Applying SWRL-F to Intercloud Constraints Analysis

Tomasz Wiktor Wlodarczyk, Chunming Rong
Department of Computer Science and Electrical Engineering, University of Stavanger, Norway
tomasz.w.wlodarczyk@uis.no

ABSTRACT

Resource description and constraints analysis emerges as a necessary element of intercloud interoperability and federation. A proper language for constraints definition is essential to efficiently support that task. We propose SWRL-F as a possible solution. We explain its advantages basing on a practical example that compares benefits of fuzzy logic based solution to standard crisp solutions. The analysis suggests that fuzzy logic solution based on SWRL-F provides a useful balance between simplicity and expressivity.

KEYWORDS: intercloud, services, SWRL-F, SWRL, Fuzzy Logic, constraint, OWL, composition, outsourcing

1. INTRODUCTION

Intercloud constraint analysis is one of the key challenges to enable interoperation between clouds. Without proper technologies and tools that interoperation will be at least limited.

Similar problems have been tackled earlier in the domain of service computing as described in the Related Work. One of the most common solutions that emerged was Web Service Description Language that provided the most basic tool for a standardized service description. Various semantic extensions to WSDL emerged with time. Analysis of related work suggests that a solution to these problems can be found in a proper application of semantic technologies.

In this paper we propose applying SWRL-F, a fuzzy extension to SWRL, to help with constraints analysis in an intercloud scenario. This could albo be possible extended to service computing in general. SWRL-F is well grounded in the standard semantic technologies and it provides a small but useful fuzzy logic extension that might help with constraints analysis. SWRL-F can later be embedded in an application to provide a stand-alone solution.

We provide example ontology with a set of simple fuzzy rules in the context of a generic scenario, with vocabulary based on AWS (Amazon Web Services[1]), and a sample of two types of crisp rules to visualize the basic benefits of fuzzy logic solution.

Related Work. Bernstein et al. [1] present and discuss challenges of providing implicit ways to enable clouds resources and services to be exported or caused to interoperate. They mention the necessity of a tool to find if the service description of another cloud meets the interest. RDF and OWL are indicated as possible tools for that task.

A parallel can be constructed between cloud service description and general service description. In such a case, technologies such as WSDL[2], WSDL-S[3] and OWL-S[4] should be investigated. They have been well researched before and most probably they could find application also in the cloud context.

For instance, Czerwinski et al. [2] describe using XML for description and querying in service discovery. Klusch et al. [3] describe using semantic technologies for service discovery and matchmaking basing on similarity computation. Kuster et al. [4] describe an approach to service discovery, matchmaking and composition using elements of fuzzy logic; though, not fuzzy rules.

In an earlier work [5] we have proposed an architecture in which semantic technologies are the enabling tool for the inter-enterprise collaboration based on cloud infrastructure. SWRL-F is an experimental fuzzy logic extension of SWRL that was introduced in [11] and later extended in [12].

Contributions. This paper extends previous work on application of fuzzy logic to service matching by analyzing specific new scenario related to intercloud outsourcing and by integrating fuzzy reasoning directly with ontology-based constraints description.

[1] http://aws.amazon.com/
[2] http://www.w3.org/TR/wsdl
[3] http://www.w3.org/Submission/WSDL-S/
[4] http://www.w3.org/Submission/OWL-S/

Organization of the Paper. After the Introduction, in Section 2 we describe and clarify all the main terms and concepts used in the paper. Analyzed scenario with all basics is described in Section 3. Section 4 provides analysis of fuzzy logic solution, and Section 5 compares fuzzy solution to two types of crisp solutions. We conclude the main points in Section 6.

2. BACKGROUND

In this section we describe and clarify all the main terms and concepts used in the paper.

Description Logic (DL) and Web Ontology Language (OWL). Description Logic is formal knowledge representation language. In terms of expressiveness and efficiency in decision problems it lays between prepositional logic and first-order predicate logic [6]. Web Ontology Language is intended to provide a language that can be used to describe the classes and relations between them that are inherent in Web documents and applications. OWL DL is a sublanguage of OWL that supports maximum expressiveness without losing computational completeness [7].

Semantic Web Rule Language (SWRL). SWRL is a combination of the OWL DL and OWL Lite sublanguages with the Unary/Binary Datalog RuleML sublanguages of the Rule Markup Language. SWRL includes a high-level abstract syntax for Horn-like rules in both the OWL DL and OWL Lite sublanguages of OWL [8].

Protégé. It is a free, open source ontology editor and knowledge-base framework. The Protégé platform allows to model ontologies via the Protégé-OWL editor [9].

Fuzzy logic (FL). FL is a form of multi-valued logic, which is derived from the fuzzy set theory introduced by Zadeh [10]. Fuzzy set extends binary set by adding a degree of membership of an element to a set. If we take for example a variable describing age. We can define several sets describing this variable e.g. young, middle aged, old. In binary set theory a particular value would either belong to one or more of such sets or not. Such information has a limited value. In fuzzy set theory a particular value could belong to each set with different degree of membership. This can provide more valuable information. FL allows using non-numeric linguistic variables. They can facilitate expression of knowledge and rules, as they make them easier to understand. This way one can formulate rules in a form: IF person is old THEN risk of cancer is high.

3. SCENARIO

In this section we present a basic constraints analysis scenario. It will serve as a basis for further comparison between fuzzy and crisp solutions.

Let us assume that a computational job can be described with two parameters: priority and workload. It might seem that this assumption is too simplistic. However, it might happen in many scenarios where there are many different jobs and describing them with bigger detail would be inefficient or impractical. Moreover, the purpose of this assumption is to demonstrate the applicability of SWRL-F and not necessarily to model a precise situation. Nevertheless, with growing amount of parameters, as we later try to demonstrate, relative usability of SWRL-F increases.

The job is to be outsourced to another cloud. We have to determine which of available instances should be used. For the sake of the example we can use vocabulary based on instances available in AWS. One can group set of standard and high-cpu instances and represent them approximately on one axis according to growing computational efficiency, but also growing price. In such a case a relation between priority and workload should determine the instance that will be chosen.

Figure 1. demonstrates simple OWL class implementing description of such a scenario that includes Instance, Priority and Workload.

4. FUZZY SOLUTION

In this section we present solution to the scenario using fuzzy matching in SWRL-F.

In Figure 2. we present definition of Job Priority, in Figure 3. definition of Job Workload and in Figure 4. definition of Job Instance. Those values are defined using FuzzyValue class, which is part of SWRL-F ontology. Priority and Workload are defined using Singleton Fuzzy Set so in fact their definition is crisp. Nevertheless, all the reasoning performed on them is fuzzy due to fuzziness in the rules. This would be a common situation in many applications that provide approximate value for those parameters in order to find appropriate instance basing on fuzzy rules. It bears close similarity with fuzzy control system approach.

Moreover, those values could also be given in as fuzzy sets and it would not change further conclusions. However, the comparison with crisp solution would become increasing more difficult. Modeling complex fuzzy calculations using Math or even Eval functions would become significantly more complicated.

Detailed definition of FuzzyTerms is omitted for the sake of space, but it is fairly straightforward and becomes clear analyzing the rules. In general, Priority and Workload terms are defined basing on typical qualitative values as Urgent, Regular or High, Small respectively. Instance terms are defined basing on example terminology from AWS e.g. m1.xlarge or c1.medium.

Basing on these definitions we can create a set of simple rules interrelating Job Priority and Workload with appropriate Instance. Seven example fuzzy rules are presented in Figure 5.

Let us inspect first three rules with more detail. First rules states that urgent job with small workload should be performed on c1.medium instance.

Job(?j) ∧ hasPriority(?j, ?p) ∧ fuzzymatch(?p, UrgentPriority) ∧ hasWorkload(?j, ?w) ∧ fuzzymatch(?w, SmallWorkload) ∧ hasInstance(?j, ?i) → fuzzymatch(?i, c1.medium)

Second rule states that urgent job with high workload should be performed on c1.xlarge instance.

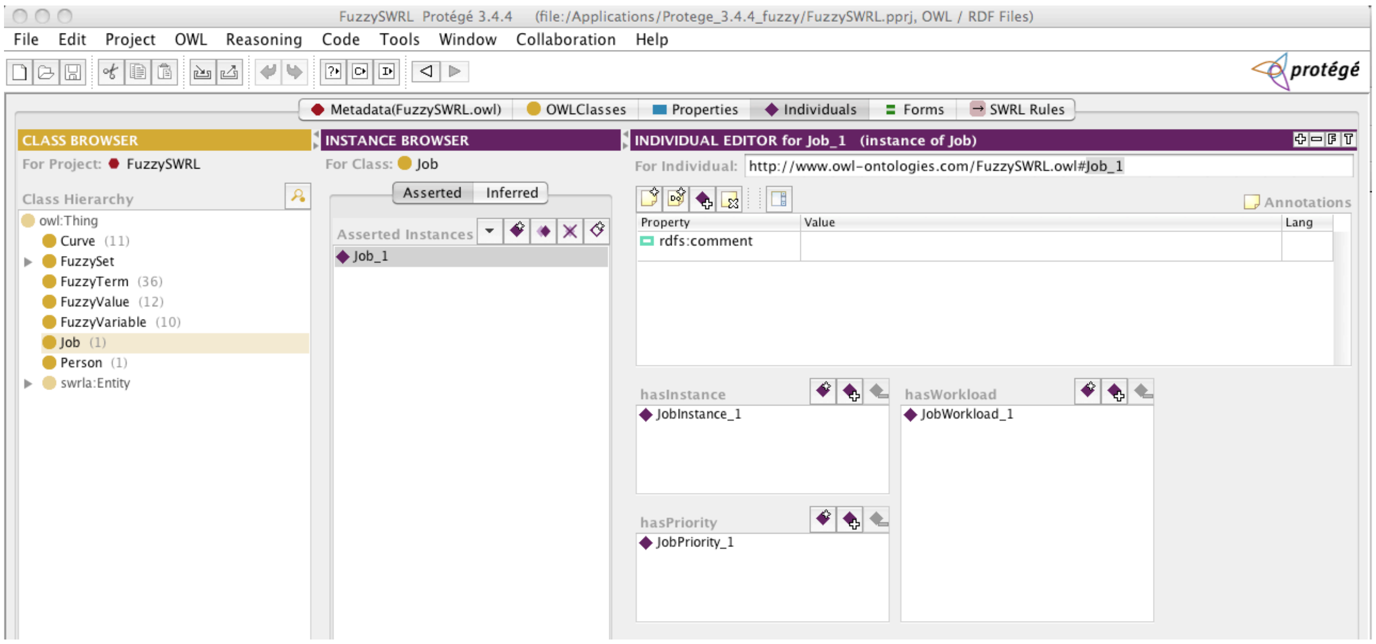
Job(?j) ∧ hasPriority(?j, ?p) ∧ fuzzymatch(?p, UrgentPriority) ∧ hasWorkload(?j, ?w) ∧ fuzzymatch(?w, HighWorkload) ∧ hasInstance(?j, ?i) → fuzzymatch(?i, c1.xlarge)

Third rule states that regular job with high workload should be performed on m1.xlarge instance.

Job(?j) ∧ hasPriority(?j, ?p) ∧ fuzzymatch(?p, RegularPriority) ∧ hasWorkload(?j, ?w) ∧ fuzzymatch(?w, HighWorkload) ∧ hasInstance(?j, ?i) → fuzzymatch(?i, m1.xlarge)

As the actual value can match to some extent more than one term, all three rules can have influence on the final choice of instance if the priority is in between urgent and regular, workload is between high and small. Actually even more rules from Figure 3. could have the influence. Depending on the particular definition of fuzzy sets and terms. This overlapping of rules with fuzzy terms mimics typical human decision-making process and it is automatically handled by SWRL-F implementation. It is important to stress that such overlap is impossible in standard SWRL. Therefore, any potential overlap has to be foreseen and embedded into each rule separately, making the process more difficult and error prone. What we try to demonstrate in the next section.

Moreover, a person that is familiar to some extent with SWRL can easily understand the sense of the rules. Even if he is not familiar with the particular scenario or fuzzy logic at all. Data (or numerical values) are encapsulated in Fuzzy Terms creating clear and readable rules, which separate data from the analysis. The benefit is easier rule management, in particular for large scenarios.


FuzzySWRL Protégé 3.4.4 (file:/Applications/Protege_3.4.4_fuzzy/FuzzySWRL.pprj, OWL / RDF Files)
File Edit Project OWL Reasoning Code Tools Window Collaboration Help
protégé
Metadata(FuzzySWRL.owl) OWLClasses Properties Individuals Forms SWRL Rules
CLASS BROWSER
INSTANCE BROWSER
INDIVIDUAL EDITOR for Job_1 (instance of Job)
Asserted Inferred
Asserted Instances
Annotations
Property Value Lang
For Project: FuzzySWRL

Class Hierarchy
- owl:Thing
  - Curve (11)
  - FuzzySet
  - FuzzyTerm (36)
  - FuzzyValue (12)
  - FuzzyVariable (10)
  - Job (1)
  - Person (1)
  - swrla:Entity

For Class: Job

Job_1

For Individual: http://www.owl-ontologies.com/FuzzySWRL.owl#Job_1

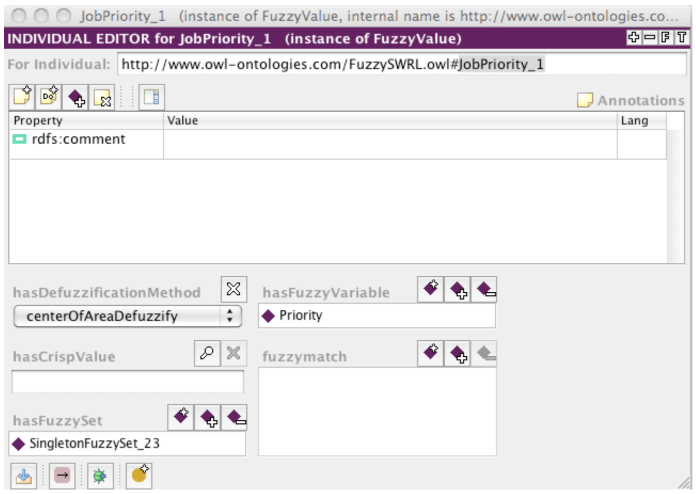
rdfs:comment

hasInstance: JobInstance_1

hasWorkload: JobWorkload_1

hasPriority: JobPriority_1

Figure 1. Basic Job Definition


JobPriority_1 (instance of FuzzyValue, internal name is http://www.owl-ontologies.co...

INDIVIDUAL EDITOR for JobPriority_1 (instance of FuzzyValue)

For Individual: http://www.owl-ontologies.com/FuzzySWRL.owl#JobPriority_1

Annotations

Property	Value	Lang
rdfs:comment		

hasDefuzzificationMethod: centerOfAreaDefuzzify

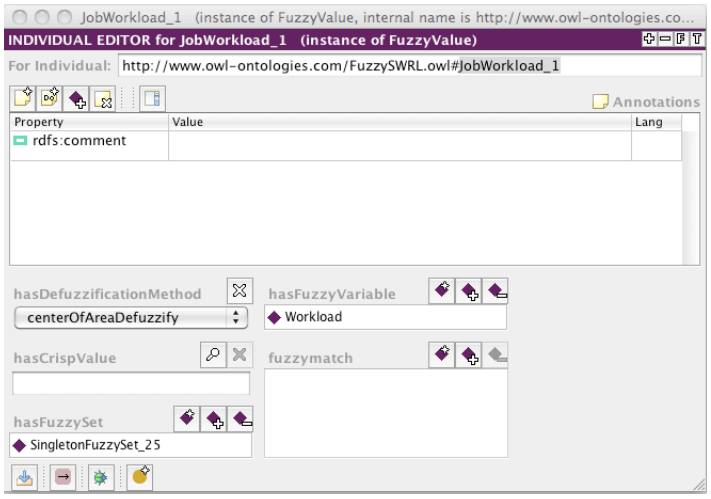
hasFuzzyVariable: Priority

hasCrispValue:

fuzzymatch:

hasFuzzySet: SingletonFuzzySet_23

Figure 2. Job Priority Definition


JobWorkload_1 (instance of FuzzyValue, internal name is http://www.owl-ontologies.co...

INDIVIDUAL EDITOR for JobWorkload_1 (instance of FuzzyValue)

For Individual: http://www.owl-ontologies.com/FuzzySWRL.owl#JobWorkload_1

Annotations

Property	Value	Lang
rdfs:comment		

hasDefuzzificationMethod: centerOfAreaDefuzzify

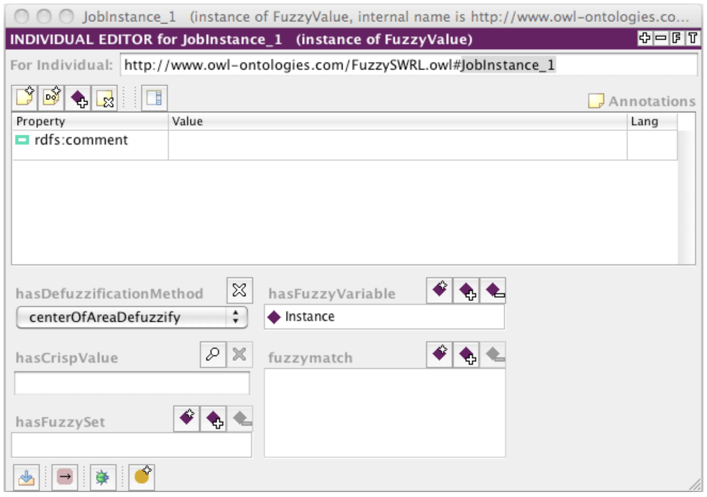
hasFuzzyVariable: Workload

hasCrispValue:

fuzzymatch:

hasFuzzySet: SingletonFuzzySet_25

Figure 3. Job Workload Definition


JobInstance_1 (instance of FuzzyValue, internal name is http://www.owl-ontologies.co...

INDIVIDUAL EDITOR for JobInstance_1 (instance of FuzzyValue)

For Individual: http://www.owl-ontologies.com/FuzzySWRL.owl#JobInstance_1

Annotations

Property	Value	Lang
rdfs:comment		

hasDefuzzificationMethod: centerOfAreaDefuzzify

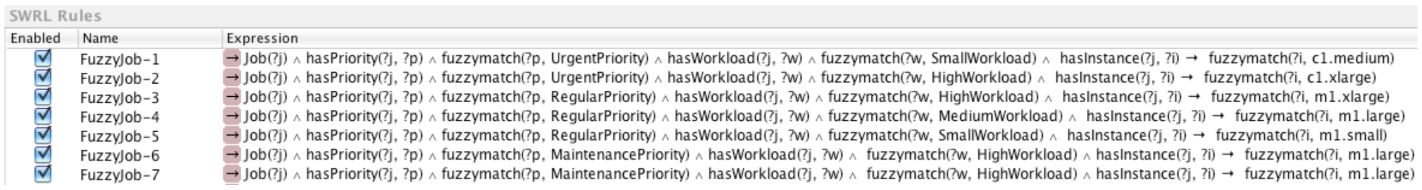
hasFuzzyVariable: Instance

hasCrispValue:

fuzzymatch:

hasFuzzySet:

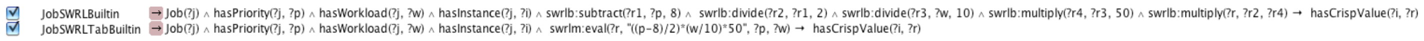
Figure 4. Job Instance Definition



SWRL Rules

Enabled	Name	Expression
✓	FuzzyJob-1	→ Job(?j) ∧ hasPriority(?j, ?p) ∧ fuzzymatch(?p, UrgentPriority) ∧ hasWorkload(?j, ?w) ∧ fuzzymatch(?w, SmallWorkload) ∧ hasInstance(?j, ?i) → fuzzymatch(?i, c1.medium)
✓	FuzzyJob-2	→ Job(?j) ∧ hasPriority(?j, ?p) ∧ fuzzymatch(?p, UrgentPriority) ∧ hasWorkload(?j, ?w) ∧ fuzzymatch(?w, HighWorkload) ∧ hasInstance(?j, ?i) → fuzzymatch(?i, c1.xlarge)
✓	FuzzyJob-3	→ Job(?j) ∧ hasPriority(?j, ?p) ∧ fuzzymatch(?p, RegularPriority) ∧ hasWorkload(?j, ?w) ∧ fuzzymatch(?w, HighWorkload) ∧ hasInstance(?j, ?i) → fuzzymatch(?i, m1.xlarge)
✓	FuzzyJob-4	→ Job(?j) ∧ hasPriority(?j, ?p) ∧ fuzzymatch(?p, RegularPriority) ∧ hasWorkload(?j, ?w) ∧ fuzzymatch(?w, MediumWorkload) ∧ hasInstance(?j, ?i) → fuzzymatch(?i, m1.large)
✓	FuzzyJob-5	→ Job(?j) ∧ hasPriority(?j, ?p) ∧ fuzzymatch(?p, RegularPriority) ∧ hasWorkload(?j, ?w) ∧ fuzzymatch(?w, SmallWorkload) ∧ hasInstance(?j, ?i) → fuzzymatch(?i, m1.small)
✓	FuzzyJob-6	→ Job(?j) ∧ hasPriority(?j, ?p) ∧ fuzzymatch(?p, MaintenancePriority) ∧ hasWorkload(?j, ?w) ∧ fuzzymatch(?w, HighWorkload) ∧ hasInstance(?j, ?i) → fuzzymatch(?i, m1.large)
✓	FuzzyJob-7	→ Job(?j) ∧ hasPriority(?j, ?p) ∧ fuzzymatch(?p, MaintenancePriority) ∧ hasWorkload(?j, ?w) ∧ fuzzymatch(?w, HighWorkload) ∧ hasInstance(?j, ?i) → fuzzymatch(?i, m1.large)

Figure 5. Fuzzy Rules



✓	JobSWRLBuiltin	→ Job(?j) ∧ hasPriority(?j, ?p) ∧ hasWorkload(?j, ?w) ∧ hasInstance(?j, ?i) ∧ swrlb:subtract(?r1, ?p, 8) ∧ swrlb:divide(?r2, ?r1, 2) ∧ swrlb:divide(?r3, ?w, 10) ∧ swrlb:multiply(?r4, ?r3, 50) ∧ swrlb:multiply(?r, ?r2, ?r4) → hasCrispValue(?i, ?r)
✓	JobSWRLTabBuiltin	→ Job(?j) ∧ hasPriority(?j, ?p) ∧ hasWorkload(?j, ?w) ∧ hasInstance(?j, ?i) ∧ swrlm:eval(?r, "((p-8)/2)*(w/10)*50", ?p, ?w) → hasCrispValue(?i, ?r)

Figure 6. Crisp Rules

5. CRISP SOLUTION

In this section we present alternative crisp solutions the simulate the first two fuzzy rules.

In Figure 4. One can see two rules. First rule utilizes standard SWRL Math builtin, second rule swrlm builtin of SWRLTab. Numerical values used in the following two rules correspond with definition of fuzzy sets from the Section 4. Let us look at each of those rules in detail.

5.1. SWRL Math Builtin

The following rule utilizes standard SWRL Math Builtin.

Job(?j) ∧ hasPriority(?j, ?p) ∧ hasWorkload(?j, ?w) ∧ hasInstance(?j, ?i) ∧ swrlb:subtract(?r1, ?p, 8) ∧ swrlb:divide(?r2, ?r1, 2) ∧ swrlb:divide(?r3, ?w, 10) ∧ swrlb:multiply(?r4, ?r3, 50) ∧ swrlb:multiply(?r, ?r2, ?r4) → hasCrispValue(?i, ?r)

It intends to represent calculations similar to the first two rules from the fuzzy example. However, it is greatly simplified as expressing fuzzification and deffuzification using Math builtin would be too difficult to be worth the effort.

The problem is the necessity to explicitly contain all the calculations inside the rule. Using Math Builtin we need to apply several different operations to provide similar result as in SWRL-F with only one operation. To achieve exactly the same result as in SWRL-F it would require even more calculations in the body of the rule. This challenge was not attempted here, as it starts to reach out of the scope of the paper.

Moreover, rules based on Math Builtin have to contain all constraints related to the Job Instance in one rule. That could be divided into separate rules in SWRL-F. The result is additional growth in complexity of the rule, which might effectively prevent constructing rules with more than a couple of constraints.

5.2. SWRLTab Builtin

The following rule utilizes SWRLTab builtin.

Job(?j) ∧ hasPriority(?j, ?p) ∧ hasWorkload(?j, ?w) ∧ hasInstance(?j, ?i) ∧ swrlm:eval(?r, "((p-8)/2)*(w/10)*50", ?p, ?w) → hasCrispValue(?i, ?r)

It makes the calculations much simpler thanks to eval function. This way five functions from SWRL Math Builtin can be replaced with only one function. That can help significantly with including more constraints for choosing each Job Instance.

However, this approach still requires putting all constraints calculations relating to particular Job Instance in one rule. That is particularly problematic in a setting where one constraint can influence many types of instances at the same time.

Moreover, any approach based on explicit calculations in rule body is relatively difficult to understand without knowing its meaning upfront. It is due to usage of numerical values which origin is not explicitly given in the rule.

5.3. Evaluation

As one can notice crisp rules are more difficult to create in the constraints analysis application. They usually require writing longer rules, which make modeling more complex scenarios particularly difficult. Moreover, they are more difficult to understand without an a priori knowledge, as they include explicit usage of numerical values.

They provide more limited functionality as all the constraints relating to a particular variable have to be contained in one rule. This has additional negative influence on creation process.

On the other hand, fuzzy rules require more upfront work with terms definition that was not pictured in Section 3. Basing on this initial comparison it seems that SWRL-F can improve on constraints analysis process by providing simple yet powerful tool. In particular, in the context where decision logic should be encapsulated in the ontology.

6. CONCLUSIONS

In this paper we proposed application of SWRL-F to constraints analysis in intercloud outsourcing. We presented basic ontology with a set of simple rules that provided expressive power that seemed appropriate to the task.

SWRL-F was demonstrated as a useful tool for constraints analysis in intercloud scenario if the decision logic should be encapsulated in the ontology. It provided natural, expressive and manageable way do defined large sets of interrelated constraints.

The alternative implementations using other SWRL bultins were showed to be more complex to understand and less expressive. Inability to separate decision logic from numerical values and to separate interrelated constraints into several rules made them difficult to apply in modeling of large scenarios.

This study provides only initial study of this application of SWRL-F. Promising results suggest the need for extending the work.

REFERENCES

[1] David Bernstein, Erik Ludvigson, Krishna Sankar, Steve Diamond, Monique Morrow, "Blueprint for the Intercloud - Protocols and Formats for Cloud Computing Interoperability," iciw, pp.328-336, 2009 Fourth International Conference on Internet and Web Applications and Services, 2009

[2] Steven E. Czerwinski, Ben Y. Zhao, Todd D. Hodes, Anthony D. Joseph, and Randy H. Katz. 1999. An architecture for a secure service discovery service. In Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking (MobiCom '99). ACM, New York, NY, USA, 24-35.

[3] Matthias Klusch, Benedikt Fries, and Katia Sycara. 2006. Automated semantic web service discovery with OWLS-MX. In Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems (AAMAS '06). ACM, New York, NY, USA, 915-922.

[4] Ulrich Kuster, Birgitta Kunig-Ries, Mirco Stern, and Michael Klein. 2007. DIANE: an integrated approach to automated service discovery, matchmaking and composition. In Proceedings of the 16th international conference on World Wide Web (WWW '07). ACM, New York, NY, USA, 1033-1042.

[5] Tomasz Wiktor Wlodarczyk, Chunming Rong, and Kari Anne Thorsen. 2009. Industrial Cloud: Toward Inter-enterprise Integration. In Proceedings of the 1st International Conference on Cloud Computing (CloudCom '09), Martin Gilje Jaatun, Gansen Zhao, and Chunming Rong (Eds.). Springer-Verlag, Berlin, Heidelberg, 460-471

[6] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, The Description Logic Handbook: Theory, Implementation and Applications, Cambridge University Press, 2003.

[7] "OWL Web Ontology Language Guide" Available: http://www.w3.org/TR/owl-guide/.

[8] "SWRL: A Semantic Web Rule Language Combining OWL and RuleML" Available: http://www.w3.org/Submission/SWRL/.

[9] "The Protégé Ontology Editor and Knowledge Acquisition System" Available: http://protege.stanford.edu/.

[10] L. Zadeh, "Fuzzy sets," pp. 338–353.

[11] T. W. Wlodarczyk, M. O'Connor, C. Rong, and M. A. Musen, "SWRL-F - A Fuzzy Logic Extension of the Semantic Web Rule Language" in CEUR Workshop Proceedings of 6th International Workshop on Uncertainty Reasoning for the Semantic Web (URSW), Shanghai, China, 2010

[12] T. W. Wlodarczyk, M. O'Connor, C. Rong, and M. A. Musen, "SWRL-F - A Fuzzy Logic Extension of the Semantic Web Rule Language" accepted at The International Conference on Web Intelligence, Mining and Semantics, Sogndal, Norway 2011