An Hybrid Architecture Integrating Forward Rules with Fuzzy Ontological Reasoning

Stefano Bragaglia, Federico Chesani, Anna Ciampolini, Paola Mello, Marco Montali, and Davide Sottara

University of Bologna, V.le Risorgimento 2 - Bologna, Italy
name.surname@unibo.it

Abstract. In recent years there has been a growing interest in the combination of rules and ontologies. Notably, many works have focused on the theoretical aspects of such integration, sometimes leading to concrete solutions. However, solutions proposed so far typically reason upon crisp concepts, while concrete domains require also fuzzy expressiveness.

In this work we combine mature technologies, namely the Drools business rule management system, the Pellet OWL Reasoner and the FuzzyDL system, to provide a unified framework for supporting fuzzy reasoning. After extending the Drools framework (language and engine) to support uncertainty reasoning upon rules, we have integrated it with custom operators that (*i*) exploit Pellet to perform ontological reasoning, and (*ii*) exploit FuzzyDL to support fuzzy ontological reasoning.

As a case study, we consider a decision-support system for the tourism domain, where ontologies are used to formally describe package tours, and rules are exploited to evaluate the consistency of such packages.

Topics: Fuzzy Reasoning, Rule-based Reasoning, Rules Integration with Ontologies, Decision Support Systems, eTourism.

1 Introduction

In the recent years there has been a growing interest in the combination of rules and ontologies. Notably, many works have focused on the theoretical aspects of such integration, sometimes leading to concrete solutions (e.g., [1,2,3,4,5,6]). However, solutions proposed so far typically reason upon crisp concepts, while concrete domains require also fuzzy expressiveness.

At the same time, some initiatives emerged in the last few years to implement fuzzy reasoning in both the context of ontology reasoning [7], and in the context of rule-based reasoning [8,9].

From a methodological viewpoint, the integration between the various forms of reasoning (ontological, rule-based and fuzzy-like) has been researched following two diverse approaches: *tight integration* vs. *loose integration*. The former aims to provide a single, comprehensive theoretical framework accounting for the various reasoning tasks (hence a unified language and semantics). The latter approach instead simply focuses on combining the different technologies available, to provide a functioning tool able to cope with the requirements.

M. Graña Romay et al. (Eds.): HAIS 2010, Part I, LNAI 6076, pp. 438–445, 2010.

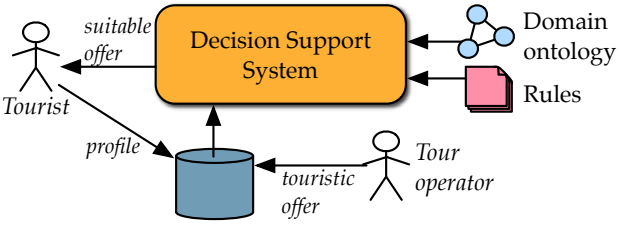

Fig. 1. Architecture of a touristic decision support system, combining ontologies with rules

In this work we focus on the integration of such diverse reasoning tasks, namely ontological, rule-base and fuzzy reasoning, following a *loose integration* approach. To this end, we have selected a set of mature technologies and, where necessary, we have properly extended them to achieve such integration.

As a case study, we consider a decision-support system (DSS) for the tourism domain, where ontologies are used to formally describe package tours, and rules are exploited to evaluate the consistency of such packages and their applicability to possible customers, as in Figure 1. The aim of our DSS is to support business tourism operators when evaluating which packages are suitable and consistent with possible customers. In this sense, the DSS helps to categorize both the customers (into a set of possible customer profiles) and the packages, and provides a matching degree between the profiles and the packages. In such scenario, two distinct kinds of knowledge come into play:

- *Ontological knowledge*, to model in a clear and unambiguous way the entities composing the tourism domain, as well as their relationships and categories.
- *(forward) Rules*, to formalize the "operational" knowledge exploited by the business operators when evaluating packages vs. customers.

For example, a rule would state that "*cultural* package tours should be preferably offered to *senior* customers". Here, the concepts of "cultural" and "senior" are examples of ontological knowledge. E.g., a senior customer could be defined as a customer whose age is higher than a certain threshold.

Moreover, notions like being "cultural" or "senior" are not crisp concepts. Tour operators implicitly interpret them as approximate concepts, which can assume several different degrees of truth. Using fuzzy reasoning, the classification of customers and offers, as well as their matching, becomes a graded matter of possibility instead of a definitive assignment. At the same time, for a given entity, the candidate classes can be ordered according to the degree of matching, so a "maximum-plausibility" classification can always be performed.

It is worth noting that fuzzy reasoning has, in this context, a twofold meaning. First of all, fuzzy ontological reasoning is needed to provide approximate reasoning at the concept level. Second, also the logical operators used inside the rules management system must be interpreted in a fuzzy manner, enabling the propagation of fuzzy truth values along the triggered rules.

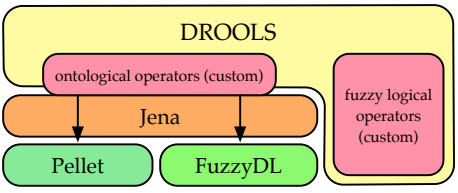

Fig. 2. The loosely coupled architecture of the system

2 Integrating Rules with Fuzzy Ontological Reasoning

Our DSS has been implemented adopting currently available technologies, preferring solutions that would offer more benefits in terms of interoperability, extensibility and support. Namely:

- *Drools Expert*[1] is a forward chaining inference rules engine based on an enhanced implementation of Charles Forgy's *Rete algorithm* [10].
- *Pellet*[2] is an OWL-DL Java-based reasoner which provides standard and advanced reasoning services for OWL ontologies.
- *Jena* and *JenaBean* are an open source Java-based framework for "semantic web" application [11].
- *FuzzyDL System* is a description logic reasoner that supports both Fuzzy Logic and fuzzy Rough Set reasoning [7].

Achieving a *loose integration* of the reasoning tools is a challenging task. Many issues are related on how interfacing the different tools, and how to provide "triggering mechanisms" from one tool to other's reasoning services. The former problem has been addressed by means of ontologies and Jena as a common representation framework. Ontologies provide explicit definitions of the concepts, while the Jena framework and its extension JenaBeans provide common APIs to access ontology files and binding of concepts to Java classes.

The latter issue, i.e. how to allow each tool to invoke reasoning services of other tools, has been addressed by choosing the Drools framework as base platform. In particular, Drools supports the definition of custom operators: we have implemented ontological operators that build the ontological inferred model and query it to evaluate some ontological operations. E.g., Drools can apply ontological queries by simply writing `Customer( this isA Single.class )` where the right operand is an ontological class defined as `Customer and (hasSpouse exactly 0) and (hasChild exactly 0)`. Drools supports fuzzy logic natively as a part of the experimental extension called Drools Chance [12] which enhance every formula by annotating it with a degree that models partial truth. Standard evaluators have been extended to return a real-valued degree instead of a boolean. A special class of fuzzy custom evaluator is formed by the *linguistic*

[1] `http://www.jboss.org/drools/drools-expert.html`
[2] `http://clarkparsia.com/pellet`

evaluators. Such evaluators are based on the concept of *fuzzy partition*: the numeric domain of a quantitative field X is partitioned using fuzzy sets [13]. Each fuzzy set S, labeled with a linguistic value is defined by its membership function $\mu_S : X \mapsto [0, 1]$, which is a generalization of the concept of characteristic function. Thus, a fuzzy partition is a set of N fuzzy sets $S_{j:1..N}$ over X such that:

$$\forall x \in X : \left[\exists\, j^\star : \mu_{S_{j^\star}}(x) > 0 \wedge \sum_{k=1}^{N} \mu_{S_k}(x) = 1 \right]$$

For example, the domain of `age` could be partitioned in $\Phi_X = \{$`young`, `mature`, `old`$\}$. In Drools, a fuzzy predicate constraint can be applied simply by writing `Customer( age seems "young" )`. The custom meta-evaluator `seems` selects the fuzzy set corresponding to the label passed as right operand and uses it to compute the membership of a `Customer` in the set of young people.

If the atomic constraints have been combined in more complex formulas using logical connectives such as `and` or `or`, the degrees resulting from the evaluation are combined using the corresponding operators. Moreover, it is possible to choose the specific implementation of each operator using the attribute "`kind`" and choosing from a predefined set of values [12]. For example, the pattern `Customer( age Seems "young" or @[kind="Max"] "mature" )` instructs the engine to combine the two degrees by taking the maximum of the two.

Eventually, the degrees are combined until modus ponens is applied to compute the truth degree corresponding to an activation of a rule: this degree is available in the consequence part, so that the effects can be conditioned on its value. By default, a rule activates and fires only if the degree is strictly greater than 0, but even this behavior can be changed on a rule-by-rule basis.

3 Examples

In the context of a national project[3], we are investigating e-tourism applications, where customers register themselves on Internet sites to receive suggestions for their vacations. To this aim, they provide biographic information such as age, gender, marital status, number and age of children and how much they are willing to spend. On this base customers are classified in different profiles, such as *singles*, *couples*, *families* and *seniors*. Similarly, tour operators submit their offers to the same sites and their touristic products get organized in categories like *nature*, *culture*, *adventure*, *hedonism*, *wellness*, *shopping*, *sports*, *spirituality* and *festivals*. Once the classification stage has ended, the system tries to match customer profiles with categories of touristic packages suggesting the most suitable offers to each customer and providing the feedback for products to each tour operator (for example, the appeal of a product in terms of potential customers it can reach). The following examples illustrates some capabilities of our system.

[3] Italian MIUR PRIN 2007 project No. 20077WWCR8.

Example 1. Consider the rule below that suggests hedonistic nonadventurous products to families (static matching). It exploits the typical rule reasoning of Drools as well as Pellet's ontological reasoning and FuzzyDL's fuzzy reasoning.

```
rule "Family Customers"
  filter "0.75"
  when
    $parent : Customer ( this isA Married.class )
    exists Customer( age seems young, isChildOf == $parent)
    $offer : Offer ( this isA Hedonism.class
                     and not isA Adventure.class )
  then
    // Suggest $offer to $parent for her Family...
end
```

The custom operator `seems`, in fact, is actually an `is-a` operator with fuzzy expressiveness that exploits FuzzyDL to decide how much the age of a customer may be considered "young" discharging any result below the `filter` threshold (`young` is a fuzzy set with a left-shoulder function that returns 1 as degree of truth until ages of 10, a value that slowly decrease to 0 for ages between 10 and 16 and then always 0). The custom operator `isA` is instead a standard `is-a` operator whose evaluation is demanded to Pellet. In this case right-side operands should be existing ontological classes: `Married`, for example, is equivalent to `Customer and (hasSpouse exactly 1)`. Finally note that standard Drools predicates such as `isChildOf` are also mapped to ontological object properties.

Thus, an instance of `Customer` with a spouse and two children (4 and 2 years old) is correctly recognized as a `Family` while another one with a 18 years old daughter is not. Similarly a product involving the entrance in a water park, which is an `Hedonistic` offer, is suitably presented to the families but one with a rafting water park is not because it is also an `Adventure` offer.

Example 2. Consider a rule that dynamically discovers the associations between customers' profile and offers' categories and provides recommendations accordingly. This rule exploits operators similar to the ones seen in Example 1, but it is worth a mention for at least two reasons:

- <Profile, Category> couples are not hard-coded in rules, but rather evaluated by an external custom operator (`matches`);
- the fuzzy results computed by FuzzyDL can be understood by Drools and used in rules' consequents.

```
rule "Matching"
  when
    $pro : Profile ( )
    $cus : Customer ( this isA $pro )
    $cat : Category ( this matches $pro )
    $off : Offer ( this isA $cat )
```

```
  then
    PriorityQueue queue = getQueue($cus);
    queue.insertOrd($off, drools.getConsequenceDegree());
end
```

In this case, the rule determines the `Profile` of and `Category` of any couple of `Customer`s and `Offer`s and then determining the strength of the association between them is demanded to `matches`. If we consider a family and a water park offer as in Example1, that couple will be evaluated as a good matching and it will produce the same results as before. On the other hand, if we consider a free-climbing offer that is an highly dangerous `Adventure` offer, the `matches` operator will likely report its association with a `Family` as less likely. By calling the `getConsequenceDegree()` method, Drools can handle the truth value being evaluated by the rule and also use it to sort offers for customers.

4 Related Works and Conclusions

Although many rule engines and ontology reasoners are currently available, only few of them support rules and DL reasoning at the same time. Jena, for example, includes a generic rule based inference engine that can easily cooperate with the supported reasoners. Hammurapi Rules[4] is another Java rule engine that leverages Java language semantics to express relationships between facts as ontologies do. Algernon[5] is an efficient and concise Protégé extension supporting both forward and backward chaining rules that stores and retrieves information in ontologies. Another solution is SweetRules[6], an integrated toolkit for semantic web revolving around RuleML and many other W3C standards that works with ontologies. Despite being remarkable, they are less easily customizable, less widespread than the components of our solution and also not supporting fuzzy reasoning.

This is mainly due to the general lack of fuzzy reasoners: even though a few are actually under development (such as DeLorean[7]), FuzzyDL system is the only mature Java-based solution we were able to find. Fuzzy logic is possibly the only type of non-boolean logic which has been integrated in mainstream, open source BRMS. Actually, there exist many tools supporting "fuzzy logic", but most of the times it must be understood as "fuzzy logic in a broad sense" [14]. Much fewer are the tools which intend fuzzy logic in the sense of a truth-functional, annotated many-valued logic. Nevertheless, two of the first and most important rule-based expert systems, Clips[8] and Jess[9], had a proper fuzzy extension (FuzzyClips[10] and FuzzyJess[11] respectively). Unfortunately, both projects are discontinued, and lack the full BRMS

[4] http://www.hammurapi.com/
[5] http://algernon-j.sourceforge.net/
[6] http://sweetrules.semwebcentral.org/
[7] http://webdiis.unizar.es/~fbobillo/delorean.php
[8] http://clipsrules.sourceforge.net/
[9] http://www.jessrules.com/
[10] http://www.nrc-cnrc.gc.ca/eng/projects/iit/fuzzy-reasoning.html
[11] http://www.csie.ntu.edu.tw/~sylee/courses/FuzzyJ/FuzzyJess.htm

capabilities offered by Drools. Other than that, an attempt to integrate logic programming and fuzzy logic has been made with FRIL [15], a prolog-like language and interpreter, but that project is currently inactive.

Concerning the problem of integrating semantic knowledge with other form of reasoning, a first approach is to exploit the common root of logic programming and description logics in first order logic by means of LP clauses. These problems have been addressed: that intersection has been named *DLP* (Description Logics Program) [2] and a method for translation has been proposed [3]. On these basis, dlpconvert [4], a tool that converts (the DLP fragment of) OWL ontologies to datalog clauses, has been developed. In [5] and [6], the authors propose techniques for reasoning on Description Logic with $\mathcal{SHOIN}$ and $\mathcal{SHIQ}$ expressiveness not based on the usual tableau algorithms but instead on bottom-up Datalog and Deductive Database inference, and top-down Prolog resolution respectively in order to deal with large data sets of individuals. Since Deductive Database deal natively with rules, extending the obtained reduction of the DL KB with a rule level appears straightforward by appending rules to the obtained KB [5]. The integration of rules and ontologies has been extensively studied in [16], where the full compatibility with OWL has been relaxed to introduce the Web Service Modeling Language, a language to be used in the WSMO framework. The MARS framework has been introduced in [17] as a framework focused on the rule layer of the Semantic Web cake. MARS follows a declarative style of modeling; its general language, however, is aimed at specifying rules for the semantic web, and it is not equipped with specific reasoning techniques for (semantic) web services.

To the best of our knowledge, our system is the first one that offer a "loose integration" of ontological, rule-based and fuzzy reasoning tools all together. It is build exploiting available technologies, starting form the Drools platform and building upon it to perform ontological and fuzzy reasoning. Currently, we are exploiting our system in the context of a national project to implement a Decision Support System for classifying customers profiles, touristic packages and the knowledge of professional operators. Future work will be devoted to investigate also the theoretical issues related to our integration, as well as extending our approach to support more ontological operators, since current implementation supports ontological isA (subClassOf and instanceOf ontological operators) and fuzzy seems (fuzzy ontological subClassOf and instanceOf). Further work will be devoted also to compare our integrated solution with other tools, taking into account both the expressivity as well as the performances.

Acknowledgements. This work has been partially supported by the Italian MIUR PRIN 2007 project No. 20077WWCR8.

References

1. Antoniou, G., Damásio, C., Grosof, B., Horrocks, I., Kifer, M., Maluszynski, J., Patel-Schneider, P.: Combining rules and ontologies: A survey. Reasoning on the Web with Rules and Semantics (2005)

2. Grosof, B., Horrocks, I., Volz, R.: Description logic programs: Combining logic programs with description logic. In: Proceedings of the 12th international conference on World Wide Web, pp. 48–57. ACM, New York (2003)
3. Hustadt, U., Motik, B., Sattler, U.: Reducing SHIQ- description logic to disjunctive datalog programs. In: Proc. KR, pp. 152–162 (2004)
4. Motik, B., Vrandečić, D., Hitzler, P., Studer, R.: Dlpconvert–converting OWL DLP statements to logic programs. In: European Semantic Web Conference 2005 Demos and Posters, Citeseer (2005)
5. Motik, B.: Reasoning in description logics using resolution and deductive databases. PhD theis, University Karlsruhe, Germany (2006)
6. Lukacsy, G., Szeredi, P., Kadar, B.: Prolog based description logic reasoning. In: Garcia de la Banda, M., Pontelli, E. (eds.) ICLP 2008. LNCS, vol. 5366, pp. 455–469. Springer, Heidelberg (2008)
7. Bobillo, F., Straccia, U.: FuzzyDL: An expressive fuzzy description logic reasoner. In: Proc. 2008 Intl. Conf. on Fuzzy Systems, FUZZ 2008 (2008)
8. Kochukuttan, H., Chandrasekaran, A.: Development of a Fuzzy Expert System for Power Quality Applications. In: Proceedings of the Twenty-Ninth Southeastern Symposium on System Theory, pp. 239–243 (1997)
9. Ozgur, N., Koyuncu, M., Yazici, A.: An intelligent fuzzy object-oriented database framework for video database applications. Fuzzy Sets and Systems 160(15), 2253–2274 (2009)
10. Forgy, C.: Rete: A fast algorithm for the many pattern/many object pattern match problem. Name: Artif. Intell. (1982)
11. Carroll, J., Dickinson, I., Dollin, C., Reynolds, D., Seaborne, A., Wilkinson, K.: Jena: implementing the semantic web recommendations. In: Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers & Posters, pp. 74–83. ACM, New York (2004)
12. Mello, P., Proctor, M., Sottara, D.: A configurable RETE-OO engine for reasoning with different types of imperfect information. IEEE TKDE - Special Issue on Rule Representation, Interchange and Reasoning in Distributed, Heterogeneous Environments (2010) (in Press)
13. Zadeh, L.A.: The concept of a linguistic variable and its application to approximate reasoning - i. Inf. Sci. 8(3), 199–249 (1975)
14. Novk, V.: Abstract: Mathematical fuzzy logic in narrow and broader sense a unified concept
15. Baldwin, J., Martin, T., Pilsworth, B.: Fril-fuzzy and evidential reasoning in artificial intelligence. John Wiley & Sons, Inc., New York (1995)
16. de Bruijn, J.: Semantic Web Language Layering with Ontologies, Rules, and Meta-Modeling. PhD thesis, University of Innsbruck (2008)
17. Behrends, E., Fritzen, O., May, W., Schenk, F.: Embedding Event Algebras and Process for ECA Rules for the Semantic Web. Fundamenta Informaticae 82(3), 237–263 (2008)