# Approaches to Uncertain or Imprecise Rules - A Survey

Matthias Nickles and Davide Sottara

[1] Department of Computer Science,
University of Bath
Bath, BA2 7AY, United Kingdom
nickles@gmx.net
[2] DEIS, Facolta di Ingegneria,
Universita di Bologna, Bologna, Italy
dsotty@gmail.com

**Abstract.** With this paper we present a brief overview of selected prominent approaches to rule frameworks and formal rule languages for the representation of and reasoning with uncertain or imprecise knowledge. This work covers selected probabilistic and possibilistic logics, as well as implementations of uncertainty and possibilistic reasoning in rule engine software.

Keywords: *Rules, Uncertainty Reasoning, Imperfect Knowledge, Semantic Web, Knowledge Engineering*

## 1 Introduction

This survey paper presents a short overview of selected more or less prominent approaches to rule frameworks and formal rule languages for the representation of and reasoning with various kinds of uncertain, imprecise or ambiguous information. These properties of information are quite different and knowledge may be affected by one or more of them at the same time. After Smets ([25]), the term "imperfection"will be used as a general concept encompassing all kinds of them, since "uncertainty", which is also sometimes used in a general way, is actually a specific type of imperfection.

The knowledge an agent has about the world can often be conveniently encoded using formulas expressed in some logic-based language. In classical logic, a formula is either true or false. However, in many practical cases the truth of a formula might be unknown, unclear or uncertain, in which case we cannot assign it a definite truth value.

There are many possible reasons for such *imperfect knowledge*. Several attempts to outline the differences and classify them in a standard framework exist: among them, the already cited survey by Smets [25] but also, for example [11, 15, 17]. More recently, the W3C *Incubator Group on Uncertainty Reasoning for the Web* has defined an ontology (see [4]) for the representation of imperfection of information on the Web which is to some degree resembled in this paper. Even if there

is much philosophical debate, it is generally accepted that such imperfection can take on the following major forms (among others):

**Uncertainty** derives from a lack of knowledge about a fact or an event, be it past, present or future, even if the actual state of the world is known to belong to some set of alternatives. Uncertainty may be *aleatory*, i.e. intrinsically present in some random phenomenon so that the knowledge gap cannot be filled, or *epistemic*, i.e. due to a partial ignorance of the agent, for example because of missing, questionable or inconsistent data. Often, but not always, the degree of uncertainty can be measured in some statistical, *objective* way. In the epistemic case, or whenever a subjective judgement is adopted, the degree of uncertainty is typically denoted as the *degree of belief* of some agent in a statement. But the subjective belief of an agent can of course also comprise statistical information.

**Imprecision** arises when knowledge is as complete as it can be, but the terms used to denote it do not allow to identify the entity that are being referred in a precise and univocal way. This imprecision may lead to *ambiguity*, when there is more than one possible interpretation, *approximation*, when a class of entities is collapsed into one representative, and *vagueness*, when the boundaries of the definition of a concept are relaxed - usually *fuzzified* - in some way.

**Inconsistency** is a property of a knowledge base with *conflicting* information, such that there is no possible world it can describe. Conflicts have to be resolved, usually removing, ignoring or modifying part or all the conflicting information. Inconsistency may be a symptom of *incorrect* or noisy information. But of course there is possibly also erroneous information which does not lead to any (apparent) inconsistency. In this survey, we do not consider formal frameworks for inconsistency handling.

Notice that an imperfect representation of knowledge may be more concise, robust and less expensive to obtain than its perfect version. Consider for example the age-related version of the *Sorite Paradox*: if a person is young on one day, they will also be young the day after, until the day when they will be old. This is a paradox in classical logic unless a different adjective is defined for every day in the life of a person, but is perfectly acceptable in fuzzy logic, where one single property, "young", has a truth degree that varies continuously with age. Given the variety of sources, most knowledge based systems are likely to have to deal with some type of imperfection in the data they process: moreover, given the robustness/conciseness trade-off, a system handling imperfection without ignoring is more applicable and powerful than an idealized system which simply prunes away uncertainty and imprecision. Of all the possible applications, we will discuss an example class, chosen because it is usually a domain of traditional (from a logical point of view), i.e., "perfect" rule-based systems:

*Complex Event Processing* (CEP) is an emerging approach [19] based on the concept of event, that is here, a message notifying that some state has changed

in a system at a certain time. Many real-world systems (e.g. a stock markets in finance, a plant in chemistry, a human body in medicine, ...) generate dozens of such events of different types at high frequencies, but only a few are usually relevant at a given time. The challenge, then, is to filter, sort and analyze the events, possibly aggregating them in higher order events at different abstraction levels, so to extract only the relevant information. Uncertain formal rules can be used to reason about such events, and *reaction rules* are particularly suitable in the context of CEP since they can be used to trigger actions (including the generation of new events) when the current events match their preconditions. Event processing, however, may be affected by imperfection in different ways:

- If some events are partially unobservable, there may be uncertainty due to the missing data. Moreover, the conditions used to detect a complex event from simpler ones may not be certain (i.e. when detecting the insurgence of a disease from its symptoms). Finally, all event-based predictions are intrinsically uncertain.
- Events may be reported with imprecision, e.g. because the measurements are unreliable. It may also be convenient, if not necessary, to express some constraints - especially the temporal ones - with some degree of vagueness (e.g., event A happens more or less contemporarily with event B).
- There may be unexpected, conflicting combinations of events, especially in case of failures.

Being a relatively novel discipline, not many real-world applications of *imperfect* CEP exist yet, possibly because commercial rule-based systems are efficient at handling imprecision, especially in the fuzzy case, but still have serious limitation when processing uncertainty.

A further set of examples and case studies related to uncertain and imprecise information on the Web can be found in [4], along with a discussion on possible solution approaches.

The remainder of this paper is structured as follows: Section 2 deals with different models of imperfection, such as fuzzy sets and probability theory, focusing on how they have been applied to extend traditional logics in the formalization of rules. Section 3 discusses whether and how mainstream rule-based systems, both academic and commercial, support imperfect rules.

## 2 Probabilistic and Possibilistic Reasoning with Rules

The different types of imperfection are the domains of different theories: in particular, uncertainty is typically handled using probabilistic approaches, while possibilistic ones are used for imprecision. The theoretical backgrounds are far beyond the scope of this work, so the main techniques will just be recalled briefly, to focus more properly on their application in logic.

### 2.1 Probabilistic Approaches

#### Probability Theory

Probability theory is the mathematical theory of *random events*. The probability of an event $A$ is represented by a real number ranging from 0 (impossible event) to 1 (certain event), and is usually denoted as $Pr(A)$. If uncertainty is aleatory, this probability is normally estimated using a "frequentist" approach, by repeating observations of events in long-run experiments and choosing the ratio of favorable outcomes over the total number of experiments as the probability. In contrast to this view, with the *Bayesian* or epistemic interpretation of probability, which is the underlying theory for most approaches in Artificial Intelligence (especially in Machine Learning), probability is a measure of the belief in some hypothesis which is not necessarily grounded in any physical properties or empirical observations (but could be). Under this view, for a rational agent, probability is typically grounded in terms of betting behavior: $Pr(A)$ is here the amount of money that a rational agent would be indifferent to betting on the occurrence of event A. In order to calculate probabilities, an agent typically starts with some personal prior belief, which is then progressively updated using the application of *Bayes' rule* as new information is acquired.

Various logics with support for probabilistic reasoning (purely statistical approaches as well as Bayesian reasoning) have been developed. In the following, we will present a subset of those first-order languages which are able to represent logical rules in the sense of Horn clauses (see below). Our survey does not aim at a complete or representative list of such formal frameworks, but just at a list of hopefully interesting example approaches and as a starting point for further reading.

One of the most simple languages for rules, and the core of *RuleML* [2], is *Datalog* [8]. A *Datalog program* allows to contain rules, that is, clauses (disjunctions of literals) with at most one positive literal (*Horn clauses*), with certain restrictions (see below). Rules can thus be written in the form $H \longleftarrow B_1 \wedge ... \wedge B_n$, where $H$ and the $B_i$ are atoms (the Datalog notation follows that of Prolog and is thus slightly different).
So-called *extensional predicates* are fully defined in an extensional manner by lists of *facts* (positive ground unit clauses). *Intensional predicates* are in contrast defined entirely by rules. Extensional predicates correspond to relations ("tables") in extensional databases, intensional predicates correspond to intensional database relations. The difference between these two type of predicates in Datalog becomes very important for probabilistic extensions of Datalog, as explained further below.

Datalog can be seen both as a *deductive database system* (a database system which can derive new data (facts) using logical inference) and as a Prolog-like logic language. But in contrast to Prolog, functions within terms are not allowed, and every variable in the head of a clause must also appear in the body of this

clause (so-called *safe rules* or *range-restricted rules*). Variants of Datalog such as *stratified Datalog* imply further restrictions. Datalog allows for very effective query evaluation, and queries are ensured to terminate. Its expressivity covers relational algebra (roughly: the core of SQL), but goes beyond it, by means of so-called *recursive queries*.

*Probabilistic Datalog* ($Datalog_P$) [13] is an extension of Datalog which additionally allows for the probabilistic weighting of facts (but not - extensionally - of rules). Informally, the idea here is that each ground fact corresponds to an event in the sense of probability theory, and rules allow for boolean combinations of events and their probabilities. However, naively applied, this approach would lead to inconsistencies and other problems, since probability theory is not *truth-functional*, that is, values of complex expressions are not necessarily functions of the values of the constituents of these expressions. This is a problem which all formal approaches which aim for a combination of probability theory and logical calculi need to be aware of.

The semantics of $Datalog_P$ programs is a probabilistic *possible-world semantics*. "Possible worlds" correspond to subsets of the least Herbrand model of the respective Datalog program, and a probability structure provides a probability distribution over all possible worlds. This uncertainty enabled possible-world semantics is typically - but not always - used for probabilistic logics, and reflects the aforementioned view of probabilities as degrees of belief. Essentially, a possible-world semantics assigns probabilities to propositions, and the probability of a certain proposition is the probability of the set of possible worlds where this particular proposition is true.

In order to deal with the absence of truth functionality, $Datalog_P$ follows two alternative directions: basic $Datalog_P$ yields probability intervals instead of "point" probabilities in case of derived event expressions (which are not given explicitly as probabilistically annotated ground facts) in order to reflect incomplete knowledge about event independence, which is the actual cause of lack of truth-functionality of probabilistic calculi. Alternatively, $Datalog_{PID}$ ("$Datalog_P$ with independence assumptions") makes the quite strong assumption of universal event independence, that is $Pr(e_1 \wedge e_2) = Pr(e_1) \cdot Pr(e_2)$ for any events $e_1$ and $e_2$. Under this assumption boolean combinations of the constituents of probabilistic event expressions become possible. Please find technical details in [13].

Approaches which combine Description Logics for the Semantic Web with Datalog/Prolog-like logic programming (and thus full rules as defined above) are described in the next section.

Whereas the logics described so far are subsets of first-order logic, in [14], Halpern presents three different probabilistic (full) first-order languages $\mathcal{L}_1$, $\mathcal{L}_2$, and $\mathcal{L}_3$. Basic $Datalog_P$ is a subset of $\mathcal{L}_2$. All three languages have a very expressive syntax, allowing for, e.g., probabilistically-weighted arbitrary first-order formulas (including rules in the sense of Horn clauses) and conditional probabilities. However, they are undecidable and their relevance is largely theoretical and

important mainly because of their influence on historically newer and practically more relevant languages.
Formulas of $\mathcal{L}_1$ take the basic form of $w_x(\varphi) \geq pr$, where $\varphi$ is a first-order formula and $pr$ is a probability. This syntax is more or less identical to the syntax of Bacchus' logic $L_p$ [6]. The semantics is quite similar, but not identical. The informal meaning of the statement form above is "The probability that a randomly chosen object x in the domain satisfies $\varphi$ is at least $pr$". This means that we do here (and also for $L_p$) *not* encounter a possible-world semantics here, but instead a "statistical semantics" (or more appropriate: *domain-frequency semantics*) which puts a probability distribution over the *domain of discourse*. This semantics is an implementation of the empirical interpretation of probabilities mentioned at the beginning of this section. Probabilities reflect here "objective" statistical proportions or frequencies.

In contrast, $\mathcal{L}_2$ uses a possible-world semantics. The syntactical form of $\mathcal{L}_2$ formulas is $lb \leq \varphi \leq up$, where $lp$ is the lower bound for the probability of $\varphi$ and $up$ is the upper bound. The semantics works much like that of $Datalog_P$, only that $\varphi$ is not restricted to facts.

Both approaches can be converted into each other - however, to understand better what the practical difference between these two languages respectively their semantics is, and why a degree-of-belief semantics is not appropriate in certain cases, and domain-frequency semantics is not adequate in other case, we look at the following example (taken from [14]):

Suppose one would like to formalize the two statements "The probability that a randomly chosen bird is greater than 0.9" and "The probability that Tweety (which is a bird) can fly is greater than 0.9", using some first-order probabilistic languages. For the second statement, a possible-world semantics which assigns a probability of 0.9 or higher to the set of those worlds in which Tweety can fly seems appropriate. However, the first statement cannot simply be formalized using a possible-world semantics, at least not in a straightforward way [14, 6]. A seeming way (among others which do not work very well) would be to attach probability 0.9 to the worlds in which $\forall x Bird(x) \rightarrow Flies(x)$ holds. However, if in all worlds there is at least a single bird which does not fly, and so the probability of this statement is zero, it could still be the case that the probability that a randomly chosen bird flies is greater than 0.9. In contrast to the possible-world semantics, a domain-frequency semantics can represent the first statement without problems, but would on the other hand have in certain contexts problems with the representation of statements like the second statement above [14, 6].

In order to allow for a simultaneous reasoning with both views of probability, Halpern introduces the language $\mathcal{L}_3$. It allows statements such as
$w(w_x(Flies(x)|Bird(x)) > .99) < 2 \wedge w(w_x(Flies(x)|Bird(x)) > .9) > .95$,
hence combining the syntactical features of $\mathcal{L}_1$ and $\mathcal{L}_2$. What essentially happens here is that agents can hold degrees of beliefs about statistically uncertain statements.

While *Bayesian networks* (also called *belief networks*) are able to represent a sort of event-conditional rules, and certain variations of Bayesian networks encode causal rules, they work only on a propositional level, and thus do not fall into our scope of interest. However, several formal approaches exist which extend Bayesian networks with first-order capabilities (relations). *Bayesian logic programs* (BLP) [16] for example can be seen as a generalization of Bayesian networks and logic programming, implementing a possible-world semantics. The logic component of BLP consists of so-called *Bayesian clauses*. A Bayesian clause is a rule of the form $A|A_1, ..., A_n$, where each $A_i$ is a universally quantified *Bayesian atom*. The main difference between Bayesian clauses and ordinary clauses (apart from the use of | instead of $:-$ as in Prolog or Datalog rules) is that the Bayesian atoms have values from a finite domain instead of boolean values. In addition to Bayesian clauses, a BLP consists of a set of conditional probability distributions over Bayesian clauses $c$ (encoding $Pr(head(c)|body(c))$) and so-called *combining rules* in order to retrieve a *combined conditional probability distribution* from the combination of the multiple different conditional probability distributions. From a BLP a Bayesian network can be easily computed and then queried using standard Bayesian inference. Another prominent example for relational extensions of Bayesian Networks are *Probabilistic Relational Models* (PRMs). However, they cannot express arbitrary quantified first-order rules.

*Multi-entity Bayesian networks* (MEBNs) [18] are another example for a formal framework which integrates first-order logic with Bayesian probability theory. In contrast to most other "relational Bayesian" approaches, they have full first-order representation power.

*Stochastic Logic Programs* (SLPs) [21] are sets of rules in form of range-restricted clauses labeled with probabilities. The resulting annotated rules are called *stochastic clauses*. The semantics of SLPs assigns a probability distribution to the atoms of each predicate in the Herbrand base of a program. SLPs are a generalization of *Hidden Markov Models* as well as stochastic grammars to first-order logic programming, and are expressive enough to encode (undirected) Bayesian networks. SLPs cannot only be manually constructed, but also learned using a combination of Inductive Logic Programming (ILP) and stochastic parameter estimation [21].
Stochastic Logic Programs encode a sort of domain-frequency semantics. It can be shown, however, that SLPs (respectively, BLPs) can be translated into BLPs (respectively, extended SLPs) [23].

A quite recent approach to the combination of first-order logic and probabilistic theory are *Markov Logic Networks* (MLNs) [24]. A MLN is a set of (unrestricted) first-order formulas with a weight (not a probability) attached to each formula. MLNs are used as "templates" from which *Markov networks* are constructed. Markov networks are graphical models for the joint distribution of a set of random variables, allowing to express conditional dependencies Bayesian networks cannot represent, and vice versa. The (ground) Markov network generated from the MLN then determines a probability distribution over possible worlds, and is used to compute probabilities of restricted formulas using proba-

bilistic inference. Machine learning algorithms allow to learn the formula weights in MLNs from relational databases.

### 2.2 Dealing with Imprecision: Possibilistic Theories

#### Fuzzy Sets

Fuzzy sets [27] are sets $X$ whose elements $x$ have varying degrees of membership, evaluated by a membership function $\mu : X \mapsto [0, 1]$. Fuzzy sets can be used to define concepts with vagueness (e.g. "old", "tall") and reason with and over them. If the membership degree can't be estimated precisely, higher order fuzzy sets can be defined, which meta-membership degrees are, in turn, fuzzy sets over the domain of membership degrees themselves, $[0, 1]$.

The concept of fuzzification, i.e. extension with vagueness, can been applied to many contexts, including logic.

#### Fuzzy Logic in a Narrow Sense

The fuzzification of first-order logic can be obtained directly by extending the underlying algebraic structure. In boolean FOL `true` and `false` are the only allowed truth values for a formula: one can obtain a fuzzy version of this logic assuming that the truth value is a member of a lattice (a partially ordered set in which any two elements always have a unique supremum and infimum), which in practice is usually the unit interval $[0, 1]$.

Like in classical logic, the formulas are still built from atoms using connectives and quantifiers, but their evaluation is delegated to complex operators which generalize the boolean ones. To preserve an axiomatic structure, only a minimal set of operators is defined primitively (e.g. the implication $\rightarrow$ and the negation $\neg$), while the others are derived according to the canonical definitions (e.g. $\neg x \vee y \Leftrightarrow x \rightarrow y$). The choice of the implication operator is strictly connected to the choice of the conjunction operator $\star$, implemented using a triangular norm, a commutative, associative, monotonic binary map on $[0, 1]$ having 1 as neutral element: in fact, it holds that $x \star z \leq y \Leftrightarrow z \leq (x \rightarrow y)$.

Interestingly, there is not a unique choice for $\star$, so, given the mutual dependencies, there exist families of operators and thus different logics according to which operators and how they are defined. However, it has been shown that all alternatives can be reduced to three basic t-norms, namely minimum, product and Lukasiewicz.

The operators, in any case, are truth functional, which makes evaluation computable efficiently. The truth value of the atoms may be given as a fact, or evaluated using the equivalent of a membership function operating on its arguments: afterwards, operators are applied to determine the truth value of formulas. For these reasons, these "mathematical" fuzzy logics are special instances of many-valued logic. A complete discussion can be found in [**?**].

**Fuzzy Logic in a Broader Sense**

When Zadeh defined fuzzy logic ([**?**]), he was more interested in adopting a formalism close to the way people think and express concepts than in defining a formal extension of mathematical logic. This vision led to the definition of linguistic variables, which values are fuzzy sets over a specific domain. For example, the variable `age` may have *young*, *mature* and *old* as values, all fuzzy sets over, say, the integer interval 0..100. This allows to write conjunctive, horn-like rules having the form "if $X_1$ is $A_1$ and ... and $X_n$ is $A_n$ then $Y$ is $B$ ". Tipically, several rules entailing information on $Y$ are written and then combined disjunctively. In order to apply the rules, the quantitative inputs $x$ (e.g. the age of a person) are fuzzified using, for example, a membership function. The fuzzified variables are matched against the qualitative constraints in the rules to entail the fuzzy conclusions $Y$, again fuzzy sets, which are defuzzified to obtain a quantitative output value. It has been shown (e.g. see again [**?**]) that a set of such fuzzy rules is equivalent to a local, declarative approximation of a relation $\Pi_{j=1}^{n} X_j \mapsto Y$. For this reason, fuzzy logic has been widely applied to the control of complex, nonlinear systems.

**Possibility theory** In possibility theory [10], a possibility distribution associates a value $\pi(x) \in [0, 1]$ to each element $x$ in a set of alternatives $X$. This value measures the compatibility of $x$ with the actual state of the world. Unlike probability theory, where relative odds are considered, in possibility theory each individual is considered separately: in fact, the possibility $\Pi(A)$ associated to a set $A$ is given by $\max(\pi(x)) : x \in A$. The dual concept of possibility is necessity, defined by negating the possibility of the complementary: $N(A) = 1 - \Pi(A)$. Possibility can be physical or epistemic, depending whether objective or subjective factors are taken into account when defining $\pi$. Notice that the membership function of a fuzzy set can be considered a possibility distribution on its domain, given the only knowledge that the actual state of the world belongs to the set.

## 3 Mainstream Software Tools and Standards

The (Semantic) Web is a heterogeneous software environment in which many systems and frameworks coexist.

We have shown that there exist different families of logic, with different expressiveness, which can be used to define and reason with rules. On the other hand, there exist many *rule engines*, with different characteristics, that a user can adopt in their projects. In modern applications, such rule engines are currently mainly used for two following tasks:

processing business rules and, as an emerging field, reasoning over Semantic Web ontologies enhanced with rules [5, 2].

The main goal of our survey is to see whether existing rule engines support imperfect rules natively, i.e. if the rule engine handles all the inference procedures typical of the respective logic.

Inevitably, however, other aspects have also to been taken into account: whether a tool is commercial or freely available, whether it is for an academic or industrial use, whether it is just a rule engine or a full fledged *Business Rule Management System* (BRMS), and its execution environment (e.g. Java vs. .NET).

Even limiting our research to mainstream projects[3], i.e. general purpose tools supported by a some community (to whatever extent), thus discarding several student projects and many ad-hoc engines built for specific applications, we have noticed that there are more than two dozens of different alternatives, but only a few with support for imperfection, and most of these work with fuzzy logic. Note that we disregard in this section logical reasoners and other implementations of probabilistic and possibilistic frameworks which are not rule engines in the usual sense.

**Imperfection in Rule-based Systems**

Fuzzy logic is perhaps the easiest type of imperfect logic to implement in a rule-based system. Inference in fuzzy logic is a generalization of the boolean case. Most importantly, the operators are truth functional, i.e. they just aggregate the degrees associated to their operands, so the complexity remains limited. However, this still means that a standard rule engine can't evaluate fuzzy rules natively without an extension. Many mainstream BRMSs, both commercial and free, include an engine, typically based on the RETE algorithm ([12]), and several additional tools giving support for rule management (remote and local), event handling, editing and reporting.

Among them, we can cite *InRule, ObjectConnections Common Knowledge, Microsoft BizTalk, Fair Isaac's Blaze Advisor, ILOG JRules, OpenRules, PegaSystems PegaRules, Open Lexicon, XpertRule KnowledgeBuilder* and *JBoss Drools*.

Adding fuzzy logic (or any other type of logic) to any of these systems would require a refactoring of the internal rule engine and, possibly, the rule language, neither of which is a simple operation, although such a process is being attempted in Drools.

The mainstream fuzzy-capable systems, instead, are open source rule shells, typically originated in an academic context, without many of the additional features of BRMSs. The most famous are possibly *FuzzyShell*, *FuzzyClips* and *FuzzyJess*; we also know of a commercial data mining tool, Scientio XMLMiner / MetaRule, which has fuzzy capabilities. FuzzyJess is one of the most used given its Java-oriented nature: it is actually a rewriting of FuzzyCLips, itself an extension of the CLIPS engine. FuzzyClips, moreover, has the merit of supporting two types of imperfection: fuzzy logic and confidence, in the form of certainty factors.

[3] Inevitably, we can't consider all existing tools, so we preventively apologize for not citing or discussion some software.

The first rule-based system to introduce uncertainty in automatic reasoning, *MYCIN* ([7]), adopted imperfect rules annotated with certainty factors to model a sort of quality score. The way of handling the factors was not theoretically very sound, so later systems used more structured approaches, even if the idea of using confidence was further developed (see for example [26]). In FuzzyClips, however, they have been introduced in a more coherent way, again truth-functional, and their evaluation proceeds in parallel with the evaluation of the fuzzy truth degrees of the formulas. Notice that all these fuzzy shells support fuzzy logic in the broader sense of the term.
In contrast, no commercial mainstream rule engine that we are aware of supports probabilistic logic. Whereas Bayesian networks have become a very popular tool for handling uncertainty and many mature software packages exist which implement Bayesian networks, hardly any product already supports any of the various probabilistic logics, even if recently at least two projects have been started, namely *Balios* and *BLOG*.

**Interoperability**

One of the limitations of the different engines is their using proprietary languages to write logic formulas and rules in particular.

However, current Web standards for knowledge representation are not able to represent imperfect probabilistic or possibilistic rules.

To achieve a good degree of interoperation, standards on rule representation and interchange are being proposed in the last few years. The *Rule Interchange Format* (RIF) [1] is a proposed W3C standard format for rule representation and interchange, based on XML. *RuleML* [2] is an initiative which develops a XML- and RDF- based markup language for rules, with Datalog-rules as the core. RuleML uses a modular approach to support different rule-based logics with different types of complexity and expressiveness, in order to promote rule interoperability between industry standards. RuleML supports various kinds of reasoning engines (e.g., forward vs backward chaining, RETE vs Prolog, ...) and leaves knowledge engineers the choice of implementation for entities and facts (e.g., objects, plain symbols, XML trees, ...). RuleML is supported by various rules engines, such as jDREW and Mandarax. A combination of the current standard ontology language OWL and RuleML is proposed to the W3C in form of the *Semantic Web Rule Language* (SWRL) [3].

The issues related to the introduction of imperfection in rule languages have recently been discussed in [9], where a module for uncertainty and fuzzy reasoning with rules is defined. This work is remarkable since it shows that most types of imperfect logic can be encoded simply by allowing truth degrees and operators to be customized using appropriate tags (degree) and attributes (kind). Such knowledge, however, should be processed by an engine capable of changing its configuration at run-time, a task that requires more than the creation of a language translator. The extensions proposed in [9] can be integrated into

RuleML, but also in a preliminary version of the W3C Rule Interchange Format (RIF) [1]. Another approach to the integration of rules based on proposed standards and fuzzy logic is *f-SWRL* [22].
As for probability theory, candidates for future standard languages will possible integrate a description logic with rules (in the sense of logic programming) and uncertainty reasoning, such as the formal framework proposed in [20].

## References


1. Rule interchange format (rif) working group, http://www.w3.org/2005/rules/wiki/rif_working_group.
2. Ruleml, http://www.ruleml.org.
3. Swrl: A semantic web rule language combining owl and ruleml, http://www.w3.org/submission/swrl/.
4. W3c uncertainty reasoning for the web incubator group, http://www.w3.org/2005/incubator/urw3/xgr-urw3.
5. G. Antoniou, C. V. Damásio, B. Grosof, I. Horrocks, M. Kifer, J. Maluszynski, and Peter. Combining Rules and Ontologies. A survey., 2005.
6. F. Bacchus. $l_p$, a logic for representing and reasoning with statistical knowledge. *Computational Intelligence*, 6:209–231, 1990.
7. B. G. Buchanan and E. H. Shortliffe. *Rule-based Expert Systems : the MYCIN experiments of the Stanford Heuristic Programming Project*. Addison-Wesley, Reading, Mass. [u.a.], 1984.
8. S. Ceri, G. Gottlob, and L. Tanca. What you always wanted to know about datalog (and never dared to ask). *IEEE Transactions on Knowledge and Data Engineering*, (1(1)):146–66, 1989.
9. C. V. Damsio, J. Z. Pan, G. Stoilos, and U. Straccia. Representing uncertainty in RuleML. *Fundam. Inf.*, 82(3):265–288, 2008.
10. D. Dubois. Possibility theory and statistical reasoning. *Computational Statistics & Data Analysis*, 51(1):47–69, 2006.
11. D. Dubois and H. Prade. Possibility theory, probability theory and Multiple-Valued logics: A clarification. *Annals of Mathematics and Artificial Intelligence*, 32(1-4):35–66, 2001.
12. C. Forgy. Rete: A fast algorithm for the many patterns/many objects match problem. *Artif. Intell*, 19(1):17–37, 1982.
13. N. Fuhr. Probabilistic datalog - a logic for powerful retrieval methods. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1995.
14. J. Y. Halpern. An analysis of first-order logics of probability. *Artificial Intelligence*, 46:311–350, 1990.
15. J. Y. Halpern. *Reasoning about Uncertainty*. The MIT Press, October 2003.
16. K. Kersting and L. D. Raedt. Bayesian logic programs. In *Proceedings of the 10th International Conference on Inductive Logic Programming*, 2000.
17. G. J. Klir. Generalized information theory. *Fuzzy Sets Syst.*, 40(1):127–142, 1991.
18. K. B. Laskey and P. C. Costa. Of klingons and starships: Bayesian logic for the 23rd century. In *Proceedings of the Twenty-first Conference on Uncertainty in Artificial Intelligence*, 2005.
19. D. C. Luckham. *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

20. T. Lukasiewicz. Probabilistic description logic programs. *International Journal of Approximate Reasoning*, 45(2):288–307, 2007.
21. S. Muggleton. Learning stochastic logic programs. In *Electronic Transactions in Artificial Intelligence*, 2000.
22. J. Z. Pan, G. B. Stamou, V. Tzouvaras, and I. Horrocks. f-swrl: A fuzzy extension of swrl. In *ICANN (2)*, pages 829–834, 2005.
23. A. Puech and Muggleton. A comparison of stochastic logic programs and bayesian logic programs. In *IJCAI03 workshop on learning statistical models from relational data, IJCAI, 2003*.
24. M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62(1-2):107–136, February 2006.
25. P. Smets. *Imperfect Information: Imprecision and Uncertainty*, pages 254, 225. 1996.
26. P. Wang. Confidence as higher order uncertainty. *null*, 1994.
27. L. A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, June 1965.