, M) \mid mealT(M) = Breakfast \wedge \\
&\qquad usesAnyIn(P, Cup, M) \wedge consumesAnyIn(P, Coffee, M) \wedge usesAnyIn(P, Tablespoon, M)) \\
&\approx \langle\langle\, Bowl: 0.97, Plate: 0.91, Knife: 0.90, Glass: 0.71, Fork: 0.32, \ldots\rangle, \\
&\quad \langle\, Cereals: 0.80, Milk: 0.80, Bread: 0.75, Juice: 0.66, Cheese: 0.61, Sausage: 0.57, \ldots\rangle\rangle
\end{aligned}
$$

since the tablespoon implies that cereals are likely to be consumed, requiring a bowl, while the probabilities for bread and associated entities are slightly reduced – as a result of a changed distribution of personalities and therefore preferences. The probability for juice increases, because, in the dataset, preferences for cereals and juice coincided.

The computation of posterior marginals is sensible if we want to control the probability threshold beyond which objects should be considered. If we are interested only in the most likely setting, MAP/MPE inference is, of course, more appropriate, as it is guaranteed to result in a consistent table setting (i.e. it would not, for instance, result in the robot providing a glass without a matching drink).

Of course, the model could also be used to compute queries that are not related to the task of table setting. For instance, we could infer the most probable identity of a person given the consumptions during a number of meals – or the most probable meal type given the things that were observed on the table.

A Simple Model of Cooking Activities. For a robotic household assistant, acquiring a notion of what it means to perform particular meal preparation tasks can be helpful. The cooking model uses data on real-world cooking activities from the CMU-MMAC dataset (De la Torre et al., 2009), which is integrated into the KnowRob knowledge processing system and is thus available for the training of models based entirely on real-world data. The training set contained 23 sequences of 16 subjects performing two types of cooking activities, *BakingBrownies* and *CookingAnOmelette*. The model, which is shown in BLN 6.2, relates these activities to the actions appearing within them. Actions are characterised by a type and a subject, and they may involve objects and locations. The categories that are used for actions and locations correspond to concepts within the KnowRob kitchen ontology. The model does not take the order




```
1  type activity, domActivityT;
2  type frame, domObjectActedOn, domLocation, domActionT, subject;
3  guaranteed domActivityT CookingAnOmelette, CookingBrownies;
4  guaranteed domLocation ToLocationNull, Bowl−Mixing, CakePan, CardbordBox−Eggs, CounterTop,
       Cupboard, DinnerPlate, Drawer, FryingPan, MeasuringCup−Big, MeasuringCup−Small, Oven,
       Refrigerator, Sink, StoveTop;
5  guaranteed domObjectActedOn ObjectActedOnNull, AerosolCan, Bowl−Mixing, CakePan, CardbordBox,
       CardbordBox−Eggs, ContainerLid, Cupboard, DinnerPlate, Drawer, Egg−Chickens, Eggshell, Fork−
       SilverwarePiece, FryingPan, Knife, MeasuringCup−Big, MeasuringCup−Small, PaperTowel, Pepper,
       PlasticBag, Refrigerator, Scissors, Spatula, Spoon−SilverwarePiece, StoveTop, TableSalt, VegetableOil
       , Water;
6  guaranteed subject P0, P1, P2, P3, P4, P5, P6, P7, P8, P9, P10, P11, P12, P13, P14, P15;
7  guaranteed domActionT Cleaning, ClosingABottle, ClosingSomething, Cracking, OpeningABottle,
       OpeningSomething, Pouring−MakingSomethingAvailable, PuttingSomethingSomewhere, Reading,
       Spraying, Stirring, Stirring−Beating, TakingSomething, TurningOffPoweredDevice,
       TurningOnPoweredDevice, Walking−Generic;
8  random domActivityT activityT(activity);
9  random domObjectActedOn objectActedOn(frame);
10 random domLocation toLocation(frame);
11 random domActionT actionT(frame);
12 random subject doneBy(frame);
13 logical boolean inActivity(frame, activity);
14 relationKey inActivity(a,_);
```

BLN 6.2: *Model of cooking activities*

of actions into consideration. Through the relational formulation, the model does, however, allow us to consider a variable number of actions in queries.

The model can, for example, be used by an autonomous robot to infer the most likely activity given individual observations on actions that have taken place, since activities are inextricably linked with (actions involving) particular types of objects (Philipose et al., 2004):

$$
\begin{aligned}
&P(activityT(Act) \mid inActivity(A1, Act) \wedge inActivity(A2, Act) \wedge \\
&\quad actionT(A1) = TakingSomething \wedge objectActedOn(A1) = Bowl\text{-}Mixing \wedge \\
&\quad objectActedOn(A2) = Egg\text{-}Chickens) \\
&= \langle\, BakingBrownies: 0.672,\ CookingAnOmelette: 0.328 \,\rangle
\end{aligned}
$$

In the above query, a bowl was fetched and some action was performed on an egg. Based on such observations, the system cannot clearly identify the activity, but it favours *BakingBrownies*, because the observed actions apparently play a more defining role in this activity. If, however, we supplied the additional evidence on an action involving a frying pan, expectations are shifted,

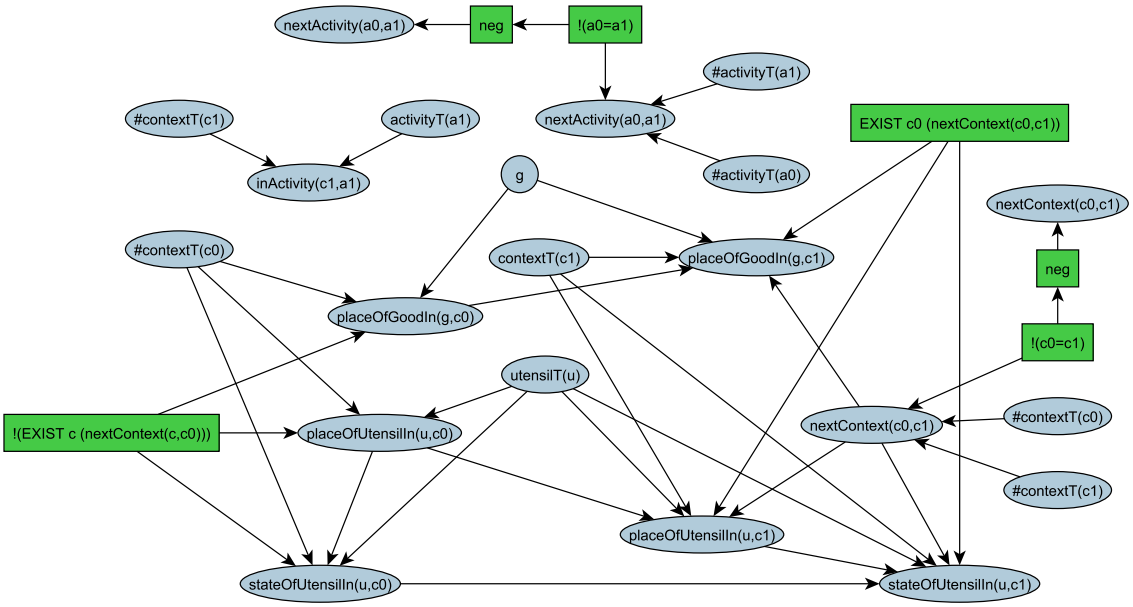
$$
\begin{aligned}
&P(activityT(Act) \mid \\
&\quad inActivity(A1, Act) = True \wedge actionT(A1) = TakingSomething \wedge \\
&\quad objectActedOn(A1) = Bowl\text{-}Mixing \wedge inActivity(A2, Act) = True \wedge \\
&\quad objectActedOn(A2) = Egg\text{-}Chickens \wedge inActivity(A3, Act) = True \wedge \\
&\quad objectActedOn(A3) = FryingPan \wedge toLocation(A3) = StoveTop) \\
&= \langle\, BakingBrownies: 0.000, CookingAnOmelette: 1.000 \,\rangle
\end{aligned}
$$

allowing the robot to ascertain that the actions serve to prepare an omelette, enabling it to decide on appropriate actions to take. It could, for instance start setting the table appropriately.

A Hierarchical Model of Activity Sequences. We now consider a model that is concerned with sequences of everyday activities, capturing the way in which objects change their location as these activities are performed as well as their state (e.g. from clean to dirty). The model is shown in BLN 6.3. Activities are dealt with at two different levels of abstraction: the very coarse level of high-level activities such as entire meals or cleaning activities, and the slightly more fine-grained level of subactivities, involving, for example, the preparation of food, setting the table for a meal, serving food, cleaning the dishes after a meal, etc. At these two levels of abstraction, the model considers a fairly complete daily schedule of kitchen activities.

As abstract random variables, the model considers the sequence of higher-level activities and lower-level action contexts (*nextActivity, nextContext*), their types (*activityT, contextT*), and the locations and states of utensils and consumable goods after an activity context (*placeOfUtensilIn, stateOfUtensilIn, placeOfGoodIn*). Activities, contexts and utensils are individual entities that may vary across instantiations of the model (while the set of goods is assumed to be fixed).

As a robotic application, consider the problem of locating objects within the environment. As activities take place, locations of objects will change. For example, assume that the sequence of activities ⟨*MakeNoodles, SetTable, ServeFood*⟩ has been




```
1  type activity, domActivityT;
2  type actionContext, domContextT;
3  type utensil, domUtensilT, domPlace, domState, domGoods;
4  guaranteed domGoods Bread, Cake, Cereals, Cheese, Coffee, Juice, Noodles, Pizza, Salad, Sausage, Soup,
       Tea, Water;
5  guaranteed domState Clean, Dirty;
6  guaranteed domPlace Cupboard, Dishwasher, Fridge, InOven, InSink, Stove, Table, Workplate;
7  guaranteed domContextT CleaningDishesInSink, Dining, MakeCoffee, MakeNoodles, MakePizza,
       MakeSalad, MakeSoup, MakeTea, PostMeal, PreMeal, ReturnDishes, ServeCoffee, ServeFood,
       ServeTea, SetTable, TableCleaning, WashingDishes;
8  guaranteed domUtensilT Bowl, CookingPot, Cup, Fork, Glass, Knife, Plate, SaladBowl, Spoon;
9  guaranteed domActivityT Breakfast, Cleaning, Dinner, Lunch, Snack;
10 random domActivityT activityT(activity);
11 random boolean inActivity(actionContext, activity);
12 random domContextT contextT(actionContext);
13 logical boolean nextActivity(activity, activity);
14 logical boolean nextContext(actionContext, actionContext);
15 random domUtensilT utensilT(utensil);
16 random domState stateOfUtensilIn(utensil, actionContext);
17 random domPlace placeOfUtensilIn(utensil, actionContext);
18 random domPlace placeOfGoodIn(domGoods, actionContext);
19 relationKey nextContext(_,c);
20 relationKey nextActivity(_,a);
21 constraint inActivity(c,a1) ^ inActivity(c,a2) => a1 = a2;
22 constraint EXIST a (inActivity(c,a));
```

BLN 6.3: *Model of activity sequences*

observed. (Such activities could have been identified, in part, by using models similar to the previous one.) Given this sequence of activities, we are interested in finding out where we are likely to find a clean plate. Furthermore, we ask for the location of a pot as well as the state it is in:

```
P(placeOfUtensilIn(U1, C3); placeOfUtensilIn(U2, C3); stateOfUtensilIn(U2, C3) |
      utensilT(U1) = Plate ∧ utensilT(U2) = CookingPot ∧
      stateOfUtensilIn(U1, C3) = Clean ∧ contextT(C1) = MakeNoodles ∧
      contextT(C2) = SetTable ∧ contextT(C3) = ServeFood ∧
      nextContext(C1, C2) = True ∧ nextContext(C2, C3) = True)
≈ ⟨⟨ Cupboard: 0.58, Table: 0.42, ...⟩,
   ⟨ Stove: 0.83, Cupboard: 0.17, ...⟩,
   ⟨ Dirty: 0.83, Clean: 0.17 ⟩⟩
```

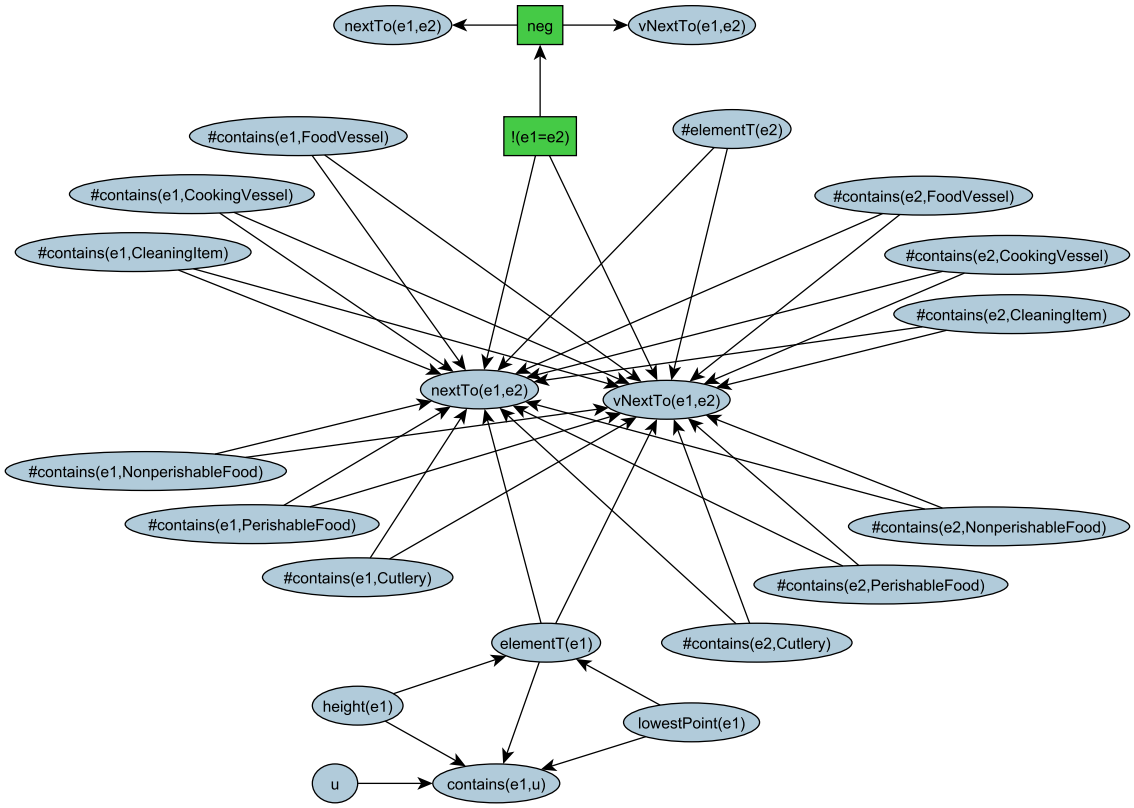
This result is certainly reasonable (given the model's implicit knowledge about the number of utensils available).

Inversely, we could provide information on the locations or states of various objects and ask for the activities that can lead to such a situation and furthermore ask for the higher-level activity that these activities are all likely to be part of:

```
P(activityT(A); contextT(C1); contextT(C2); contextT(C3) |
     inActivity(C1, A) ∧ inActivity(C2, A) ∧ inActivity(C3, A) ∧
     nextContext(C1, C2) ∧ nextContext(C2, C3) ∧
     utensilT(U1) = Plate ∧ placeOfUtensilIn(U1, C3) = Table ∧
     utensilT(U2) = CookingPot ∧ placeOfUtensilIn(U2, C3) = Heater ∧
     placeOfGoodIn(Noodles, C3) = Table)
≈ ⟨⟨ Lunch: 0.86, ...⟩,
   ⟨ MakeNoodles: 0.80, SetTable: 0.20, ...⟩,
   ⟨ SetTable: 0.80, ServeFood: 0.20, ...⟩,
   ⟨ ServeFood: 0.80, Dining: 0.20, ...⟩⟩
```

In a similar fashion, we could ask which actions transform a dirty plate on the dining table into a clean one; and the model will tell us that the plate is moved from the table to the dishwasher, which is subsequently run, and that one then typically returns the now clean plate to the cupboard.

A Model of Kitchen Layouts. General information about the environments a robotic household assistant is to operate in can also be relevant in practice. In the general context of kitchens, let us consider a model that associates the configuration of the main containers and appliances in a kitchen with the objects within. Imagine a robot that first enters a kitchen environment, performs a 3D scan of the kitchenette, identifies cupboards, dishwashers and other devices from the scan data (Rusu et al., 2009) and symbolically represents the acquired information in an ontology that has conceptual knowledge about the particular types of devices and containers (Tenorth et al., 2010). During an early orientation phase, the information thus collected can




```
1  type element, objectClass;
2  type domElementT, domHeight, domLowestPoint, objectClass;
3  guaranteed domElementT Cupboard, Drawer, Fridge, Oven, Sink, Washer;
4  guaranteed domHeight Small, Medium, Large;
5  guaranteed domLowestPoint H0, H2, H6;
6  guaranteed objectClass CleaningItem, CookingVessel, Cutlery, FoodVessel, NonperishableFood,
       PerishableFood;
7  random domElementT elementT(element);
8  random domHeight height(element);
9  random Boolean vNextTo(element, element);
10 random domLowestPoint lowestPoint(element);
11 random Boolean contains(element, objectClass);
12 random Boolean nextTo(element, element);
```

BLN 6.4: *Model of kitchen layouts*

be sufficient for the robot to determine where particular types of objects might be found in the environment.

This problem is addressed in BLN 6.4, which considers the type of kitchen fittings/elements (*elementT*), the horizontal and vertical spatial neighbourhood relation (*nextTo*, *vNextTo*) and the *contains* relation linking kitchen elements to the types of objects they contain. Note that to determine the types of objects an element contains, the model can take into account the types of objects contained in neighbouring elements

as well as the proximity to other kitchen elements. The model was trained on the configurations of 100 kitchens.

Instantiating this model for a particular environment, e.g. the kitchen shown in Figure 6.14, which represents the configuration of a robot laboratory at TUM, the model can be used to infer the storage locations for the abstract classes of objects considered by the model. We denote by e the configuration of the particular kitchen shown in Figure 6.14, i.e. the relational description involving the full extensions of the predicates of *elementT*, *height*, *lowestPoint*, *nextTo* and *vNextTo*. Given this configuration, the model's expectations regarding the locations of some of the classes of objects are as follows:

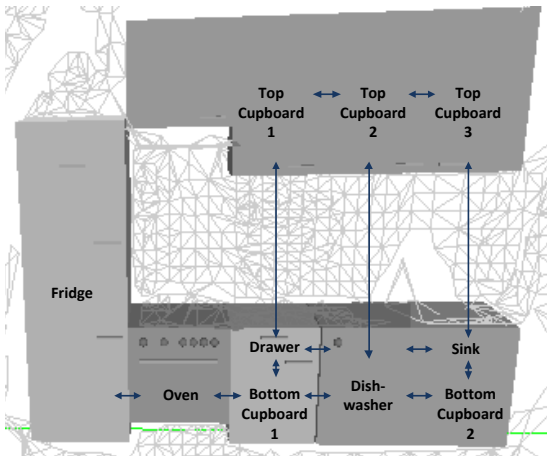


Figure 6.14: *Kitchen fittings and neighbourhood relations*

$$
\begin{aligned}
&P(\mathit{contains}(?e, \mathit{Cutlery}); \mathit{contains}(?e, \mathit{CookingVessel}); \mathit{contains}(?e, \mathit{CleaningItem});\\
&\quad \mathit{contains}(?e, \mathit{NonperishableFood}); \mathit{contains}(?e, \mathit{PerishableFood}) \mid e)\\
&\approx \langle\langle\ \mathit{Drawer}: 0.87, \mathit{Sink}: 0.23, \mathit{Dishwasher}: 0.15, \mathit{TopCupboard2}: 0.08, \ldots\rangle,\\
&\quad \langle\ \mathit{BottomCupboard1}: 0.71, \mathit{TopCupboard1}: 0.11, \mathit{Sink}: 0.11, \mathit{TopCupboard3}: 0.06, \ldots\rangle,\\
&\quad \langle\ \mathit{BottomCupboard2}: 0.99, \mathit{TopCupboard3}: 0.51, \mathit{BottomCupboard1}: 0.09, \ldots\rangle,\\
&\quad \langle\ \mathit{TopCupboard1}: 0.49, \mathit{TopCupboard2}: 0.12, \mathit{BottomCupboard1}: 0.12, \ldots\rangle,\\
&\quad \langle\ \mathit{Fridge}: 1.00, \mathit{TopCupboard3}: 0.21, \mathit{TopCupboard2}: 0.07, \mathit{TopCupboard1}: 0.01, \ldots\rangle\rangle
\end{aligned}
$$

The set of object classes considered by the model is fairly small. Note, however, that because the classes considered are rather abstract, it will suffice in cases where more specific types of objects need not be differentiated. For a search task during an initial orientation phase, a robot can reasonably use its ontology to map the kind of object it is searching for to the closest matching superclass that is considered by the model.

The given model would, of course, not scale to significantly larger sets of object classes, because the conditional distributions in the fragments for *nextTo* and *vNextTo* would become excessively large. We could, however, make some simplifying assumptions and use a combine rule to build up these conditional distributions from smaller parts.

For instance, the fragment shown in Figure 6.15 could, along with a combining rule definition for *nextTo*, be used to replace the corresponding fragment in BLN 6.4: It splits the representation of the conditional distribution across c^2 smaller distributions if c is the size of the *objectClass* domain. This will yield a significantly more compact representation, provided that, in ground models, the conditional distribution is not explicitly represented as a table but is computed on demand. Similarly, one could use a Markov logic network in which the dependency between a *nextTo* atom and the $2c$ *contains* statements involving the two neighbouring containers is represented through formulas involving only a subset of the relevant atoms – perhaps a pairwise approach analogous to Figure 6.15.

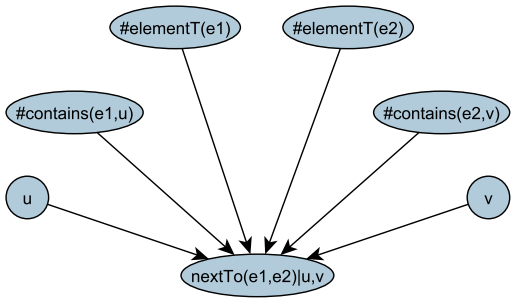


Figure 6.15: *An alternative fragment specification for* nextTo

An application of the above model is useful for a case where the robot has not yet fully explored its environment. Once the robot has familiarised itself with the storage locations of the kitchen environment and has determined their contents, it faces the new problem of having to identify a structure in the organisation of the kitchen. Solving this problem would, for example, enable it to suitably place an object which was not previously stored (a task that arises when trying to find storage locations for newly purchased groceries). Schuster et al. (2012) have shown that, for such a task, the very simple notion of semantic concept similarity (computed via the structure of an ontology) can provide a good model of organisational principles in kitchen environments. Beyond the kitchen and its containers, Vasudevan and Siegwart (2008) and Viswanathan et al. (2009) have, for instance, investigated models that consider the structure of environments at a slightly coarser level, linking, for instance, rooms with the objects they contain.

6.2.1.3 Discussion

In this section, the applicability of statistical relational models to the realm of everyday activities was investigated. While limited data availability necessitated the use of

synthetic data, the architecture that was described can allow models to be built from relational data that is physically grounded. As the examples have indicated, it is, in principle, possible to learn comprehensive models of everyday activities that are capable of answering a multitude of interesting and highly relevant queries, particularly in view of physical AI systems such as robotic household assistants. Within the limits of their applicability, the models presented in this section can produce results that largely reflect humans' understanding of common sense.[3] The learning of generalised full-joint probability distributions for everyday activities can thus be considered as a promising approach towards the construction of knowledge bases that are potentially useful to robotic household assistants and other AI systems acting in everyday settings.

Naturally, if all-encompassing models of everyday activities are to be created, it is, given the inherent limits of tractability, infeasible to consider all aspects within a single model. It would, however, be conceivable to consider the entire daily schedule of activities in many loosely coupled probabilistic models that can largely be handled independently.

The following sections address the issue of integrating such models into real-world robotic systems as well as the issue of learning models from the robot's own experience data.

6.2.2 Plan Parametrisation

The models described in the previous section are a potentially instrumental resource for an autonomous robot. This section will show how the powerful representational paradigm of statistical relational models can be transparently integrated into robot control programs, enabling robots to perform their tasks more adaptively – by parametrising plans through probabilistic reasoning processes that take context information into account (cf. Jain, Mösenlechner, and Beetz, 2009b). The integration is realised through a coupling of robot control programs with the ProbCog statistical relational reasoning system and extensions of the robot's plan language that are to

[3] This was informally confirmed in a small user study involving nine subjects, who subjectively evaluated the results given by the models on a variety of queries – in relation to the answers given by the other subjects on natural language translations of the same queries.

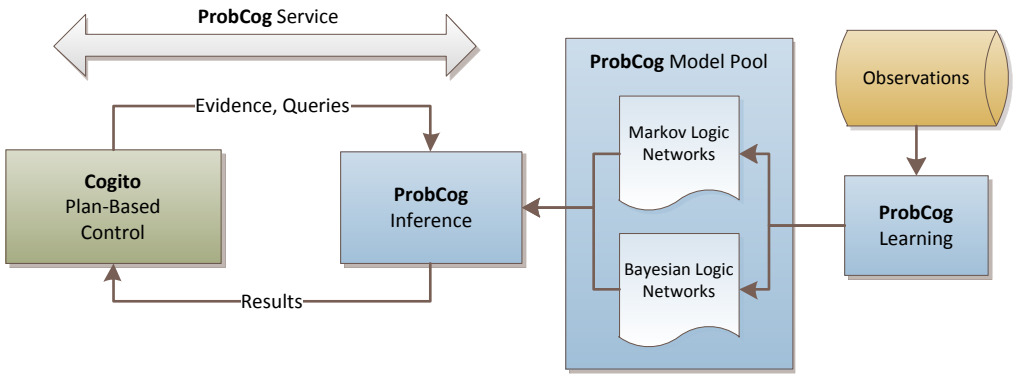


Figure 6.16: *Coupling of the plan-based control module and the probabilistic reasoning module*

support the interaction with the reasoning system as well as the interpretation of probabilistic results. The Intelligent Kitchen serves as the application context.

The basic architecture is shown in Figure 6.16. The probabilistic reasoning engine and the robot controller that makes use of it are realised as separate processes that interact via remote procedure calls. The plan-based controller uses the Lisp-based CRAM plan language, CRAM-PL. It stores known facts about entities in the robot's environment in a knowledge base which can be used to provide evidence to the probabilistic reasoner. Whenever the control program is faced with a situation in which probabilistic inference is necessary, e.g. an under-specified task, it queries the probabilistic reasoning system by issuing a request consisting of the name of the model to use as well as a list of evidence variables (taken from its knowledge base) and a list of query variables. The ProbCog reasoner, which manages a pool of probabilistic models, then processes the request by instantiating the selected model for the given set of entities, running the inference method, and finally returning the results in a reply. The robot controller then processes the returned probabilities by applying suitable operators (e.g. thresholding or argmax) and uses the processed results to parametrise its plans or modify its control program in general.

As a simple example, consider the problem of setting the table for breakfast. Assume that in its knowledge base, the robot has stored the information that exactly three people will participate, namely Anna, Bert and Dorothy – members of the family that are known to the model. To set the table, we need to know what utensils will be required at which seat; therefore if we know what utensils people are likely to

use and where they will sit, we have the information that we need. The problem translates to a probabilistic query that can be issued to BLN 6.1 (see the previous section) as follows,

P(sitsAtIn(?p, ?pl, M); usesAnyIn(?p, ?u, M) |
mealT(M, Breakfast) ∧ day(M, Thursday) ∧
takesPartIn(P1, M) ∧ name(P1, Anna) ∧
takesPartIn(P2, M) ∧ name(P2, Bert) ∧
takesPartIn(P3, M) ∧ name(P3, Dorothy))

i.e. there is a breakfast meal M, in which the three people take part, and we are interested in the probability of *sitsAtIn* atoms telling us who will sit where and *usesAnyIn* atoms telling us who will use which utensils. The query will return, for each person and place, the probability of the corresponding *sitsAtIn* atom, and, for each person and type of utensil, the probability of the corresponding *usesAnyIn* atom.

The robot can thus use its model of meal habits to infer sensible table configurations and thereby parametrise its table setting plan given the information it is given about the activity. For the query above, the model produced results that imply the configuration shown in Figure 6.17a when assuming for each person the most likely seating location and assuming that objects where the usage probability is at least 0.05 should be considered, the concrete probabilities being:

usesAnyIn(P1, Plate, M)	0.9981	usesAnyIn(P1, Pitcher, M)	0.0000
usesAnyIn(P1, Fork, M)	0.0000	**usesAnyIn(P2, Plate, M)**	0.9967
usesAnyIn(P1, Cup, M)	0.9136	usesAnyIn(P2, Fork, M)	0.0000
usesAnyIn(P1, Platter, M)	0.0000	**usesAnyIn(P2, Cup, M)**	0.8546
usesAnyIn(P1, Bowl, M)	0.0347	...	
usesAnyIn(P1, Glass, M)	0.0000	**sitsAtIn(P1, Seat1, M)**	1.0000
usesAnyIn(P1, Knife, M)	0.9981	**sitsAtIn(P2, Seat2, M)**	0.7815
usesAnyIn(P1, Spoon, M)	0.0347	...	

These results would change considerably if we removed from the evidence the identities of the three people, as illustrated in Figure 6.17b.

The probabilistic reasoning mechanisms can be integrated directly into the language that is used for high-level control, CRAM-PL. Since CRAM-PL is built on top of Lisp (Beetz et al., 2010b), the language allows to conveniently add new language constructs to this end. CRAM-PL not only allows the language to be extended with new commands but also with new special forms for the generation of variable bindings. In particular, the language can easily be extended with a new special form that binds

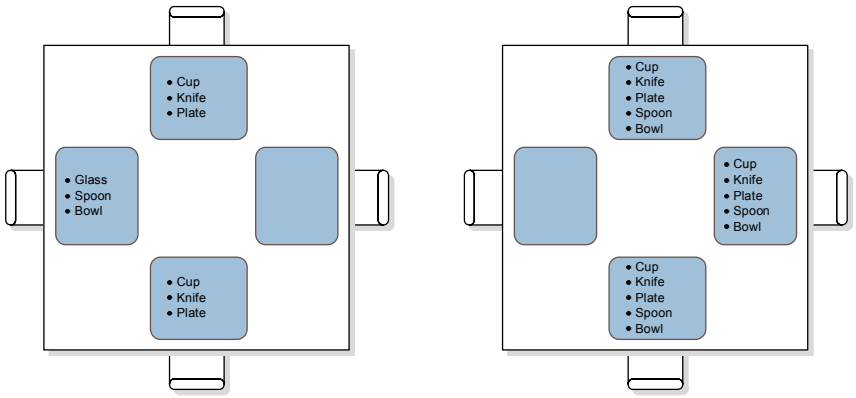


Figure 6.17: *Interpreted inference results*

variables in order for plan parametrisation to be directly based on the results of probabilistic inference. The new special form *likely-let* was added to CRAM-PL for this purpose. Like the special form *let*, it establishes a binding of variables in the current lexical context. Given a specification of the evidence, it extracts the inference results for a given set of queries and binds ground atoms (along with their probabilities) to variables. Furthermore, the results can be filtered beforehand by either applying a probability threshold or returning only the most probable atom from a particular set of ground atoms (i.e. $\arg\max$). In this way, *likely-let* transparently enables a CRAM-PL plan to draw upon the results of probabilistic inference.

As an example, consider the plan for setting the table for the case described in earlier, where the three people named Anna, Bert and Dorothy will participate and the type of the meal is breakfast. A full CRAM-PL plan for this task is given in Figure 6.18. The program queries the seating locations and the objects used by the participants by using *likely-let*. The operators *argmax* and *threshold* are applied in a post-processing step. The variable *places* is bound to a list containing the seating location with the highest probability for every person,

```
((P1 Seat1 M) (P2 Seat2 M) (P3 Seat0 M))
```

```
1  (likely-let
2    ((places
3        :query
4         '(sitsAtIn ?person ?seating-location M)
5        :argmax ?person)
6      (utensils
7        :query '(usesAnyIn ?person ?utensil M)
8        :threshold 0.05)
9      :evidence
10       '((takesPartIn P1 M) (name P1 "Anna")
11         (takesPartIn P2 M) (name P2 "Bert")
12         (takesPartIn P3 M) (name P3 "Dorothy")
13         (mealT M "Breakfast")))
14   (with-designators
15      ((table '(the entity (name kitchen-table))))
16     (for-all-matching
17         (lambda ((?person ?place ?m)
18                  (?person ?entity-type ?m))
19           (with-designators
20              ((obj (an entity (type ,entity-type)
21                               (status unused)))
22               (seat (a location (on ,table)
23                                 (place ,place))))
24             (achieve (entity-on-entity
25                        obj table
26                        seat))))
27         (cross-product places utensil))))
```

Figure 6.18: *A table setting plan that makes use of probabilistic inference (Jain, Mösenlechner, and Beetz, 2009b)*

and the variable *utensils* contains a similar but much longer list of arguments of *usesAnyIn* atoms that are filtered from the full list of results by using the *threshold* operator.[4]

The plan iterates over the matching elements of the cross product generated by combining the elements of *places* and *utensils*. The command *for-all-matching* executes the lambda body (lines 19–26) only for pairs matching the pattern *((?person ?place ?m) (?person ?entity-type ?m))* (line 17). Pictures of the robot performing this plan in simulation are shown in Figure 6.19.

As the example shows, probabilistic inference can provide a sound way of parametrising under-specified plans. The architecture described in this section effectively

[4]It should be noted that deriving the most probable seating location from posterior marginals may produce inconsistent results. In case of inconsistencies, we can either resolve conflicts in the post-processing stage of the probabilistic results or apply MAP/MPE inference instead, which is certainly appropriate for the seating locations. Using posterior marginals to determine object use, however, is somewhat more flexible, as it allows us to control the degree to which the robot should be conservative or eager in its selection of objects to consider, simply by altering the probability threshold above which we consider the statements to be relevant.

equips robot control programs with first-order probabilistic reasoning capabilities, enabling robots to react to changing circumstances and action conditions by adapting their plans appropriately. In general, an intelligent autonomous system that is to act competently in a real-world environment must accurately represent knowledge about its environment, and, given the high degree of uncertainty of real-world domains, it is clear that at least some parts of this knowledge must be probabilistic. The present architecture can be seen as a first step towards a cognitive system that represents complex knowledge in a declarative, flexible way and that is in a position to employ that knowledge in ways that support adaptive and reliable behaviour.

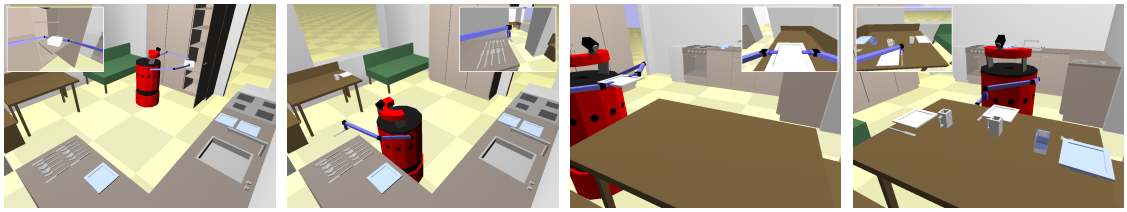


Figure 6.19: *The simulated robot performing an automatically parametrised plan for table setting*

6.2.3 A Robot System that Combines Perception, Knowledge Processing and Probabilistic Reasoning

We will now consider a fully integrated, real-world robot system that – within its particular application context – closes the perception-cognition-action loop. The K-CoPMan system for knowedge-enabled cognitive perception and manipulation explicitly integrates knowledge processing and probabilistic reasoning to solve real-world manipulation problems (Pangercic, Tenorth, Jain, and Beetz, 2010).

As an example scenario, we consider once more the task of fetching items for a meal. Assume that a single person is about to have a meal and that the robot is to jointly set the table with that person or continue setting a table that has already been partially set. The robot is to fetch items that are likely to be required for the meal but are yet missing from the table's current setup. The robot is to solve this task by augmenting its knowledge about the meal at hand with the perceptual input it gathers by observing the table, appropriately abstracting its observations into a symbolic format

that allows it reason probabilistically about the items that are most likely to be missing.

The system architecture integrates three main components:

- The K-CoPMan perception server (developed by Dejan Pangercic) provides image and point cloud processing for the detection of objects already placed on the table. In particular, the system's perceptual capabilities hinge upon its capability to find horizontal planes (Klank et al., 2009), to find clusters representing object hypotheses upon these planes, and to identify objects based on CAD model matching and SURF (speeded up robust feature) matching (Ulrich et al., 2009). Furthermore, the perception server supports manipulation capabilities by reconstructing object surfaces (Blodow et al., 2009), which are useful for grasping.

- The KnowRob knowledge processing system (Tenorth and Beetz, 2009) integrates general encyclopaedic knowledge about classes of objects, knowledge about the robot's particular environment, and the perceptual data that is gathered by the robot. Furthermore, it provides an interface to the control program that executes the actual robot actions.

- The ProbCog system for statistical relational reasoning provides the probabilistic reasoning capabilities.

The KnowRob system is integral in linking the three components. At its core, KnowRob is a logical knowledge processing system based on a Prolog reasoner. Logical knowledge is stored chiefly using the web ontology language (OWL), which allows for abstract modelling in terms of classes of objects and instances thereof.

In addition to pure logical reasoning, KnowRob supports so-called *computable predicates*, which can be evaluated using arbitrary mechanisms, triggering computations as side effects. Such computable predicates can be used to make explicit the implicit knowledge that is available in the robot's low-level data structures and can furthermore serve as an interface to non-logical reasoning processes. In particular, computable predicates are used to spawn computations within the perception server and the ProbCog system for probabilistic reasoning.

On the perception side, the system abstracts away from perceptual mechanisms and stores the results in a symbolic format that is suitable for logical knowledge processing. Furthermore, by having a model of the robot's environment and linking the objects within to semantic categories, the system attains the ability to trigger perception routines that serve a particular purpose in the robot's tasks. For instance, the system can direct the perception server to identify objects that are positioned on the particular table that serves as the dining table in the environment, and it is furthermore able to establish that the objects thus recognized indeed have the corresponding spatial relation with the dining table.

On the probabilistic reasoning side, the system can trigger a concrete probabilistic inference process in the ProbCog system, constructing an evidence database based on the very information that is stored within the logical knowledge base itself, e.g. information on objects that were perceived in the scene. The results of probabilistic inference can be post-processed as required by the application and can subsequently be returned to the knowledge processing system, thus implementing a computable predicate. In the table setting problem defined above, one such predicate is the *neededObjectsForMeal* predicate, which is to be true for classes of objects that are most likely to be required on the table.

Within the KnowRob system, the information conveyed by such a predicate can be combined with other information in order to determine, through the definition of the predicate *missingObjects*, the set of objects that is to be acted upon. As this predicate contains precisely the task-relevant information for the control program, it can be queried by the Cogito planner to parametrise a plan that will achieve that the objects that are most likely to be missing will be fetched and positioned accordingly.

Figure 6.20 illustrates the implementation of the *missingObjects* predicate. Its computation includes background knowledge stored within the KnowRob system, perceptual processing provided by the perception server and probabilistic reasoning provided by the ProbCog server. The first three conditions identify the table that serves as the dining table in the kitchen, as indiciated by the robot's environment model. The fourth condition triggers the perception routines that classify objects atop this table and binds the results to the set of *Perceived* object classes. The fifth condition computes the set of object classes that are likely to be relevant based on probabilistic reasoning, binding the result to *Needed*. The remaining lines construct the set

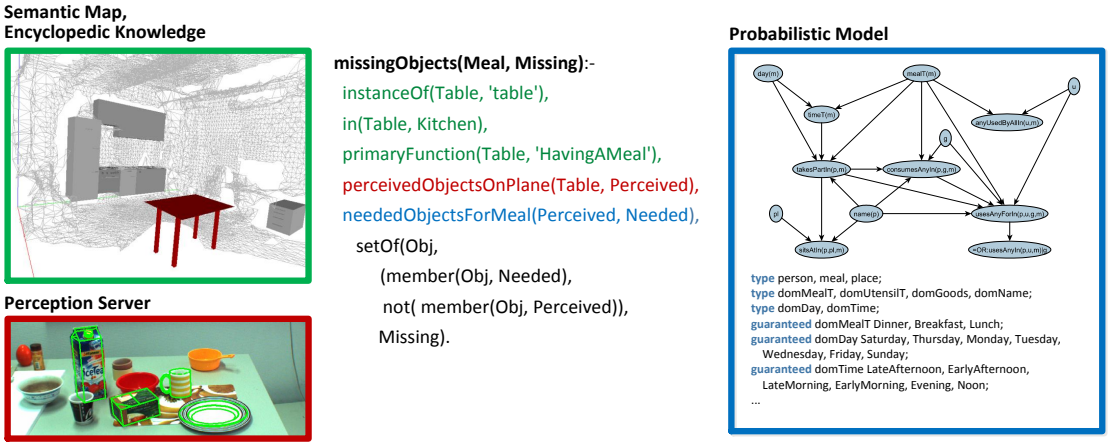


Figure 6.20: *Illustration of the computation of the predicate* missingObjects*, which is used to infer objects that are likely to be required. Its computation involves logical knowledge processing, perception routines and probabilistic inference.*

for which *missingObjects* is true by computing the set difference between the needed and perceived objects.

Using this architecture, several experiments were performed on table scenes in order to illustrate the results of the computation described above. Figure 6.21 shows four scenes to which the system was applied, indicating the setup that was presented to the robot and the objects that it contained as well as the inference results as visualised through KNOWROB.

The probabilistic computations that were necessary for the application can again be reasonably reduced to queries involving the BLN model 6.1, which was described in Section 6.2.1. Assuming the scenario that was outlined earlier, where a person partially sets a table and asks the robot to fetch additional items for the meal, the probabilistic inference task can straightforwardly be formulated. For instance, for the third scene in Figure 6.21, the query for potentially missing entities translates to the following query,

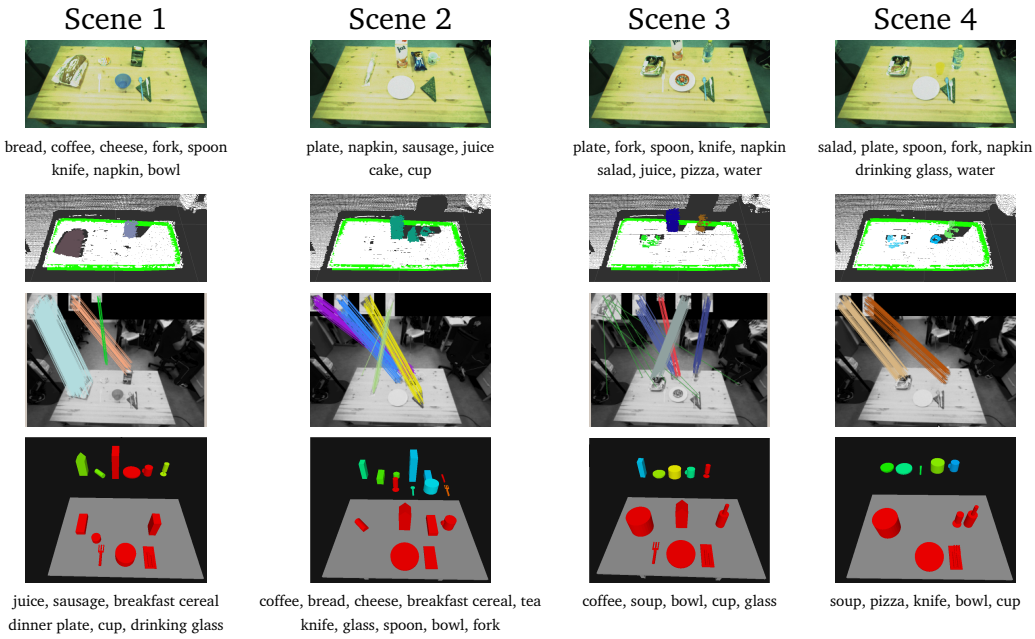


Figure 6.21: *Scenes to which the* K-CoPMan *system was applied in order to infer missing objects in the table setting scenario. The top row shows a colour image of the scene; the objects therein are listed below for clarity. The two middle rows illustrate the regions of interest extracted by the perception server and SURF matching. The bottom images show* KnowRob*'s visualisation of abstracted percepts (red objects on the table's surface) and the inferred missing objects (positioned behind the table), which are also listed below for convenience. (Pangercic, Tenorth, Jain, and Beetz, 2010)*

$$
\begin{aligned}
&P(usesAnyIn(P, ?u, M), consumesAnyIn(P, ?f, M) \mid mealT(M) = Lunch \wedge \\
&\quad usesAnyIn(P, Plate, M) \wedge usesAnyIn(P, Knife, M) \wedge \\
&\quad usesAnyIn(P, Fork, M) \wedge usesAnyIn(P, Spoon, M) \wedge \\
&\quad usesAnyIn(P, Napkin, M) \wedge consumesAnyIn(P, Salad, M) \wedge \\
&\quad consumesAnyIn(P, Pizza, M) \wedge consumesAnyIn(P, Juice, M) \wedge \\
&\quad consumesAnyIn(P, Water, M) \wedge takesPartIn(P, M)) \\
&\approx \langle\langle\ Glass: 1.00, Bowl: 0.85, Cup: 0.51, \ldots\rangle, \\
&\quad \langle\ Soup: 0.82, Coffee: 0.41, Tea: 0.14, \ldots\rangle\rangle
\end{aligned}
$$

where P is the person participating in the meal M who is assumed to be using/consuming the objects that were detected, and we ask for the probabilities of corresponding *usesAnyIn* and *consumesAnyIn* atoms. The results above (listed in order of probability) are certainly sensible, given that the presence of a spoon generally im-

plies that something like soup is likely to be consumed, and therefore that a bowl is likely to be required. Also, a glass is necessary for the drinks to be consumed.

Based on the results thus obtained, the robot's control program can initiate a plan to fetch the appropriate items, and this fetch task was also part of the real-world demonstration of the system. Notably, for the fetch task, the system can again draw upon the reasoning methods in KnowRob and ProbCog in order to infer, for instance, likely storage locations of the objects it requires.

Although the application of the K-CoPMan system that was described above is (like all robotic applications) limited in scope, the principles behind it are highly general and carry over to manifold applications – in mobile manipulation and beyond. The tight coupling of logical knowledge representation, perception, actuation and logical as well as probabilistic reasoning is an important building block of cognitive robotics, and the approach put forth in the K-CoPMan system represents a promising direction for the development of cognitive capabilities. The K-CoPMan system thus follows the principle of achieving cognitive capabilities through an integration of AI methods – an idea that was first brought into focus in the 1960s with the development of Shakey the robot (Nilsson, 1984).

From the perspective of artificial intelligence, it would, of course, be desirable for the robot to autonomously acquire data that it could later use to learn models. This problem is addressed in the next section.

6.2.4 Learning from Logged Execution Traces

In robot learning, the learning and actuation phases are often strictly decoupled, because robots typically lack an awareness of their own actions that would enable them to collect task-relevant data that could later be leveraged for learning tasks. Thus far, we have seen few efforts towards declarative specifications of the pieces of information that can potentially be made available while a robot is performing its tasks. One exception is the robot learning language RoLL (Kirsch, 2009), which allows to specify learning problems using data collected during robot plan execution. Beyond the previous applications of RoLL, which involved models such as decision trees and feed-forward neural networks, statistical relational models are an attractive

representational framework for abstract information that can be contained in the execution traces of control programs.

Robot control programs written in CRAM-PL are designed to collect data about plan execution, potentially enabling the robot to reflect upon its experience and its own behaviour (Mösenlechner et al., 2010). As previously mentioned, plans are hierarchically structured, breaking down task execution into manageable subplans which monitor their own execution and frequently involve parallel execution. Apart from the representation of the structure itself, plans need to refer to entities that are to be interacted with or are otherwise relevant to the robot's tasks. For this purpose, CRAM-PL uses *designators*, which represent symbolic descriptions of entities that can be resolved at runtime in order to determine a concrete entity that satisfies the task requirements given the robot's current beliefs about the state of the world. Therefore, highly relevant pieces of information about plans are contained within logged data about task execution and the designators involved. This section gives a brief overview of the structure of execution trace data and furthermore presents a simple model that was learnt from logged data.[5]

6.2.4.1 The Structure of Execution Trace Data

As in RoLL (Kirsch, 2009), *raw experience* collected during plan execution is differentiated from *abstract experience* that is suitable for learning. CRAM provides access to the data that is collected during execution through a Prolog knowledge base. Specifically, it provides the following predicates describing tasks:

- *goalTask*(*task*) characterises tasks that are associated with declarative goals and *task-goal*(*task*, *goal*) relates a task to its goal,
- *subtask*(*task*, *task*) establishes the hierarchy of tasks,
- *task-outcome*(*task*, *outcome*) associates a task with its final status (e.g. *SUCCEEDED* or *FAILED*), and *task-result*(*task*, *result*) associates a task with its task-specific result,
- *task-started-at*(*task*, *time*) and *task-ended-at*(*task*, *time*) provide information about a task's duration.

[5] The model was developed in collaboration with Simon Grötzinger and also appears in his Bachelor's thesis (Grötzinger, 2011).

Furthermore, several predicates characterise the entities that are relevant to tasks, in particular objects referred to via designators:

- *desig-description*(*designator*, *description*) associates a designator with its complex description, which is typically given by a set of attribute-value pairs,
- *equated-desigs(designator, designator)* is true for pairs of designators that are considered equal,
- *pose-stamped*(*pose*, *frame*, *stamp*, *origin*, *orientation*) associates with a pose identifier its defining attributes.

Access to the Prolog knowledge base of individual execution traces is provided by means of a ROS (robot operating system) service.

Using ProbCog's relational data collection library, the data contained in the Prolog knowledge bases can be easily transformed into relational data that is suitable for the learning of statistical relational models. Specifically, the data was transformed

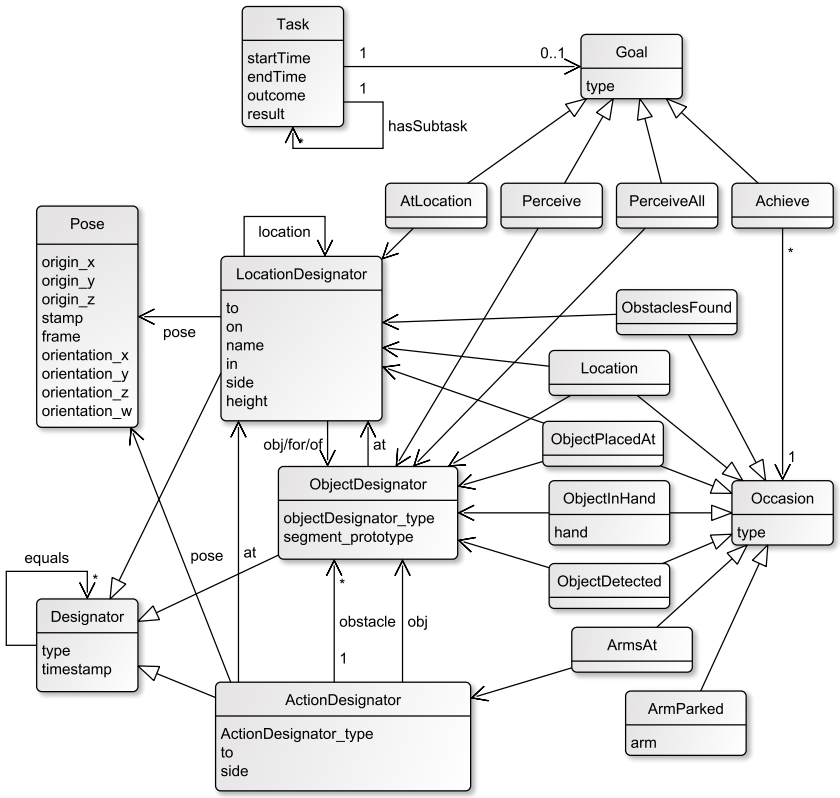


Figure 6.22: *Structure of execution trace data*

to conform to the schema shown in Figure 6.22: A *Task* may be composed of an arbitrary number of subtasks and may be associated with a *Goal*. Several types of goals are differentiated to enable associations with goal-specific entities:

- *AtLocation* goals achieve that all subtasks are performed at a particular location, which is specified in an associated *LocationDesignator*.
- *Perceive* and *PerceiveAll* goals try to detect one or all of the objects that match a given *ObjectDesignator*, returning one or more matching designators as a result.
- *Achieve* goals represent general goals that are to bring about a particular *Occasion*. The achievement of an occasions can, for instance, involve an object being placed at a particular location (*ObjectPlacedAt* occasion with associated object and location designators), repositioning the robot base (*Location* occasion with associated *LocationDesignator*), or repositioning a robot arm (*ArmsAt* occasion with associated *ActionDesignator* specifying more detailed parameters including relevant objects and obstacles – given via object designators).

Location and action designators are associated with a *Pose* that defines a 3D position and orientation relative to some coordinate frame. The influence of the poses referenced by location and action designators may be dependent on the purpose of the associated goal/task. For instance, the requirements for observing a position (*to=SEE*) are different from physically reaching a position with one's arms (*to=REACH*).[6]

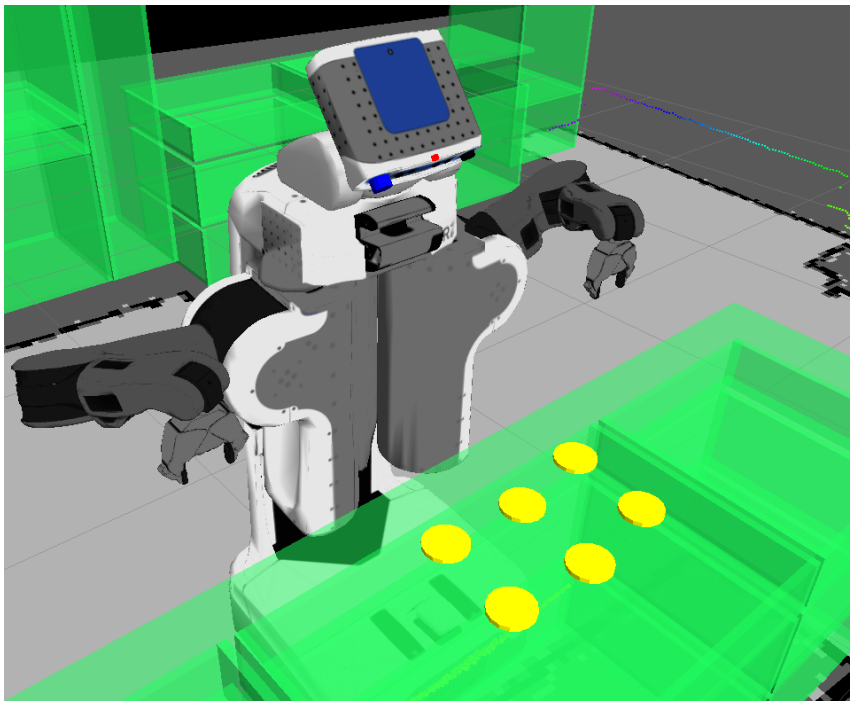
6.2.4.2 A Simple Task Model

Since the collection of comprehensive execution traces using CRAM-PL is a recent development, there is not yet a large dataset of execution traces that could be leveraged for the learning of probabilistic models. Therefore, a simple prototype is presented that illustrates the main principles. The experiment treats a basic pick and place task, which involves the PR2 robot TUM-James picking and placing a small plastic bottle from and onto a counter top. As illustrated in Figure 6.23a, the object is fetched from and placed in six predetermined areas on the surface of the counter. Twenty-seven such pick and place tasks encompassing a total of 1257 individual lower-level tasks

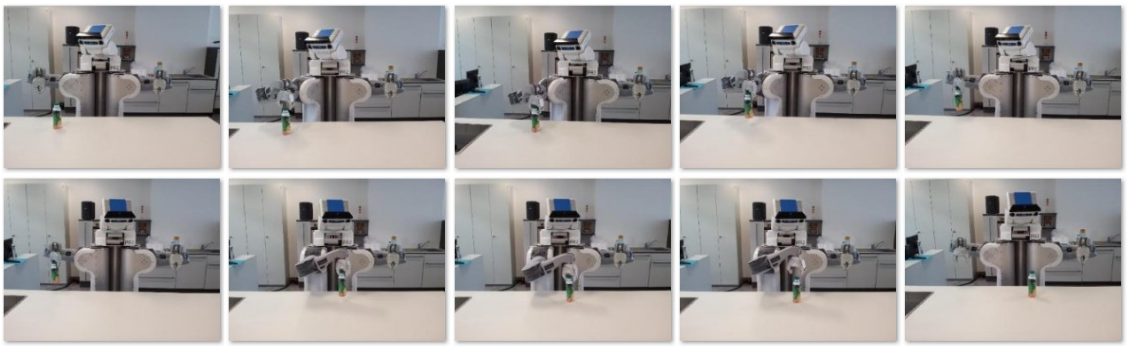
[6] In CRAM-PL, such attribute value definitions are represented as tuples, which in Lisp notation leads to particularly readable expressions such as *(a location (to REACH) ...)*.

were recorded using CRAM-PL's execution trace recording capabilities. The real robot performing a pick and place task is shown in Figure 6.23b. Whenever an action failed, the CRAM plan made the robot retry the action, sometimes with a different parametrisation (e.g. using a different destination for a place task).

Figure 6.24 visualises the task tree for a single CRAM-PL plan, generated from the relational data that was extracted. It is a minimal example in which all subtasks succeeded. The four main subtasks the plan is composed of are shown in the four



(a) Illustration of the task, highlighting the six regions on the counter top



(b) A pick and place task being performed by TUM-James as part of the experiment

Figure 6.23: *Pick and place experiment for the recording of execution trace data*

excerpts. First, the robot's arms are returned to the parking position (red), and then a perception task is triggered to determine the objects on the counter top, which, prior to the actual perception task, results in the robot being positioned appropriately (green). The perceived set of objects is returned as a list of object designators. Next, the robot performs the pick up action, which necessitates appropriately repositioning its arm for the grasp (light blue). The final subtask places the object in one of the other designated regions on the counter top (dark blue).

To take the specific properties of the pick and place task into consideration, the data structure that is shown in Figure 6.22 is augmented with additional high-level data. Thus *high-level tasks* are introduced, which can either be pick or place tasks or combinations of both (pick-and-place). High-level pick tasks are associated with tasks

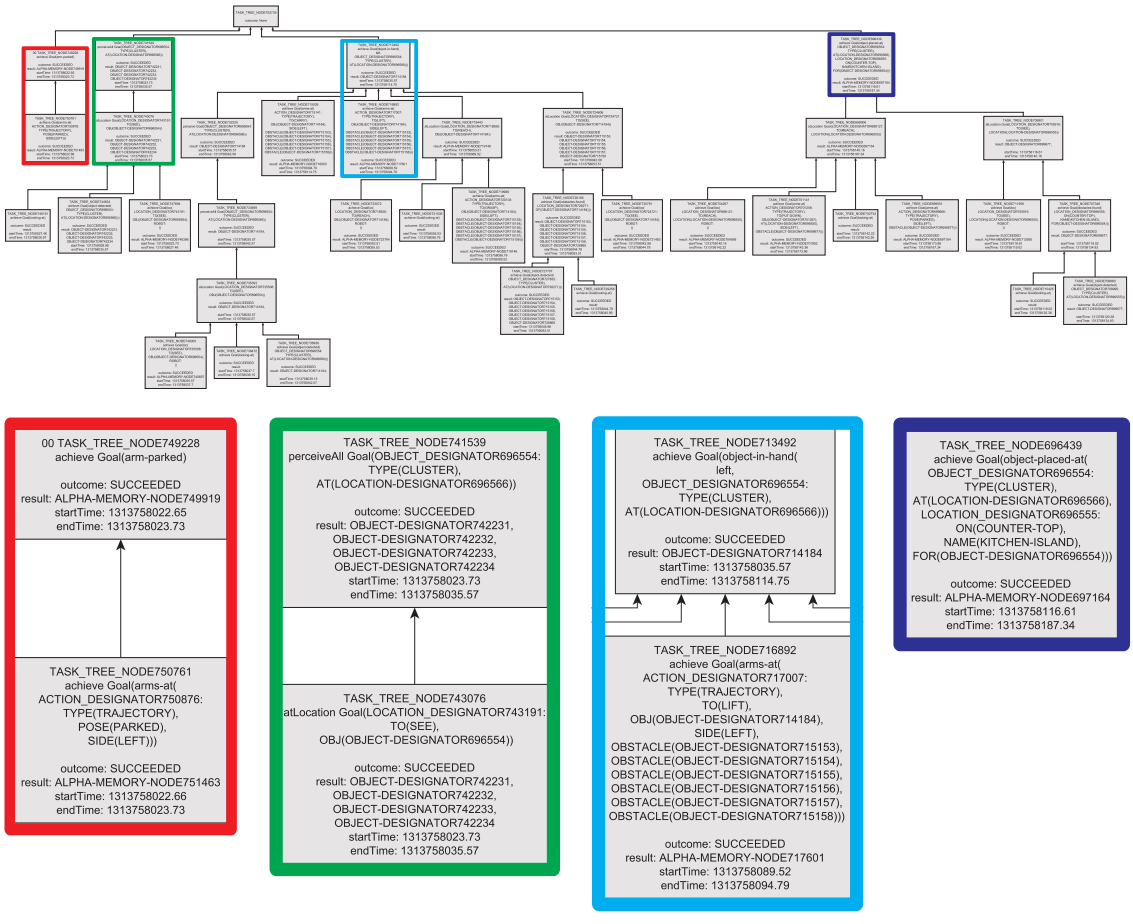
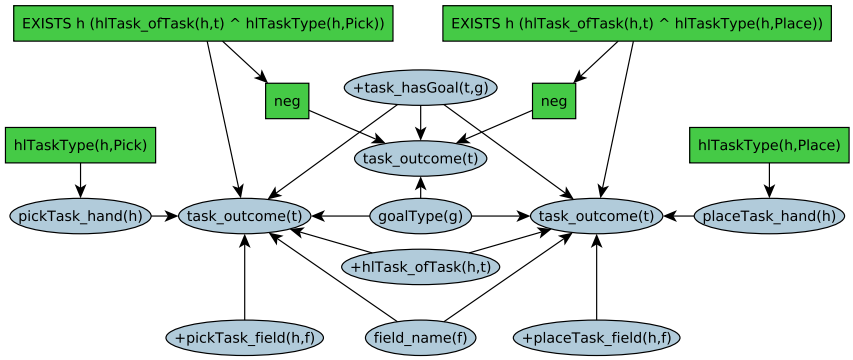


Figure 6.24: *Visualisation of execution trace data recorded by a* CRAM-PL *control program which performed one pick and place task*

that achieve an *ObjectInHand* occasion. Likewise, place tasks are associated with tasks that achieve an *ObjectPlacedAt* occasion. Both pick and place tasks are associated with one of the six fields, which is identified based on the spatial information given by the respective designators/poses. The resulting model's declarations and fragment network structure are shown in BLN 6.5.

This model can subsequently be used to reason about the individual types of tasks. For instance, we can extract the probability of success depending on the type of the associated goal:




```
1  type field, task, hlTask, goal;
2  type domField_name, domHlTaskType, domArmParkedOccasion_arm, domGoalType, domTask_outcome;
3  guaranteed domField_name FrontLeft, FrontMiddle, FrontRight, BackLeft, BackMiddle, BackRight;
4  guaranteed domHlTaskType Place, Pick, PandP;
5  guaranteed domArmParkedOccasion_arm Left, Right;
6  guaranteed domGoalType Achieve, AtLocation, PerceiveAll, Perceive;
7  guaranteed domTask_outcome SUCCEEDED, FAILED;
8  random domField_name field_name(field);
9  random domTask_outcome task_outcome(task);
10 random domArmParkedOccasion_arm pickTask_hand(hlTask);
11 random domArmParkedOccasion_arm placeTask_hand(hlTask);
12 random domGoalType goalType(goal);
13 logical domHlTaskType hlTaskType(hlTask);
14 logical boolean pickTask_field(hlTask, field);
15 logical boolean placeTask_field(hlTask, field);
16 logical boolean hlTask_ofTask(hlTask, task);
17 logical boolean task_hasGoal(task, goal);
18 relationKey task_hasGoal(t,_);
19 relationKey hlTask_ofTask(_,t);
20 relationKey pickTask_field(h,_);
21 relationKey placeTask_field(h,_);
```

BLN 6.5: *Model of pick and place experiments learnt from execution traces*

$$
\begin{aligned}
&P(task\_outcome(T) \mid task\_hasGoal(T, G) = True \ \wedge \ goalType(G) = \textbf{\textit{Achieve}}) \\
&\quad \approx \langle \ SUCCEEDED: 0.95, FAILED: 0.05 \ \rangle \\
&P(task\_outcome(T) \mid task\_hasGoal(T, G) = True \wedge goalType(G) = \textbf{\textit{AtLocation}}) \\
&\quad \approx \langle \ SUCCEEDED: 0.93, FAILED: 0.07 \ \rangle \\
&P(task\_outcome(T) \mid task\_hasGoal(T, G) = True \wedge goalType(G) = \textbf{\textit{Perceive}}) \\
&\quad \approx \langle \ SUCCEEDED: 1.00, FAILED: 0.00 \ \rangle \\
&P(task\_outcome(T) \mid task\_hasGoal(T, G) = True \wedge goalType(G) = \textbf{\textit{PerceiveAll}}) \\
&\quad \approx \langle \ SUCCEEDED: 1.00, FAILED: 0.00 \ \rangle
\end{aligned}
$$

For a pick or place task, we can reason about the probability of success depending the field with which the task was associated or the arm/gripper used by the robot:

$$
\begin{aligned}
&P(task\_outcome(T) \mid task\_hasGoal(T, G) = True \wedge \ hlTask\_ofTask(H, T) \wedge \\
&\qquad hlTaskType(H) = \textbf{\textit{Pick}} \wedge pickTask\_field(H, F) = True) \\
&\quad \approx \langle \ SUCCEEDED: 0.85, FAILED: 0.15 \ \rangle \\
&P(task\_outcome(T) \mid task\_hasGoal(T, G) = True \wedge \ hlTask\_ofTask(H, T) \wedge \\
&\qquad hlTaskType(H) = \textbf{\textit{Place}} \wedge placeTask\_field(H, F) = True) \\
&\quad \approx \langle \ SUCCEEDED: 0.80, FAILED: 0.20 \ \rangle \\
&P(task\_outcome(T) \mid task\_hasGoal(T, G) = True \wedge hlTask\_ofTask(H, T) = True \ \wedge \\
&\qquad hlTaskType(H) = \textbf{\textit{Pick}} \ \wedge \ pickTask\_field(H, F) = True \ \wedge \\
&\qquad pickTask\_hand(H) = \textbf{\textit{Right}} \ \wedge \ field\_name(F) = \textbf{\textit{FrontLeft}}) \\
&\quad \approx \langle \ SUCCEEDED: 0.60, FAILED: 0.40 \ \rangle \\
&P(task\_outcome(T) \mid task\_hasGoal(T, G) = True \wedge hlTask\_ofTask(H, T) = True \ \wedge \\
&\qquad hlTaskType(H) = \textbf{\textit{Pick}} \ \wedge \ pickTask\_field(H, F) = True \ \wedge \\
&\qquad pickTask\_hand(H) = \textbf{\textit{Left}} \ \wedge \ field\_name(F) = \textbf{\textit{FrontLeft}}) \\
&\quad \approx \langle \ SUCCEEDED: 1.00, FAILED: 0.00 \ \rangle \\
&P(task\_outcome(T) \mid task\_hasGoal(T, G) = True \wedge hlTask\_ofTask(H, T) = True \ \wedge \\
&\qquad hlTaskType(H) = \textbf{\textit{Place}} \ \wedge \ placeTask\_field(H, F) = True \ \wedge \\
&\qquad pickTask\_hand(H) = \textbf{\textit{Right}} \ \wedge \ field\_name(F) = \textbf{\textit{BackMiddle}}) \\
&\quad \approx \langle \ SUCCEEDED: 0.80, FAILED: 0.20 \ \rangle \\
&P(task\_outcome(T) \mid task\_hasGoal(T, G) = True \wedge hlTask\_ofTask(H, T) = True \ \wedge \\
&\qquad hlTaskType(H) = \textbf{\textit{Place}} \ \wedge \ placeTask\_field(H, F) = True \ \wedge \\
&\qquad pickTask\_hand(H) = \textbf{\textit{Left}} \ \wedge \ field\_name(F) = \textbf{\textit{FrontRight}}) \\
&\quad \approx \langle \ SUCCEEDED: 1.00, FAILED: 0.00 \ \rangle
\end{aligned}
$$

The model can also be applied to determine the abstract parametrisation of an action. For instance, it can be used to determine the arm/gripper to use given that a pick or place task is to succeed given the field involved,

$$
\begin{aligned}
&P(pickTask\_hand(H) \mid task\_hasGoal(T,G) = True \ \wedge \\
&\qquad task\_outcome(T) = SUCCEEDED \ \wedge \ hlTask\_ofTask(H,T) = True \ \wedge \\
&\qquad hlTaskType(H) = \textbf{\textit{Pick}} \ \wedge \ pickTask\_field(H,F) = True \ \wedge \\
&\qquad field\_name(F) = \textbf{\textit{FrontLeft}}) \\
&\quad \approx \langle \ Left: 0.58, Right: 0.42 \ \rangle
\end{aligned}
$$

$$
\begin{aligned}
&P(\mathit{placeTask\_hand}(H) \mid \mathit{task\_hasGoal}(T,G) = \mathit{True} \wedge \\
&\qquad \mathit{task\_outcome}(T) = \mathit{SUCCEEDED} \wedge \mathit{hlTask\_ofTask}(H,T) = \mathit{True} \wedge \\
&\qquad \mathit{hlTaskType}(H) = \textbf{\textit{Place}} \wedge \mathit{pickTask\_field}(H,F) = \mathit{True} \wedge \\
&\qquad \mathit{field\_name}(F) = \textbf{\textit{FrontRight}}) \\
&\quad \approx \langle\, \mathit{Left}\text{: } 0.51, \mathit{Right}\text{: } 0.49 \,\rangle
\end{aligned}
$$

or the fields for which pick or place tasks succeed:

$$
\begin{aligned}
&P(\mathit{field\_name}(F) \mid \mathit{task\_hasGoal}(T, G) = \mathit{True} \wedge \\
&\qquad \mathit{task\_outcome}(T) = \mathit{SUCCEEDED} \wedge \mathit{hlTask\_ofTask}(H, T) = \mathit{True} \wedge \\
&\qquad \mathit{hlTaskType}(H) = \textbf{\textit{Pick}} \wedge \mathit{placeTask\_field}(H, F) = \mathit{True} \wedge \\
&\qquad \mathit{placeTask\_hand}(H) = \textbf{\textit{Left}}) \\
&\quad \approx \langle\, \mathit{FrontLeft}\text{: } 0.21, \mathit{FrontMiddle}\text{: } 0.14, \mathit{FrontRight}\text{: } 0.21 \\
&\qquad \mathit{BackLeft}\text{: } 0.21, \mathit{BackMiddle}\text{: } 0.21, \mathit{BackRight}\text{: } 0.00 \,\rangle \\
&P(\mathit{field\_name}(F) \mid \mathit{task\_hasGoal}(T, G) = \mathit{True} \wedge \\
&\qquad \mathit{task\_outcome}(T) = \mathit{SUCCEEDED} \wedge \mathit{hlTask\_ofTask}(H, T) = \mathit{True} \wedge \\
&\qquad \mathit{hlTaskType}(H) = \textbf{\textit{Place}} \wedge \mathit{placeTask\_field}(H, F) = \mathit{True} \wedge \\
&\qquad \mathit{placeTask\_hand}(H) = \textbf{\textit{Right}}) \\
&\quad \approx \langle\, \mathit{FrontLeft}\text{: } 0.25, \mathit{FrontMiddle}\text{: } 0.18, \mathit{FrontRight}\text{: } 0.25, \\
&\qquad \mathit{BackLeft}\text{: } 0.12, \mathit{BackMiddle}\text{: } 0.20, \mathit{BackRight}\text{: } 0.00 \,\rangle
\end{aligned}
$$

Of course, the size of the training database in this experiment was too small to permit truly informative predictions, and the task was too simple for the full expressiveness of relational models to be leveraged. However, as more data is collected on a greater variety of tasks, the resulting models could become a valuable resource for robots and enable them to make informed decisions online that are based on their own past experience. Indeed, the autonomous learning of statistical relational models from execution trace data is a most promising direction for future work.

6.3 Plan Assessment in Manufacturing

In today's world of ubiquitous standardisation, people seek to distinguish themselves from the masses. In reaction to this desire for individualism, the industry has adopted the notion of *mass customisation*, i.e. the bulk production of highly customised, individualised products that are realised through flexible manufacturing processes that can adapt themselves in order to produce countless variations of a particular type of product. This production paradigm is typically made possible through computer-aided techniques that help to sustain profitability. Indeed, the widespread deployment of AI techniques such as planning and scheduling for the generation of production plans and the allocation of resources may well become a necessity if manufacturers are to efficiently organise their production processes in the light of the increasingly unmanageable complexity of the production process.

Owing to the flexible manner in which production plans need to be generated, mass customisation is likely to result in a multitude of problematic conditions that could potentially occur during production, and it may be economical to flexibly treat these conditions rather than to preclude their occurrence entirely by implementing a fail-safe production pipeline. A system is envisaged that employs planning and scheduling capabilities based on very abstract models to keep planning and scheduling tractable. Observations made at execution time, which were not available during the planning/scheduling phase, may indicate that certain factory stations are operating outside of normal parameters, potentially causing the production of any products allocated to them to fail. Given these online observations, one can, based on probabilistic models of station behaviour, try to predict the probability with which the production of individual items will succeed and to furthermore diagnose the most probable state of the entire production plant at any point in time. The combination of these two inference problems has been termed the *plan assessment problem* (Maier et al., 2009). The problem arises in scenarios where the system is sufficiently rigid to allow offline planning of actions to be feasible but is, at the same time, not free from uncertainties. In an online phase, pre-planned actions are executed in order to achieve given production goals, whose reachability needs to be continuously monitored and evaluated given the observations made. Solving the plan assessment problem is a precondition for informed decision-making in a production plant and therefore an important

step in the realisation of what might rightfully be called a “cognitive” factory in the future.

In the applications described below, only the plan assessment problem itself is considered, not the planning and scheduling phase that precedes it. In particular, the prediction problem of computing the probability with which production goals will be reached is addressed. We shall assume to be given a schedule $\mathscr{S}$ that defines, for a particular set of products to be manufactured, the allocation of the factory's stations to concrete products for each point in time.

6.3.1 A Statistical Relational Model of Production Plant Behaviour

As an exemplary production plant, we shall consider the demonstration scenario of the Cognitive Factory that is studied by the CoTeSys cluster of excellence (Zäh et al., 2009).[7] The test-bed, an iCim3000-based Festo Flexible Manufacturing System, consists of conveyor transports, storage, machining and assembly. In particular, the production of toy mazes that are composed of an alloy base plate, a small metal ball and an acrylic glass cover fixed by metal pins (see Figure 6.25 left) is studied. The manufacturing process involves cutting the labyrinth groove into the base plate, drilling the fixation holes, inserting the ball into the labyrinth, placing the glass cover onto the base plate and finally pushing the pins in place to fixate the cover.

This production plant serves as the basis for hypothetical example scenarios, where a planning and scheduling system determines the automated production of toy mazes. In the abstract, simplified scenarios, we shall disregard transportation processes and focus exclusively on the machining (i.e. cutting and drilling) and assembly actions, where a plant can consist of any number of machining and assembly stations that perform these actions.

During production, products can get flawed as a result of being worked on by faulty stations. However, the observations that are made by the plant do not immediately indicate the root cause of a fault. For instance, when the pins are pushed into the holes in order to fixate the glass cover of the maze, the assembly station measures the force that is required for the operation and triggers an alarm if too much force

[7] http://www.cotesys.de/research/demonstration-scenarios.html

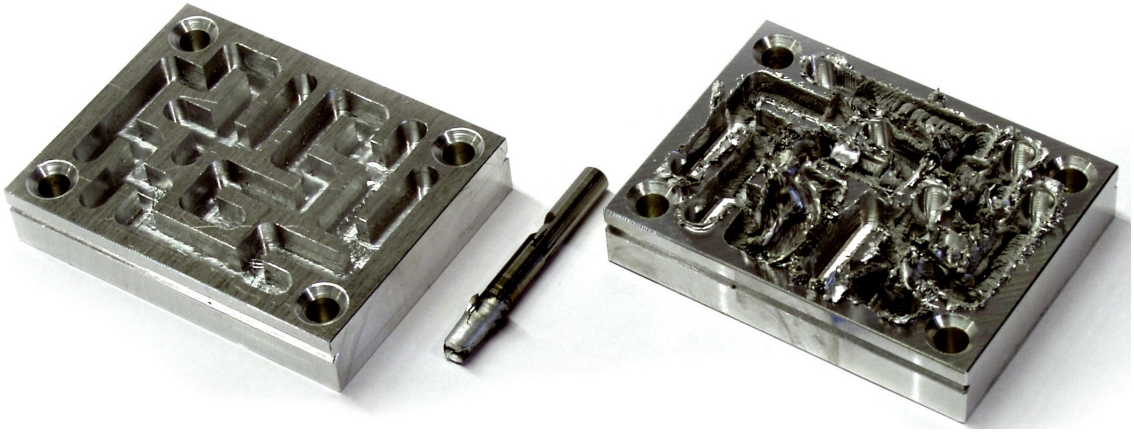
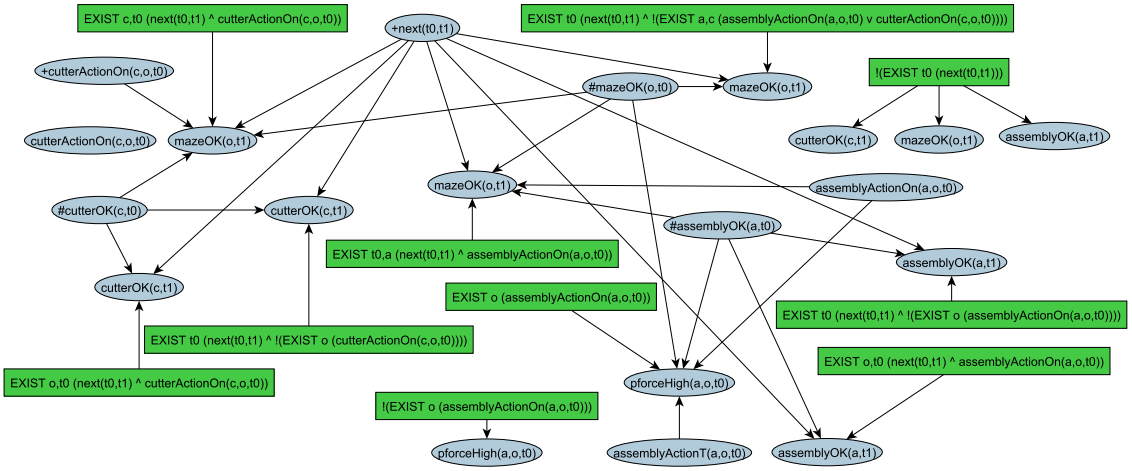


Figure 6.25: *The fully cut base plate of the toy maze that is produced in the example factory (left) and the effect of a worn out cutter on its grooves (right).*

is applied. At least two reasons for the alarm being triggered can be differentiated. First, the assembly station's calibration could be off, causing a misalignment of the gripper holding the pin and the baseplate's holes. Second, the cutter of the machining station that previously processed the baseplate could be worn out, causing the holes that were drilled to be malformed (see Figure 6.25 right).

BLN 6.6 is a Bayesian logic network for this manufacturing scenario (Maier, Jain, Waldherr, and Sachenbacher, 2010). The abstract model generalises over the set of time steps (type *time*), machining stations (type *cutter*) and assembly stations (*assembly*). In the fixed domain *asyAction*, we differentiate the assembly actions of attaching the glass cover and of inserting the pins for fixation. The schedule $\mathscr{S}$ of manufacturing actions determines the sequence of time steps (*next*) and the allocation of products to stations (*cutterActionOn*, *assemblyActionOn*). The corresponding predicates are declared as logical with relation keys that allow us to establish connections between entities via functional links. The actual set of random variables in an instantiated model comprises Boolean variables describing the state of products (*mazeOK*) and stations (*assemblyOK*, *cutterOK*) as well as Boolean variables for the observation of alarms caused by excessive forces being measured at assembly stations (*pforceHigh*).




```
1  type cutter, assembly, product, time, asyAction;
2  guaranteed asyAction AssemblePins, AssembleCover;
3  logical Boolean next(time, time);
4  logical Boolean cutterActionOn(cutter, product, time);
5  logical Boolean assemblyActionOn(assembly, product, time);
6  logical asyAction assemblyActionT(assembly, product, time);
7  relationKey next(t, _);
8  relationKey next(_, t);
9  relationKey cutterActionOn(_, o, t);
10 relationKey assemblyActionOn(_, o, t);
11 random Boolean cutterOK(cutter, time);
12 random Boolean assemblyOK(assembly, time);
13 random Boolean mazeOK(product, time);
14 random Boolean pforceHigh(assembly, product, time);
```

BLN 6.6: *Bayesian logic network that models a class of manufacturing scenarios with machining (cutter) and assembly stations as well as their interaction with maze products.*

In the top right hand corner of the fragment network, the initial state of assembly stations, machining stations and maze products is modelled in three fragments that have no parents beyond the precondition for the initial state. In order to adapt the ground models to the input that is given by the schedule $\mathscr{S}$, the fragment network makes extensive use of precondition nodes. For instance, the temporal evolution of the state of the product is influenced by the stations that are used to process it: Depending on the station that worked on a particular product (maze) in the previous time step, the set of parents for the applicable *mazeOK* fragment includes the state of the respective machining or assembly stations (cf. the two *mazeOK(o,t1)* fragments in BLN 6.6). Furthermore, the temporal evolution of the state of factory stations is modelled in four fragments – two for each type of station, because the case where a

station is idle is differentiated from the case where it processes a product. Particularly machining stations are subject to higher degradation if they are used to process a product.

The model can be instantiated for an arbitrary factory configuration of assembly and machining stations and for an arbitrary schedule that is to produce any given number of products over the course of a specific number of time steps. The structure of the ground models that result are heavily dependent on the schedule $\mathscr{S}$, for the template model will instantiate different conditional distributions in each time step as the allocation of products to factory stations changes.

6.3.2 Confidence Bounds for Approximate Plan Assessment

We can use the approximate, sampling-based inference techniques described in Section 5.4 in order to solve the plan assessment problem of inferring the probability with which the production of individual products will succeed. In the manufacturing domain, there are high demands on the reliability of the inference results that are obtained and we thus make use of the confidence interval-based approach from Section 5.4.5.2.

We consider three (hypothetical) scenarios that involve variations of the same schedule and for which different sets of observations are given. In each case, the factory is equipped with two machining stations (*Mach0* and *Mach1*) and one assembly station (*Assy0*). The scenarios cover the production of up to four mazes (*Maze0*–*Maze3*) in up to twelve time steps (*T1*–*T12*). *Maze0* and *Maze2* are cut on *Mach0*, while *Maze1* and *Maze3* are cut on *Mach1*. The production of *Maze0/1/2/3* is completed by *T5/9/7/12*. The scenarios are further characterised as follows:

- *Scenario 1*: *Mach0* is faulty (but this is not known to the system). The production of *Maze3* is omitted, thus *T9* is the last time point considered. Observations were made up to time *T4*, and at *T4*, a force alarm is observed.
- *Scenario 2*: *Assy0* is faulty (but this is not known to the system). Observations were made up to time *T8*. The pin assembly is done at time *T4*, *T6* and *T8* for *Maze0*, *Maze2* and *Maze1* respectively, and a force alarm is consequently observed at all three points in time.

- *Scenario 3*: *Mach0* is faulty (but this is not known to the system). As in scenario 2, observations are made up to time *T8*, but an alarm is observed only at *T4* and *T6* but not at time *T8*, since *Maze1* was machined on the fully functional *Mach1*.

Several sampling algorithms were run on the instantiations of the BLN implied by the three scenarios described above in order to compute confidence intervals for the success probabilities of the products. Results obtained using likelihood weighting (after 10,000 samples) are summarised in Table 6.3. Scenario 1 is not entirely conclusive: While the production of *Maze0* and *Maze2* will almost certainly fail, the model is uncertain with respect to the outcome of *Maze1*. In scenario 2, all products – including the yet unfinished *Maze3* – are very likely to be damaged, which is certainly sensible given the fact that the observations do strongly indicate that the only assembly station, through which all products must pass, is faulty. In scenario 3, the production of the yet unfinished *Maze3* is very likely to succeed, since, given the observations, it is very likely that only *Mach0* is faulty.

	Scenario 1	Scenario 2	Scenario 3
mazeOK(M0,T5)	[0.000, 0.000]	[0.000, 0.003]	[0.000, 0.003]
mazeOK(M1,T9)	[0.587, 0.606]	[0.000, 0.003]	[0.997, 1.000]
mazeOK(M2,T7)	[0.035, 0.042]	[0.000, 0.003]	[0.000, 0.003]
mazeOK(M3,T12)	–	[0.002, 0.011]	[0.911, 0.922]
cutterOK(*Mach0*,T9)	[0.283, 0.300]	[0.635, 0.654]	[0.000, 0.001]
cutterOK(*Mach1*,T9)	[0.849, 0.862]	[0.653, 0.671]	[0.931, 0.941]
assemblyOK(Assy0,T9)	[0.671, 0.690]	[0.197, 0.213]	[0.989, 0.993]

Table 6.3: *Plan assessment results after running likelihood weighting for 10,000 steps. Intervals span a coverage probability of* $\gamma = 0.95$.

From these results, the factory could sensibly derive appropriate actions: In scenario 2, production should be halted completely, whereas in scenario 3, the production of *Maze3* should be continued. In uncertain cases such as scenario 1, it may be most sensible to actively gather further information, i.e. to adapt the schedule in order to unambiguously determine the stations that are at fault (Kuhn et al., 2008).

Given that sampling methods allow to obtain any-time approximations, one can also investigate the estimated quality of the approximations that is obtained with varying numbers of samples – as reflected in the size of the intervals. Table 6.4 compares the intervals obtained using likelihood weighting after 100, 2500 and 10000 samples respectively with the exact results obtained using variable elimination. As the number

γ		Number of samples			Exact
		100	2500	10000	
0.95	mazeOK(M2,T7)	$[0.002, 0.054]$	$[0.035, 0.051]$	$[0.035, 0.042]$	0.0387
	mazeOK(M1,T9)	$[0.593, 0.772]$	$[0.591, 0.629]$	$[0.587, 0.606]$	0.6041
	mazeOK(M0,T5)	$[0.000, 0.029]$	$[0.000, 0.001]$	$[0.000, 0.000]$	0.0000
0.999	mazeOK(M2,T7)	$[0.010, 0.162]$	$[0.030, 0.057]$	$[0.035, 0.048]$	0.0387
	mazeOK(M1,T9)	$[0.359, 0.677]$	$[0.573, 0.637]$	$[0.594, 0.626]$	0.6041
	mazeOK(M0,T5)	$[0.000, 0.066]$	$[0.000, 0.003]$	$[0.000, 0.001]$	0.0000
runtime ($\gamma = 0.950$)		0.06 s	1.61 s	6.24 s	58.76 s
runtime ($\gamma = 0.999$)		0.06 s	1.52 s	6.00 s	58.76 s

Table 6.4: *Confidence intervals for the success probabilities of mazes in scenario 1 obtained with likelihood weighting after having drawn 100, 2500 and 10000 samples respectively. The runtime variations are due only to the fact that sampling is randomised. Exact results were obtained using variable elimination. (Maier, Jain, Waldherr, and Sachenbacher, 2010)*

of samples increases, we observe that, for each posterior, the range spanned by the interval quickly decreases and converges to a narrow interval around the true posterior. Sufficiently accurate results could be obtained within very few seconds, while the computation of exact results via variable elimination took almost one minute.

The impact of different sampling techniques was also investigated. Likelihood weighting, backward simulation and SampleSearch (with chronological backtracking) were applied to each of the scenarios, using the confidence interval size-based termination criterion. The sampling process was stopped when the interval size for confidence level $\gamma = 0.95$ dropped below $s_{\max}$ for $s_{\max} \in \{0.025, 0.01, 0.0025\}$. For each experiment, the number of samples that was required and the time taken is listed in Table 6.5. We observe that even though the three scenarios are not entirely dissimilar, performance varies widely. While even a highly accurate estimate for scenario 2 can be computed within as few as two seconds, it took even the most efficient method around 7 minutes to gather sufficiently many samples for scenario 3. SampleSearch seems to work best on scenarios 1 and 2, while the simpler likelihood weighting performs best on scenario 3.

$s_{\max}$	likelihood weighting			backward simulation			SampleSearch		
0.0250	5900	320	2020	5820	200	–	6180	200	1380
	3.61 s	0.90 s	3.82 s	1.23 s	5.44 s	–	0.84 s	0.06 s	13.27 s
0.0100	36800	1000	11800	37280	300	–	38440	660	5080
	22.42 s	2.67 s	21.47 s	7.00 s	7.56 s	–	4.76 s	0.16 s	42.08 s
0.0025	588020	17420	185820	588860	1200	–	614700	6460	181320
	384.38 s	50.06 s	362.08 s	111.95 s	31.75 s	–	79.70 s	1.40 s	1551.25 s

Table 6.5: *Number of samples and runtime required to reach, for confidence level $\gamma = 0.95$, a confidence interval of size at most $s_{\max}$ for all query variables. Three sampling-based algorithms are compared on scenarios 1/2/3 (sub-columns). Backward simulation was unable to produce any approximations for scenario 3, because the problem was, apparently, too ill-conditioned. (Maier, Jain, Waldherr, and Sachenbacher, 2010)*

6.3.3 Translating AI Engineering Models into Statistical Relational Models

While the framework of Bayesian logic networks is highly flexible, the BLN representation language may not be the most practical formalism in an engineering context. For the specification of production facilities and the modelling of manufacturing processes, languages such as the *reactive model-based programming language* (RMPL) (Williams et al., 2001) have been specifically developed to meet the needs of engineers. RMPL addresses the challenge of modelling and monitoring systems composed of complex components, merging constructs from synchronous programming languages, qualitative modelling, Markov models and constraint programming. The operational semantics of RMPL are specified via a mapping to hierarchical constraint-based hidden Markov models (Williams et al., 2001), which, in turn, can be compactly encoded as probabilistic hierarchical constraint automata (PHCA) (Mikaelian et al., 2005). Unfortunately, the existing methods developed specifically for PHCA focus exclusively on diagnosis and do not support the computation of posterior marginals that is required for the plan assessment problem. In the following, I will outline an automated translation from PHCA to BLNs, which was developed in collaboration with Paul Maier in order to fill this computational gap (Maier, Jain, and Sachenbacher, 2011).

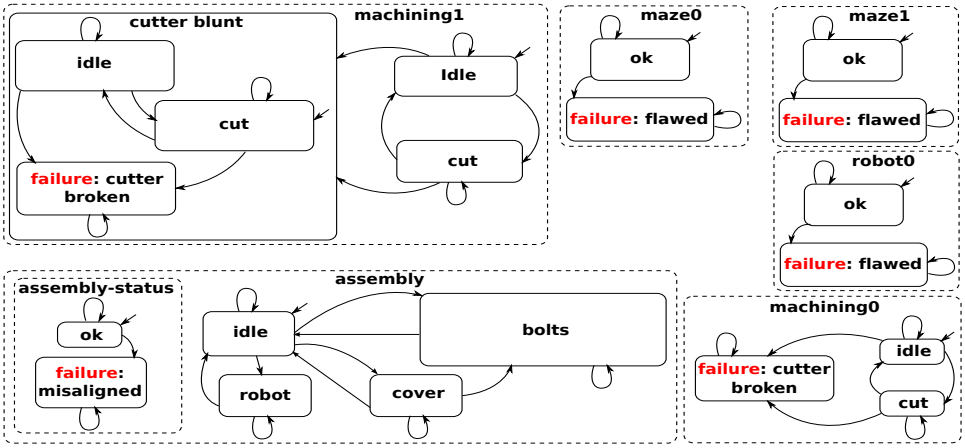
In essence, a *probabilistic hierarchical constraint automaton* (PHCA) is a probabilistic template model that is capable of describing the evolution of several coupled temporal chains. The automaton states of a PHCA are called *locations*; *primitive loca-*

tions are differentiated from *composite locations*, which can be hierarchically decomposed. The behaviour of a composite location is modelled within a sub-automaton, such that whenever a composite location is transitioned to, the system's behaviour is simultaneously characterised by the sub-automaton at a more fine-grained level. Every sub-automaton is itself a PHCA, which can potentially be recursively decomposed further. In addition to locations, PHCA models use a set of *variables* Π which represents *observations*, external *commands* and dependent internal variables. These variables serve, in particular, to establish interactions between system components. For a particular set of time steps and system components, a PHCA can be procedurally unfolded by reproducing its template components appropriately. At any point in time, the state of the automaton is given by a *marking* of its locations. The system's non-deterministic behaviour is captured by two probabilistic components. The first is a probability distribution over subsets of the PHCA's set of locations that characterises the initial marking for the first time step. The second represents, for each location, a probability distribution over transitions leading to other locations that are consistent with the current state (marking). Consistency is captured in *transition guard constraints* that are represented as finite domain constraints over the set of variables Π. Furthermore, a set of *behaviour constraints* captures admissible interactions between variable settings and location markings.

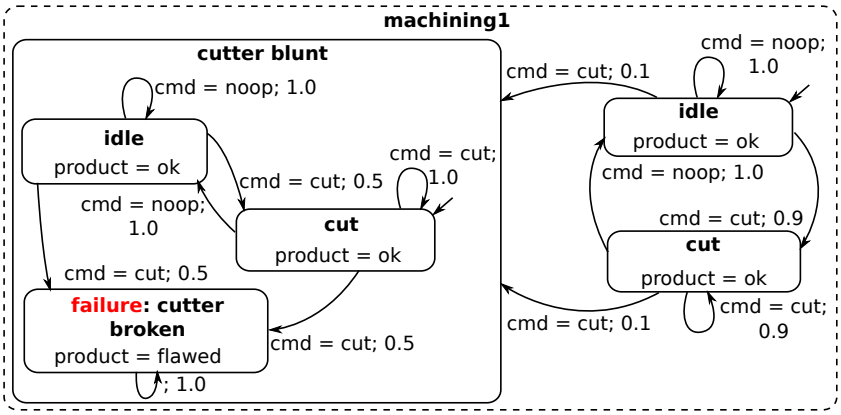
An illustration of a PHCA is given in Figure 6.26. The model is a variation of the scenario that was previously described in Section 6.3.1 with the concrete extension described in Section 6.3.2: It is concerned with the same factory consisting of two machining and one assembly station, but in addition to the maze products, it now also considers the production of a robot arm.

In the application thus considered, the command variables of the PHCA are always given as evidence, as they stand for operations that are to be carried out on the factory's stations and are, therefore, fully specified by the schedule $\mathscr{S}$. Consequently, some of the PHCA's transition guard constraints may become unsatisfiable and the corresponding transitions can be removed from the instantiated model entirely.

The translation from PHCA to BLNs is particularly natural, since BLNs provide support for both conditional distributions and logical constraints. The two probabilistic components of a PHCA can be directly mapped to the former, and finite domain constraints can be mapped to the latter. Likewise, any information on commands and



(a) The full automaton



(b) A detailed image of the automaton for "machining1", which contains the composite location "cutter blunt"

Figure 6.26: *A probabilistic hierarchical constraint automaton (PHCA) for our example factory. The full automaton consists of six components (i.e. sub-automata of the top-level PHCA), three of which are concerned with the behaviour of the factory's stations (two machining, one assembly station). Three further components model the evolution of products (two mazes, one robot arm). Note that the representation does not contain an observation model. The observations are modelled only in the PHCA's constraints, where one can state that an observation is consistent with particular states. The observation model is thus quasi-deterministic. (Maier, Jain, and Sachenbacher, 2011)*

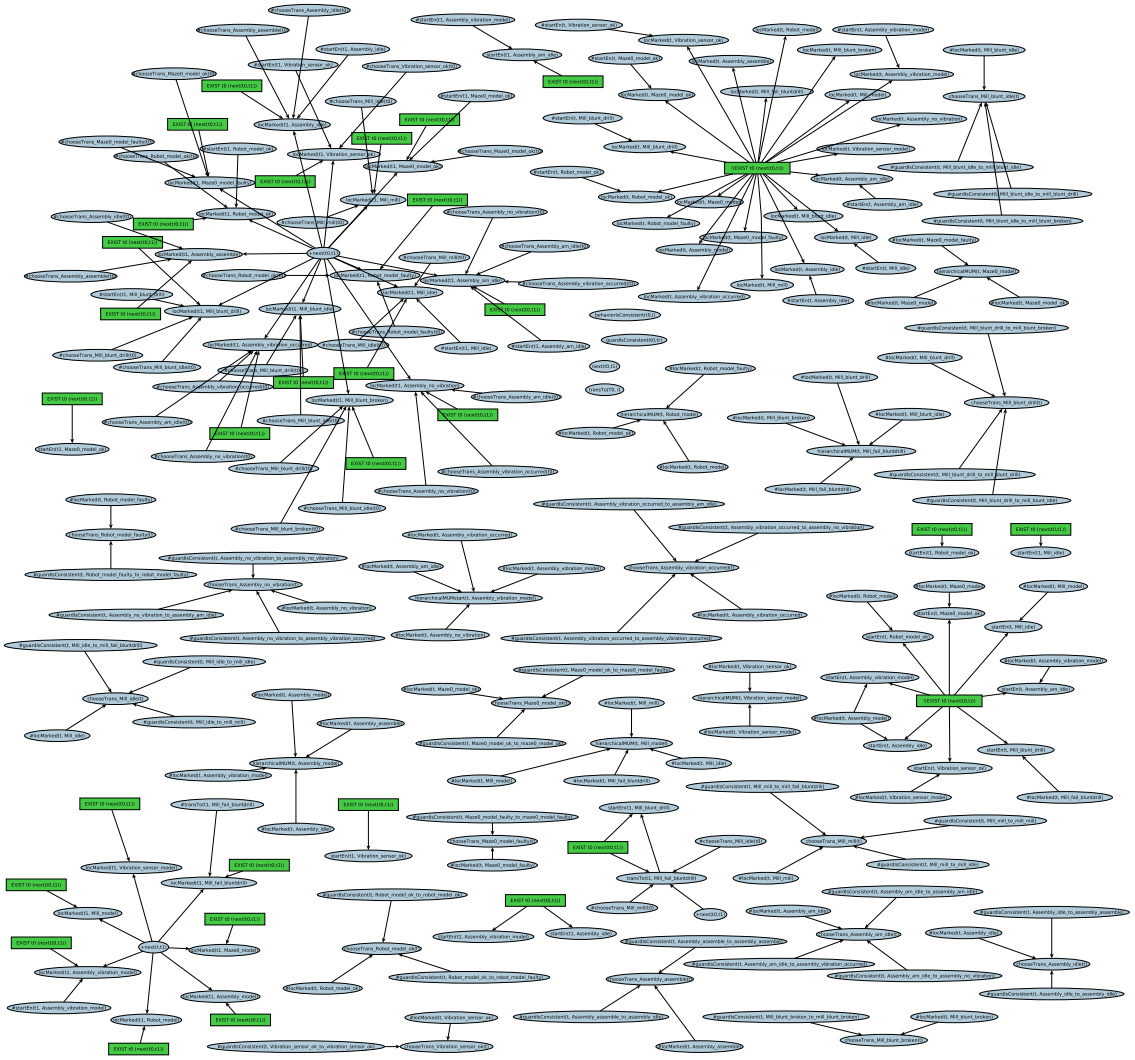
observations can be directly mapped to an evidence database. Through optimisations in the BLN instantiation process, a high level of compactness can be achieved, since

constraints and random variables that are known to be irrelevant a priori will not be considered in the ground model.

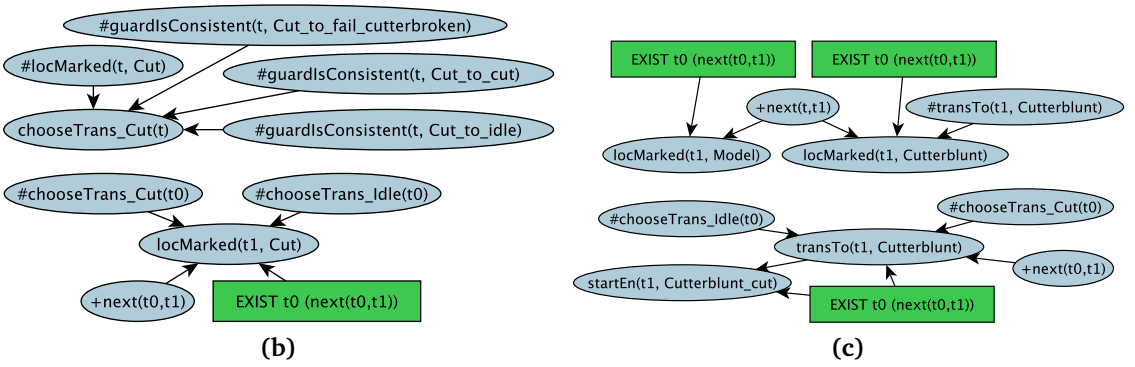
To give the reader an idea of what the fragment networks that result from the translation might look like, one such network is depicted in Figure 6.27a. Exemplarily, let us consider the probabilistic choice of transitions in more detail. Figure 6.27b shows two relevant fragments for the component “machining1” that is shown in Figure 6.26b. The top fragment is concerned with choosing a transition away from the location “cut” (for a case where it is marked) depending on the consistency of the constraints that guard the transitions; *chooseTrans_Cut(t)* maps to the set of outgoing transitions from “cut” accordingly. The bottom fragment is concerned with the marking of the location “cut”: If, in the current time step, a transition to cut has been chosen (which can occur from either “cut“ itself or “idle”), the location “cut” must be marked in the next step. Figure 6.27c shows fragments used for the translation of the composite location “cutter blunt”. The bottom fragments encode the *transTo* predicate, which is to be true if the location “cutter blunt” is transitioned to, and the *startEn* predicate for the sub-location “cut” of “cutter blunt”, which is to be true if the sub-location is enabled as a result of the composite location being transitioned to, thus handling the hierarchical interactions of markings.

In contrast to the model described in Section 6.3.1, which did not include any logical constraints, the automated translation from PHCA models involves comparatively large numbers of logical constraints. For instance, the BLN model shown in Figure 6.26a is supplemented by a total of 67 constraints in first-order logic. The constraints cover general axioms, requiring, for instance, the behaviour consistency of states that are marked, as well as instance-specific constraints describing the conditions under which a particular location’s behaviour is consistent. Such constraints can be used to couple the states of locations associated with factory stations and products. Furthermore, logical formulas are used to encode the transition guard constraints that establish the dependency on commands. For instance, a transition from “idle” to “cut” in Figure 6.27c is possible if and only if the command that is issued is “cut”.

All in all, the concepts of PHCA models can, for the most part, be straightforwardly mapped to BLNs. The translation that was sketched above is loosely based on the constraint optimization formulation for PHCA models (Mikaelian et al., 2005), which



(a)



(b) (c)

Figure 6.27: *Fragment network structure of a BLN that resulted from the translation of the PHCA for the example factory model (a) and two excerpts of it (b).*

instance	time steps	PHCA size	# BN nodes	ACE comp.	ACE eval.
fm1	6	11/6/27	1106	0.37 s	0.09 s
fm2	9	15/8/33	2122	0.93 s	0.18 s
fm3	9	17/8/33	2292	0.36 s	0.07 s
fm2(long $\mathscr{S}$)	19	15/8/33	4444	1128.04 s	OOM
fm3(long $\mathscr{S}$)	33	18/8/35	8878	1.34 s	0.16 s
sm	8	8/4/22	1080	0.87 s	0.03 s

Table 6.6: *For five variations of the factory model (fm) shown in Figure 6.26a (fm) and a satellite diagnosis model (sm), the table lists the number of time steps considered, the size of the original PHCA (no. of primitive locations/no. of composite locations/no. of transitions), the number of nodes in the ground auxiliary Bayesian network instantiated from the BLN obtained via the automated translation process, and the compilation and evaluation times required by ACE to solve the instances exactly. (Maier, Jain, and Sachenbacher, 2011)*

gives rise to a largely logic-based translation. However, the translation occasionally makes explicit use of BLN concepts to yield a more compact and computationally efficient representation. A more detailed account of the translation from PHCA models to BLNs is provided by Maier, Jain, and Sachenbacher (2011).

The translation to BLNs was tested on six models, five of which are variations of the factory model that was described above. All factory models use one assembly and one or two machining stations. In factory models 1, 2 and 3, we consider one, two and three products respectively. There are also minor differences in the sensor configurations. For models 2 and 3, we additionally consider variations with a longer production schedule (long $\mathscr{S}$). The sixth model is concerned with diagnosing hard- and software faults in a satellite's camera module (Mikaelian et al., 2005). Exact inference was used to compute posterior marginals for several variables of interest (i.e. product success probabilities at various points in time for the factory models), using ACE as the inference engine, which proved to be efficient for this particular application. The results are summarised in Table 6.6. For the most part, correct solutions could be computed quite quickly; results were equivalent to solutions computed based on an enumeration of possible worlds using branch and bound search on the constraint optimization formulation of the problems (Maier et al., 2011). For one of the instances, however, the 2GB memory limit of the machine was insufficient to reach a solution (out of memory error, OOM).

The automated translation of engineering models such as PHCA to a first-order probabilistic language such as BLNs is clearly a worthwhile endeavour. It opens up the possibility of leveraging state-of-the-art methods developed for standard graphical models (such as ACE) and furthermore leaves the gate open for an exploitation of regularities that result from an application of a first-order model using lifted inference.

Chapter 7

Conclusion

This thesis has explored statistical relational models for high-level knowledge representation, learning and reasoning in the context of cognitive technical systems. The techniques presented in this work are implemented in the ProbCog system, which integrates two complementary first-order probabilistic languages: Markov logic networks and Bayesian logic networks.

Markov logic networks are a particularly attractive, general framework, which has garnered much attention in recent years due to its conceptual simplicity and representational power. Since knowledge engineering in Markov logic networks is, however, not a straightforward task, this work has investigated a number of practically significant knowledge engineering issues. As a particularly relevant problem, I have analysed the soundness of generalisation across domains in detail, identifying common conditions under which shallow transfer invariably fails – with correspondingly severe consequences for knowledge engineering practice and the applicability of generalised models. Two conceptually distinct solutions to this problem were presented: the augmentation of the language with probability constraints, which can be enforced during inference using the algorithm IPFP-M, and the learning of adaptive Markov logic networks (AMLNs), which phrase parameters as functions over attributes of instantiations and therefore can dynamically adapt probabilistic parameters to the particular domain at hand, extending the class of probabilistic models that can be represented in practice.

The second representation language prominently portrayed in this work, Bayesian logic networks, is a novel formalism that is geared towards practical applicability. It is the first language to be based on the framework of mixed networks with probabilistic

and deterministic constraints. While it integrates many principles of other first-order probabilistic languages that are based on directed graphical models, it particularly facilitates the representation of logical knowledge through the inclusion of constraints that are specified in first-order logic, which enables the compact representation of deterministic principles that apply at a global level. Furthermore, BLNs are supported by a fully-featured, free and open-source implementation. I have given an overview of learning and inference methods and have specifically explored sampling-based inference as a practical method for approximate inference; in particular, novel approaches to sample searching were investigated, and the question of approximating solution quality for sampling-based inference was addressed.

Regarding the integration of first-order probabilistic languages in cognitive technical systems, I have advocated a decidedly inhomogeneous approach where cognitive capabilities result from the interplay of many cognitive resources that are linked via an interface layer. In practice, a potentially important problem is therefore the integration of cognitive components and the question of how probabilistic beliefs can be efficiently passed on between different probabilistic reasoning components of a technical system – i.e. the problem of probabilistic information interchange. Especially since inference in all-encompassing probabilistic models can be expected to be intractable, the use of multiple loosely coupled probabilistic models suggests itself. In this work, I have investigated the use of uncertain evidence for the integration of beliefs, focusing specifically on the problem of soft evidential update, where evidence is given in the form of probability distributions that are believed to hold. Being strongly related to the imposition of probability constraints, the problem can, in principle, be handled using methods such as IPFP-M, albeit inefficiently. As a more practical solution, this work has presented MC-SAT-PC, the first Markov chain Monte Carlo method for soft evidential update, explaining its relationship to iterative proportional fitting and demonstrating its superior performance in practice.

The general applicability of the techniques that were presented in this work was demonstrated in two types of technical systems: an autonomous robotic kitchen assistant and a cognitive factory. In robotics, many high-level reasoning problems can be phrased as inference problems in statistical relational models. In robot perception, two applications were considered: the first addressed the categorisation of tabletop objects based on fused point cloud and image data in a Bayesian logic network; the

second treated the problem of keeping track of observation-object associations over time, using soft evidential update in Markov logic networks to handle uncertain beliefs on, for instance, previous associations. While the solution that was presented is not fully mature, it does indicate that the highly general framework of statistical relational models can flexibly be applied to structured problems in perception. The use of statistical relational models can avoid the time-consuming development of custom-tailored algorithms. Furthermore, statistical relational models can easily be adapted to even significant changes in the assumptions made by the model – simply by modifying the set of probabilistic and deterministic constraints appropriately.

In the context of robot control, the use of statistical relational models as flexible cognitive resources for decision making, plan parametrisation, prediction and diagnosis was explored. The approach to cognitive robotics advocated in this work revolves around the cognitive robot abstract machine (CRAM) and its flexible mechanisms for the provision of reasoning capabilities, particularly through extensions of the high-level control language CRAM-PL. The example scenario that was considered involved an autonomous robotic kitchen assistant. In the household domain, models of everyday activities are potentially relevant to many high-level tasks. Therefore, the applicability of statistical relational models to everyday activities and environments was indicated in a number of example models, including models of meal habits and activity sequences. In order to apply such models in practice and leverage the first-order probabilistic reasoning capabilities in robot control, an extension of CRAM-PL that allows to transparently integrate inference results for the parametrisation of plans was presented, using the task of setting the table as an example. Furthermore, with the K-CoPMan system, a complete robotic application that integrates perception, logical knowledge processing and first-order probabilistic reasoning – closing the cognitive perception-action loop in a real robot system – was presented. The K-CoPMan system establishes physically grounded representations of perceived scenes and uses its abstract knowledge about the scene to infer actions based on a statistical relational model, enabling it to reasonably adapt its actions to the circumstances at hand in a principled manner. Moreover, this work has presented an initial experiment that leveraged the data that is collected as plans are carried out by the cognitive robot abstract machine.

Experience-based learning being an important concept for cognition-enabled robotics, the learning of statistical relational models based on relational data gathered during task execution is certainly an interesting direction for future work. Indeed, one of the most promising ideas for the equipment of autonomous robots with comprehensive knowledge processing and reasoning capabilities is the lifelong autonomous learning of joint probability distributions over the variations and parametrisations of robot control programs, the behaviour they generate and the situation-dependent effects they bring about.

Another application that was addressed in this work considered an entirely different type of technical system: a cognitive factory that assesses its production plans based on observations made during production. Bayesian logic networks were used as a probabilistic framework for the representation of factory station behaviour and the complex interactions between the production schedule, the products being produced and the state of the factory. Furthermore, a translation from AI engineering models to Bayesian logic networks was presented, which enables many inference techniques to straightforwardly be applied to this class of models.

Of course, statistical relational models are but one of many resources that can – and must – be leveraged for cognitive capabilities to be implemented in technical systems. At present, however, artificial intelligence is a highly fragmented field of research, and too few efforts are made at combining the much-needed insights from sub-disciplines as diverse as perception, learning, reasoning, planning, language processing and control towards an end: the development of cognition in technical systems such as autonomous robots. Only through the integration of manifold state-of-the-art techniques can we hope for significant progress to be made. This thesis has specifically addressed one piece of the integration puzzle; it has introduced novel techniques pertaining to probabilistic knowledge representation, reasoning and learning and has presented ways of applying these techniques within cognitive technical systems, demonstrating their practical viability.

List of Prior Publications

The work presented in this document is partly based on prior publications. Sections of this work that drew upon content from prior publications cited the respective publications where appropriate. A complete list of publications that were (co-)authored during my research as a doctoral candidate is provided below.

Journal Articles

Michael Beetz, Dominik Jain, Lorenz Mösenlechner, Moritz Tenorth, Lars Kunze, Nico Blodow, and Dejan Pangercic. Cognition-Enabled Autonomous Robot Control for the Realization of Home Chore Task Intelligence. *Proceedings of the IEEE, Special Issue on Quality of Life Technology*, 100(8):2454–2471, 2012.

Moritz Tenorth, Dominik Jain, and Michael Beetz. Knowledge Representation for Cognitive Robots. *Künstliche Intelligenz*, 24(3):233–240, 2010.

Michael Beetz, Moritz Tenorth, Dominik Jain, and Jan Bandouch. Towards Automated Models of Activities of Daily Life. *Technology and Disability*, 22(1-2):27–40, 2010a.

Michael Beetz, Dominik Jain, Lorenz Mösenlechner, and Moritz Tenorth. Towards Performing Everyday Manipulation Activities. *Robotics and Autonomous Systems*, 58(9):1085–1095, 2010b.

Conference and Workshop Papers

Martin Schuster, Dominik Jain, Moritz Tenorth, and Michael Beetz. Learning Organizational Principles in Human Environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3867–3874, 2012.

Dominik Jain, Klaus von Gleissenthall, and Michael Beetz. Bayesian Logic Networks and the Search for Samples with Backward Simulation and Abstract Constraint Learning. In *KI 2011: Advances in Artificial Intelligence, 34th Annual German Conference on AI*, volume 7006 of *Lecture Notes in Computer Science*, pages 144–156. Springer, 2011.

Dominik Jain. Knowledge Engineering with Markov Logic Networks: A Review. In *DKB 2011: Proceedings of the Third Workshop on Dynamics of Knowledge and Belief*, 2011.

Paul Maier, Dominik Jain, and Martin Sachenbacher. Compiling AI Engineering Models for Probabilistic Inference. In *KI 2011: Advances in Artificial Intelligence, 34th Annual German Conference on AI*, volume 7006 of *Lecture Notes in Computer Science*, pages 191–203. Springer, 2011.

William R. Murray and Dominik Jain. Modeling Cognitive Frames for Situations with Markov Logic Networks. In *Proceedings of the 8th International NLPCS Workshop: Human-Machine Interaction in Translation, Copenhagen Studies in Language 41*, pages 167–178. Samfundslitteratur, 2011.

Dominik Jain, Andreas Barthels, and Michael Beetz. Adaptive Markov Logic Networks: Learning Statistical Relational Models with Dynamic Parameters. In *19th European Conference on Artificial Intelligence (ECAI)*, pages 937–942, 2010.

Dominik Jain and Michael 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.

Nico Blodow, Dominik Jain, Zoltan-Csaba Marton, and Michael Beetz. Perception and Probabilistic Anchoring for Dynamic World State Logging. In *10th IEEE-RAS International Conference on Humanoid Robots*, pages 160–166, 2010.

Dejan Pangercic, Moritz Tenorth, Dominik Jain, and Michael Beetz. Combining Perception and Knowledge Processing for Everyday Manipulation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1065–1071, 2010.

Paul Maier, Dominik Jain, Stefan Waldherr, and Martin Sachenbacher. Plan Assessment for Autonomous Manufacturing as Bayesian Inference. In *KI 2010: Advances in Artificial Intelligence, 33rd Annual German Conference on AI*, volume 6359 of *Lecture Notes in Computer Science*, pages 263–271. Springer, 2010.

Moritz Tenorth, Lars Kunze, Dominik Jain, and Michael Beetz. KNOWROB-MAP – Knowledge-Linked Semantic Object Maps. In *10th IEEE-RAS International Conference on Humanoid Robots*, pages 430–435, 2010.

Dominik Jain, Lorenz Mösenlechner, and Michael Beetz. Equipping Robot Control Programs with First-Order Probabilistic Reasoning Capabilities. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3626–3631, 2009a.

Dominik Jain, Paul Maier, and Gregor 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*, 2009b.

Zoltan Csaba Marton, Radu Bogdan Rusu, Dominik Jain, Ulrich Klank, and Michael Beetz. Probabilistic Categorization of Kitchen Objects in Table Settings with a Composite Sensor. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4777–4784, 2009.

Michael Beetz, Jan Bandouch, Dominik Jain, and Moritz Tenorth. Towards Automated Models of Activities of Daily Life. In *First International Symposium on Quality of Life Technology – Intelligent Systems for Better Living*, 2009.

Michael Beetz, Freek Stulp, Bernd Radig, Jan Bandouch, Nico Blodow, Mihai Dolha, Andreas Fedrizzi, Dominik Jain, Uli Klank, Ingo Kresse, Alexis Maldonado, Zoltan Marton, Lorenz Mösenlechner, Federico Ruiz, Radu Bogdan Rusu, and Moritz Tenorth. The Assistive Kitchen – A Demonstration Scenario for Cognitive Technical Systems. In *IEEE 17th International Symposium on Robot and Human Interactive Communication (RO-MAN), Muenchen, Germany*, pages 1–8, 2008. Invited paper.

Dominik Jain, Lorenz Mösenlechner, and Michael Beetz. Equipping Robot Control Programs with First-Order Probabilistic Reasoning Capabilities. In *Proceedings of the 1st International Workshop on Cognition for Technical Systems*, 2008.

Dominik Jain, Bernhard Kirchlechner, and Michael Beetz. Extending Markov Logic to Model Probability Distributions in Relational Domains. In *KI 2007: Advances in Artificial Intelligence, 30th Annual German Conference on AI*, volume 4667 of *Lecture Notes in Computer Science*, pages 129–143. Springer, 2007.

Other

Dejan Pangercic, Koppany Mathe, Zoltan-Csaba Marton, Lucian Cosmin Goron, Monica-Simona Opris, Martin Schuster, Moritz Tenorth, Dominik Jain, Thomas Ruehr, and Michael Beetz. A Robot that Shops for and Stores Groceries. AAAI Video Competition (AIVC 2011), 2011.

Dominik Jain, Stefan Waldherr, and Michael Beetz. Bayesian Logic Networks. Technical report, IAS Group, Fakultät für Informatik, Technische Universität München, 2009.

Bibliography

Jean-Raymond Abrial. Data Semantics. In *IFIP TC2 Working Conference on Data Base Management*, pages 1–59. Elsevier Science Publishers (North-Holland), 1974.

Michael L. Anderson. Embodied Cognition: A Field Guide. *Artificial Intelligence*, 149(1): 91–130, 2003.

Dragomir Anguelov, Ben Taskar, Vassil Chatalbashev, Daphne Koller, Dinkar Gupta, Geremy Heitz, and Andrew Ng. Discriminative Learning of Markov random fields for Segmentation of 3D Scan Data. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 169–176, 2005.

Franz Baader and Ulrike Sattler. Expressive Number Restrictions in Description Logics. *Journal of Logic and Computation*, 9(3):319–350, 1999.

Fahiem Bacchus, Adam J. Grove, Joseph Y. Halpern, and Daphne Koller. From Statistical Knowledge Bases to Degrees of Belief. *Artificial Intelligence*, 87(1-2):75–143, 1996.

Jan Bandouch, Florian Engstler, and Michael Beetz. Evaluation of Hierarchical Sampling Strategies in 3D Human Pose Estimation. In *Proceedings of the 19th British Machine Vision Conference (BMVC)*, 2008.

Leonard E. Baum and Ted Petrie. Statistical Inference for Probabilistic Functions of Finite State Markov Chains. *The Annals of Mathematical Statistics*, 37(6):1554–1563, 1966.

Michael Beetz, Jan Bandouch, Dominik Jain, and Moritz Tenorth. Towards Automated Models of Activities of Daily Life. In *First International Symposium on Quality of Life Technology – Intelligent Systems for Better Living*, 2009.

Michael Beetz, Martin Buss, and Bernd Radig. Learning from Humans – Cognition-enabled Computational Models of Everyday Activity. *Künstliche Intelligenz*, 2010a.

Michael Beetz, Dominik Jain, Lorenz Mösenlechner, Moritz Tenorth, Lars Kunze, Nico Blodow, and Dejan Pangercic. Cognition-Enabled Autonomous Robot Control for the Realization of Home Chore Task Intelligence. *Proceedings of the IEEE, Special Issue on Quality of Life Technology*, 100(8):2454–2471, 2012.

Michael Beetz, Lorenz Mösenlechner, and Moritz Tenorth. CRAM – A Cognitive Robot Abstract Machine for Everyday Manipulation in Human Environments. In *IEEE/RSJ International*

Conference on Intelligent Robots and Systems, pages 1012–1017, 2010b.

Michael Beetz, Freek Stulp, Bernd Radig, Jan Bandouch, Nico Blodow, Mihai Dolha, Andreas Fedrizzi, Dominik Jain, Uli Klank, Ingo Kresse, Alexis Maldonado, Zoltan Marton, Lorenz Mösenlechner, Federico Ruiz, Radu Bogdan Rusu, and Moritz Tenorth. The Assistive Kitchen – A Demonstration Scenario for Cognitive Technical Systems. In *IEEE 17th International Symposium on Robot and Human Interactive Communication (RO-MAN), Muenchen, Germany*, pages 1–8, 2008. Invited paper.

Michael Beetz, Moritz Tenorth, Dominik Jain, and Jan Bandouch. Towards Automated Models of Activities of Daily Life. *Technology and Disability*, 22(1-2):27–40, 2010c.

Detlev Blanke. Leibniz und die Lingua Universalis. *Sitzungsberichte der Leibniz-Sozietät*, 13 (5):27–35, 1996.

Hannah Blau, Neil Immerman, and David Jensen. A Visual Language for Querying and Updating Graphs. Technical Report 037, Computer Science, University of Massachusetts Amherst, 2002.

Nico Blodow, Dominik Jain, Zoltan-Csaba Marton, and Michael Beetz. Perception and Probabilistic Anchoring for Dynamic World State Logging. In *10th IEEE-RAS International Conference on Humanoid Robots*, pages 160–166, 2010.

Nico Blodow, Radu Bogdan Rusu, Zoltan Csaba Marton, and Michael Beetz. Partial View Modeling and Validation in 3D Laser Scans for Grasping. In *9th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2009.

Peter Bonasso, James Firby, Erann Gat, David Kortenkamp, David Miller, and Marc Slack. Experiences with an Architecture for Intelligent, Reactive Agents. *Journal of Experimental and Theoretical Artificial Intelligence*, 9(1), 1997.

John S. Breese. Construction of Belief and Decision Networks. *Computational Intelligence*, 8: 624–647, 1992.

Matthias Bröcheler, Lilyana Mihalkova, and Lise Getoor. Probabilistic Similarity Logic. In *UAI 2010, Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, 2010.

Rodney A. Brooks. Elephants Don't Play Chess. *Robotics and Autonomous Systems*, 6:3–15, 1990.

Rodney A. Brooks. Intelligence without Representation. *Artificial Intelligence*, 47(1-3):139–159, 1991.

Peter Carbonetto, Jacek Kisyński, O De Freitas, and David Poole. Nonparametric Bayesian Logic. In *UAI 2005, Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*, 2005.

Anthony R. Cassandra, Leslie P. Kaelbling, and James A. Kurien. Acting Under Uncertainty: Discrete Bayesian Models for Mobile-Robot Navigation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1996.

Nick Chater, Joshua B. Tenenbaum, and Alan Yuille. Probabilistic Models of Cognition: Conceptual Foundations. *Trends in Cognitive Sciences, Special Issue: Probabilistic Models of Cognition*, 10(7):287 – 291, 2006.

Mark Chavira and Adnan Darwiche. Compiling Bayesian Networks with Local Structure. In *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, pages 1306–1312. Professional Book Center, 2005.

Gregory F. Cooper. The Computational Complexity of Probabilistic Inference Using Bayesian Belief Networks. *Artificial Intelligence*, 42(2-3):393–405, 1990.

Vítor Santos Costa, David Page, Maleeha Qazi, and James Cussens. CLP(BN): Constraint Logic Programming for Probabilistic Knowledge. In *UAI '03, Proceedings of the 19th Conference in Uncertainty in Artificial Intelligence*, pages 517–524. Morgan Kaufmann, 2003.

David R. Cox, Daniel Commenges, Anthony C. Davison, Patty J. Solomon, and Suzan R. Wilson. *The Oxford Dictionary of Statistical Terms*. Oxford University Press, 2003.

Ingemar J. Cox and Sunita L. Hingorani. An Efficient Implementation of Reid's Multiple Hypothesis Tracking Algorithm and Its Evaluation for the Purpose of Visual Tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(2):138–150, 1996.

Imre Csiszár. I-Divergence Geometry of Probability Distributions and Minimization Problems. *The Annals of Probability*, 3(1):146–158, 1975.

Jesse Davis and Pedro Domingos. Deep Transfer via Second-Order Markov Logic. In *Proceedings of the 26th Annual International Conference on Machine Learning*, page 28. ACM, 2009.

Fernando De la Torre, Jessica Hodgins, Javier Montano, Sergio Valcarcel, and Justin Macey. Guide to the Carnegie Mellon University Multimodal Activity (CMU-MMAC) Database. Technical report, CMU-RI-TR-08-22, Robotics Institute, Carnegie Mellon University, 2009.

Luc De Raedt, Paolo Frasconi, Kristian Kersting, and Stephen Muggleton, editors. *Probabilistic Inductive Logic Programming – Theory and Applications*, volume 4911 of *Lecture Notes in Computer Science*, 2008. Springer.

Luc De Raedt, Angelika Kimmig, and Hannu Toivonen. ProbLog: A Probabilistic Prolog and its Application in Link Discovery. In *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 2462–2467, 2007.

Rodrigo de Salvo Braz, Eyal Amir, and Dan Roth. Lifted First-Order Probabilistic Inference. In *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, pages 1319–1325. Professional Book Center, 2005.

Rodrigo de Salvo Braz, Eyal Amir, and Dan Roth. A Survey of First-Order Probabilistic Models. In *Innovations in Bayesian Networks*, volume 156 of *Studies in Computational Intelligence*, pages 289–317. Springer, 2008.

Thomas Dean and Keiji Kanazawa. A Model for Reasoning about Persistence and Causation. *Computational Intelligence*, 5(2):142–150, 1989.

Rina Dechter and Daniel Frost. Backjump-Based Backtracking for Constraint Satisfaction Problems. *Artificial Intelligence*, 136(2):147–188, 2002.

Rina Dechter, Kalev Kask, and Robert Mateescu. Iterative Join-Graph Propagation. In *UAI '02, Proceedings of the 18th Conference in Uncertainty in Artificial Intelligence*, pages 128–136, 2002.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society, Series B*, 39:1–38, 1977.

Kenji Doya, Shin Ishii, Alexandre Pouget, and Rajesh P. N. Rao, editors. *Bayesian Brain*. MIT Press, 2007.

Ivan P. Fellegi and Alan B. Sunter. A Theory for Record Linkage. *Journal of the American Statistical Association*, 64(328):1183–1210, 1969.

Daan Fierens, Hendrik Blockeel, Maurice Bruynooghe, and Jan Ramon. Logical Bayesian Networks and Their Relation to Other Probabilistic Logical Models. In *BNAIC 2005 – Proceedings of the Seventeenth Belgium-Netherlands Conference on Artificial Intelligence*, pages 343–344. Koninklijke Vlaamse Academie van Belie voor Wetenschappen en Kunsten, 2005.

Richard O. Fikes and Nils J. Nilsson. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. Technical Report 43r, AI Center, SRI International, 1971.

James Firby. An Investigation into Reactive Planning in Complex Domains. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 202–206, 1987.

Jens Fisseler. Toward Markov Logic with Conditional Probabilities. In *Proceedings of the Twenty-First International Florida Artificial Intelligence Research Society (FLAIRS) Conference*, pages 643–648, 2008.

Nir Friedman, Lise Getoor, Daphne Koller, and Avi Pfeffer. Learning Probabilistic Relational Models. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 1300–1309. Morgan Kaufmann, 1999.

Robert M. Fung and Kuo-Chu Chang. Weighing and Integrating Evidence for Stochastic Simulation in Bayesian Networks. In *UAI '89: Proceedings of the Fifth Annual Conference on Uncertainty in Artificial Intelligence*, pages 209–220. North-Holland Publishing Co., 1990.

Robert M. Fung and Brendan Del Favero. Backward Simulation in Bayesian Networks. In *UAI '94: Proceedings of the Tenth Annual Conference on Uncertainty in Artificial Intelligence*, pages 227–234, 1994.

Erann Gat. ESL: A Language for Supporting Robust Plan Execution in Embedded Autonomous Agents. In *AAAI Fall Symposium: Issues in Plan Execution*, 1996.

Erann Gat. On Three-Layer Architectures. In P. Bonasso, D. Kortenkamp, and R. Murphy, editors, *Artificial Intelligence and Mobile Robots*, pages 195–210. MIT Press, Cambridge, MA, 1998.

Stuart Geman and Donald Geman. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

Lise Getoor. *Learning Statistical Models from Relational Data*. PhD thesis, Stanford University, 2001.

Lise Getoor, Nir Friedman, Daphne Koller, Avi Pfeffer, and Benjamin Taskar. Probabilistic Relational Models. In Lise Getoor and Ben Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, 2007.

Lise Getoor and Ben Taskar, editors. *Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning)*. MIT Press, 2007.

Vibhav Gogate and Rina Dechter. SampleSearch: A Scheme that Searches for Consistent Samples. In *Eleventh International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2007.

Noah D. Goodman, Vikash K. Mansinghka, Daniel M. Roy, Keith Bonawitz, and Joshua B. Tenenbaum. Church: a language for generative models. In *UAI 2008, Proceedings of the 24th Conference in Uncertainty in Artificial Intelligence*, pages 220–229. AUAI Press, 2008.

Thomas L. Griffiths, Charles Kemp, and Josh B. Tenenbaum. *The Cambridge Handbook of Computational Cognitive Modeling*, chapter Bayesian Models of Cognition. Cambridge University Press, 2008.

David B. Grimes, Rawichote Chalodhorn, and Rajesh P. N. Rao. Dynamic Imitation in a Humanoid Robot through Nonparametric Probabilistic Inference. In *Proceedings of Robotics: Science and Systems (RSS)*. MIT Press, 2006.

Simon Grötzinger. Learning Probabilistic Models of Robot Behaviour from Logged Execution Traces. Bachelor's Thesis, Department of Informatics, Technische Universität München. Michael Beetz, supervisor, Dominik Jain, advisor, 2011.

Rahul Gupta, Ajit A. Diwan, and Sunita Sarawagi. Efficient Inference with Cardinality-Based Clique Potentials. In *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007)*, pages 329–336, 2007.

Bernd Gutmann, Manfred Jaeger, and Luc De Raedt. Extending ProbLog with Continuous Distributions. In *Inductive Logic Programming – 20th International Conference*, volume 6489 of *Lecture Notes in Computer Science*, pages 76–91. Springer, 2010.

Theodore Hailperin. Probability Logic. *Notre Dame Journal of Formal Logic*, 25(3):198–212, 1984.

Joseph Y. Halpern. An Analysis of First-Order Logics of Probability. *Artificial Intelligence*, 46: 311–350, 1990.

David Heckerman. A Tutorial on Learning with Bayesian Networks. Technical report, Microsoft Research, Redmond, Washington, 1995.

Michael C. Horsch and David Poole. A Dynamic Approach to Probabilistic Inference Using Bayesian Betworks. In *UAI '90: Proceedings of the Sixth Annual Conference on Uncertainty in Artificial Intelligence*, pages 155–161. Elsevier, 1990.

François Félix Ingrand, Raja Chatila, Rachid Alami, and Frédérick Robert. PRS: A High Level Supervision and Control Language for Autonomous Mobile Robots. In *IEEE International Conference on Robotics and Automation*, pages 43–49, 1996.

Manfred Jaeger. Relational Bayesian Networks. In *UAI '97: Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, pages 266–273. Morgan Kaufmann, 1997.

Manfred Jaeger. Model-Theoretic Expressivity Analysis. In *Probabilistic Inductive Logic Programming*, volume 4911 of *Lecture Notes in Computer Science*, pages 325–339. Springer, 2008.

Dominik Jain. Knowledge Engineering with Markov Logic Networks: A Review. In *DKB 2011: Proceedings of the Third Workshop on Dynamics of Knowledge and Belief*, 2011.

Dominik Jain, Andreas Barthels, and Michael Beetz. Adaptive Markov Logic Networks: Learning Statistical Relational Models with Dynamic Parameters. In *19th European Conference on Artificial Intelligence (ECAI)*, pages 937–942, 2010.

Dominik Jain, Bernhard Kirchlechner, and Michael Beetz. Extending Markov Logic to Model Probability Distributions in Relational Domains. In *KI 2007: Advances in Artificial Intelligence, 30th Annual German Conference on AI*, volume 4667 of *Lecture Notes in Computer Science*, pages 129–143. Springer, 2007.

Dominik Jain, Paul Maier, and Gregor 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.

Dominik Jain, Lorenz Mösenlechner, and Michael Beetz. Equipping Robot Control Programs with First-Order Probabilistic Reasoning Capabilities. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3626–3631, 2009b.

Dominik Jain, Klaus von Gleissenthall, and Michael Beetz. Bayesian Logic Networks and the Search for Samples with Backward Simulation and Abstract Constraint Learning. In *KI 2011: Advances in Artificial Intelligence, 34th Annual German Conference on AI*, volume 7006 of *Lecture Notes in Computer Science*, pages 144–156. Springer, 2011.

Dominik Jain, Stefan Waldherr, and Michael Beetz. Bayesian Logic Networks. Technical report, IAS Group, Fakultät für Informatik, Technische Universität München, 2009c.

Henry Kautz, Bart Selman, and Yueyen Jiang. A General Stochastic Approach to Solving Problems with Hard and Soft Constraints. In *The Satisfiability Problem: Theory and Applications*, pages 573–586. American Mathematical Society, 1996.

Kristian Kersting, Babak Ahmadi, and Sriraam Natarajan. Counting Belief Propagation. In *UAI 2009, Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, 2009.

Kristian Kersting and Luc De Raedt. Bayesian Logic Programming: Theory and Tool. In Lise Getoor and Ben Taskar, editors, *Introduction to Statistical Relational Learning*, chapter 10. MIT Press, 2007.

Young-Gyun Kim and Marco Valtorta. Soft Evidential Update for Communication in Multiagent Systems and the Big Clique Algorithm, 2000.

Alexandra Kirsch. Robot Learning Language – Integrating Programming and Learning for Cognitive Systems. *Robotics and Autonomous Systems Journal*, 57(9):943–954, 2009.

Ulrich Klank, Dejan Pangercic, Radu Bogdan Rusu, and Michael Beetz. Real-time CAD Model Matching for Mobile Manipulation and Grasping. In *9th IEEE-RAS International Conference on Humanoid Robots*, pages 290–296, 2009.

David C. Knill and Alexandre Pouget. The Bayesian Brain: The Role of Uncertainty in Neural Coding and Computation. *Trends in Neurosciences*, 27(12):712–719, 2004.

Stanley Kok and Pedro Domingos. Learning the Structure of Markov Logic Networks. In *ICML '05: Proceedings of the 22nd International Conference on Machine learning*, pages 441–448. ACM Press, 2005.

Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

Daphne Koller, Alon Levy, and Avi Pfeffer. P-CLASSIC: A Tractable Probabilistic Description Logic. In *Proceedings of the AAAI Fourteenth National Conference on Artificial Intelligence*, pages 390–397, 1997.

Daphne Koller and Avi Pfeffer. Object-Oriented Bayesian Networks. In *UAI '97: Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, pages 302–313. Morgan Kaufmann, 1997.

Daphne Koller and Avi Pfeffer. Probabilistic Frame-Based Systems. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 580–587. AAAI Press, 1998.

Konrad P. Körding and Daniel M. Wolpert. Bayesian Integration in Sensorimotor Learning. *Nature*, 427:244–247, 2004.

Lukas Kuhn, Bob Price, Johan de Kleer, Minh Binh Do, and Rong Zhou. Pervasive Diagnosis: The Integration of Diagnostic Goals into Production Plans. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, pages 1306–1312. AAAI Press, 2008.

John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289, 2001.

Scott Langevin and Marco Valtorta. Performance Evaluation of Algorithms for Soft Evidential Update in Bayesian Networks: First Results. In *Second International Conference on Scalable Uncertainty Management*, volume 5291 of *Lecture Notes in Computer Science*, pages 284–297. Springer, 2008.

Kathryn B. Laskey. MEBN: A Language for First-Order Bayesian Knowledge Bases. *Artificial Intelligence*, 172(2-3):140–178, 2008.

Steffen L. Lauritzen and David J. Spiegelhalter. Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, 50(2), 1988.

John J. Leonard and Hugh F. Durrant-Whyte. Simultaneous Map Building and Localization for an Autonomous Mobile Robot. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 1442–1447, 1991.

Xiao Li. On the Use of Virtual Evidence in Conditional Random Fields. In *EMNLP '09: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1289–1297. Association for Computational Linguistics, 2009.

Marco Lippi and Paolo Frasconi. Prediction of Protein Beta-Residue Contacts by Markov Logic Networks with Grounding-Specific Weights. *Bioinformatics*, 25(18):2326–2333, 2009.

Sebastian Loh, Matthias Thimm, and Gabriele Kern-Isberner. On the Problem of Grounding a Relational Probabilistic Conditional Knowledge Base. In *Proceedings of the 14th International Workshop on Non-Monotonic Reasoning (NMR'10)*, 2010.

Daniel Lowd and Pedro 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, 2007a.

Daniel Lowd and Pedro Domingos. Recursive Random Fields. In *20th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 950–955, 2007b.

Paul Maier, Dominik Jain, and Martin Sachenbacher. Compiling AI Engineering Models for Probabilistic Inference. In *KI 2011: Advances in Artificial Intelligence, 34th Annual German Conference on AI*, volume 7006 of *Lecture Notes in Computer Science*, pages 191–203. Springer, 2011.

Paul Maier, Dominik Jain, Stefan Waldherr, and Martin Sachenbacher. Plan Assessment for Autonomous Manufacturing as Bayesian Inference. In *KI 2010: Advances in Artificial Intelligence, 33rd Annual German Conference on AI*, volume 6359 of *Lecture Notes in Computer Science*, pages 263–271. Springer, 2010.

Paul Maier, Martin Sachenbacher, Thomas Rühr, and Lukas Kuhn. Constraint-Based Integration of Plan Tracking and Prognosis for Autonomous Production. In *KI 2009: Advances in Artificial Intelligence, 32nd Annual German Conference on AI*, volume 5803 of *Lecture Notes in Computer Science*, pages 403–410. Springer, 2009.

Kim Marriott, Nicholas Nethercote, Reza Rafeh, Peter J. Stuckey, Maria Garcia De La Banda, and Mark Wallace. The Design of the Zinc Modelling Language. *Constraints*, 13(3):229–267, 2008.

Zoltan Csaba Marton, Lucian Cosmin Goron, Radu Bogdan Rusu, and Michael Beetz. Reconstruction and Verification of 3D Object Models for Grasping. In *Proceedings of the 14th International Symposium on Robotics Research (ISRR09)*, 2009a.

Zoltan Csaba Marton, Radu Bogdan Rusu, and Michael Beetz. On Fast Surface Reconstruction Methods for Large and Noisy Datasets. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2009b.

Zoltan Csaba Marton, Radu Bogdan Rusu, Dominik Jain, Ulrich Klank, and Michael Beetz. Probabilistic Categorization of Kitchen Objects in Table Settings with a Composite Sensor. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4777–4784, 2009c.

Robert Mateescu and Rina Dechter. Mixed Deterministic and Probabilistic Networks. *Annals of Mathematics and Artificial Intelligence*, 54(1-3):3–51, 2008.

Andrew McCallum, Karl Schultz, and Sameer Singh. FACTORIE: Probabilistic Programming via Imperatively Defined Factor Graphs. In *Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems*, pages 1249–1257. Curran Associates, Inc., 2009.

Drew McDermott. The 1998 AI Planning Systems Competition. *AI Magazine*, 21(2):35–55, 2000.

Lilyana Mihalkova. *Learning with Markov Logic Networks: Transfer Learning, Structure Learning, and an Application to Web Query Disambiguation*. PhD thesis, Department of Computer Sciences, University of Texas at Austin, Austin, TX, 2009.

Tsoline Mikaelian, Brian C. Williams, and Martin Sachenbacher. Model-Based Monitoring and Diagnosis of Systems with Software-Extended Behavior. In *Proceedings of the Twentieth National Conference on Artificial Intelligence*. AAAI Press, 2005.

Brian Milch, Bhaskara Marthi, Stuart J. Russell, David Sontag, Daniel L. Ong, and Andrey Kolobov. BLOG: Probabilistic Models with Unknown Objects. In *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, pages 1352–1359. Professional Book Center, 2005.

Brian Milch and Stuart J. Russell. First-Order Probabilistic Languages: Into the Unknown. In *Inductive Logic Programming, 16th International Conference, ILP 2006*, volume 4455 of *Lecture Notes in Computer Science*, pages 10–24. Springer, 2006.

Brian Milch, Luke S. Zettlemoyer, Kristian Kersting, Michael Haimes, and Leslie Pack Kaelbling. Lifted Probabilistic Inference with Counting Formulas. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, 2008.

Lorenz Mösenlechner, Nikolaus Demmel, and Michael Beetz. Becoming Action-aware through Reasoning about Logged Plan Execution Traces. In *IEEE/RSJ International Conference on Intelligent RObots and Systems.*, pages 2231–2236, 2010.

Jun Nakanishi, Jay A. Farrell, and Stefan Schaal. Composite Adaptive Control with Locally Weighted Statistical Learning. *Neural Networks*, 18(1):71–90, 2005.

Jennifer Neville and David Jensen. Relational Dependency Networks. *Journal of Machine Learning Research*, 8:653–692, 2007.

Allen Newell and Herbert A. Simon. Computer Science as Empirical Inquiry: Symbols and Search. *Communications of the ACM*, 19(3):113–126, 1976.

Nils J. Nilsson. Shakey the Robot. Technical Note 323, SRI International, Menlo Park, California, 1984.

Nils J. Nilsson. Probabilistic Logic. *Artificial Intelligence*, 28(1):71–87, 1986.

Rong Pan, Yun Peng, and Zhongli Ding. Belief Update in Bayesian Networks Using Uncertain Evidence. In *Proceedings of the 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2006)*, pages 441–444, 2006.

Dejan Pangercic, Moritz Tenorth, Dominik Jain, and Michael Beetz. Combining Perception and Knowledge Processing for Everyday Manipulation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1065–1071, 2010.

James D. Park and Adnan Darwiche. Complexity Results and Approximation Strategies for MAP Explanations. *Journal of Artificial Intelligence Research (JAIR)*, 21:101–133, 2004.

Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

Yun Peng and Zhongli Ding. Modifying Bayesian Networks by Probability Constraints. In *UAI '05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence*, pages 26–29, 2005.

Yun Peng and Shenyong Zhang. Integrating Probability Constraints into Bayesian Nets. In *19th European Conference on Artificial Intelligence*, pages 981–982, 2010.

Avi Pfeffer. IBAL: A Probabilistic Rational Programming Language. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 733–740. Morgan Kaufmann, 2001.

Avi Pfeffer, Daphne Koller, Brian Milch, and Ken T. Takusagawa. SPOOK: A System for Probabilistic Object-Oriented Knowledge Representation. In *UAI '99: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 541–550, 1999.

Matthai Philipose, Kenneth P. Fishkin, Mike Perkowitz, Donald J. Patterson, Dieter Fox, Henry A. Kautz, and Dirk Hähnel. Inferring Activities from Interactions with Objects. *IEEE Pervasive Computing*, 3(4):50–57, 2004.

David Poole. First-Order Probabilistic Inference. In *IJCAI*, pages 985–991, 2003.

Hoifung Poon and Pedro Domingos. Sound and Efficient Inference with Probabilistic and Deterministic Dependencies. In *Proceedings of the Twenty-First National Conference on Arti-*

ficial Intelligence. AAAI Press, 2006.

Ingmar Posner, Mark Cummins, and Paul Newman. Fast Probabilistic Labeling of City Maps. In *Proceedings of Robotics: Science and Systems*. MIT Press, 2008.

Till Quack, Vittorio Ferrari, Bastian Leibe, and Luc J. Van Gool. Efficient Mining of Frequent and Distinctive Feature Configurations. In *IEEE 11th International Conference on Computer Vision (ICCV)*. IEEE Computer Society, 2007.

Matthew Richardson and Pedro Domingos. Markov Logic Networks. *Machine Learning*, 62 (1-2):107–136, 2006.

Sebastian Riedel. Improving the Accuracy and Efficiency of MAP Inference for Markov Logic. In *UAI 2008, Proceedings of the 24th Conference in Uncertainty in Artificial Intelligence, Helsinki, Finland, July 9-12, 2008*, pages 468–475. AUAI Press, 2008.

Dan Roth. On the Hardness of Approximate Reasoning. *Artificial Intelligence*, 82(1-2):273–302, 1996.

Reuven Y. Rubinstein. *Simulation and the Monte Carlo Method*. John Wiley & Sons, Inc., 1981.

Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, Englewood Cliffs, New Jersey, 1995.

Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, Upper Saddle River, New Jersey, second edition, 2003.

Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, Andreas Holzbach, and Michael Beetz. Model-based and Learned Semantic Object Labeling in 3D Point Cloud Maps of Kitchen Environments. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2009.

Silvio Savarese and Fei-Fei Li. 3D Generic Object Categorization, Localization and Pose Estimation. In *IEEE 11th International Conference on Computer Vision (ICCV)*. IEEE Computer Society, 2007.

Thorsten Schmitt, Robert Hanek, Michael Beetz, Sebastian Buck, and Bernd Radig. Cooperative Probabilistic State Estimation for Vision-based Autonomous Mobile Robots. *IEEE Transactions on Robotics and Automation*, 18(5), 2002.

Ruwen Schnabel, Roland Wahl, and Reinhard Klein. Efficient RANSAC for Point-Cloud Shape Detection. *Computer Graphics Forum*, 26(2):214–226, 2007.

Dirk Schulz, Wolfram Burgard, Dieter Fox, and Armin B. Cremers. Tracking Multiple Moving Targets with a Mobile Robot using Particle Filters and Statistical Data Association. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1665–1670, 2001.

Martin Schuster, Dominik Jain, Moritz Tenorth, and Michael Beetz. Learning Organizational Principles in Human Environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3867–3874, 2012.

Bart Selman, Henry A. Kautz, and Bram Cohen. Local Search Strategies for Satisfiability Testing. In *Proceedings of the Second DIMACS Challange on Cliques, Coloring, and Satisfiability*, 1993.

Ross D. Shachter. Probabilistic Inference and Influence Diagrams. *Operations Research*, 36 (4):589–604, 1988.

Ross D. Shachter and Mark A. Peot. Simulation Approaches to General Probabilistic Inference on Belief Networks. In *UAI '89: Proceedings of the Fifth Annual Conference on Uncertainty in Artificial Intelligence*, pages 221–234. North-Holland, 1989.

Parag Singla and Pedro Domingos. Discriminative Training of Markov Logic Networks. In *Proceedings of the Twentieth National Conference on Artificial Intelligence*, pages 868–873. AAAI Press, 2005.

Parag Singla and Pedro Domingos. Entity Resolution with Markov Logic. In *In ICDM*, pages 572–582. IEEE Computer Society Press, 2006.

Parag Singla and Pedro Domingos. Markov Logic in Infinite Domains. In *UAI 2007, Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence*, pages 368–375. AUAI Press, 2007.

Parag Singla and Pedro Domingos. Lifted First-Order Belief Propagation. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, pages 1094–1099. AAAI Press, 2008.

Brian C. Smith. *Reflection and Semantics in a Procedural Language*. PhD thesis, Massachusetts Institute of Technology, 1982.

David J. Spiegelhalter and Steffen L. Lauritzen. Sequential Updating of Conditional Probabilities on Directed Graphical Structures. *Networks*, 20(5):579–605, 1990.

Charles Sutton and Andrew McCallum. An Introduction to Conditional Random Fields for Relational Learning. In Lise Getoor and Ben Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, 2007.

Moritz Tenorth, Jan Bandouch, and Michael Beetz. The TUM Kitchen Data Set of Everyday Manipulation Activities for Motion Tracking and Action Recognition. In *IEEE International Workshop on Tracking Humans for the Evaluation of their Motion in Image Sequences (THEMIS), in conjunction with ICCV2009*, 2009.

Moritz Tenorth and Michael Beetz. KnowRob – Knowledge Processing for Autonomous Personal Robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4261–4266, 2009.

Moritz Tenorth, Lars Kunze, Dominik Jain, and Michael Beetz. KNOWROB-MAP – Knowledge-Linked Semantic Object Maps. In *10th IEEE-RAS International Conference on Humanoid Robots*, pages 430–435, 2010.

Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT Press, Cambridge, 2005.

Marc Toussaint. Probabilistic Inference as a Model of Planned Behavior. *Künstliche Intelligenz*, 3/09:23–29, 2009.

Rudolph Triebel, Kristian Kersting, and Wolfram Burgard. Robust 3D Scan Point Classification Using Associative Markov Networks. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 2603–2608, 2006.

Alan M. Turing. On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42:230–265, 1936.

Markus Ulrich, Christian Wiedemann, and Carsten Steger. CAD-Based Recognition of 3D Objects in Monocular Images. In *IEEE International Conference on Robotics and Automation*, pages 1191–1198, 2009.

Peter Van Roy, Per Brand, Denys Duchier, Seif Haridi, Martin Henz, and Christian Schulte. Logic Programming in the Context of Multiparadigm Programming: the Oz Experience. *Theory and Practice of Logic Programming*, 3(6):717–763, 2003.

Shrihari Vasudevan and Roland Siegwart. Bayesian Space Conceptualization and Place Classification for Semantic Maps in Mobile Robotics. *Robotics and Autonomous Systems*, 56(6): 522–537, 2008.

Pooja Viswanathan, David Meger, Tristram Southey, James J. Little, and Alan K. Mackworth. Automated Spatial-Semantic Modeling with Applications to Place Labeling and Informed Search. In *CRV 2009, Sixth Canadian Conference on Computer and Robot Vision*, pages 284–291. IEEE Computer Society, 2009.

Wei Wei, Jordan Erenrich, and Bart Selman. Towards Efficient Sampling: Exploiting Random Walk Strategies. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence*, pages 670–676. AAAI Press, 2004.

Brian C. Williams, Seung Chung, and Vineet Gupta. Mode Estimation of Model-based Programs: Monitoring Systems with Complex Behavior. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 579–590. Morgan Kaufmann, 2001.

Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, San Francisco, second edition, 2005.

Xu Hong Xiao, Hian Beng Lee, and Gee Wah Ng. Learning Bayesian Network Parameters from Soft Data. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 17 (2):281–294, 2009.

Pingkun Yan, Saad M. Khan, and Mubarak Shah. 3D Model Based Object Class Detection in an Arbitrary View. In *IEEE 11th International Conference on Computer Vision (ICCV)*. IEEE Computer Society, 2007.

Changhe Yuan and Marek J. Druzdzel. An Importance Sampling Algorithm Based on Evidence Pre-propagation. In *UAI '03, Proceedings of the 19th Conference in Uncertainty in Artificial Intelligence*, pages 624–631. Morgan Kaufmann, 2003.

Michael F. Zäh, Michael Beetz, Kristina Shea, Gunther Reinhart, K. Bender, Christian Lau, Martin Ostgathe, W. Vogl, Mathey Wiesbeck, Marco Engelhard, Christoph Ertelt, Thomas Rühr, M. Friedrich, and S. Herle. The Cognitive Factory. In H. A. ElMaraghy, editor, *Changeable and Reconfigurable Manufacturing Systems*, pages 355–371. Springer, 2009.

Nevin Lianwen Zhang and David Poole. Exploiting Causal Independence in Bayesian Network Inference. *Journal of Artificial Intelligence Research (JAIR)*, 5:301–328, 1996.

Ciyou Zhu, Richard H. Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-BFGS-B: Fortran Subroutines for Large-Scale Bound-Constrained Optimization. *ACM Transactions On Mathematical Software*, 23(4):550–560, 1997.

Index