University of New Brunswick

Combining Rules, Taxonomies and Probabilities in the Extended OO jDREW Rule Engine

Research Exposition

for

UNB Faculty of Computer Science

By

Harold Boley, Benjamin Larry Craig, Judy Zhao,

Greg Sherman, Tshering Dema

Abstract OO jDREW extensions are discussed for top-down and bottom-up rule execution, a Findall solutions predicate for the top-down engine, dual rule syntax, light-weight rule-ontology combination, and importing knowledge bases from Web sources. A design is presented for introducing uncertainty management to knowledge representation with rules, and exemplified with a Currency Exchange example.

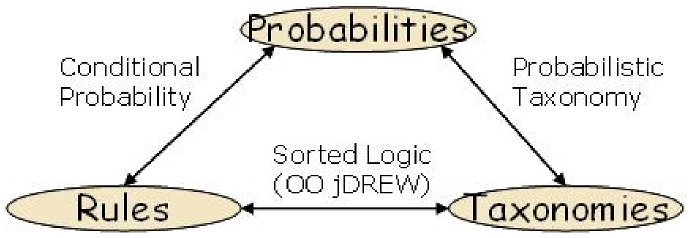

Figure 1: Our Approach for bridging probabilities, rules and taxonomies

1 Introduction

There are two kinds of knowledge bases (KBs) that can be expressed in the RuleML interchange language and its POSL presentation syntax <http://www.ruleml.org> as implemented in OO jDREW <http://www.jdrew.org/oojdrew>: Rule KBs (e.g., in propositional logic, Datalog, and Horn logic with negation as failure) and the `subClassOf` taxonomy backbone of ontology KBs (e.g., of frame-based languages, RDFS, and description logic).

We present OO jDREW extensions including top-down and bottom-up rule execution, a Findall solutions predicate for the top-down engine, dual rule syntax, light-weight rule-ontology combination and importing KBs from Web sources.

Extending our efforts to bridge rules and taxonomies use OO jDREW as a reasoning engine for RuleML and POSL, we then present a design to introduce uncertainty management into knowledge representation in both KB categories and work on bridging probabilities, rules and taxonomies, as shown in Fig. 1.

As a starting point, we first introduce probabilities to knowledge representation in rules exemplified with a Currency Exchange example. This effort will include the following three main steps:

- Extend POSL to allow additional probabilistic information;
- Convert a Bayesian Network into a probabilistic POSL KB that can be read into probabilistically extended OO jDREW;
- Query and reasoning based on the converted Bayesian Network using extended OO jDREW.

2 Extended OO jDREW Features

2.1 Top-Down and Bottom-up Rule Execution

OO jDREW allows support for two different types of reasoning namely bottom up and top down. Bottom-up execution is used to infer all derivable knowledge from a set of clauses and is also known as forward reasoning. Top-down execution is used to solve a query on the KB and is sometimes called backward reasoning. Having both modes of execution is useful depending on what user applications require.

2.2 Findall Solutions Predicate for the Top-Down engine

OO jDREW has Prolog like primitives, one of these primitives is the findall solutions predicate. The findall solutions allows the top-down engine to find all the solutions to a selected query. The answer is then returned as a container of all the solutions to the given query.

2.3 Dual Rule Syntax

The engine contains parsers for both the POSL (Positional Slotted Language) and RuleML. POSL, is an ASCII language combining Prolog and F-logic, which can be used as a shorthand for the XML-based RuleML. RuleML is the XML based language that describes the rule mark up language. POSL is used as a human-readable syntax for RuleML while RuleML XML is used as an interchange language.

2.4 Light-Weight Rule-Ontology Combination

OO jDREW contains support for an order-sorted type system which specifies a taxonomy of classes. Having a type system allows the user to take advantage employing a taxonomy in their rules. Also using types can restrict the search space during queries that can improve query times. Ontologies are defined as Resource Description Framework schema in the OO jDREW engine. A new feature recently added to OO jDREW is the capability to create ontolgoies using POSL instead of the XML syntax of RDFS. The POSL syntax uses the predicate subsumes(class1, class2) to define that class1 is a subclass of class2. Also the POSL syntax allows for querying based on sub class, super class, great lower bound and least upper bound of classes.

2.5 Importing KBs from Web sources

In order to improve the flexibility of OO jDREW the feature of importing rule bases and ontolgoies from web-based sources has been implemented. This feature is vital in allowing OO jDREW to become more web-ready. The convenience of reading web-sources is important because these sources are continuously updated and a user of OO jDREW will not be required to update their KBs when these sources change.

3 From Propositional to Probabilistic Rules

It is well known that there are two important quantities in a currency trading market, the interest rate and the inflation rate of a currency, which affect two other quantities, the supply and exchange rate of the currency. From finance theory, we know that when the interest rate of a currency is not rising (i.e., stagnating or falling) and its inflation rate is rising, then the supply of this currency on the currency trading market tends to be rising. Furthermore, when the supply is not rising (again, stagnating or falling), then the exchange rate for this currency tends to be rising. This knowledge can be roughly formalized in Propositional Logic (with negation as failure: naf) using four nullary predicates expressing the rising of these quantities (Interest: Interest Rate; Inflation: Inflation Rate; Supply: Supply on Market; Exchange: Currency Exchange rate). In this initial model, there are two facts and rules, which can be written in POSL as follows (the empty parenthesis pairs are usually omitted in Propositional Logic, but prepare the following steps):

```
Interest().
Inflation().
Supply() :- naf(Interest()), Inflation().
Exchange() :- naf(Supply()).
```

In a step towards a more realistic model, we introduce two arguments, time and currency, for all four predicates, obtaining a KB in Datalog is shown as follows (parentheses enclose two constants or variables, the latter being prefixed in POSL by a question mark):

```
Interest("2008-03-17", CAD).
Inflation("2008-03-17", CAD).
Supply(?T,?C) :- naf(Interest(?T,?C)), Inflation(?T,?C).
Exchange(?T,?C) :- naf(Supply(?T,?C)).
```

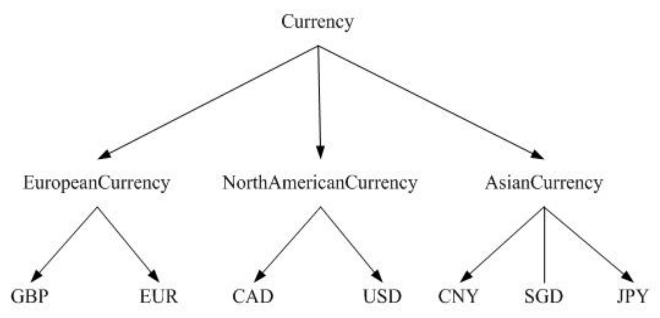

Figure 2: Taxonomy of Currency

For the next step, suppose we have the taxonomy of currencies shown in Fig. 2. KB in Datalog combined with taxonomy, obtaining a KB in Order-Sorted Datalog, is shown as follows, where the colon infix associates a variable with its type (a class from the taxonomy) and the question mark refers to an anonymous variable:

```
Inflation("2008-03-17", ?:CAD).
Interest("2008-03-17", ?:CAD).
Supply(?T, ?C:NorthAmericanCurrency) :-
   naf(Interest(?T, ?C:NorthAmericanCurrency)),
   Inflation(?T, ?C:NorthAmericanCurrency).
Exchange(?T:Date, ?C:NorthAmericanCurrency) :-
   naf(Supply(?T:Date, ?C:NorthAmericanCurrency)).
```

For the next step, notice that such rules are not always completely true in real situations. For example, we should only state that the conditional probability for a currency exchange rate known to go up if the supply of this currency goes up is p. In traditional probability theory, it has the form prob(Exchange|Supply) = p. Based on the causal relationships between the interest rate, the inflation rate, the supply of a currency on an exchange market and the exchange rate, we can construct a Bayesian network as shown in Fig. 3.

We regard a conditional probability like prob(Exchange | ¬Supply) = p as a probabilistic rule Exchange() :- neg(Supply()) / p. The neg represents a classical negation of a logical atom. The Bayesian network then becomes a probabilistic KB with the four rows of the Supply table becoming four rules, and the two rows of the Exchange table becoming two rules (in Probabilistic Propositional Logic):

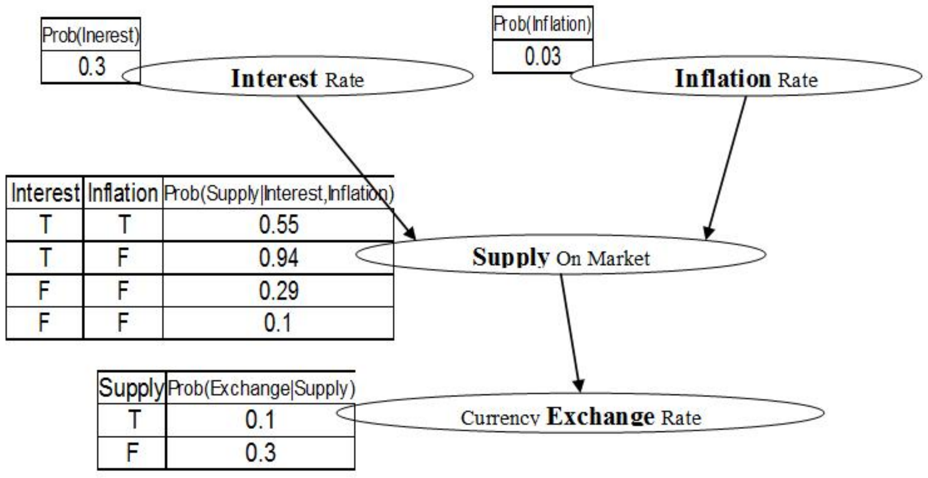

Figure 3: A Network of Currency Trading Market

```
Interest() / 0.3.
Inflation() / 0.03.
Supply() :- Interest(), Inflation() / 0.55.
Supply() :- Interest(), neg(Inflation()) / 0.94.
Supply() :- neg(Interest()), Inflation() / 0.29.
Supply() :- neg(Interest()), neg(Inflation()) / 0.1.
Exchange() :- Supply() / 0.1.
Exchange() :- neg(Supply()) / 0.3.
```

With such a KB, we will extend OO jDREW to do several kinds of probabilistic querying and reasoning based on the product rule, the theorem of total probability, Bayes' rule and Bayesian Networks. For example, the implementation of independent conjoined probabilities is planned as an extension of the OO jDREW Java iterator for conjunction processing.

4 Conclusion

The versions of the Currency Exchange example prior to the introduction of probabilities can already be run in the current OO jDREW, which can be used online with Java Web Start (the OO jDREW sources are freely available under GNU LGPL). The implementation of OO jDREW probability management on the Java level will follow the design presented in this paper. OO jDREW is one of the engines used by the distributed Rule Responder architecture <http://resp onder.ruleml.org>. RuleML and the OO jDREW Open Source Initiative <http://wiki.ruleml.org/OO_jDREW> have also been working with OMG's PRR <http://www.omg.org/docs/dtc/07-11-04.pdf> and W3C's RIF <http://www.w3.org/2005/rules/ wiki/RIF_Working _Group>. Our companion paper in these proceedings describes the XSLT-based RDF2POSL translator along with the eTourism OO jDREW use case eTourPlan.