draft – 2011-06-01

# Probabilistic Logics in Expert Systems: Approaches, Implementations, and Applications*

Gabriele Kern-Isberner[1]
Christoph Beierle[2], Marc Finthammer[2], Matthias Thimm[1]

[1]Dept. of Computer Science, TU Dortmund, 44221 Dortmund, Germany
[2]Dept. of Computer Science, FernUniversität in Hagen, 58084 Hagen, Germany

**Abstract.** The handling of uncertain information is of crucial importance for the success of expert systems. This paper gives an overview on logic-based approaches to probabilistic reasoning and goes into more details about recent developments for relational, respectively first-order, probabilistic methods like Markov logic networks, and Bayesian logic programs. In particular, we will feature the maximum entropy approach as a powerful and elegant method that combines convenience with respect to knowledge representation with excellent inference properties. We briefly describe some systems for probabilistic reasoning, and go into more details on the KReator system as a versatile toolbox for probabilistic relational learning, modelling, and inference. Moreover, we will illustrate applications of probabilistic logics in various scenarios.

## 1 Introduction

Real world applications of expert systems (and other computational systems, too) usually have to struggle with the problem that both background knowledge and information on the situation at hand are neither complete nor certain. For instance, in a medical domain, the physician may know that most patients suffering from appendicitis also complain about abdominal pain, but in some cases, the patients show other atypical symptoms; however, these relationships can not be further specified in a satisfactory way. In the special case of the patient she is just facing, she is not even sure whether he feels abdominal pain as he is a boy of three years of age.

Probabilistic logics offer a rich framework to represent and process uncertain information, and are linked to statistics and machine learning in a natural way. Knowledge can be extracted from data, expressed in a suitable probabilistic formalism like Bayesian networks [26], and used for uncertain reasoning by applying inference mechanisms. Completeness of knowledge can be achieved by presupposing additional assumptions like conditional independence of variables, like in most probabilistic networks [26], or by making use of the information-theoretical principles of optimum entropy [16]. In both ways, a full probability distribution

* The research reported here was partially supported by the Deutsche Forschungsgemeinschaft (grants KE 1413/2-2 and BE 1700/7-2).

is generated from partial knowledge, on the base of which probabilities for arbitrary queries can be computed.

Most of the standard probabilistic approaches applied today make use of some type of probabilistic networks and propositional logic. While network techniques are of major importance to allow for local computations, the restriction to propositional logic makes probabilistic knowledge representation inadequate for domains in which relationships between objects are in the focus of investigation. So, especially in the last decade, a multitude of probabilistic approaches based on first-order logic have been brought forth. Markov logic networks [5], Bayesian logic programs [19], and similar relational probabilistic approaches [11] aim at generalising established propositional probabilistic methods to first-order knowledge representation. This turns out to be not an easy task, as the complexity of knowledge representation raises substantially, so that new inference techniques have to be devised. Moreover, the probabilistic semantics of open formulas is not at all clear. For example, the following conditional probabilistic formulas express commonsense knowledge about the relationships between elephants and their keepers which are usually good (elephants like their keepers), but also take exceptions into regard – elephants tend not to like keeper *fred*, except for the good natured elephant *clyde*:

$$\begin{array}{l}(\mathit{likes}(X,Y)\,|\,\mathit{elephant}(X),\ \mathit{keeper}(Y))\,[0.8]\\(\mathit{likes}(X,\mathit{fred})\,|\,\mathit{elephant}(X))\,[0.3]\\\mathit{likes}(\mathit{clyde},\mathit{fred})[0.9]\end{array}$$

A schematic grounding of all rules of this knowledge base would cause conflicts with respect to elephant *clyde* and keeper *fred*. Moreover, both statistical (or population-based, respectively) information and subjective views are addressed, as the first two formulas involve all elephants (and keepers), while the third one only considers situations involving *clyde* and *fred*.

With many interesting new applications like social networks, hard computational problems, and challenging theoretical questions, the area of relational probabilistic knowledge representation has witnessed very active research work recently, providing a lot of new approaches and techniques. However, comparisons between approaches and evaluations with respect to computational and representational issues are not easily done, since each approach uses its own logical framework. Hence, there is an increasing demand for tools that support the investigation of different approaches in example or real world scenarios.

This paper aims to give an overview on the area of probabilistic knowledge representation, starting with standard propositional approaches and introducing into relational probabilistic knowledge representation by sketching some major approaches. As a special focus of the paper, we feature approaches that are based on the principle of maximum entropy as an elegant and powerful methodology that provides an excellent framework for commonsense and uncertain reasoning. Suitable systems are described briefly, and their use for applications in medical and biochemical domains is illustrated.

This paper is organized as follows. Sec. 2 provides details on Markov and Bayesian networks, and on the maximum entropy approach, both in propositional and first-order frameworks. In Sec. 3, the systems SPIRIT, MECoRe, Alchemy, ProbCog, and KREATOR are presented, and applications of these systems are illustrated in Sec. 4. Finally, Sec. 5 concludes this paper.

## 2 Approaches

According to these approaches, Popular propositional approaches to probabilistic logic use probabilistic networks, heavily relying on the notion of conditional independence. According to these approaches, Probabilistic conditional logic employs the principle of maximum entropy and is also based on propositional logic, while Bayesian logic programs, Markov logic networks, and relational maximum entropy approaches support a relational setting.

### 2.1 Propositional Approaches

**Probabilistic Networks** Conditional probabilities are often used in knowledge representation to describe causal or diagnostic dependencies [25]. A well-known framework which relies heavily on the notions of conditional probability and conditional independence are Bayesian networks. A Bayesian network $BN$ for a set of propositions $A$ is a tuple $BN = (A, E, P)$ such that $(A, E)$ is a directed acyclic graph and $P$ is a probability function that obeys

$$\{a\} \perp\!\!\!\perp_{\mathrm{P}} \mathsf{nd}(a) \mid \mathsf{pa}(a) \quad \text{(for every } a \in A\text{)}, \tag{1}$$

expressing that each vertex $a$ is conditionally independent of its non-descendants $\mathsf{nd}(a)$, given the values of its parents $\mathsf{pa}(a)$. Condition (1) is also called the *local Markov property*. Due to this property, the probability function $P$ can be decomposed into conditional probability functions for each node $a \in A$.

*Example 1.* We adapt an example on medical diagnosis, cf. [25]. Consider the propositions $A = \{a, b, c, d, e\}$ with the informal interpretations

| | | | |
|---|---|---|---|
| $a$ | cancer | $b$ | increased serum calcium level |
| $c$ | brain tumor | $d$ | coma |
| $e$ | headache | | |

and a Bayesian network $BN_{\mathrm{med}} = (A, E, P)$ with $(A, E)$ given as depicted in Fig. 1. It follows that $P$ has to adhere to the conditional independence $\{b\} \perp\!\!\!\perp_{\mathrm{P}} \{c\} \mid \{a\}$ (among others). Moreover, the probability of a possible world such as $ab\overline{c}\overline{d}e$ can be written as

$$P(ab\overline{c}\overline{d}e) = P(e \mid \overline{c}) \cdot P(\overline{d} \mid b\overline{c}) \cdot P(\overline{c} \mid a) \cdot P(b \mid a) \cdot P(a).$$

Therefore, $P$ can be completely described by e. g. the following assignments[1]:

$$\begin{aligned}
P(a) &= 0.20 & & \\
P(b \mid a) &= 0.80 & P(b \mid \overline{a}) &= 0.20 \\
P(c \mid a) &= 0.20 & P(c \mid \overline{a}) &= 0.05 \\
P(e \mid c) &= 0.80 & P(e \mid \overline{c}) &= 0.60 \\
P(d \mid b \wedge c) &= 0.80 & P(d \mid b \wedge \overline{c}) &= 0.90 \\
P(d \mid \overline{b} \wedge c) &= 0.70 & P(d \mid \overline{b} \wedge \overline{c}) &= 0.05
\end{aligned}$$

Note that the probabilities of negated variables derive from the above equations via e. g. $P(\overline{e} \mid c) = 1 - P(e \mid c)$. By only defining the above conditional probabilities the function $P$ can be compactly stored.

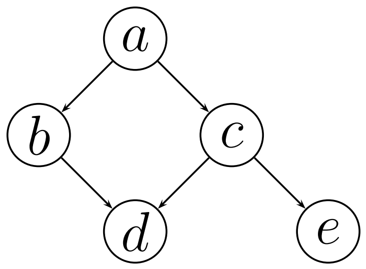


**Fig. 1.** The graph $(A, E)$ from Ex. 1

**Probabilistic Conditional Logic and Maximum Entropy** Conditional logic [22] is a knowledge representation formalism that concentrates on the role of *conditionals* or *if-then-rules*. A conditional of the form $(\psi \mid \phi)$ connects some detached pieces of information $\phi, \psi$ and represents a rule "*If $\phi$ then (usually, probably) $\psi$*". A *probabilistic conditional* is an expression of the form $(\psi \mid \phi)[d]$ with propositional formulas $\phi$ and $\psi$ and $d \in [0, 1]$.

*Example 2.* The well-known penguin example that illustrates the problem of exceptions in subclasses can be represented as a knowledge base $\mathcal{R}$ with $\mathcal{R} = \{r_1, r_2, r_3\}$ with

$$r_1 = (bird \mid peng)[1.0] \qquad r_2 = (fly \mid bird)[0.9] \qquad r_3 = (fly \mid peng)[0.01].$$

A probability function $P$ satisfies a probabilistic conditional

$$P \models (\psi \mid \phi)[d] \quad \text{if and only if} \quad P(\psi \mid \phi) = d \text{ and } P(\phi) > 0. \tag{2}$$

[1] The numbers have been arbitrarily chosen and may not describe the real world.

Reasoning in probabilistic conditional logic can be performed by maximizing entropy. The entropy $H(P)$ of a probability function $P$ is defined via $H(P) = -\sum_{\omega} P(\omega) \log P(\omega)$ with $0 \cdot \log 0 = 0$. By selecting $P^* = \arg\max_{P \models \mathcal{R}} H(P)$ as the (unique) model of the knowledge base $\mathcal{R}$ with maximal entropy, one obtains a probability function that both satisfies all conditionals in $\mathcal{R}$ and adds as few additional information (in the information-theoretic sense) as possible. For a consistent knowledge base $\mathcal{R}$ the maximum entropy model $P^*$ is uniquely determined, cf. [16]. Model-based probabilistic inference via $P^*$ shows excellent logical properties [16], and has been proved to be most adequate for commonsense reasoning [24].

## 2.2 Relational Approaches

**Bayesian Logic Programs** *Bayesian logic programming* combines logic programming and Bayesian networks [19]. The basic structure for knowledge representation in Bayesian logic programs are *Bayesian clauses* which model probabilistic dependencies between *Bayesian atoms* as in the following BLP corresponding to an example given in [4]:

$$\begin{array}{llll} c_1: & (\mathit{likes}(X,Y) \mid \mathit{elephant}(X),\ \mathit{keeper}(Y)) & c_3: & \mathit{likes}(\mathit{clyde},\mathit{fred}) \\ c_2: & (\mathit{likes}(X,\mathit{fred}) \mid \mathit{elephant}(X)) & & \end{array}$$

For each Bayesian clause $c$, a function $\mathsf{cpd}_c$ must be defined, expressing the conditional probability distribution $P(\mathsf{head}(c) \mid \mathsf{body}(c))$ and thus partially describing an underlying probability distribution $P$. For instance, $\mathsf{cpd}_{c_1}(\mathsf{true},\mathsf{true},\mathsf{true}) = 0.8$ would express our subjective belief that $\mathit{likes}(X,Y)$ is true with probability 0.8 if $\mathit{elephant}(X)$ and $\mathit{keeper}(Y)$ are true. In order to aggregate probabilities that arise from applications of different Bayesian clauses with the same head, BLPs make use of *combining rules*. Semantics are given to Bayesian logic programs via transformation into propositional forms, i. e. into Bayesian networks [25] (see [19] for details).

**Markov Logic Networks** *Markov logic* [5] establishes a framework which combines Markov networks [25] with first-order logic to handle a broad area of statistical relational learning tasks. The Markov logic syntax complies with first-order logic where each formula is quantified by an additional weight value, e.g.

$$\begin{array}{lr} (\mathit{elephant}(X) \wedge \mathit{keeper}(Y) \Rightarrow \mathit{likes}(X,Y), & 2.2) \qquad (\mathit{likes}(\mathit{clyde},\mathit{fred}),\ \infty) \\ (\mathit{elephant}(X) \Rightarrow \mathit{likes}(X,\mathit{fred}), & -0.8) \end{array}$$

Semantics are given to sets of Markov logic formulas by a probability distribution over propositional possible worlds that is calculated as a log-linear model over weighted ground formulas. The fundamental idea in Markov logic is that first-order formulas are not handled as hard constraints (which are indicated by weight $\infty$), but each formula is more or less softened depending on its weight. A *Markov* *logic network (MLN)* $L$ is a set of weighted first-order logic formulas $(F_i,\, w_i)$ together with a set of constants $C$. The semantics of $L$ is given by a ground Markov network $M_{L,C}$ constructed from $F_i$ and $C$ [6]. The standard semantics of Markov networks [25] is used for reasoning, e.g. to determine the consequences of $L$ (see [6] for details).

**Relational Maximum Entropy** The syntax of *relational probabilistic conditional logic* (RPCL) [33] has already been used in the representation of the elephant-keeper-example from the introduction and employs conditionals of the form $(B \,|\, A)[x]$ with first-order formulas $A, B$ and $x \in [0,1]$. A conditional $(B \,|\, A)[x]$ represents a constraint on a probability distribution $P : \Omega \to [0,1]$ on the set of possible worlds $\Omega$ and expresses that the conditional probability of $B$ given $A$ is $x$. In order to interpret conditionals containing free variables several relational semantics have been proposed, see [33, 21]. The *grounding semantics* [21] uses a *grounding operator* $\mathcal{G}$, e. g. universal instantiation, that translates a set $\mathcal{R}$ of conditionals with free variables into a set of ground conditionals. Then, a probability distribution $P$ *$\mathcal{G}$-satisfies* $\mathcal{R}$, denoted by $P \models_{\mathcal{G}} \mathcal{R}$, iff $P(B' \,|\, A') = x$ for every ground $(B' \,|\, A')[x] \in \mathcal{G}(\mathcal{R})$. Both *averaging* and *aggregating semantics* [18, 33] do not require a grounding operator but interpret the intended probability $x$ of a conditional with free variables only as a guideline for the probabilities of its instances, but the actual conditional probabilities of the instantiated formulas may differ from $x$. More precisely, for the averaging semantics, a probability distribution $P$ *$\varnothing$-satisfies* $\mathcal{R}$, denoted by $P \models_{\varnothing} \mathcal{R}$, iff for every $(B \,|\, A)[x] \in \mathcal{R}$ it holds that $P(B_1 \,|\, A_1) + \ldots + P(B_n \,|\, A_n) = nx$ where $(B_1 \,|\, A_1), \ldots, (B_n \,|\, A_n)$ are the ground instances of $(B \,|\, A)$. In the aggregating semantics, a probability function $P$ *$\odot$-satisfies* $\mathcal{R}$, denoted by $P \models_{\odot} \mathcal{R}$, iff for every $(B \,|\, A)[x] \in \mathcal{R}$ it holds that $P(B_1 \wedge A_1) + \ldots + P(B_n \wedge A_n) = x(P(A_1) + \ldots + P(A_n))$ where $(B_1 \,|\, A_1), \ldots, (B_n \,|\, A_n)$ are the ground instances of $(B \,|\, A)$. Note that all these three semantics are extensions of classical probabilistic semantics for propositional probabilistic conditional logic [15]. Based on any of these semantical notions the *principle of maximum entropy* ([23, 15], see also Sec. 2.1), can be used for reasoning. By employing this principle one can determine the unique probability distribution that is the optimal model for a consistent knowledge base $\mathcal{R}$ in an information-theoretic sense via

$$P_{\mathcal{R}}^{ME_\circ} = \arg \max_{P \models_\circ \mathcal{R}} \mathcal{H}(P) \tag{3}$$

with $\circ$ being one of $\mathcal{G}$, $\varnothing$, or $\odot$. We abbreviate the approaches of reasoning based on the principle of maximum entropy with grounding, averaging, and aggregating semantics with $\mathrm{ME}_{\mathcal{G}}$, $\mathrm{ME}_{\varnothing}$, and $\mathrm{ME}_{\odot}$, respectively. We say that a formula $(B \,|\, A)[x]$ is $\circ$-inferred from $\mathcal{R}$ iff $P_{\mathcal{R}}^{ME_\circ} \models_\circ (B \,|\, A)[x]$ with $\circ$ being one of $\mathcal{G}$, $\varnothing$, or $\odot$.

## 3 Systems

There are various implementations of probabilistic logic. In the following, we give brief overviews on a selection of systems and toolboxes for probabilistic logics based on a propositional or a relational setting.

### 3.1 SPIRIT and Probabilistic Reasoning under Maximum Entropy

Reasoning in probabilistic conditional logic by employing the principle of maximum entropy [23, 15] requires solving the numerical optimization problem $P^* = \arg\max_{P \models \mathcal{R}} H(P)$ (cf. Sec 2.1). SPIRIT [29] is an expert system shell[2] implementing maximum entropy reasoning and solving this optimization problem. In order to tame the complexity of the optimization task which grows exponentially in the number of variables, SPIRIT generates a junction-tree of variable clusters, allowing to represent the global probability distribution by a product of marginal distributions. SPIRIT has been used successfully in various application domains, like medical diagnosis, project risk management, or credit scoring.

MEcore [8] is another system implementing reasoning for propositional probabilistic conditional logic under maximum entropy. While it does not employ a junction-tree modelling, but a straight-forward representation of the complete probability distribution, its focus is on flexibly supporting different basic knowledge and belief management functions like revising or updating probabilistic beliefs, or hypothetical reasoning in what-if mode.

### 3.2 Alchemy

Markov logic is implemented in the Alchemy system[3] [20]. Alchemy provides a wide range of functionalities for statistical relational learning and probabilistic logic inference. In particular, the consequences of a Markov logic network $L$ defined via the ground Markov network $M_{L,C}$ (cf. Sec. 2.2) can be determined. With respect to learning, both weight learning as well as learning the structure of an MLN is supported. Applications of MLN realized with Alchemy include classifications tasks and social network modelling. In Sec. 4, we will report on some experiments using MLNs and Alchemy in medical diagnosis.

### 3.3 ProbCog

The *ProbCog* (Probabilistic Cognition for Cognitive Technical Systems) system[4] [13] is a software suite for statistical relational learning. ProbCog currently supports three knowledge representation approaches: Bayesian Logic Networks (BLNs), Adaptive Markov Logic Networks (AMLNs), and Markov Logic Networks (MLNs). For each approach, ProbCog provides several learning and inference algorithms, implemented either in Java or Python. ProbCog provides a

[2] `http://www.fernuni-hagen.de/BWLOR/spirit/index.php`
[3] `http://alchemy.cs.washington.edu/`
[4] `http://ias.cs.tum.edu/research-areas/knowledge-processing/probcog`

sophisticated framework for relational data, which features, amongst others, a unified data model (which allows data conversion for all integrated approaches ) and the generation of synthetic data (for learning experiments). The main focus of the ProbCog suite is on providing a comprehensive library of algorithms and powerful data structures for statistical relational learning, but it also includes some graphical interfaces for learning and querying, respectively.

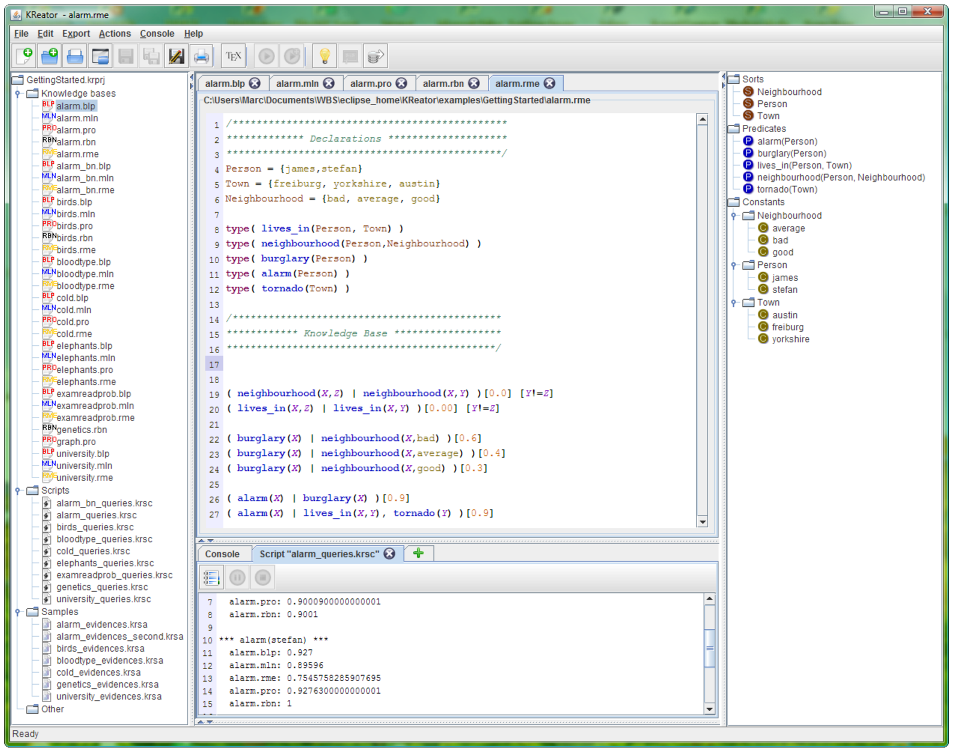
## 3.4 KReator

KReator [32] is a toolbox for representing, learning, and automated reasoning with various approaches combining relational first-order logic with probabilities[5].

The implementation of KReator is done in Java and mirrors its objective to support different approaches to relational probabilistic knowledge representation and reasoning. It strictly separates the internal logic and the user interface, employing an abstract command structure allowing easy modifications on both sides. In order to support the implementation of other approaches, KReator features a large library on first-order logic and basic probabilistic methods. Among others this library contains classes for formulæ, rules, conditionals, and various methods to operate on these. There is also a rudimentary implementation of Prolog available that can be used for specifying background knowledge as e. g. in BLPs. This integrated library is designed to support a fast implementation of specific approaches to statistical relational learning. The task of integrating a new approach into the KReator system is supported by a small set of interfaces that have to be implemented in order to be able to access the new approach from the user interface. There are interfaces for knowledge bases (which demands e. g. support for querying), file writers and parsers (for reading and writing the specific syntax of an approach), and learner. One thing to note is that both file writers and parsers have to work on strings only, all the cumbersome overhead of file operations and I/O is handled by KReator. With the help of a plugin-like architecture the developer of a new approach only has to be concerned with connecting her approach to KReator using these interfaces. Then all the benefits of an integrated development environment as provided by KReator are immediately accessible. Currently, KReator supports knowledge representation using BLPs, MLNs, the relational maximum entropy approach RME, as well as Relational Bayesian Networks [12], and Probabilistic Prolog [27]; support for Logical Bayesian Networks [7], Probabilistic Relational Language [10], and Relational Bayesian Networks [14] is in preparation.

Performing inference on MLNs is done using the Alchemy software package [20], a console-based tool for processing Markov logic networks. For BLPs, a reasoning component was implemented within KReator. To process ground $\mathrm{ME}_{\mathcal{G}}$ knowledge bases, KReator uses a so-called ME-adapter to communicate with a MaxEnt-reasoner. Currently, such adapters are supplied for the SPIRIT reasoner [29] and for MECoRe [8] which are tools for processing (propositional) conditional probabilistic knowledge bases using maximum entropy methods.

[5] `http://kreator.cs.tu-dortmund.de/`



```
/*********************************************
************* Declarations ********************
*********************************************/
Person = {james,stefan}
Town = {freiburg, yorkshire, austin}
Neighbourhood = {bad, average, good}

type( lives_in(Person, Town) )
type( neighbourhood(Person,Neighbourhood) )
type( burglary(Person) )
type( alarm(Person) )
type( tornado(Town) )

/*********************************************
************ Knowledge Base ******************
*********************************************/


( neighbourhood(X,Z) | neighbourhood(X,Y) )[0.0]  [Y!=Z]
( lives_in(X,Z) | lives_in(X,Y) )[0.00]  [Y!=Z]

( burglary(X) | neighbourhood(X,bad) )[0.6]
( burglary(X) | neighbourhood(X,average) )[0.4]
( burglary(X) | neighbourhood(X,good) )[0.3]

( alarm(X) | burglary(X) )[0.9]
( alarm(X) | lives_in(X,Y), tornado(Y) )[0.9]
```

```
  alarm.pro: 0.9000900000000001
  alarm.rbn: 0.9001

*** alarm(stefan) ***
  alarm.blp: 0.927
  alarm.mln: 0.89596
  alarm.rme: 0.7545758285907695
  alarm.pro: 0.9276300000000001
  alarm.rbn: 1
```

**Fig. 2.** Graphical user interface of the KReator system

ProbCog and KReator share some similarities with respect to their general approach to gather different knowledge representation approaches within one software framework, e. g. both systems feature some sort of unified data model for evidence or sample data. But the primary application focus of both systems differs significantly: ProbCog is developed for its intended practical application and integration in cognitive technical systems. So its primary focus is on providing a versatile and efficient framework for that specific purpose, therefore some sort of unified graphical user interface to the framework is not needed. In contrast, KReator's focus is on the typical workflow of a knowledge engineer, researcher, or developer. Therefore, KReator collects different approaches in an integrated graphical development environment (see Fig. 2 for a screenshot of the KReator user interface) to provide easy access to typical tasks and provides a plugin interface to support the study and development of further approaches.

## 4 Applications

In the following subsections, we will present three practical application scenarios of some of the afore described systems. All three applications cover settings from the medical domain. The first one illustrates ME-reasoning using a fictitious example, whereas the other ones describe learning experiments involving Markov logic networks and real-world data from medical studies.

## 4.1 Knowledge Processing with the MEcore system

In this section, we will illustrate how MECoRe can process incomplete, uncertain knowledge expressed by a probabilistic knowledge base using a fictitious example from the medical domain. This example is taken from [8] and discusses the general treatment of a patient who suffers from a perilous bacterial infection. The infection will probably cause permanent neurological damage or even death if it is not treated appropriately. There are two antibiotics available that might be capable of ending the infection, provided that the bacteria are not resistant to the specific antibiotic. It must also be considered that each antibiotic might cause a life-threatening allergic reaction that could be especially dangerous for an already weakened patient. The resistance of the bacteria to a specific antibiotic can be tested, but each test is very time-consuming.

**Building Up the Knowledge Base** The construction of the knowledge base starts with the definition of some binary variables that describe aspects concerning antibiotic A:

`med_A`: The patient is treated with antibiotic A.
`effect_A`: Antibiotic A is effective against the bacteria.
`allergic_A`: The patient is allergic to antibiotic A.
`resistance_A`: The bacteria are resistant to antibiotic A.
`posResT_A`: The test result suggests a resistance to antibiotic A.

Analogously, there are also five variables concerning antibiotic B. A three-valued variable `outcome` describes the three possible outcomes of the treatment:

`outcome`=`healthy`: The infection is treated successfully and the patient is healthy again.
`outcome`=`impaired`: The patient overcomes the infection but suffers a permanent damage to the nervous system.
`outcome`=`dead`: The infection is not treated effectively and the patient dies.

The available knowledge summarizing the previously made experiences about the infection and the two antibiotics is modeled by the knowledge base `medKB` $= \{R_1, \ldots, R_{22}\}$ consisting of the probabilistic rules given in Fig. 3.

The first four rules express very obvious correlations between the variables: $R_1$ and $R_2$ say that if a certain antibiotic is not administered or the bacteria are resistent to it, then this antibiotic has no effect. $R_3$ and $R_4$ assure that if the bacteria are not resistant to a certain antibiotic, then this antibiotic is effective if—and only if—it is administered. The facts $R_5$ to $R_9$ integrate statistical information available for antibiotic A and antibiotic B, i. e. some a priori probabilities, into the knowledge base: antibiotic B is twice as likely as antibiotic A to cause an allergic reaction ($R_5$, $R_6$); and the resistance to antibiotic B is

$R_1$ : ($\neg$effect_A | $\neg$med_A $\vee$ resistance_A)[1.00]
$R_2$ : ($\neg$effect_B | $\neg$med_B $\vee$ resistance_B)[1.00]
$R_3$ : (effect_A $\Leftrightarrow$ med_A | $\neg$resistance_A)[1.00]
$R_4$ : (effect_B $\Leftrightarrow$ med_B | $\neg$resistance_B)[1.00]
$R_5$ : (allergic_A)[0.10]
$R_6$ : (allergic_B)[0.20]
$R_7$ : (resistance_A)[0.01]
$R_8$ : (resistance_B)[0.09]
$R_9$ : (med_A $\wedge$ med_B)[0.00001]
$R_{10}$: (outcome = dead | $\neg$med_A $\wedge$ $\neg$med_B)[0.10]
$R_{11}$: (outcome = healthy | $\neg$med_A $\wedge$ $\neg$med_B)[0.10]
$R_{12}$: (posResT_A | resistance_A)[0.97]
$R_{13}$: ($\neg$posResT_A | $\neg$resistance_A)[0.99]
$R_{14}$: (posResT_B | resistance_B)[0.90]
$R_{15}$: ($\neg$posResT_B | $\neg$resistance_B)[0.80]
$R_{16}$: (outcome = dead | med_A $\wedge$ allergic_A)[0.99]
$R_{17}$: (outcome = dead | med_B $\wedge$ allergic_B)[0.40]
$R_{18}$: (outcome = healthy | effect_A)[0.8]
$R_{19}$: (outcome = healthy | effect_B)[0.7]
$R_{20}$: (allergic_A | med_A)[0.10]
$R_{21}$: (outcome = dead | effect_B)[0.09]
$R_{22}$: (outcome = healthy | med_B $\wedge$ allergic_B)[0.001]

**Fig. 3.** Probabilistic rules in the knowledge base medKB

nine times higher compared to antibiotic A ($R_7$, $R_8$). It has occurred very rarely that somebody administers both antibiotics to the patient ($R_9$). $R_{10}$ and $R_{11}$ model the prognosis for the patient if no antibiotic is administered. The result of a resistance-test, testing the resistance of the bacteria to an antibiotic, always includes some error, but the test regarding antibiotic A is very reliable ($R_{12}$, $R_{13}$); whereas the test concerning antibiotic B has a somewhat lower sensitivity ($R_{14}$) and a considerably lower specificity ($R_{15}$).

The rules $R_{16}$ to $R_{19}$ express special knowledge about antibiotic A and antibiotic B, respectively: The allergic reaction caused by antibiotic A is most likely lethal ($R_{16}$), whereas the chance of surviving an allergy to antibiotic B is more likely than to die of it ($R_{17}$). If antibiotic A is effective, then the patient has a good chance to become healthy again ($R_{18}$), whereas the effectiveness of antibiotic B is somewhat lower ($R_{19}$). The following knowledge is available for antibiotic A only: $R_{20}$ makes clear that the a priori probability of an allergy to antibiotic A (expressed by $R_5$ with equal probability) is not affected by the administration of antibiotic A. There is also some exclusive knowledge about antibiotic B: If antibiotic B is effective, there still remains some risk to die of the infection ($R_{21}$). If the patient survives an allergic reaction caused by antibiotic B, it is very unlikely that he will become healthy again ($R_{22}$).

**Computing an Initial Epistemic State** In MECoRe, the computation of an epistemic state incorporating the knowledge expressed by the knowledge base `medKB` can be initiated by the command:

(1) $\texttt{currState} := \texttt{epstate.initialize}(\texttt{medKB});$

The calculated epistemic state `currState` represents the incomplete knowledge expressed by `medKB` inductively completed in an entropy-optimal way.

A closer look at `medKB` reveals that some additional rules can be logically deduced from the existing rules since they hold in all models satisfying `medKB`. For instance, a literal of the three-valued variable `outcome` makes up the conclusion of several rules. Hence, two rules with identical premise and an `outcome` literal as conclusion directly imply a corresponding third rule, e. g. $R_{10}$ and $R_{11}$ imply $(\texttt{outcome} = \texttt{impaired} \mid \neg\texttt{med\_A} \wedge \neg\texttt{med\_B})[0.8]$. Appropriate queries to MECoRe in `currState` yield these expected probabilities since reasoning at optimum entropy is compatible with classical probabilistic consequences.

**Query** Suppose we want to know the patient's chances in each case of treatment, i. e. for each of the four possible options of medical administration: no antibiotic, antibiotic A only, antibiotic B only, both antibiotics. This can be expressed by a set of twelve query formulas (i. e. conditionals of the form e. g. $(\texttt{outcome} = \texttt{healthy} \mid \texttt{med\_A} \wedge \neg\texttt{med\_B})$) which we will denote by `medQueries`. While using classical probabilistic consequences does not yield informative answers for `medQueries`, MECoRe infers the following probabilities from `currState`:

| | healthy | impaired | dead |
|---|---|---|---|
| no antibiotic | 0.10 | 0.80 | 0.10 |
| only A | 0.79 | 0.06 | 0.15 |
| only B | 0.65 | 0.23 | 0.12 |
| A and B | 0.94 | 0.02 | 0.04 |

These results clearly suggest that the combined administration of both antibiotics would be the best treatment. It offers a high chance of healing accompanied by a minimal risk of permanent neurological damage or death. However, a closer look at the knowledge base reveals that it implies that there is almost no possible drug interaction. For instance, asking for the degree of belief for the conditional

$\texttt{C}_{\texttt{int}} : (\texttt{dead} \mid \texttt{med\_A} \wedge \texttt{med\_B} \wedge \neg\texttt{allergic\_A} \wedge \neg\texttt{allergic\_B})$

in `currState` yields the inferred drug interaction probability 0.01.

**Incorporation of New Knowledge** Suppose that later on, the doctors learn to know from an outside source that there is a severe risk (0.25) of a deadly drug interaction between both antibiotics. Executing

(2) $\texttt{currState.update}(\texttt{medKB}, \texttt{C}_{\texttt{int}}[0.25]);$

incorporates this new knowledge into the current epistemic state as if it had been available already in `medKB`. In fact, this kind of belief change is a *genuine revision* (cf. [17]) which in MECoRe can also be more easily expressed by

(2') `currState.revise(`$\texttt{C}_{\texttt{int}}$`[0.25]);`

Now, asking the `medQueries` again, the probabilities have changed considerably (cf. Fig. 4(a)): With the knowledge about a deadly drug interaction, the probabilities show that the administration of antibiotic A maximizes the patient's chance to become healthy again.

| **(a)** | healthy | impaired | dead |
|---|---|---|---|
| no antibiotic | 0.10 | 0.80 | 0.10 |
| only A | 0.79 | 0.06 | 0.15 |
| only B | 0.65 | 0.23 | 0.12 |
| A and B | 0.70 | 0.02 | 0.28 |

| **(b)** | healthy | impaired | dead |
|---|---|---|---|
| no antibiotic | 0.10 | 0.80 | 0.10 |
| only A | 0.79 | 0.06 | 0.15 |
| only B | 0.69 | 0.21 | 0.10 |
| A and B | 0.76 | 0.02 | 0.22 |

| **(c)** | healthy | impaired | dead |
|---|---|---|---|
| no antibiotic | 0.10 | 0.80 | 0.10 |
| only A | 0.43 | 0.15 | 0.42 |
| only B | 0.65 | 0.23 | 0.12 |
| A and B | 0.32 | 0.05 | 0.63 |

| **(d)** | healthy | impaired | dead |
|---|---|---|---|
| no antibiotic | 0.10 | 0.80 | 0.10 |
| only A | 0.43 | 0.15 | 0.42 |
| only B | 0.54 | 0.26 | 0.20 |
| A and B | 0.20 | 0.04 | 0.76 |

**Fig. 4.** Probabilities for `medQueries` infered by MEcoRe

**What-If-Analysis** It has to be noticed that the knowledge used for generating the epistemic state `currState` says that no resistance tests have been performed, i. e. for neither of the antibiotics any resistance test results are available. A what-if-analysis can be used to analyze what changes would occur if a negative resistance-test result concerning antibiotic B was known. That is, could this test result make antibiotic B the better choice for treatment? In MEcoRe, such a what-if-analysis is accomplished by

(3) `currState.whatif((¬posResT_B)[1.0], medQueries);`

delivering the results shown in Fig. 4(b). The probabilities show that even a negative resistance-B test would not change the general decision to administer antibiotic A. This result is, amongst others, caused by the low resistance-B test specificity.

Another what-if-analysis revealing the effects of a positive resistance-A-test

(4) `currState.whatif((posResT_A)[1.0], medQueries);`

yields the probabilities given in Fig. 4(c). This shows that a test-result suggesting the resistance to antibiotic A would change the situation: In this case, a treatment with antibiotic B becomes the only that offers a realistic healing chance. This is not surprising, because a resistance-test result concerning antibiotic A is very reliable. So it is clearly advisable to perform the time-consuming resistance-A test.

In case of a positive resistance-A-test result, would it also be helpful to test the resistance to antibiotic B? That is, could an additional positive resistance-B-test change the decision to administer antibiotic B? Hypothetical reasoning

(5) `currState.whatif(((posResT_A)[1.0], (posResT_B)[1.0]), medQueries);`

yields the results shown in Fig. 4(d), indicating that even a positive resistance-B-test would not change the decision to administer antibiotic B. So it is not helpful to perform a resistance-B test in any situation, since its result would never change the decision that had been made without knowing the test result.

## 4.2 Diagnosis of Lung Cancer

This section is based on [9], reporting on a case study of using probabilistic relational modelling and learning as provided by MLNs and the MLN system Alchemy [20] in the field of biomedical diagnosis. The idea behind this diagnostical setting is to support diagnosis of bronchial carcinoma on the basis of the substances a person exhales [2, 1]. In this setting, the focus is on the early detection of bronchial carcinoma by ion mobility spectrometry, a non-invasive diagnostic method which delivers results within a few minutes and can be applied at low costs.

**Ion Mobility Spectrometry** In order to determine chemical substances in gaseous analytes, ion mobility spectrometry (IMS) can be used [2]. This method relies on characterizing substances in gases by their ion mobility. Figure 5 illustrates the working principle of an ion mobility spectrometer. After ionisation, ion swarms enter the drift region through an ion shutter. The time needed to pass the drift region is called *drift time*, and the ion mobility is inversely proportional to the drift time. An ion mobility spectrum is obtained by mapping the drift time to the signal intensity measured at the Faraday plate (cf. Fig. 5). If the gaseous analyte contains various substances, they may reach the Faraday plate at the same time. In order to avoid this, a multi capillary column is used for the pre-separation of different substances [2] so that they enter the spectrometer at different time points, called *retention times*; for more detailed descriptions of ion mobility spectrometry and its working principle we refer to [1] or [2].

Thus, applying ion mobility spectrometry to gaseous analytes yields IMS spectra where a peak in such a spectrum corresponds to a particular substance. The determination of peaks in a measurement requires sophisticated processing of the raw spectra, see [2, 3] for details. Peak objects taken from two different measurements that correspond to the same substance occur at corresponding areas in their respective so-called *heat maps*, and in order to identify such corresponding peaks, they can be mapped to *peak clusters* [2, 9]. In our case study, we investigated an IMS database consisting of 158 measurements obtained from screening the breath of 158 patients out of which 82 had lung cancer (*bronchial carcinoma, bc*), yielding a database $D_{bc}$ with 33 peak clusters, in the following referred to by the identifiers $pc0, \ldots, pc32$. For each peak cluster $pc_i$, $P(bc|pc_i)$ denotes the conditional probability that a measurement having a peak belonging to $pc_i$ stems from a person having bronchial carcinoma. For applying methods of probabilistic relational modelling and learning to $D_{bc}$, we use a logic representation of $D_{bc}$ (for convenience, also referred to as $D_{bc}$) involving the predicates

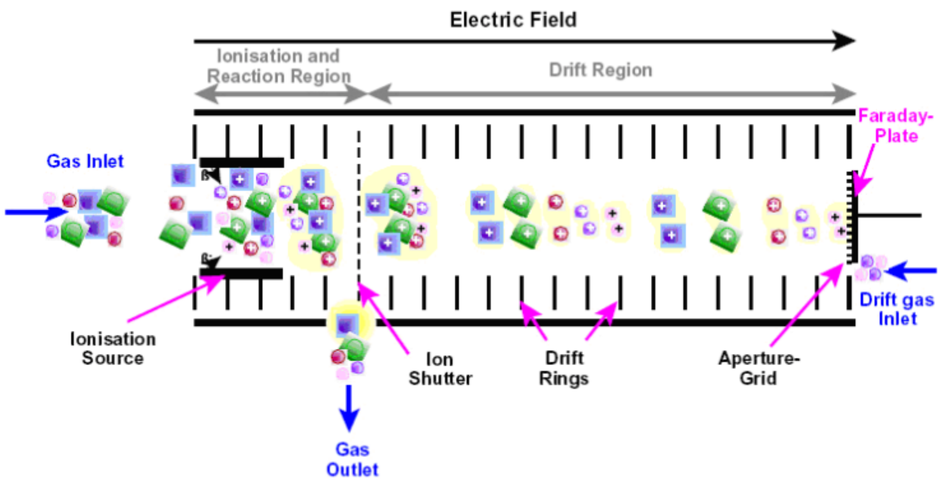


**Fig. 5.** Schematic overview of an ion mobility spectrometer (from [2])

$bc(M)$ indicating that measurement $M$ belongs to a person having lung cancer and $pcInM(PC, M)$ stating that peak cluster $PC$ occurs in measurement $M$.

In the following, we present different setups to learn MLNs from the data set $D_{bc}$. Our goal is to calculate the probability that a certain measurement $m$ is from some person with a bronchial carcinoma, given the information for each of the 33 peak clusters whether or not it is contained in measurement $m$. That is, we want to calculate the conditional probability of $bc(m)$, given the truth values of the literals $pcInM(pc0, m)$, ..., $pcInM(pc32, m)$. This conditional probability helps to classify patients with respect to suffering from lung cancer. The corresponding classification task can be realized with MLNs. We use the software package Alchemy [20] which provides several sophisticated algorithms to perform (structure and parameter) learning and inference of MLNs. A learned MLN is validated in terms of classification accuracy, defined as the proportion of the correctly predicted (positive and negative) results on the total number of measurements in a testing set; these values are determined as the average accuracy of all tests in a 10-fold cross-validation.

**Learning Logic Rules with the ILP System Aleph** In a first learning setup, we use the inductive logic programming (ILP) system *Aleph* [31] for learning first-order logic rules from the data set. Besides other parameters, Aleph allows to make detailed specifications about which atoms may appear in the body or head of a rule. As we want to predict whether or not the measurement M belongs to a patient having bronchial carcinoma, we require that heads of the rules learned by Aleph must contain the *bc* predicate, whereas their body must consist of one or more atoms of the *pcInM* predicate, with a constant in the first argument.

This way, the rules predict the value of $bc(M)$, given the values of some of the $pcInM(pc_i, M)$. The two rules

$$\begin{array}{ll} R_1\text{:} & pcInM(pc5, M) \wedge pcInM(pc8, M) \qquad\qquad\qquad\qquad \Rightarrow bc(M) \\ R_2\text{:} & pcInM(pc7, M) \wedge pcInM(pc17, M) \wedge pcInM(pc31, M) \Rightarrow bc(M) \end{array}$$

are examples of the 11 rules learned with Aleph [9]. The premises of all 11 rules consist of conjunctions of at most three positive *pcInM* literals. From the 33 different peak clusters found in the data set, only 18 occur in the rule set, so the other 15 peak clusters seem to carry no useful information with regard to lung cancer according to the Aleph result.

**Learning Weights of Aleph Formulas with Alchemy** In a subsequent step, we take the Aleph implications as logical base structure of an MLN and learn appropriate weights for them from the data set using Alchemy. For instance, the resulting weights for the rules $R_1$ and $R_2$ above are 4.596 and 6.004, respectively. Evaluating the MLN prediction performance results in an accuracy of 78%.

If we take the implications as if-then-rules, we can determine the conditional probabilities of these rules under the distribution induced by the MLN, i. e. we use Alchemy to calculate the conditional probability of a rule's consequent ground atom given its premise ground atoms as evidence. E. g., for rule $R_1$, Alchemy determines the probability $P(bc(m)|pcInM(pc5, m) \wedge pcInM(pc8, m)) = 0.9800$ in the MLN; for $R_2$ we get 0.996. In fact, the conditional probabilities of all rules are not exactly 1.0, as expected, but rather close to it (see [9]). This is due to the fact that Alchemy performs approximate inference and thereby, as a side-effect, prevents overfitting.

The learned MLN allows to draw some conclusions between peak clusters (i. e. the occurrence of substances in a measurement) and bronchial carcinoma. E. g., formula $R_2$ relates the combined occurrence of peak clusters *pc7*, *pc17*, and *pc31* in a measurement $M$ to the presence of bronchial carcinoma. Because of the positive (and relative high) weight of this formula, the combined occurrence of these peak clusters can be interpreted as an indicator for bronchial carcinoma. Likewise, there are also formulas relating the combined occurrence of peak clusters to the absence of bronchial carcinoma.

**Simple Classification with MLNs** In a further learning setup, we predefine the formula structure of a quite simple MLN: The MLN consists of the 33 implications $pcInM(pc0, M) \Rightarrow bc(M), \ldots, pcInM(pc32, M) \Rightarrow bc(M)$. Since the Alchemy syntax allows to express such "partially grounded" formulas in a compact way, the whole predefined structural Alchemy input merely consists of a single line. With this MLN structure, we follow a straightforwardly modelled classification approach: To classify the *bc* state of a measurement, we consider each peak cluster separately, leaving out any connections or dependencies among them. To some extent, this approach resembles Naive Bayes classification, where explicit independence assumptions among classifying attributes are made. The evaluation of the learned MLN revealed quite a high accuracy of 88% [9], although the enforced MLN structure lacks any connections between peak clusters, suggesting that those connections are not of great importance for classifying the measurements regarding *bc*.

**MLN Structure Learning** In this learning setup, we make use of Alchemy's structure learning feature to learn an MLN from scratch. Alchemy does not allow to make detailed specifications about the formulas to be learned, i. e. we cannot impose the requirement that the $pcInM(\_,\_)$ atoms have a constant in the first argument. As a consequence, Alchemy's structure learning algorithm produces no useful results when applied to $D_{bc}$ without any further information. So we modify the relational modelling in some aspect by replacing the binary predicate $pcInM(PC, M)$ by 33 unary predicates $pc0(M), \ldots, pc32(M)$.

Using this setup, the structure (and weight) learning with Alchemy starts from an empty MLN and results in an MLN with 89 formulas (including 34 atomic formulas for all 34 predicates) [9]. The evaluation of this MLN shows an accuracy of 90%. Compared to the previous results, this MLN models much more connections among the peak clusters and their combined influence regarding $bc(M)$. Only 13 of the 55 non-atomic formulas involve a *bc* literal, so the other 42 formulas express connections among the peak clusters regardless of the $bc(M)$ state, and the formulas contain both positive and negative peak cluster literals. So compared to the previous results, this MLN exhibits more complex and subtle connections among the occurrences of peak clusters and the $bc(M)$ state. Here are two examples for the learned formulas:

$$R_{61}: \quad (\neg pc10(M) \wedge pc14(M) \wedge \neg pc18(M) \wedge pc21(M) \Rightarrow \quad bc(M), \quad 7.15)$$
$$R_{44}: \quad (pc17(M) \wedge pc28(M) \Rightarrow pc21(M), \quad 5.05)$$

$R_{61}$ relates the combined occurrence of peak clusters *pc14* and *pc21* and the explicit absence of peak clusters *pc10* and *pc18* in a measurement to bronchial carcinoma. With a lower, but still relatively high weight, $R_{44}$ implies that a measurement containing peak clusters *pc17* and *pc28* also contains peak cluster *pc21*. In other words, the system has learned the relationship that the occurrence of the two substances indicated by peak clusters *pc17* and *pc28* in a measurement $M$ leads to the presence of the substance identified by *pc21* in the same measurement. Such a relation can provide interesting insights into the general composition of substances in typical measurements.

## 4.3 Predicting Allergic Diseases of Children

In this section, another application of MLNs for modelling and learning in the medical domain is presented. In [30], MLNs were employed to analyze the correlations between allergic diseases of children and certain environmental factors. The data used in this analysis has been extracted from the KiGGS study of the Robert Koch-Institut [28]. The KiGGS study is a long term study which covers the health situation of 17.000 children (and adolescents) in Germany. It considers a multitude of attributes for every child concerning medical or social aspects. For the experiments described in [30], 13 of these attributes had been chosen which represent well-known risk factors for allergies, e. g. *"the child has a pet at* *home"*, *"the child lives in an urban environment"*, or *"a parent suffers from an allergy"*. Each such attribute was modelled by a corresponding MLN predicate, e. g. $hasPet(X)$, $urban(X)$. Together with the information whether or not a child is allergic (represented by an $isAllergic(X)$ predicate) this allowed to model the data from the study as MLN learning data, i. e. as data samples in terms of ground atoms. The extracted and preprocessed learning data from the study consisted of about 8.000 data samples, covering allergic respectively non-allergic children in equal parts. In all experiments, subsets of these data samples were used as actual training and testing data (performing a 5-fold cross-validation).

Several learning experiments were performed on this learning data using the algorithms of the Alchemy software package [20] for learning and inference. The goal of all experiments was to learn an MLN which can predict the risk of a child to be allergic given the presence (or absence, respectively) of each of the 13 risk factors. The learning experiments included parameter (i. e. weight) learning using a predefined MLN formula structure which consisted of 13 implications of the form e. g. $hasPet(X) \Rightarrow isAllergic(X)$. In another experiment, Alchemy's structure learning algorithm was applied to learn an MLN (formulas and weights) from scratch. The evaluation of the learned MLNs was carried out by using several of Alchemy's (approximate) inference algorithms. Additionally, the software PyMLNs (which is part of the ProbCog suite [13]) was used to perform exact inference on some MLNs in order to evaluate the deviation compared to the approximate results. The experiments showed that the results of the various Alchemy algorithms were quite similar and that there were no significant difference compared to the exact results.

Overall, the quality of the learned MLNs in terms of classification accuracy turned out to be not as good a expected. For various experiment settings, the MLNs resulting from structure as well as from parameter learning provide an accuracy of about 61% in predicting a child to be allergic. This could be improved by focusing on formulas the probabilities of which were significantly different from 0.5. However, further investigations into the evaluation of the quality of learned MLNs for prediction tasks in this domain will be necessary.

## 5 Summary and Conclusion

This paper gives a brief overview on the state of the art in probabilistic reasoning, and illustrates the relevance of probabilistic methods for expert systems by describing their applications in various scenarios. The main advantage of probabilistic formalisms is an accurate handling of uncertainty which pervades all real world problems. Degrees of uncertainty can be conveniently obtained from statistical data and processed via probabilistic networks. Moreover, we go into more details on novel approaches combining probability theory and first-order logic which provide more expressive frameworks for probabilistic reasoning. The problem of incompleteness of knowledge is addressed by describing the information-theoretical principle of maximum entropy which might also be applied in first-order settings. Altogether, probabilistic frameworks provide suitable and rich environments for learning, modelling, and reasoning in expert systems.

## References

1. Baumbach, J., Bunkowski, A., Lange, S., Oberwahrenbrock, T., Kleinbölting, N., Rahmen, S., Baumbach, J.I.: IMS$^2$ – An integrated medical software system for early lung cancer detection using ion mobility spectometry data of human breath. J. of Integrative Bioinformatics 4(3) (2007)
2. Baumbach, J.I., Westhoff, M.: Ion mobility spectometry to detect lung cancer and airway infections. Spectroscopy Europe 18(6), 22–27 (2006)
3. Bödeker, B., Vautz, W., Baumbach, J.I.: Peak finding and referencing in MCC/IMS-data. International Journal for Ion Mobility Spectrometry 11(1-4), 83–87 (2008)
4. Delgrande, J.P.: On First-Order Conditional Logics. Artificial Intelligence 105(1–2), 105–137 (1998)
5. Domingos, P., Lowd, D.: Markov Logic: An Interface Layer for Artificial Intelligence. Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan and Claypool, San Rafael, CA (2009)
6. Domingos, P., Richardson, M.: Markov logic: A unifying framework for statistical relational learning. In: Getoor, L., Taskar, B. (eds.) Introduction to Statistical Relational Learning. MIT Press (2007)
7. Fierens, D., Blockeel, H., Ramon, J., Bruynooghe, M.: Logical Bayesian Networks. In: Dzeroski, S., Blockeel, H. (eds.) Proceedings of the 3nd international workshop on multi-relational data mining. pp. 19–30 (2004)
8. Finthammer, M., Beierle, C., Berger, B., Kern-Isberner, G.: Probabilistic reasoning at optimum entropy with the MEcore system. In: Lane, H.C., Guesgen, H.W. (eds.) Proceedings 22nd International FLAIRS Conference, FLAIRS'09. AAAI Press, Menlo Park, California (2009)
9. Finthammer, M., Beierle, C., Fisseler, J., Kern-Isberner, G., Baumbach, J.I.: Using probabilistic relational learning to support bronchial carcinoma diagnosis based on ion mobility spectrometry. International Journal for Ion Mobility Spectrometry 13, 83–93 (2010)
10. Getoor, L., Grant, J.: Prl: A probabilistic relational language. Machine Learning 62(1), 7–31 (2006)
11. Getoor, L., Taskar, B. (eds.): Introduction to Statistical Relational Learning. MIT Press (2007)
12. Jaeger, M.: Relational Bayesian Networks: A Survey. Electronic Transactions in Artificial Intelligence 6 (2002)
13. Jain, D., Mösenlechner, L., Beetz, M.: Equipping Robot Control Programs with First-Order Probabilistic Reasoning Capabilities. In: International Conference on Robotics and Automation (ICRA). pp. 3130–3135 (2009)
14. Jain, D., Waldherr, S., Beetz, M.: Bayesian Logic Networks. Tech. rep., IAS Group, Fakultät für Informatik, Technische Universität München (2009)
15. Kern-Isberner, G.: Characterizing the principle of minimum cross-entropy within a conditional-logical framework. Artificial Intelligence 98, 169–208 (1998)
16. Kern-Isberner, G.: Conditionals in nonmonotonic reasoning and belief revision. No. 2087 in Lecture Notes in Computer Science, Springer (2001)

17. Kern-Isberner, G.: Linking iterated belief change operations to nonmonotonic reasoning. In: Brewka, G., Lang, J. (eds.) Proceedings 11th International Conference on Knowledge Representation and Reasoning, KR'2008. pp. 166–176. AAAI Press, Menlo Park, CA (2008)
18. Kern-Isberner, G., Thimm, M.: Novel Semantical Approaches to Relational Probabilistic Conditionals. In: Proc. Twelfth International Conference on the Principles of Knowledge Representation and Reasoning (KR'10). pp. 382–392 (2010)
19. Kersting, K., De Raedt, L.: Bayesian logic programming: Theory and tool. In: Getoor, L., Taskar, B. (eds.) Introduction to Statistical Relational Learning. MIT Press (2007)
20. Kok, S., Singla, P., Richardson, M., Domingos, P., Sumner, M., Poon, H., Lowd, D., Wang, J.: The Alchemy System for Statistical Relational AI: User Manual. Department of Computer Science and Engineering, University of Washington (2008)
21. Loh, S., Thimm, M., Kern-Isberner, G.: On the problem of grounding a relational probabilistic conditional knowledge base. In: Proceedings of the 14th International Workshop on Non-Monotonic Reasoning (NMR'10). Toronto, Canada (May 2010)
22. Nute, D., Cross, C.: Conditional Logic. In: Gabbay, D., Guenther, F. (eds.) Handbook of Philosophical Logic, vol. 4, pp. 1–98. Kluwer Academic Publishers (2002)
23. Paris, J.: The uncertain reasoner's companion – A mathematical perspective. Cambridge University Press (1994)
24. Paris, J.: Common sense and maximum entropy. Synthese 117, 75–93 (1999)
25. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann (1998)
26. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann (1998)
27. Raedt, L.D., Kimmig, A., Gutmann, B., Kersting, K., Costa, V.S., Toivonen, H.: Probabilistic Inductive Querying Using ProbLog. Tech. Rep. CW 552, Department of Computer Science, Katholieke Universiteit Leuven, Belgium (June 2009)
28. Robert Koch-Institut: *Public Use File KiGGS, Kinder- und Jugendgesundheitssurvey 2003-2006*, Berlin (2008)
29. Rödder, W., Reucher, E., Kulmann, F.: Features of the expert-system-shell SPIRIT. Logic Journal of the IGPL 14(3), 483–500 (2006)
30. Schmaußer-Hechfellner, E.: Probabilistische logikbasierte Wissensmodellierung mit statistischen medizinischen Daten unter Verwendung von Lern- und Inferenzverfahren für Markov-Logik-Netze. Bachelor Thesis, Dept. of Computer Science, FernUniversität in Hagen (2011), (in German)
31. Srinivasan, A.: The Aleph Manual. `www.comlab.ox.ac.uk/activities/machinelearning/Aleph/` (2007)
32. Thimm, M., Finthammer, M., Loh, S., Kern-Isberner, G., Beierle, C.: A system for relational probabilistic reasoning on maximum entropy. In: Guesgen, H.W., Murray, R.C. (eds.) Proceedings 23rd International FLAIRS Conference, FLAIRS'10. pp. 116–121. AAAI Press, Menlo Park, California (2010)
33. Thimm, M., Kern-Isberner, G., Fisseler, J.: Relational probabilistic conditional reasoning at maximum entroy. In: Proceedings ECSQARU-11 (2011), (to appear)